

Universidad Católica de Salta
Facultad de Ingeniería



Reconocimiento de Entidades en Relatos de
Denuncias mediante Aprendizaje Automático
y Procesamiento de Lenguaje Natural

Gustavo Ariel Rivera Madrazo
Ingeniería en Informática

Año: 2021

Título

Ingeniero en Informática

Profesor Guía

Nombre: Dr. Eduardo Xamena

Firma: _____

Tribunal Evaluador

Presidente:

Firma: _____

I Miembro:

Firma: _____

II Miembro:

Firma: _____

Alumno

Nombre: Gustavo Ariel Rivera Madrazo

Firma: _____

Fecha de Exposición

_____/_____/_____

Dedicatorias y Agradecimientos

En primer lugar, a Dios, por su presencia infalible en cada momento de mi vida y por regalarme la dicha de finalizar una etapa más del camino.

Al Dr. Eduardo Xamena por resolver todas mis dudas, aconsejarme y guiarme a lo largo del proyecto de grado con su conocimiento, motivación y gran calidad humana.

A la universidad, sus autoridades y profesores de la Facultad de Ingeniería, por las valiosas enseñanzas que me fueron moldeando para llegar a ser un profesional.

A mis padres y hermanos por su confianza y apoyo constantes en el transcurso de la carrera y en toda mi vida.

A mis compañeros de cursada y amigos, por compartir la alegría de los logros conseguidos y brindarme su apoyo en las dificultades.

ÍNDICE DE CONTENIDO

ABSTRACT	4
1 INTRODUCCIÓN.....	5
1.1 DESCRIPCIÓN DEL PROBLEMA	5
1.2 MOTIVACIÓN	5
1.3 ORGANIZACIÓN DEL TRABAJO	5
1.4 CRITERIOS DE ÉXITO	6
2 ESTADO DE LA CUESTIÓN	7
2.1 ANTECEDENTES	7
2.2 CONCEPTOS PREVIOS	8
2.2.1 Metodología CRISP-DM.....	8
2.2.2 Aprendizaje Automático.....	9
2.2.2.1 Aprendizaje supervisado.....	10
2.2.2.2 Redes Neuronales	11
2.2.2.3 Aprendizaje Profundo.....	11
2.2.2.4 Evaluación	12
2.2.3 Procesamiento del Lenguaje Natural.....	14
2.2.4 Extracción de Información	14
2.2.4.1 Proceso de anotación	15
2.2.4.2 Reconocimiento de Entidades Nombradas	15
2.2.4.3 Desambiguación de Entidades Nombradas y Vinculación.....	15
2.2.4.4 Extracción de Relaciones.....	16
2.2.5 Conceptos de spaCy	16
3 PROBLEMA Y SOLUCIÓN PROPUESTA	19
3.1 OBJETIVOS Y ALCANCES	19
3.1.1 Objetivo General.....	19
3.1.2 Objetivos específicos.....	19
3.1.3 Alcances del trabajo.....	19
3.2 RESTRICCIONES Y ESPECIFICACIONES	20
3.3 SOLUCIÓN PROPUESTA	20
3.4 HERRAMIENTAS UTILIZADAS	20
3.5 ALTERNATIVAS TECNOLÓGICAS DE SOLUCIÓN AL PROBLEMA.....	25
3.6 ARQUITECTURA	26
3.7 METODOLOGÍA	27
3.7.1 Fase 1: Comprensión del negocio.....	27
3.7.1.1 Objetivos del proyecto.....	27
3.7.1.2 Evaluación de la situación	27
3.7.1.3 Plan de proyecto	27
3.7.2 Fase 2: Comprensión de los datos	27
3.7.2.1 Recolección de los datos.....	27
3.7.2.2 Descripción de los datos	27
3.7.2.3 Exploración de los datos.....	28
3.7.3 Fase 3: Preparación de los datos.....	30
3.7.3.1 Selección de los datos.....	30
3.7.3.2 Anotación de los datos.....	30
3.7.3.3 División y formateo de los datos	31
3.7.4 Fase 4: Modelado	34
3.7.4.1 Técnica de modelado	34
3.7.4.2 Construcción de modelos	36

3.7.4.3 Evaluación de modelos	39
3.7.5 Fase 5: Evaluación	40
3.7.5.1 Evaluación de resultados	40
3.7.5.2 Revisión del proceso	50
3.7.6 Fase 6: Implementación	51
3.7.6.1 Desarrollo del sistema	51
3.7.6.2 Anonimización de relatos	51
3.7.6.3 Informe final del proyecto	52
3.8 SISTEMA DESARROLLADO	52
3.8.1 Pestaña INICIO	52
3.8.2 Pestaña DENUNCIAS	53
3.8.3 Pestaña VISUALIZACIONES	56
3.8.4 Modelo Utilizado	58
3.9 ANÁLISIS FODA	59
3.9.1 Estrategias	60
3.10 PLAN DE PROYECTO	62
3.10.1 Estructura de Desglose de Trabajo	62
3.10.2 Lista de Actividades	66
3.10.3 Recursos Requeridos	67
3.10.4 Cronograma del Proyecto	69
3.11 GESTIÓN DE COSTOS Y ADQUISICIONES	71
3.11.1 Costos de Recursos Humanos	71
3.11.2 Costos de Hardware	72
3.11.3 Costos de Software	73
3.11.4 Línea Base de Costos	73
3.11.4.1 Análisis Costo-Beneficio	74
3.12 GESTIÓN DE RIESGOS	76
3.13 ANÁLISIS DE FACTIBILIDAD	77
4 CONCLUSIONES Y FUTURAS LÍNEAS DE INVESTIGACIÓN	79
BIBLIOGRAFÍA	81
ANEXO 1 – TERMINOLOGÍA	83
ANEXO 2 – INSTALACIÓN Y EMPLEO DE BRAT	84
INSTALAR WSL UBUNTU 20.04	84
INSTALAR BRAT	84
AGREGAR RELATOS PARA ANOTAR	85
ANOTAR RELATOS	85
LUEGO DE ANOTAR LOS RELATOS	86
CAMBIAR TIPOS DE ENTIDADES	86
ANEXO 3 – PAQUETES UTILIZADOS	87
FRONTEND – LADO DEL CLIENTE	87
BACKEND – LADO DEL SERVIDOR	88
ANEXO 4 – SCRIPTS DE PYTHON	90
SCRIPT GENERAR_TXTS.PY	90
SCRIPT ELIMINAR_TXTS_SIN_ARCHIVO_ANN.PY	91
SCRIPT CONTAR_TOTAL_DE_ENTIDADES.PY	91
SCRIPT DETERMINAR_MEJOR_SEED_PARA_BALANCEO.PY	92
SCRIPT DIVIDIR_RELATOS_TRAIN_VALID_TEST.PY	96
SCRIPT BRAT_ANNOTATION_TO_SPACY_ANNOTATION.PY	97

SCRIPT GENERARARCHIVOSPACY.PY	98
SCRIPT MATRIZDeCONFUSION.PY	99
ANEXO 5 – ARCHIVO CONFIG.CFG MODELO 5	103

ÍNDICE DE IMÁGENES

Imagen 1: Fases del modelo de referencia CRISP-DM.....	9
Imagen 2: Proceso de anotación, entrenamiento y validación – Aprendizaje Supervisado	10
Imagen 3: Ciclo de entrenamiento de un modelo.....	12
Imagen 4: Resumen que genera el buscador de Google al buscar "Jorge Rojas"	14
Imagen 5: Logotipo de Windows 10	20
Imagen 6: Logotipos de Windows, Linux y Ubuntu	21
Imagen 7: Logotipo de MongoDB	21
Imagen 8: Logotipos de Python, Flask y spaCy	22
Imagen 9: Ejemplo de anotación en Brat.....	22
Imagen 10: Logotipo de React	23
Imagen 11: Logotipo de Recharts.....	23
Imagen 12: Logotipo de Visual Studio Code	23
Imagen 13: Logotipos de Git y GitHub.....	24
Imagen 14: Logotipo de Heroku.....	24
Imagen 15: Logotipo de Netlify	25
Imagen 16: Arquitectura del Proyecto.....	26
Imagen 17: Histograma del largo de los relatos de denuncias	29
Imagen 18: Zoom al histograma del largo de los relatos de denuncias	30
Imagen 19: Formato JSON para relatos utilizado por spaCy 2	33
Imagen 20: Estructura del modelo entrenado	35
Imagen 21: Apartado Initialize del archivo config.cfg.....	36
Imagen 22: Configuración de métricas en el archivo config.cfg.....	39
Imagen 23: Resultados de entrenamiento y validación del modelo 1	40
Imagen 24: Resultados de entrenamiento y validación del modelo 2	41
Imagen 25: Resultados de entrenamiento y validación del modelo 3	42
Imagen 26: Resultados de entrenamiento y validación del modelo 4	42
Imagen 27: Resultados de entrenamiento y validación del modelo 5	43
Imagen 28: Resultados de entrenamiento y validación del modelo 6	43
Imagen 29: Resultados de entrenamiento y validación del modelo 7	44
Imagen 30: Resultados de entrenamiento y validación del modelo 8	44
Imagen 31: Resultados de entrenamiento y validación del modelo 9	45
Imagen 32: Resultados de entrenamiento y validación del modelo 10	46
Imagen 33: Resultados de entrenamiento y validación del modelo 11	46
Imagen 34: Resultados de entrenamiento y validación del modelo 12	47
Imagen 35: Matriz de confusión del modelo 5	48
Imagen 36: Matriz de confusión del modelo 4	49
Imagen 37: Matriz de confusión del modelo 12	49
Imagen 38: Sistema web - Pestaña Inicio - Presentación	52
Imagen 39: Sistema web - Pestaña Inicio - Datos del Alumno	53
Imagen 40: Sistema web - Pestaña Inicio - Funcionalidades	53
Imagen 41: Sistema web - Pestaña Denuncias – Listado de denuncias.....	54
Imagen 42: Sistema web - Pestaña Denuncias - Cargar denuncia.....	54
Imagen 43: Sistema web - Pestaña Denuncias - Texto de una denuncia.....	54
Imagen 44: Sistema web - Pestaña Denuncias - Entidades de una denuncia	55
Imagen 45: Sistema web - Pestaña Denuncias - Resumen de una denuncia	55
Imagen 46: Sistema web - Pestaña Visualizaciones – Gráfico de Torta	56
Imagen 47: Sistema web - Pestaña Visualizaciones – Visualizaciones disponibles.....	56
Imagen 48: Sistema web - Pestaña Visualizaciones – Gráfico de Barra	57
Imagen 49: Sistema web - Pestaña Visualizaciones – Gráfico de Línea.....	57

Imagen 50: Sistema web - Pestaña Visualizaciones – Gráfico Mapa de Árbol	58
Imagen 51: Estructura de Desglose del Trabajo para el proyecto	62
Imagen 52: Listado de tareas con sus duraciones y diagrama de Gantt – Fases 1, 2 y 3	69
Imagen 53: Listado de tareas con sus duraciones y diagrama de Gantt – Fases 4, 5 y 6	70
Imagen 54: Listado de tareas con sus duraciones, fechas, relaciones y recursos – Fases 1, 2 y 3	70
Imagen 55: Listado de tareas con sus duraciones, fechas, relaciones y recursos – Fases 1, 2 y 3	70

ÍNDICE DE TABLAS

Tabla 1: Diferentes resultados posibles en una predicción de dos clases.....	13
Tabla 2: Número ideal de entidades de cada tipo para los conjuntos de entrenamiento, validación y prueba.....	32
Tabla 3: Número de entidades de cada tipo para los conjuntos de entrenamiento, validación y prueba utilizados.	32
Tabla 4: Configuraciones de modelos entrenados	38
Tabla 5: Resumen de puntajes obtenidos para los 12 modelos entrenados	47
Tabla 6: Precision, recall y F1 score por tipo de entidad del modelo 5.....	48
Tabla 7: Recursos Humanos requeridos para el proyecto	68
Tabla 8: Costo estimado de Recursos Humanos	72
Tabla 9: Costo estimado de Hardware.....	73
Tabla 10: Costos del Proyecto en pesos argentinos.....	74
Tabla 11: Costos del Proyecto en dólares.....	74
Tabla 12: Flujo de fondos del proyecto	76
Tabla 13: Análisis de riesgos del Proyecto.....	77

ABSTRACT

En el presente trabajo se desarrolló un sistema de reconocimiento y visualización de entidades en relatos de denuncias provistas por el Ministerio Público Fiscal de la provincia de Salta. Los cinco tipos de entidades identificadas en los relatos son: persona, barrio, Documento Nacional de Identidad (DNI), teléfono y patente.

Se emplearon tecnologías web para el desarrollo del sistema y una librería de Python para procesamiento del lenguaje natural, de nombre spaCy, con la cual se entrenó un modelo cuya tarea es analizar los relatos suministrados y extraer las entidades de los tipos mencionados en el párrafo anterior.

Los resultados obtenidos cumplen con los objetivos y criterios de éxito propuestos, presentando así la posibilidad de implementación del sistema, o al menos del modelo entrenado, en fiscalías y dependencias policiales.

1 INTRODUCCIÓN

1.1 Descripción del problema

En la actualidad, la gestión de la información relacionada con reportes de eventos delictivos es llevada a cabo mayormente de forma manual en las distintas dependencias de la justicia del ámbito salteño. Desde las búsquedas de información puntual acerca de un sospechoso, hasta la clasificación de un reporte por tipo de delito, requieren un esfuerzo cognitivo puramente humano. Al día de hoy, sin embargo, existen herramientas y metodologías que pueden ayudar a los agentes de la justicia en estas tareas, haciéndolas mucho más eficientes.

El trabajo a realizar busca aportar una herramienta de reconocimiento y visualización de entidades, dirigida en primera instancia al Ministerio Público Fiscal (MPF), pero con potencial utilidad para otras dependencias del ámbito judicial y policial. En el MPF, entre fiscalías y dependencias policiales, se reciben alrededor de 350 denuncias por día, resultando sumamente complejo poder realizar un análisis de todas ellas solamente con personas.

Las entidades reconocidas, extraídas, almacenadas y utilizadas por el sistema desarrollado incluyen: personas, barrios, números de DNI/CUIL/CUIT, patentes de vehículos y números de teléfono. Dichas entidades, se destacan en los relatos de las denuncias, para que el usuario pueda realizar las correcciones que considere necesarias. Además, se generan resúmenes y gráficos sobre la cantidad de entidades encontradas de cada tipo. Estos datos, pueden ser utilizados por los usuarios como información que apoye a la toma de decisiones.

1.2 Motivación

En un mundo, donde lo que sobra es información, resultan imprescindibles visualizaciones adecuadas que faciliten la interpretación de tantos datos. Una visualización bien lograda permite reducir el tiempo y el esfuerzo requeridos para identificar datos de interés. Por ello, se considera de suma importancia la existencia de herramientas de visualización flexibles que faciliten el trabajo de las personas, en este caso, de los policías, fiscales y abogados, que día a día deben lidiar con numerosos documentos judiciales en una tarea que muchas veces resulta tediosa y que podría verse beneficiada por la solución aquí propuesta.

1.3 Organización del trabajo

El trabajo desarrollado ha sido organizado de la siguiente manera:

En el capítulo 2 se menciona el estado del arte, los antecedentes de tecnologías similares a la propuesta en el proyecto. También, se desarrolla un breve marco teórico con los conceptos previos necesarios para abordar el documento.

En el capítulo 3, en las secciones 3.1 y 3.2, se definen los objetivos y restricciones del proyecto. Entre las secciones 3.3 a 3.8 se desarrolla la parte técnica del proyecto, indicando la solución propuesta, herramientas utilizadas, alternativas a dichas herramientas, arquitectura del proyecto y cada una de las fases de la metodología de trabajo elegida junto con el sistema web desarrollado para implementar el modelo entrenado. Entre las secciones

3.9 a 3.13 se presenta la planificación del proyecto incluyendo un análisis FODA, plan del proyecto, actividades, recursos, costos, beneficios, riesgos y análisis de factibilidad.

En el capítulo 4, se presentan las conclusiones a las que se arribó y las líneas de investigación propuestas a futuro, para dar continuidad al trabajo aquí iniciado.

1.4 Criterios de éxito

Los criterios de éxito del presente proyecto son:

- Construir un sistema para la búsqueda y visualización de entidades extraídas de relatos de denuncias, cuyo rendimiento alcance o supere un puntaje del 70%, en la métrica escogida, sobre los relatos de prueba empleados.
- Seguir adecuadamente la metodología propuesta.
- Conseguir que el costo-beneficio del proyecto permita recuperar la inversión en un plazo menor o igual a 3 años.

2 ESTADO DE LA CUESTIÓN

2.1 Antecedentes

En la actualidad existen diversas herramientas empleadas para la visualización de datos como lo son Tableau¹ y Power BI², u otras que implementan Procesamiento de Lenguaje Natural (PLN), Recuperación de Información (RI) o Extracción de Información (EI) ya sean traductores automáticos de texto, asistentes conversacionales, buscadores, anotadores, generadores de resúmenes, analizadores de sentimientos, entre otros. Incluso en el ámbito judicial o legal, encontramos el software Prometea desarrollado por el Ministerio Público Fiscal de la Ciudad Autónoma de Buenos Aires y la Facultad de Derecho de la UBA. Sin embargo, la cantidad de herramientas que realicen ambas cosas, es decir, aplicar algunas de estas técnicas, puntualmente la extracción de información de un texto (entidades) y luego plasmarla en gráficas interactivas es mucho menor.

Es necesario mencionar el trabajo realizado por The Information Interfaces Group, un grupo de investigadores sobre Interacción Humano-Computadora o HCI por sus siglas en inglés, en la Escuela de Computación Interactiva de Georgia Tech, Estados Unidos. Este grupo ya ha realizado más de 10 proyectos, todos expuestos en su página web³ sobre visualización y extracción de información. Dos de los más recientes e interesantes son:

- Jigsaw: es un sistema de análisis visual que ayuda a los analistas e investigadores a explorar, analizar y dar sentido a colecciones de documentos mediante una serie de visualizaciones que representan diferentes aspectos de los mismos. Se enfoca particularmente en presentar las entidades importantes identificables (personas, lugares, organizaciones, etc.) y sus conexiones directas o indirectas. El procesamiento textual extrae las entidades importantes de los documentos y luego las visualizaciones ayudan al analista a explorar las relaciones y conexiones entre las entidades. El sistema incluye una variedad de visualizaciones como listas, gráficos, vistas temporales y basadas en conexiones, así como vistas del texto de un documento individual y la colección de documentos en su conjunto. La última versión del software, la 0.54 del 4 de marzo de 2016, está disponible para su descarga en la página web⁴.
- NL4DV⁵: conjunto de herramientas para la visualización de datos basada en lenguaje natural. Es un paquete de Python que toma como entrada un conjunto de datos tabular y una consulta en lenguaje natural sobre ese conjunto de datos. En respuesta, el kit de herramientas devuelve una especificación analítica modelada como un objeto JSON que contiene atributos de datos, tareas analíticas y una lista de especificaciones Vega-Lite relevantes para la consulta de entrada. Al hacerlo, NL4DV ayuda a los desarrolladores de visualización que pueden no tener experiencia en PLN, lo que les permite crear nuevas ILN (Interfaces de Lenguaje Natural) de visualización o incorporar una entrada de lenguaje natural dentro de sus sistemas existentes (Narechania, Srinivasan, & Stasko, 2020).

¹ <https://www.tableau.com/es-es>

² <https://powerbi.microsoft.com/es-es/>

³ <https://www.cc.gatech.edu/gvu/ii/index.html>

⁴ <https://www.cc.gatech.edu/gvu/ii/jigsaw/>

⁵ <https://www.cc.gatech.edu/~john.stasko/papers/infovis20-nl4dv.pdf>

- Media Monitoring of the Past⁶: es una suite de monitoreo de medios para periódicos históricos impulsada por objetivos de investigación en lingüística computacional, historia y diseño. El marco tecnológico desarrollado permite extraer, procesar, vincular y explorar datos de archivos de medios impresos a través de un conjunto de herramientas de procesamiento del lenguaje natural (NLP) dedicadas a los medios impresos históricos, interfaces de visualización para la exploración activa y el análisis crítico de los corpus de periódicos de manera transparente.

2.2 Conceptos previos

2.2.1 Metodología CRISP-DM

La metodología CRISP-DM⁷ (Chapman, y otros, 2000) provee un ciclo de vida iterativo para procesos de minería de datos independientes del dominio de aplicación y las tecnologías empleadas.

Esta metodología cubre las etapas de un proyecto, sus tareas respectivas y las relaciones entre estas tareas a través de las 6 fases identificadas en la imagen 1. La secuencia de fases puede variar, los resultados obtenidos como salida de una fase determinan qué tareas realizar. Las flechas indican las dependencias más importantes y frecuentes.

Las seis fases son:

- Fase 1, Comprensión del negocio
Busca comprender los objetivos del proyecto y requerimientos desde el punto de vista del negocio que dan origen a la necesidad de utilizar la minería de datos. En esta fase se definen los objetivos a perseguir y se elabora un plan de proyecto.
- Fase 2, Comprensión de los datos
Comienza con la colección de datos inicial y continúa con las actividades que permiten familiarizarse con los datos, identificar los problemas de calidad, descubrir conocimiento preliminar sobre los datos, y/o descubrir subconjuntos interesantes para formar hipótesis en cuanto a la información oculta.
- Fase 3, Preparación de los datos
Cubre todas las actividades necesarias para construir el conjunto final de datos (los datos que se utilizarán en las herramientas de modelado) a partir de los datos brutos iniciales. Las tareas incluyen la selección de tablas, registros y atributos, así como la transformación y la limpieza de datos para las herramientas que modelan.
- Fase 4, Modelado
Se seleccionan y aplican las técnicas de modelado que sean pertinentes al problema (cuanto más, mejor), y se calibran sus parámetros a valores óptimos. Típicamente hay varias técnicas para el mismo tipo de problema de minería de datos. Algunas técnicas tienen requerimientos específicos sobre la forma de los datos. Por

⁶ impresso. Media Monitoring of the Past. Supported by the Swiss National Science Foundation under grant CR- SII5_173719, 2019. <https://impresso-project.ch>.

⁷ <https://the-modeling-agency.com/crisp-dm.pdf>

lo tanto, casi siempre en cualquier proyecto se acaba volviendo a la fase de preparación de datos.

- Fase 5, Evaluación

En esta etapa del proyecto, se han construido uno o varios modelos que parecen alcanzar calidad suficiente desde una perspectiva de análisis de datos. Antes de proceder al despliegue final del modelo, es importante evaluar a fondo y revisar los pasos ejecutados para crearlo, comparar el modelo obtenido con los objetivos de negocio. Un objetivo clave es determinar si hay una cuestión importante de negocio que no haya sido considerada suficientemente. Al final de esta fase, se debería obtener una decisión sobre la aplicación de los resultados del proceso de análisis de datos.

- Fase 6, Implementación

Consiste en la aplicación del modelo dentro de los procesos de negocio de una organización. Incluye el desarrollo de un plan de despliegue, su ejecución o supervisión y la elaboración de un reporte final con los resultados y la documentación del proceso completo. A fin de utilizar el modelo, se elaboró una aplicación web. Ésta se encuentra detallada en la sección 3.8.

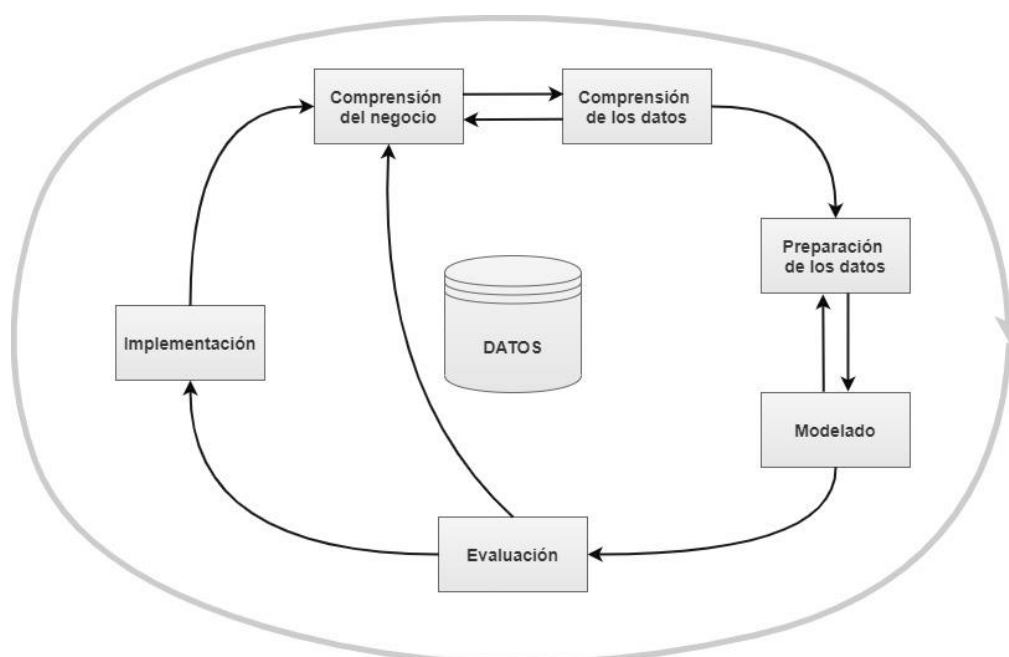


Imagen 1: Fases del modelo de referencia CRISP-DM

2.2.2 Aprendizaje Automático

El aprendizaje automático consiste en diseñar algoritmos que extraigan automáticamente información valiosa de un set de datos. El énfasis aquí está en lo "automático", es decir, el aprendizaje automático se preocupa por las metodologías de propósito general que se pueden aplicar a muchos conjuntos de datos, mientras se produce algo significativo. Hay tres conceptos que están en el centro del aprendizaje automático: datos, un modelo y aprendizaje (Deisenroth, Faisal, & Ong, 2020, pág. 11).

Dado que el aprendizaje automático se basa intrínsecamente en datos, estos son el núcleo del aprendizaje automático. El objetivo del aprendizaje automático es diseñar metodologías de propósito general para extraer patrones valiosos de los datos, idealmente sin mucha experiencia en un dominio específico. Por ejemplo, dado un gran corpus de documentos (por ejemplo, libros en muchas bibliotecas), los métodos de aprendizaje automático se pueden usar para encontrar automáticamente temas relevantes que se comparten entre documentos (Hoffman, Blei, & Back, 2010). Para lograr este objetivo, se diseñaron modelos que suelen estar relacionados con el proceso que genera los datos, similares al conjunto de datos que se proporciona. Por ejemplo, en una configuración de regresión, el modelo describiría una función que mapea entradas a salidas de valor real. Parafraseando a Mitchell (1997) :Se dice que un modelo aprende de los datos si su desempeño en una tarea dada mejora después de que se toman en cuenta los datos. El objetivo es encontrar buenos modelos que generalicen bien los datos aún no vistos, que pueden interesarnos en el futuro. El aprendizaje puede entenderse como una forma de encontrar automáticamente patrones y estructura en los datos optimizando los parámetros del modelo (Deisenroth, Faisal, & Ong, 2020, pág. 11).

Otra definición posible de Aprendizaje Automático enuncia: “El aprendizaje automático es una disciplina en la que definimos un programa, no escribiéndolo enteramente nosotros mismos, sino aprendiendo de los datos” (Howard & Gugger, 2020, pág. 112)

2.2.2.1 Aprendizaje supervisado

Hay tres clases principales de técnicas de aprendizaje automático: aprendizaje supervisado, aprendizaje no supervisado y aprendizaje reforzado. En el aprendizaje supervisado, el conjunto de entrenamiento consta de pares de entrada y salida deseada, y el objetivo es aprender un mapeo entre los espacios de entrada y salida (Simeone, 2018, pág. 2).

En este proyecto se realizan tareas de reconocimiento de entidades. Para este tipo de tareas suele emplearse aprendizaje supervisado, indicándole al modelo cuales son las entidades que debería reconocer (salida) en un texto dado (entrada). Los pares de entrada y salida deseada se obtienen mediante el proceso de anotación y luego se suministran al modelo como lotes de entrenamiento y validación para que automáticamente modifique sus parámetros y mejore sus predicciones (imagen 2). Estos conceptos se desarrollan más adelante en este mismo capítulo.

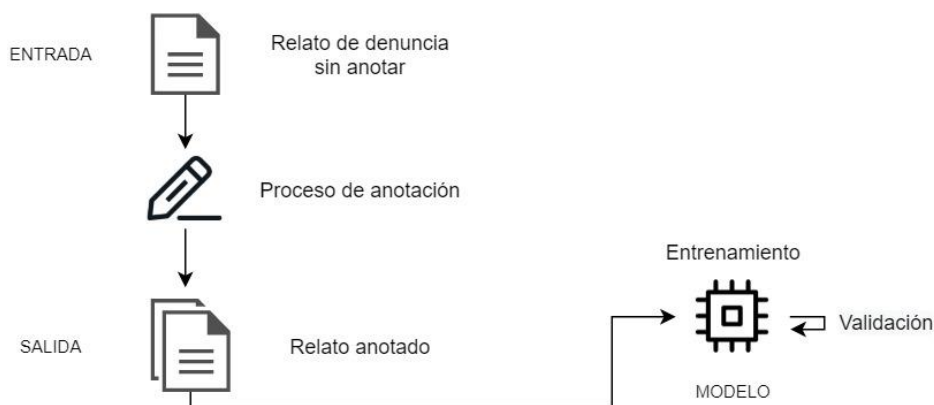


Imagen 2: Proceso de anotación, entrenamiento y validación – Aprendizaje Supervisado

2.2.2.2 Redes Neuronales

Una red neuronal es un tipo particular de modelo de aprendizaje automático. Las redes neuronales son especiales debido a su alta flexibilidad, esto significa, que pueden resolver una gama inusualmente amplia de problemas con solo encontrar los pesos correctos (Howard & Gugger, 2020, págs. 78,79). Los pesos o parámetros, son variables, cuyos valores se definen previo al entrenamiento del modelo. Por cada valor diferente que se asigne a cada peso, se obtiene un modelo distinto a los demás, cuyas predicciones, una vez entrenado, pueden variar significativamente. Generalmente estos pesos se inicializan con un valor aleatorio y luego se van ajustando automáticamente a través de una técnica matemática conocida como *descenso de gradiente estocástico (SGD)*. Estos conceptos se desarrollan un poco más en la sección 2.2.2.3.

2.2.2.3 Aprendizaje Profundo

El Aprendizaje profundo es una especialidad dentro del aprendizaje automático que utiliza redes neuronales con múltiples capas. La clasificación de imágenes es un ejemplo representativo (también conocido como reconocimiento de imágenes). Comenzamos con datos etiquetados: un conjunto de imágenes para las que hemos asignado una etiqueta a cada imagen, indicando lo que representa. Nuestro objetivo es producir un programa, llamado modelo, que, dada una nueva imagen, hará una predicción precisa sobre lo que representa esa nueva imagen (Howard & Gugger, 2020, pág. 112).

Cada modelo comienza con una elección de *arquitectura*, una plantilla general que permite saber cómo funciona ese tipo de modelo internamente. El proceso de *entrenar* (o *ajustar*) el modelo consiste en encontrar un conjunto de *parámetros*, *valores* (o *pesos*) que especializan esa arquitectura general en un modelo que funciona bien para los tipos particulares de datos que utilizamos. Para definir cómo rinde un modelo ante una sola predicción, necesitamos definir una *función de pérdida*, que determine que tan bien o que tan mal realizamos una predicción (Howard & Gugger, 2020, pág. 112).

Para que el proceso de entrenamiento sea más rápido, podríamos empezar con un *modelo pre entrenado* (un modelo que ya ha sido entrenado con los datos de otra persona). Luego podemos adaptarlo a nuestros datos entrenándolos un poco más con ellos, en un proceso llamado *ajuste fino* (Howard & Gugger, 2020, pág. 113).

La herramienta spaCy, una de las utilizadas en este proyecto (sección 3.4), hace uso de estos modelos pre entrenados en los *pipelines* que ofrece para el idioma Español⁸. En la página de la referencia podemos ver que las fuentes (sources) utilizadas son UD_Spanish_AnCora⁹ (datos en español del corpus AnCora), wikiNER¹⁰ y los vectores de fastText¹¹. Con ellos se entrenan los modelos de tamaño medio, grande y con Transformers, los cuales pueden ser luego utilizados por quienes los descarguen para continuar el entrenamiento con datos propios y de esta manera agilizar este proceso.

⁸ <https://spaCy.io/models/es>

⁹ https://github.com/UniversalDependencies/UD_Spanish-AnCora

¹⁰ https://figshare.com/articles/dataset/Learning_multilingual_named_entity_recognition_from_Wiki_pedia/5462500

¹¹ <https://fasttext.cc/docs/en/english-vectors.html>

Cuando entrenamos un modelo, una preocupación clave es asegurarnos de que nuestro modelo *generaliza*: aprende lecciones generales de nuestros datos que también se aplican a nuevos elementos que encontrará, por lo que puede hacer buenas predicciones sobre esos elementos. El riesgo es que, si entrenamos mal nuestro modelo, en lugar de aprender lecciones generales, memoriza lo que ya ha visto, y entonces hará malas predicciones sobre nuevos datos. Tal fracaso es llamado *sobreajuste* (Howard & Gugger, 2020, pág. 113).

Para evitar esto, siempre dividimos nuestros datos en dos partes, *el conjunto o lote de entrenamiento* y *el conjunto o lote de validación*. Entrenamos el modelo mostrándole solo el conjunto de entrenamiento, y luego evaluamos qué tan bien se desempeña en los elementos del conjunto de validación. De esta manera, verificamos si las lecciones que el modelo aprende, del conjunto de entrenamiento, son lecciones que se generalizan al conjunto de validación. Para que una persona pueda evaluar qué tan bien está funcionando el modelo en el conjunto de validación en general, es que definimos una *métrica*. Durante el proceso de entrenamiento, cuando el modelo ha visto todos los elementos del conjunto de entrenamiento, decimos que ha transcurrido una *época* (Howard & Gugger, 2020, pág. 113).

Todos estos conceptos se aplican al aprendizaje automático en general. Se aplican a todo tipo de esquemas para definir un modelo entrenándolo con datos. Lo que distingue al aprendizaje profundo es una clase particular de arquitecturas: las arquitecturas basadas en redes neuronales (Howard & Gugger, 2020, pág. 113).

En la imagen 3 se muestra el proceso de entrenamiento de un modelo, con una arquitectura genérica. En este proyecto las entradas están dadas por el texto de los relatos y las etiquetas por las entidades anotadas en esos relatos.

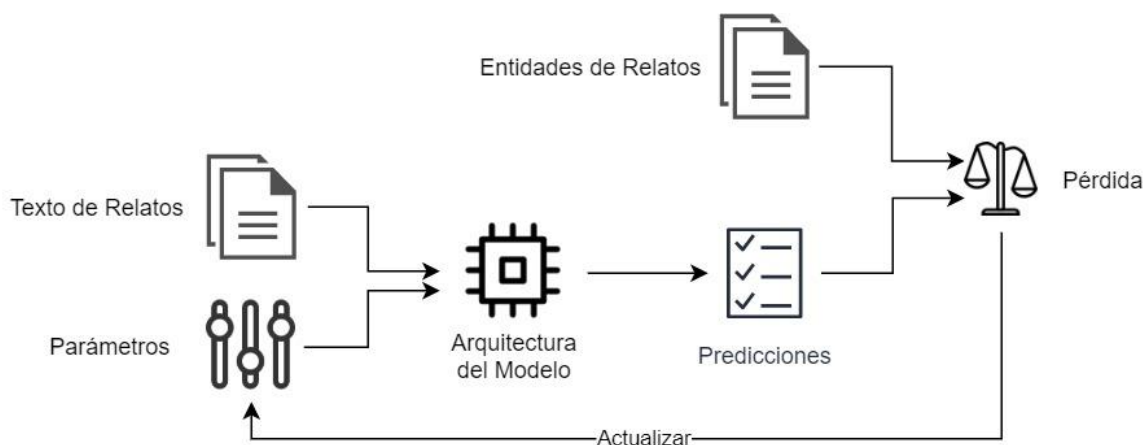


Imagen 3: Ciclo de entrenamiento de un modelo.

2.2.2.4 Evaluación

A la hora de evaluar los modelos de Aprendizaje Automático, incluidos los modelos de Reconocimiento de Entidades que se utilizan en este proyecto, se pueden utilizar diferentes métricas que ayuden a tener una noción de cómo se está desempeñando el modelo entrenado. Dentro de las métricas más populares encontramos cuatro: *accuracy*, *precision*, *recall* y *F1 score*.

Para una tarea de clasificación, como el reconocimiento de entidades donde se busca predecir si una palabra pertenece o no a una determinada clase, puedo encontrarme en alguna de las cuatro situaciones planteadas en la tabla 1.

		PREDICHO	
		Positivo	Negativo
REAL	Positivo	Verdadero Positivo (TP)	Falso Negativo (FN)
	Negativo	Falso Positivo (FP)	Verdadero Negativo (TN)

Tabla 1: Diferentes resultados posibles en una predicción de dos clases

Los verdaderos positivos (TP) y los verdaderos negativos (TN) son clasificaciones correctas. Un falso positivo (FP) es cuando el resultado se predice incorrectamente como positivo cuando en realidad es negativo. Un falso negativo (FN) es cuando el resultado se predice incorrectamente como negativo cuando en realidad es positivo. (Witten, Frank, Hall, & Pal, 2017, pág. 180). A partir de estos cuatro conceptos podemos definir las cuatro métricas mencionadas anteriormente:

Accuracy: se usa cuando la variable de salida es categórica o discreta. Denota la fracción de veces que el modelo predice correctamente en comparación al total de predicciones que hace (Vajjala, Majumder, Gupta, & Surana, 2020, pág. 115). La fórmula es:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision: muestra cuán precisas o exactas son las predicciones del modelo, es decir, dados todos los casos positivos, ¿cuántos clasifica el modelo correctamente? (Vajjala, Majumder, Gupta, & Surana, 2020, pág. 115). La fórmula es:

$$Precision = \frac{TP}{TP + FP}$$

Recall: es complementario a la métrica Precision. Captura qué tan bien el modelo puede predecir a la clase positiva, es decir, dadas todas las predicciones positivas que hace, ¿cuántas de ellas son realmente positivas? (Vajjala, Majumder, Gupta, & Surana, 2020, pág. 115). La fórmula es:

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

F1 score: Combina las métricas Precision y Recall para proporcionar una única métrica, que también captura el equilibrio entre esas dos métricas, es decir, integridad y exactitud. (Vajjala, Majumder, Gupta, & Surana, 2020, pág. 115). La fórmula es:

$$F1\ score = \frac{2 * Precision * Recall}{Precision + Recall}$$

2.2.3 Procesamiento del Lenguaje Natural

El procesamiento del lenguaje natural (PLN) es un campo en la intersección de la Informática, la Inteligencia Artificial y la Lingüística. Se trata de construir sistemas que puedan analizar, modelar y comprender el lenguaje humano. Toda aplicación inteligente que involucre lenguaje humano tiene algo de PLN detrás (Vajjala et al., 2020, p. 11,27,28).

2.2.4 Extracción de Información

Muchos documentos breves describen un tipo particular de objeto o evento, combinando entidades en un compuesto de nivel superior que representa todo el contenido del documento. La tarea de identificar la estructura compuesta, que a menudo se puede representar como una plantilla con espacios que se llenan con piezas individuales de información estructurada, se denomina *extracción de información* (EI). Una vez que se han encontrado las entidades, se analiza el texto para determinar las relaciones entre ellas. (Witten, Frank, Hall, & Pal, 2017, pág. 513).

Otra definición para extracción de información la presenta como la tarea de PLN que consiste en extraer información relevante de documentos de texto. Un ejemplo de EI que se utiliza en aplicaciones del mundo real son las breves descripciones que vemos a la derecha cuando buscamos el nombre de una figura popular en Google (Vajjala, Majumder, Gupta, & Surana, 2020, pág. 209). En la imagen 4 se ve un ejemplo de estos resúmenes que genera el buscador de Google, en ese caso para el cantante Jorge Rojas. En esa tarea de extracción de información se resumen algunas fotografías del artista, datos personales, canciones y álbumes.

The image shows a Google search result for 'Jorge Rojas'. It features a grid of photos at the top. Below the photos, the name 'Jorge Rojas' is displayed with the title 'Cantautor'. To the left, there are links to his social media profiles: YouTube, Spotify, YouTube Music, and Deezer. To the right, a summary of his personal information is provided: 'Nacimiento: 4 de marzo de 1972 (edad 49 años), Cutral Co', 'Cónyuge: Valeria Ojeda (m. 2012)', 'Hijos: Lautaro Rojas, Belén Rojas, Nicolás Rojas, María Lucía Rojas Ojeda, Franco Rojas', and 'Hermanos: Lucio Rojas, Alfredo Rojas'. Below this, there are sections for 'Canciones' (listing 'Uno Mismo', 'El Secreto de Tu Vida', and 'De Eso Se Trata') and 'Álbumes' (listing 'Aniversario', 'Sinfónico (En Vivo)', 'Mi Cantar', and 'Mi Voz y Mi Sangre'). At the bottom, there is a section for 'Otras personas también buscan' with links to Soledad Pastorutti, Chaqueño Palavecino, Luciano Pereyra, and Abel Pintos.

Imagen 4: Resumen que genera el buscador de Google al buscar "Jorge Rojas".

Cuando se compara con fuentes de información estructurada como bases de datos o tablas o fuentes semiestructuradas como páginas web (que tienen algún marcado), el texto es una forma de datos no estructurados. Por ejemplo, en una base de datos, sabemos dónde buscar algo según su esquema. Sin embargo, en gran medida, los documentos de texto generalmente comprenden texto de flujo libre sin un esquema establecido. Esto hace que EI sea un problema desafiante. Los textos pueden contener varios tipos de información. En la mayoría de los casos, extraer información que tiene un patrón fijo (p. ej., direcciones, números de teléfono, fechas, etc.) es relativamente sencillo utilizando técnicas de extracción basadas en patrones como las expresiones regulares, aunque el texto en sí se considera datos no estructurados. Sin embargo, extraer otra información (por ejemplo, nombres de personas, relaciones entre diferentes entidades en el texto, detalles de un evento del calendario, etc.) puede requerir un procesamiento de lenguaje más avanzado (Vajjala, Majumder, Gupta, & Surana, 2020, pág. 209).

2.2.4.1 Proceso de anotación

Este proceso consiste en etiquetar el texto empleado como entrada durante el entrenamiento con información referente a las entidades y sus relaciones identificadas en dicho texto. Este proceso se puede realizar manualmente o a través de softwares especializados para dicha tarea, como lo es Brat. Durante el etiquetado se debe indicar: la posición de inicio y fin de una entidad en el texto, y su tipo.

2.2.4.2 Reconocimiento de Entidades Nombradas

El Reconocimiento de Entidades Nombradas (REN) se refiere a la tarea de EI de identificar las entidades en un documento. Las entidades suelen ser nombres de personas, ubicaciones y organizaciones, y otras cadenas especializadas, como expresiones monetarias, fechas, productos, nombres/números de leyes o artículos, etc. REN es un paso importante en el proceso de varias aplicaciones de PLN que involucran la extracción de información (Vajjala et al., 2020, p. 221).

En este proyecto se implementan técnicas de reconocimiento de entidades a través de la librería spaCy. Sin embargo, como se mencionó en el párrafo anterior, este es un paso más, dentro de la extracción de información.

2.2.4.3 Desambiguación de Entidades Nombradas y Vinculación

La desambiguación de entidad nombrada (DEN) se refiere a la tarea de PLN de lograr exactamente esto: asignar una identidad única a las entidades mencionadas en el texto. También es el primer paso para avanzar hacia tareas más sofisticadas para abordar un escenario más avanzado mediante la identificación de relaciones entre entidades. REN y DEN juntos se conocen como vínculo o enlace de entidad nombrada (EEN) (Vajjala, Majumder, Gupta, & Surana, 2020, pág. 232). Cuando dice: “asignar una identidad única” se refiere a que, si en un mismo texto aparecen las entidades: Esteban y Dr. Esteban, por ejemplo, si refieren a la misma persona, deberían considerarse como una única entidad de tipo persona.

Entonces, ¿cómo construimos un sistema de EI para realizar EEN? Así como REN identifica entidades y sus tipos utilizando información contextual codificada por una variedad de características, EEN también se basa en el contexto. Sin embargo, requiere ir

más allá del etiquetado POS (Part Of Speech)¹² en términos del preprocesamiento de PLN necesario. Como mínimo, EEN necesita alguna forma de análisis para identificar elementos lingüísticos como sujeto, verbo y objeto. Además, también puede necesitar una resolución de coreferencia para resolver y vincular múltiples referencias a la misma entidad en una gran base de conocimientos enciclopédica (por ejemplo, Wikipedia). Por lo general, esto se modela como un problema de aprendizaje automático supervisado y se evalúa en términos de precisión, recall y F1 score en conjuntos de pruebas estándar. Claramente, aprender un modelo EEN requiere la presencia de un gran conjunto de datos anotados, así como algún tipo de recurso enciclopédico para vincular. Además, EEN es una tarea de PLN mucho más especializada en comparación con REN (Vajjala, Majumder, Gupta, & Surana, 2020, pág. 233).

2.2.4.4 Extracción de Relaciones

La extracción de relaciones (ER) es la tarea de EI que se ocupa de extraer entidades y relaciones entre ellas de documentos de texto. Es un paso importante en la creación de una base de conocimientos y también es útil para mejorar la búsqueda y desarrollar sistemas de respuesta a preguntas. En ER, además de identificar qué entidades existen (REN) y eliminar la ambigüedad (DEN), necesitamos modelar el proceso de extracción de las relaciones entre ellas considerando las palabras que conectan las entidades en una oración, su sentido de uso, etc. Además, una pregunta importante que debe resolverse es: ¿qué constituye una “relación”? Las relaciones pueden ser específicas de un dominio determinado. Por ejemplo, en el ámbito médico, las relaciones podrían incluir el tipo de lesión, la ubicación de la lesión, la causa de la lesión, el tratamiento de la lesión, etc. En el ámbito financiero, las relaciones podrían significar algo completamente diferente. Algunas relaciones genéricas entre personas, ubicaciones y organizaciones son: ubicadas en, es parte de, fundador de, padre de, etc. (Vajjala, Majumder, Gupta, & Surana, 2020, pág. 236).

2.2.5 Conceptos de spaCy

En este proyecto se utiliza la librería spaCy para trabajar con la mayoría de los conceptos previamente introducidos. La manera de trabajar de esta librería requiere una breve explicación teórica que se desarrolla a continuación.

Un concepto esencial en spaCy es el de *token*. Este se menciona por primera vez en el proceso de *tokenización*. En su página web¹³ se define al proceso de tokenización como la tarea que consiste en dividir un texto en segmentos con significado llamados tokens. Esto se consigue aplicando reglas específicas a cada lenguaje.

Fórmula de Aprendizaje Profundo para modelos de NLP

La empresa desarrolladora de spaCy describe en un artículo¹⁴ cuál es la fórmula que utilizan los modelos de aprendizaje profundo que son estado del arte. Esta consiste en 4 pasos: *embed*, *encode*, *attend*, *predict*, que en español serían algo similar a: inscrutar, codificar, atender y predecir. Citando a Matthew Honnibal, el autor del artículo:

¹² Etiquetado POS: implica asignar una etiqueta a cada palabra de una oración. La etiqueta indica a que parte de la oración pertenece dicha palabra: sustantivo, adjetivo, verbo, pronombre, etc.

¹³ <https://spaCy.io/usage/linguistic-features#tokenization>

¹⁴ Consultado en fecha: 14/08/2021 <https://explosion.ai/blog/deep-learning-formula-nlp>

“Las representaciones de palabras incrustadas (*word embeddings*), también conocidas como "vectores de palabras", son ahora una de las tecnologías de procesamiento del lenguaje natural más utilizadas. Las incrustaciones de palabras le permiten tratar palabras individuales como unidades de significado relacionadas, en lugar de identificadores completamente distintos. Sin embargo, la mayoría de los problemas de PLN requieren la comprensión de tramos de texto (*spans*) más largos, no solo palabras individuales. Ahora existe una solución simple y flexible que está logrando un rendimiento excelente en una amplia gama de problemas. Después de incrustar el texto en una secuencia de vectores, se utilizan RNN bidireccionales (Redes Neuronales Recurrentes Bidireccionales, un tipo de Red Neuronal eficiente para tareas de PLN) para codificar los vectores en una matriz de frases. Las filas de esta matriz pueden entenderse como vectores de token: son sensibles al contexto de oración del token. La última pieza del rompecabezas se llama mecanismo de atención. Esto le permite reducir la matriz de oraciones a un vector de oraciones, listo para la predicción”.

Para mayor detalle sobre cada uno de estos pasos se puede leer el artículo citado. Lo importante es saber que los dos componentes de spaCy utilizados en este proyecto: Tok2Vec y NER son en realidad modelos de redes neuronales que siguen una estructura similar a la indicada en el artículo anterior. Puntualmente, dentro de la documentación de spaCy, las 5 arquitecturas de redes neuronales empleadas en este proyecto son:

- *TransitionBasedParser.v2*¹⁵: crea un modelo analizador basado en transiciones. Se puede aplicar a REN o análisis de dependencia. El análisis sintáctico basado en transiciones es un enfoque de la predicción estructurada en el que la tarea de predecir la estructura se asigna a una serie de transiciones de estado. Está compuesta por 3 sub redes neuronales que reciben los nombres de *tok2vec*, *lower* y *upper*.
- *Tok2VecListener.v1*¹⁶: un oyente se utiliza como subcapa dentro de un componente como *EntityRecognizer* (el que se utiliza para la tarea de NER). Por lo general, tendrá varios oyentes que se conectarán a un único componente Tok2Vec ascendente que se encuentra anteriormente en el flujo. Las capas de escucha actúan como *proxies*, pasando las predicciones del componente Tok2Vec a componentes descendentes y comunicando gradientes ascendentes. En este caso solamente se utiliza un componente adicional que nos permite realizar la tarea de NER, por lo cual solo se emplea un oyente en dicho componente.
- *Tok2Vec.v2*¹⁷: construye un modelo tok2vec a partir de dos subredes: una para incrustar (*embedding*) y otra para codificar (*encoding*).
- *MultiHashEmbed.v2*¹⁸: construye una capa de incrustación (*embedding*) que incrusta por separado una serie de atributos léxicos mediante la incrustación de hash, concatena los resultados y los pasa a través de una subred de avance para crear una representación mixta.

¹⁵ <https://spaCy.io/api/architectures#TransitionBasedParser>

¹⁶ <https://spaCy.io/api/architectures#Tok2VecListener>

¹⁷ <https://spaCy.io/api/architectures#Tok2Vec>

¹⁸ <https://spaCy.io/api/architectures#MultiHashEmbed>

- *MaxoutWindowEncoder.v2*¹⁹: codifica el contexto utilizando convoluciones con activación máxima (*maxout activation*), normalización de capa y conexiones residuales. Estos últimos son conceptos normalmente utilizados en un tipo de red neuronal conocido como Redes Neuronales Convolucionales.

Al final del proyecto, en el Anexo 5 se puede consultar el archivo de configuración empleado para entrenar el modelo que obtuvo los mejores puntajes. Allí se pueden encontrar cada una de las 5 arquitecturas de la lista anterior, con sus respectivos parámetros.

¹⁹ <https://spaCy.io/api/architectures#MaxoutWindowEncoder>

3 PROBLEMA Y SOLUCIÓN PROPUESTA

3.1 Objetivos y alcances

3.1.1 Objetivo General

Desarrollar un sistema para el reconocimiento de entidades en relatos de denuncias, gestionadas por el Ministerio Público Fiscal de la Provincia de Salta, mediante técnicas de Reconocimiento de Entidades, su posterior procesamiento y visualización.

3.1.2 Objetivos específicos

- Generar un modelo para identificar las entidades existentes en relatos de denuncias.
- Aplicar técnicas de Reconocimiento de Entidades y visualización.
- Desarrollar un sistema web que sirva de interfaz al usuario para interactuar con la plataforma de Reconocimiento de Entidades.

3.1.3 Alcances del trabajo

El proyecto consiste en el desarrollo de un sistema de visualización de información referente a relatos de denuncias para la toma de decisiones en el Ministerio Público Fiscal, a través del empleo de técnicas y herramientas de aprendizaje profundo aplicadas a la Extracción de Información sobre textos no estructurados.

El sistema brindará una interfaz para la gestión de relatos de denuncias y de las entidades identificadas automáticamente en ellos. Además, ofrecerá cuatro tipos de gráficos interactivos que resuman información sobre la cantidad de entidades registradas.

El sistema no contempla el re entrenamiento del modelo a partir de nuevos textos cargados, ya sea automáticamente o a elección del usuario, sino que esta tarea, en caso de considerarse necesaria, debería ser realizada por el científico de datos. Esto se debe a que, si aparecen relatos cuya estructura no se asemeja a los utilizados en el entrenamiento, es muy posible que el sistema no sepa generalizar adecuadamente y por ende no identifique las entidades de manera óptima. Por ello, cada vez que se haga un re entrenamiento, incluyendo estos nuevos relatos, lo ideal sería realizar diversas pruebas, con ajustes de parámetros hasta hallar los valores que mejoren la capacidad del modelo para detectar las entidades.

A pesar de lo descrito en el párrafo anterior, sería posible implementar una funcionalidad en el sistema, que permita el re entrenamiento automático del modelo, utilizando todos los relatos almacenados en la base de datos, pero sin modificar los parámetros de la red neuronal. También se podría permitir al usuario seleccionar los nuevos parámetros desde la interfaz y observar los resultados de validación del modelo. Aún sin incluir esta funcionalidad de elegir los parámetros, los resultados obtenidos al re entrenar el modelo debería ser más que aceptable debido a la alta eficiencia de la arquitectura elegida por spaCy.

El sistema tampoco contempla la desambiguación de entidades ni la extracción de relaciones entre entidades.

En una segunda etapa del proyecto, posterior a este trabajo, sería necesario incorporar las funcionalidades mencionadas en los párrafos anteriores, permitiendo a los usuarios re entrenar el modelo cuando lo consideren necesario y acceder a mejor información plasmada en nuevos gráficos (grafos) que les permitan realizar análisis más profundos y de mayor calidad.

3.2 Restricciones y especificaciones

Especificaciones legales:

- Nombres de personas, documentos, teléfonos, direcciones, lugares y fechas que se muestren en el proyecto, deberán ser anonimizados como medida de protección de dichos datos.
- Se estableció un acuerdo de confidencialidad para no difundir los datos proporcionados por el Ministerio Público Fiscal de Salta.

3.3 Solución propuesta

Se propone, como solución, el desarrollo de un sistema que permita obtener distintos tipos de visualizaciones sobre la información extraída de denuncias.

El sistema permitirá la carga de nuevos relatos de denuncias para identificar las entidades presentes en estos. Dichas entidades se almacenarán en una base de datos a partir de la cual se construirán distintos tipos de visualizaciones.

3.4 Herramientas utilizadas

WINDOWS 10 HOME

El sistema operativo con el cual se trabajó es Windows 10 Home²⁰. No obstante, todas las herramientas utilizadas pueden instalarse en distribuciones Linux y en Mac OS sin ningún problema. En la imagen 5 se presenta el logo de Windows 10.



Imagen 5: Logotipo de Windows 10

²⁰ <https://www.microsoft.com/es-ar/d/windows-10/home/d76qx4bznwk4?activetab=pivot:overviewtab>

SUBSISTEMA DE WINDOWS PARA LINUX (WSL)

El Subsistema de Windows para Linux permite a los desarrolladores ejecutar un entorno de GNU/Linux, incluida la mayoría de herramientas de línea de comandos, utilidades y aplicaciones, directamente en Windows, sin modificar y sin la sobrecarga de una máquina virtual tradicional o una configuración de arranque dual (Loewen & olprod, 2021). Una de las opciones para conseguirlo es instalarlo directamente desde la Microsoft Store (tienda oficial de Microsoft).

Para esta solución se utilizó el WSL para Ubuntu 20.04.2 LTS. La imagen 6 muestra los logos de los sistemas combinados con esta tecnología WSL.

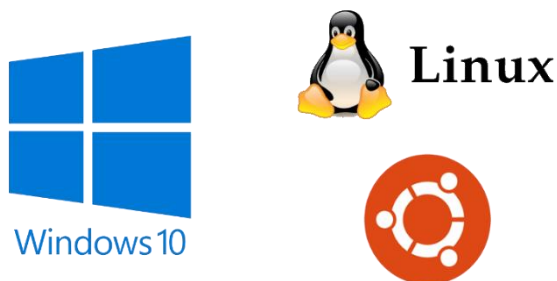


Imagen 6: Logotipos de Windows, Linux y Ubuntu

MONGODB

MongoDB es una base de datos de documentos que ofrece una gran escalabilidad y flexibilidad, y un modelo de consultas e indexación avanzado. Almacena los datos en documentos flexibles similares a JSON por lo que los campos pueden entre documentos y la estructura de los datos puede cambiarse con el tiempo (mongodb, 2021).

Para desplegar y administrar la base de datos MongoDB se utilizó su servicio de hosting en la nube: MongoDB Atlas²¹. El servicio se conoce como “Database as a Service” (base de datos como servicio). El plan gratuito se incluye en la categoría “Shared” y se corresponde al cluster M0 cuyos recursos son: memoria RAM y CPUs virtuales compartidos (no indica cuantos), además de un límite de almacenamiento de 512 MB. Por este último motivo, no es posible cargar gran cantidad de denuncias en el sistema actual. El logo de mongoDB se muestra en la imagen 7.



Imagen 7: Logotipo de MongoDB

²¹ <https://www.mongodb.com/cloud/atlas>

PYTHON-FLASK-SPACY

Python es un potente lenguaje de programación fácil de aprender. Tiene estructuras de datos eficientes de alto nivel y un enfoque simple pero efectivo para la programación orientada a objetos. La elegante sintaxis y la escritura dinámica de Python, junto con su naturaleza interpretada, lo convierten en un lenguaje ideal para la creación de scripts y el rápido desarrollo de aplicaciones en muchas áreas en la mayoría de las plataformas (Python Software Foundation, 2021).

En este proyecto, para el desarrollo del backend del sistema, se utilizan Flask definido por Pallets Projects (The Pallets Projects, 2021) como: “un marco de aplicación web WSGI ligero. Está diseñado para que la puesta en marcha sea rápida y sencilla, con la capacidad de escalar a aplicaciones complejas. Comenzó como una simple envoltura de Werkzeug y Jinja y se ha convertido en uno de los marcos de aplicaciones web de Python más populares”.

Además, para las tareas de PLN se utiliza spaCy, una biblioteca de software de código abierto para procesamiento avanzado de lenguaje natural en Python (Explosion, 2021).



Imagen 8: Logotipos de Python, Flask y spaCy

BRAT

Brat es un proyecto de código abierto desarrollado en colaboración por investigadores de varios grupos interesados en la anotación de texto (Brat, 2021). Como se mencionó en la sección referente al proceso de anotación, esta tarea es necesaria para resaltar e indicar, manualmente al modelo, cuales son las entidades presentes en los textos, sus tipos y relaciones. Brat nos facilita esta tarea a través de una interfaz gráfica con diversas funcionalidades.

1	QUE EN FECHA Y HORA TOMO EL COLECTIVO CORREDOR 1 A EN LA PARADA QUE SE ENCUENTRA EN AVDA.
2	SAN LEANDRO Y PEATONAL ALVEAR, AGLOMERANDOSE MUCHAS GENTES PARA SUBIR, CASI TODO EL TRAYECTO LO HIZO DE
	PIE POR LA GRAN CANTIDAD DE GENTE QUE SUBIO, CUANDO INGRESO AL BARRIO ALMAGRO UNA SRA.
3	LE DIO EL ASIENTO, AL SENTARSE SE DIO CON LA NOVEDAD QUE PERSONAS DESCONCIDAS DEL INTERIOR DE LA MOCHILA DEL BOLSILLO DEL FRETE LE SUTRAJERON SU CARNET DEL UPS (EL RENOVADO CON SU FOTOGRAFIA Y UNO VIEJO) Y OTRO
	CARNET A NOMBRE DE SU MARIDO EL SR. MILHOUSE VAN HOUTEN
4	DNI N° 33.753.907, ASI MISMO LE SUTRAJERON UN TELEFONO CELULAR MCA.
5	SAMSUNG GALAXY DE COLOR NEGRO NO RECEURDA EL MODELO, NO POSEE FACTURA DEL MISMO YA SE LO REGALARON PARA EL DIA DE LA MADRE, SUS ANTEOJOS RECETADOS, PAPELES VARIOS LOS QUE SE ENCONTRABA SU RECIBO DE SUELDO DEL MES DE MARZO Y RECETAS A NOMBRE DE LA DENUNCIANTE Y LA SUMA DE \$ 200 (DOCIENTOS PESOS).
6	RADICA LA PRESENTE A LOS SOLOS FINES DE SER PRESENTADA EN EL UPS A FIN DE QUE LE HAGAN ENTRGA DE OTRO CARNET.-

Imagen 9: Ejemplo de anotación en Brat

REACT

Para el desarrollo del sistema web, en el frontend, se utilizó React²², una biblioteca de JavaScript de código abierto, mantenida por Facebook y la comunidad de software libre. Específicamente se utilizó la solución que lleva por nombre Create React App, en cuyo sitio web se define como “una forma oficialmente compatible de crear aplicaciones React de una sola página. Ofrece una configuración de construcción moderna sin configuración” (Facebook, 2021).



Imagen 10: Logotipo de React

RECHARTS

Recharts es una biblioteca de gráficos componible construida sobre componentes de React. Al ser componible permite crear gráficos rápidamente con componentes desacoplados y reutilizables. Está construido sobre elementos SVG con una dependencia ligera en submódulos de D3JS. Permite personalizar los gráficos ajustando los “props” de los componentes y pasando componentes personalizados (Recharts Group, 2021).



Imagen 11: Logotipo de Recharts

VISUAL STUDIO CODE

Visual Studio Code es un editor de código fuente ligero, pero potente, que se ejecuta en el escritorio y está disponible para Windows, macOS y Linux. Viene con soporte integrado para JavaScript, TypeScript y Node.js y tiene un rico ecosistema de extensiones para otros lenguajes (como C ++, C #, Java, Python, PHP, Go) y entornos de ejecución (como .NET y Unity) (Microsoft, 2021).



Imagen 12: Logotipo de Visual Studio Code

²² <https://es.reactjs.org/>

GIT – GITHUB

Git es un sistema de control de versiones distribuido de código abierto y gratuito diseñado para manejar todo, desde proyectos pequeños hasta muy grandes, con velocidad y eficiencia. Es fácil de aprender y ocupa poco espacio con un rendimiento increíblemente rápido (Torvalds, 2021).

GitHub²³ es una plataforma de desarrollo colaborativo (repositorio) para alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de ordenador.



Imagen 13: Logotipos de Git y GitHub

HEROKU

Heroku es una plataforma en la nube que permite a las empresas construir, entregar, monitorear y escalar aplicaciones; es la forma más rápida de pasar de la idea a la URL, evitando todos esos dolores de cabeza de infraestructura (Salesforce, 2021). En este proyecto fue utilizado para desplegar el *backend* del sistema, es decir, los archivos correspondientes al servidor de Flask y el modelo de spaCy.



Imagen 14: Logotipo de Heroku

NETLIFY

Netlify es una plataforma todo en uno para automatizar proyectos web modernos. Permite reemplazar la infraestructura de alojamiento, integración continua y canal de implementación con un solo flujo de trabajo. También permite integrar funciones dinámicas como funciones sin servidor, autenticación de usuarios y manejo de formularios a medida que crecen los proyectos (Netlify, 2021).

²³ <https://github.com/>

En este proyecto fue utilizado para desplegar el *frontned* del sistema, es decir, los archivos de React.



Imagen 15: Logotipo de Netlify

3.5 Alternativas tecnológicas de solución al problema

Previo a realizar el trabajo con la tecnología descrita anteriormente, se analizaron distintas alternativas tecnológicas que se muestran a continuación:

Para el sistema web:

- *Frontend:*
 - Angular o VueJs: ambos son frameworks, similares a React ya que están basados en JavaScript. Personalmente no los escogí por falta de experiencia en estos, pero ambos gozan de una excelente reputación.
 - HTML5: es posible realizar el frontend sin necesidad de acudir a frameworks, lo cual no es recomendable para la escalabilidad de los proyectos ya que el trabajo a realizar sería mucho mayor.
- *Backend:*
 - PHP: junto con Apache son una alternativa a Flask. Debido a la mayor integración de Flask con spaCy y al desconocimiento de las tecnologías de PHP no se optó por esta alternativa.
 - NodeJS: al estar basado en JavaScript al igual que React, quizás sea la mejor alternativa de backend para aplicaciones que utilicen JavaScript o frameworks basados en este último para el frontend. Sin embargo, debido a que spaCy corre sobre Python, resultaba más sencillo integrarlo con Flask.
- Base de datos:
 - SQL Server: requiere de más recursos del ordenador para funcionar, además de tener licencia paga para producción.
 - PostgreSQL o MySQL: excelentes alternativas gratuitas para bases de datos relacionales. Debido a que se tenía mayor experiencia trabajando con el ORM flask-pymongo (para conectar mongoDB con Flask) y con MongoDB Atlas para el despliegue de la base de datos, se optó por la opción no relacional.

- Anotación:
 - Prodigy: excelente herramienta desarrollada por Explosion.ai²⁴ (los mismos autores de spaCy). Presenta una interfaz agradable y una gran variedad de funcionalidades para anotar texto, audio e imágenes. Para utilizarlo debe pagarse la licencia por única vez.
 - Doccano: buena alternativa a Brat ya que también es gratuito y presenta una interfaz más agradable a la vista. A diferencia de Brat, solamente sirve para anotar entidades, lo cual para este proyecto es suficiente, pero además presenta un problema al intentar cargar archivos en lote para anotarlos. Debido a este último problema esta alternativa fue rechazada.
- Reconocimiento de Entidades:
 - Tensorflow/Pytorch: librerías de código abierto para aprendizaje automático. Se utilizan para diversas tareas que abarcan desde el campo de la Visión Artificial hasta el Procesamiento del Lenguaje Natural. Son las mejores alternativas del mercado para el entrenamiento, validación y prueba de modelos orientados a diversas tareas, entre ellas el Reconocimiento de Entidades Nombradas. A diferencia de spaCy, estas manejan un menor nivel de abstracción, motivo por el cual no se las utilizó.

3.6 Arquitectura

Para el lado del servidor, el sistema consiste en una API realizada en Flask, conectada con la base de datos MongoDB a través del ORM PyMongo, además un script de Python que, mediante spaCy, hace uso del modelo de reconocimiento de entidades previamente entrenado a partir de relatos de denuncias anotadas en el sistema anotador Brat. La API se consulta desde el lado del cliente (frontend) el cual fue desarrollado con React y Recharts, este último para las visualizaciones. La imagen 16 ilustra los elementos mencionados y sus interacciones.

Como se observa en la imagen, el anotador Brat requiere de un entorno Linux para funcionar, por ello se instaló en el WSL de Windows 10. El resto del desarrollo fue completamente realizado en Windows.

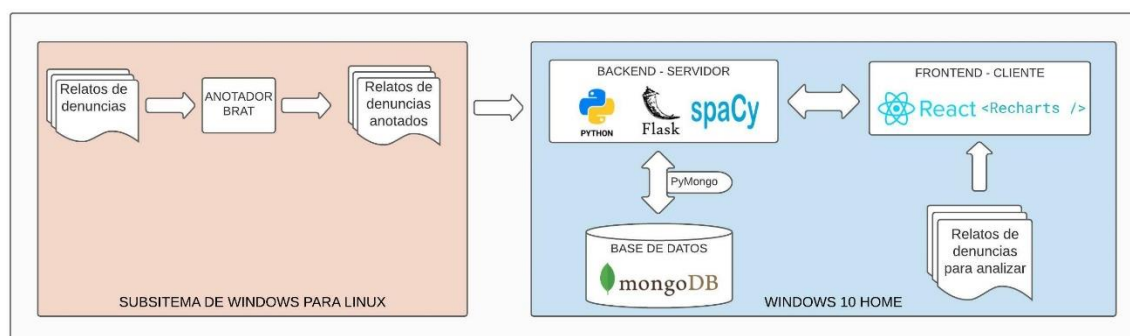


Imagen 16: Arquitectura del Proyecto

²⁴ <https://explosion.ai/>

3.7 Metodología

A continuación, se presenta cómo se fueron realizando las distintas fases de la metodología CRISP-DM las cuales se describieron brevemente en la sección 2.2.1:

3.7.1 Fase 1: Comprensión del negocio

3.7.1.1 Objetivos del proyecto

Tanto el objetivo general del proyecto, como sus correspondientes objetivos específicos y alcance, se definieron en la sección 3.1.

3.7.1.2 Evaluación de la situación

Para el proyecto se requirieron los recursos software indicados en la sección 3.4 y los recursos hardware y humanos detallados en la sección 3.10.3.

Todas las restricciones, suposiciones y demás factores tenidos en cuenta se desarrollan en la sección 3.2 y 3.13, correspondiente esta última al análisis de factibilidad del proyecto.

3.7.1.3 Plan de proyecto

El plan del proyecto se encuentra desarrollado en la sección 3.10.

3.7.2 Fase 2: Comprensión de los datos

3.7.2.1 Recolección de los datos

Previo acuerdo de confidencialidad con personal de Informática del Ministerio Público Fiscal de la Provincia de Salta, se obtuvo de ellos un lote de 35628 relatos de denuncias, todas ellas con fecha de registro en el año 2016. Además de los relatos, dicho lote contenía otras características de las denuncias detalladas a continuación, en la Descripción de los datos.

3.7.2.2 Descripción de los datos

Los 35628 registros se consiguieron en formato CSV (valores separados por coma). Cada registro presentaba los siguientes atributos:

- *denunciaId*: identificador numérico de la denuncia.
- *Anio*: se refiere al año en que se registró la denuncia. En este conjunto de datos todas tienen año 2016.
- *idModalidad*: identificador numérico de la modalidad del delito.
- *Modalidad*: tipo de modalidad denunciada.
- *tipoLugarId*: identificador numérico del tipo de lugar donde ocurrió el hecho denunciado.
- *TipoLugar*: tipo de lugar donde ocurrió el hecho denunciado.

- *relato*: conocimiento transmitido por el denunciante, en detalle, respecto al hecho denunciado.

3.7.2.3 Exploración de los datos

De todos los datos descriptos en el título anterior, solamente el Id de la denuncia, el año y el relato presentan valores para todas las denuncias. 2509 denuncias no tienen especificado el *idModalidad* y/o la *Modalidad* y 4490 denuncias no tienen especificado el *tipoLugarId* y/o el *TipoLugar*.

Valores que toma el atributo *Modalidad* en este conjunto de datos:

No Corresponde, 'De Celulares', 'Ocasionales', 'Arrebatadores', 'Descuideros', 'Violentadores', 'Planificadores', nan, 'Golpes de puño', 'Motochorros', 'Taxi y Remises', 'Descuidista', 'Cuenteros', 'Punguistas', 'Patotas', 'De Motos', 'De Automotores', 'Falsos policías o inspectores', 'Sirviente, empleado u obrero', 'De bicicletas', 'Escalamiento', 'Mecheras', 'Escruchante', 'Atracadores', 'Carteristas', 'Boqueteros', 'Dormidoras', 'De Documentos en general', 'De Joyas, Piezas artísticas, etc.', 'Estafadores comerciales', 'De Armas de fuego', 'De Billetes, títulos, etc.', 'Bicichorros', 'Cuatreros', 'Reducidores', 'Extorsionistas', 'Violencia escolar o estudiantil', 'Delitos informáticos', 'Mercadería de tránsito', 'Maleteros', 'Sogueros', 'Violadores', 'Dormidera', 'Encubridores', 'Hileros o cableros', 'Salidera Bancaria', 'Repartidor o fletero', 'Pescadores', 'De Cheques', 'Violentador de cajas fuertes'.

Valores que toma el atributo *TipoLugar* en este conjunto de datos:

nan, 'CALLE', 'TERMINAL DE ÓMNIBUS', 'DESCAMPADO', 'EDIFICIO PÚBLICO', 'IGLESIA', 'PARQUE O PLAZA', 'OBRA EN CONSTRUCCION', 'HOSPITAL', 'DEP.POLICIAL', 'AEROPUERTO', 'DOMICILIO PARTICULAR', 'COLECTIVO', 'COMERCIO', 'TAXI', 'PARADA DE COLECTIVO', 'VEHICULO PARTICULAR', 'ALOJAMIENTO', 'CONSULTORIO MEDICO', 'SUPERMERCADO', 'LOCAL BAILABLE', 'CAMPING/COMPLEJO', 'ESTABLECIMIENTO EDUCATIVO', 'PLAYA DE ESTACIONAMIENTO', 'FINCA', 'CANCHA DE FUTBOL', 'REMIS', 'CEMENTERIO', 'BANCO', 'EMPRESA', 'TRANSPORTE DE MERCADERIA', 'FARMACIA', 'CENTRO DE SALUD', 'CAMION DE CAUDAL', 'MOVIL POLICIAL', 'CÁRCEL'.

Tanto el atributo de modalidad como el de tipo de lugar pueden resultar de gran utilidad para tareas de Clasificación de denuncias, en donde podríamos enseñarle a un modelo con estas denuncias para que aprenda a colocar las etiquetas de lugar y modalidad automáticamente. En lo que refiere a la tarea de identificación de entidades resulta complicado encontrarles utilidad, ya que, como se verá en la sección 3.7.3.3, la estructura que utilizan los anotadores como Brat y la librería spaCy, para los archivos de entrenamiento y validación, no permiten incluir más información además del contenido del relato, las entidades que aparecen en el mismo, sus posiciones y tipos. Aún así estos datos, en caso de estar presentes en la denuncia que se cargue, podrían mostrarse en la interfaz del sistema a fin de aportar más datos a los usuarios, que puedan utilizar en sus análisis.

Si bien los atributos Modalidad y TipoLugar podrían aportar algún contexto para identificar las entidades de los relatos, la estructura mencionada en la sección 3.7.3.3 no permitiría incluirlos en el proceso de entrenamiento. Sin embargo, una manera de tener en cuenta el contexto es la que se plantea como continuación del proyecto aquí desarrollado, estamos hablando de la extracción de relaciones entre entidades.

Debido a lo planteado en los párrafos anteriores y a la intención de mantener las fases de modelado y evaluación lo más simples posible, los únicos atributos que se tuvieron en cuenta para continuar el trabajo fueron: *denunciaId*, para identificar cada denuncia, y *relato*, en base al cual se realizó el análisis.

Sobre el atributo *relato*: un detalle no menor a considerar es el hecho de que todos los relatos de denuncias se encontraban escritos completamente en mayúsculas. Tras consultar al proveedor de estos datos, se aclaró que el sistema de denuncias, por medio del cual personal policial registra las mismas, automáticamente convierte el contenido a mayúsculas y lo almacena de este modo.

Los relatos varían bastante entre sí:

- En algunos se tienen en cuenta los acentos, en otros no.
- Algunos presentan gran cantidad de errores de tipeo u ortográficos. Otros en menor cantidad y algunos no presentan errores.
- Las expresiones utilizadas por los distintos registrantes varían: por ejemplo, para nombrar a la patente algunos utilizan esa palabra y otros emplean dominio o matrícula.
- Algunos relatos utilizan un gran número de abreviaturas, mayoritariamente para referirse a títulos de personas como: sr., sra., sgo., aye., entre otros.
- El número de entidades por relato, tanto para personas, barrios, números de DNI/CUIL/CUIT, patentes de vehículos y números de teléfono, varía demasiado. Incluso, un gran número de relatos no presentan entidades de esos tipos.
- Al cargar las 35628 denuncias en una planilla de Excel y agregar una nueva columna que contenga la cantidad de caracteres de los relatos utilizando la función *LARGO* o *LONG* es posible obtener los siguientes datos:
 - El relato con menor cantidad de caracteres presenta un total de 104.
 - El relato con mayor cantidad de caracteres presenta un total de 12775.
 - La distribución de frecuencias del largo de los relatos se observa en la imagen 17. Un zoom del mismo gráfico se observa en la imagen 18.

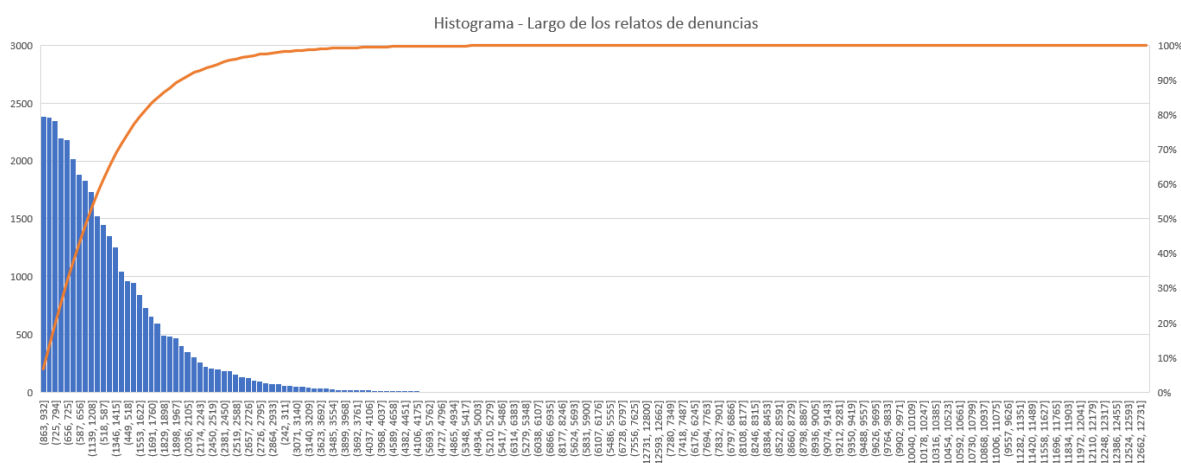


Imagen 17: Histograma del largo de los relatos de denuncias

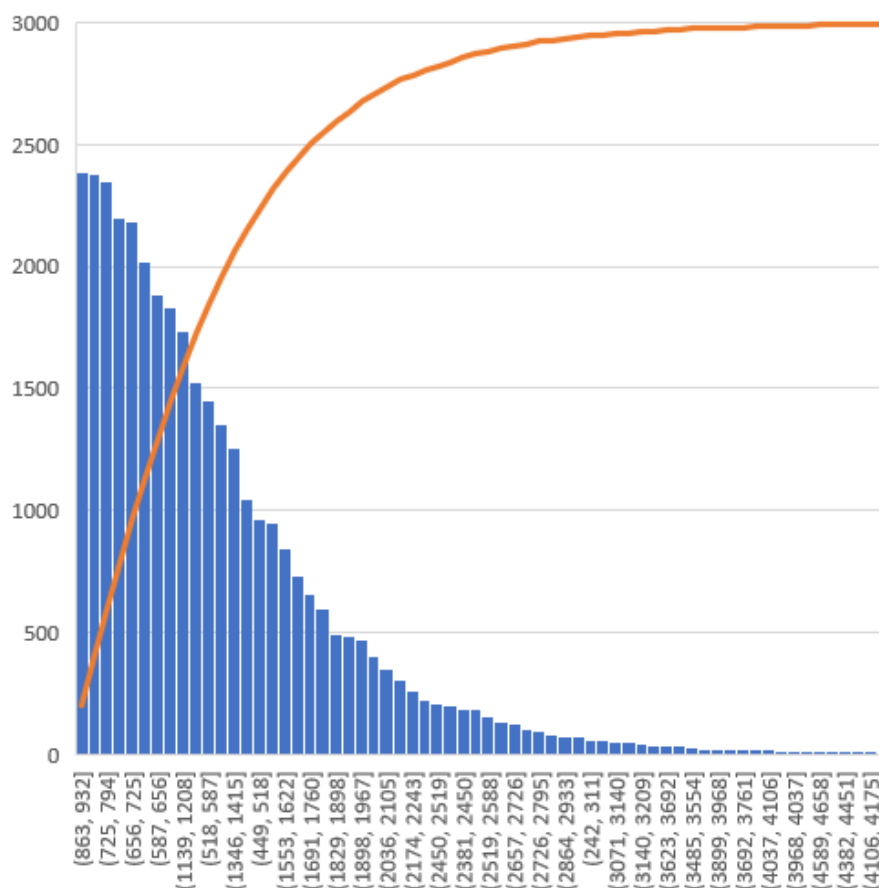


Imagen 18: Zoom al histograma del largo de los relatos de denuncias

- De las imágenes anteriores se puede observar que aproximadamente el 80% de los relatos tiene una longitud entre 400 y 1800 caracteres

3.7.3 Fase 3: Preparación de los datos

3.7.3.1 Selección de los datos

Del total de 35628 denuncias, se seleccionaron 450 para las etapas de entrenamiento, validación y prueba de modelos, y otras 70 las cuales pasaron por un proceso de anonimización a fin de ser cargadas en el sistema.

Procedimiento:

A través del script de Python: *generarTxts.py* (Anexo 4) se cargaron las 35628 denuncias, se mezclaron aleatoriamente utilizando el método *Sample* de la librería de Python, *Pandas*, con parámetros *frac=1* y *random_state=0*. Luego se tomaron las 1000 primeras denuncias a partir de las cuales se generaron 1000 archivos de texto cuyo nombre se correspondía con el campo *denunciaId* y cuyo contenido era el relato.

3.7.3.2 Anotación de los datos

Los 1000 archivos de extensión *.txt* con los relatos de las denuncias se cargaron en la carpeta *data*, dentro del directorio de instalación del anotador Brat (en el WSL Ubuntu 20.04). Para comenzar a anotar, Brat requiere que la colección de archivos *.txt* añadidos, contengan cada uno un archivo de extensión *.ann* con el mismo nombre. Es en este último donde se registran las entidades identificadas en el texto del archivo *.txt*.

En el anexo 2 se encuentra la guía de instalación y configuración de Brat, allí se indica cómo generar los archivos .ann.

El proceso de anotación consistió en leer gran parte de los 1000 relatos cargados a fin de encontrar 450 que contengan al menos una entidad de tipo persona, barrio, DNI, patente y/o teléfono.

Tras anotar los 450 relatos planificados, en la carpeta *data* seguían quedando 1000 archivos de extensión .txt y 1000 .ann. Los archivos .ann con entidades registradas tenían un tamaño mayor a 0KB, dato que se utilizó para filtrar y copiar solo los 450 archivos .ann escritos al directorio de trabajo en Windows, junto con los 1000 archivos de extensión .txt (1450 archivos en total). Para eliminar los archivos .txt sin su correspondiente par .ann se utilizó el script *eliminar_txts_sin_archivo_ann.py* (Anexo 4), tras lo cual quedaron 450 archivos txt y 450 ann, 900 en total. Cada archivo .ann contenía la siguiente información:

- Número de entidad
- Tipo de entidad
- Índice del carácter de inicio de la entidad en el archivo .txt correspondiente
- Índice del carácter final de la entidad en el archivo .txt correspondiente
- Texto marcado como entidad

3.7.3.3 División y formateo de los datos

Debido a que la cantidad de entidades encontradas por relato varía, fue necesario identificar la cantidad total de entidades de cada tipo existentes en el total de los 450 relatos. De esta manera, al momento de dividir los relatos en conjuntos de entrenamiento, validación y prueba, cada grupo tendría una cantidad proporcional de entidades de cada tipo.

Para lograr lo expresado en el párrafo anterior se realizó lo siguiente:

1. Considerando que el número de relatos por conjunto sería de 350, 50 y 50 para entrenamiento, validación y prueba respectivamente, el porcentaje sobre el total de 450 relatos es aproximadamente: 78%, 11% y 11% respectivamente.
2. Se determinó cuántas entidades de cada tipo existían en el lote de 450 relatos. Para ello se utilizó el script *contarTotalDeEntidades.py* (Anexo 4). Como resultado se obtuvo lo siguiente:
 - 803 entidades de tipo persona
 - 183 entidades de tipo barrio
 - 133 entidades de tipo teléfono
 - 130 entidades de tipo patente
 - 79 entidades de tipo DNI
 - En total 1328 entidades
3. Se calculó, de manera aproximada, el número de entidades ideal de cada tipo para cada conjunto, para ello se tomó el 78%, 11% y 11% de los valores obtenidos en el paso 2 para cada uno de los 3 conjuntos. Los resultados se presentan en la tabla 2.

Tipo de Entidad	Entrenamiento	Validación	Prueba
Persona	626	89	88
Barrio	143	20	20
Teléfono	104	15	14
Patente	101	15	14
DNI	62	9	8

Tabla 2: Número ideal de entidades de cada tipo para los conjuntos de entrenamiento, validación y prueba.

- Se ejecutó el script *determinarMejorSeedParaBalanceo.py* (Anexo 4) el cual prueba 10.000 semillas diferentes para dividir aleatoriamente los 450 relatos en los 3 grupos mencionados de 350, 50 y 50. Por cada una de estas pruebas se cuenta la cantidad de cada tipo de entidad existente en los 3 grupos y por medio de una métrica definida en el script se calcula la diferencia con la cantidad ideal representada en la tabla del punto 3. De esta manera, se determina cual es la semilla que consigue la menor diferencia, resultando en la semilla 7448. Los resultados para dicha semilla fueron los de la tabla 3.

Tipo de Entidad	Entrenamiento	Validación	Prueba
Persona	629	87	87
Barrio	138	22	23
Teléfono	104	15	14
Patente	100	16	14
DNI	60	10	9

Tabla 3: Número de entidades de cada tipo para los conjuntos de entrenamiento, validación y prueba utilizados.

- Sabiendo la cantidad de relatos por grupo y la semilla a utilizar, se ejecutó el script *dividirRelatosTrainValidTest.py* (Anexo 4) para obtener las 3 carpetas.

Debido a que la librería de Python, spaCy necesita un solo archivo .json para el conjunto de entrenamiento y otro para validación, fue necesario realizar una conversión del formato obtenido de Brat (archivos .txt y .ann) al formato JSON.

A través del script *bratAnnotationTospaCyAnnotation.py* (Anexo 4) se obtuvieron 3 archivos de extensión .json a partir de los 450 archivos .txt y de los 450 archivos .ann. Los 3 archivos .json se corresponden a 350, 50 y 50 relatos con sus entidades identificadas para los conjuntos de entrenamiento, validación y prueba respectivamente.

Hasta la versión 2 de spaCy se utilizaban dos archivos en formato JSON, uno para entrenamiento y otro para validación, conteniendo una lista de objetos compuestos por el texto a analizar, en este caso el relato de la denuncia, y por una lista de las entidades identificadas en dicho texto, de las cuales se indicaba el índice del carácter de inicio y el de fin, más el tipo de entidad. En la imagen 19 vemos dicha estructura. Los archivos de extensión .json que obtuvimos al ejecutar el script *bratAnnotationTospaCyAnnotation.py* presentan esa estructura.

```
1  [
2      [
3          "Relato de denuncia 1",
4          {
5              "entities": [
6                  [
7                      100,
8                      118,
9                      "Persona"
10                 ],
11                 [
12                     132,
13                     142,
14                     "DNI"
15                 ],
16                 [
17                     266,
18                     283,
19                     "Barrio"
20                 ],
21             ],
22         }
23     ],
24     [
25         "Relato de denuncia 2",
26         {
27             "entities": [
28                 [
29                     183,
30                     190,
31                     "Patente"
32                 ],
33             ],
34         }
35     ]
36 ]
```

Imagen 19: Formato JSON para relatos utilizado por spaCy 2

En la versión 3 de spaCy se utilizan dos archivos de extensión .spacy, nuevamente uno para entrenamiento del modelo y otro para validación. Estos archivos pueden ser generados a partir de los obtenidos previamente en formato JSON, tarea que se realizó con el script *generarArchivospaCy.py* (Anexo 4).

En este caso no fue necesario limpiar los relatos de las denuncias ya que el sistema debe aprender a identificar las entidades a pesar de los errores de tipeo y de ortografía de los registrantes.

A pesar de haber tomado la consideración de tener cantidades proporcionales de cada tipo de entidad en cada conjunto de relatos, lo ideal sería tener la misma cantidad (o lo más similar posible) de entidades de cada tipo en cada conjunto, es decir, si en total tenemos 800 entidades persona, sería recomendable tener igual número o similar de entidades barrio, DNI, patente y teléfono, lo cual en este trabajo no se consiguió.

Una manera de lidiar con el problema planteado en el párrafo anterior podría ser el detectar aquellos relatos que contengan solamente los tipos de entidades cuya cantidad necesitamos aumentar o al menos que la mayoría de las entidades sean de esos tipos. Una vez seleccionados se podrían aplicar técnicas de aumento de datos (*Data Augmentation*):

“El PLN tiene un gran número de técnicas a través de las cuales podemos tomar un pequeño conjunto de datos y usar algunos trucos para crear más datos. Estos trucos también se denominan *aumento de datos* e intentan explotar las propiedades del idioma para crear

texto que sea sintácticamente similar a los datos del texto de origen. Pueden parecer trucos, pero funcionan muy bien en la práctica” (Vajjala, Majumder, Gupta, & Surana, 2020, pág. 76). Algunas de estas técnicas se introducen en la página de la referencia anterior y siguientes. Ellas son: reemplazo de sinónimos, traducción inversa, reemplazo de palabras basado en TF-IDF, cambio de bigramas, reemplazo de entidades, agregar ruido a los datos, y otras técnicas más avanzadas.

En este proyecto no se implementaron técnicas de aumento de datos debido a que la intención fue tomar el lote aleatorio de denuncias y evaluar los resultados de entrenamiento del modelo. Como se verá en la fase de Evaluación, los resultados alcanzados resultaron satisfactorios, cumpliendo con el criterio de éxito de obtener un rendimiento igual o superior al 70% en la métrica escogida, F1-Score. Los resultados fueron buenos aún para aquellos tipos de entidades presentes en menor medida.

Si los resultados de la evaluación no hubieran superado el criterio de éxito, se habría intentado aplicar alguna técnica de aumento de datos. Aún con buenos resultados no se aplicaron las técnicas debido a la complejidad añadida que habría supuesto para el trabajo, significando más tiempo en las tareas de preparación de datos. Estas técnicas serían convenientes aplicarlas, junto con desambiguación de entidades y la extracción de relaciones, en la segunda etapa del sistema, propuesta como continuación a este proyecto.

3.7.4 Fase 4: Modelado

3.7.4.1 Técnica de modelado

La técnica de modelado consistió en realizar distintos tipos de entrenamientos con los relatos anotados, organizados en lotes de dos tamaños diferentes: 150 y 450.

En las pruebas realizadas sobre 150 relatos, estos se dividieron en 96, 24 y 30 relatos para entrenamiento, validación y prueba respectivamente. El criterio seguido para esta división consistió en tomar un 20% del total y asignarlo para el lote de prueba. Sobre el 80% restante tomar un 20% para el lote de validación y lo que quedé para el de entrenamiento.

En las pruebas realizadas sobre 450 relatos, estos se dividieron en 350, 50 y 50 relatos para entrenamiento, validación y prueba respectivamente. Si se siguiese el mismo criterio que en el párrafo anterior hubieran quedado 90 relatos para prueba, 52 para validación y 288 para entrenamiento. Debido a que en este caso se deseaba designar la mayor cantidad posible de relatos para entrenamiento fue que se optó por seguir un criterio diferente: designar alrededor de un 80% de los relatos (77,78% para ser exactos) para entrenar al modelo, y el 20% restante dividirlo equitativamente entre validación y prueba del modelo obteniendo así las cantidades de 350, 50 y 50 relatos.

Partiendo de estos datos la técnica consistió en entrenar un total de 12 modelos: 2 de ellos para las pruebas con 150 relatos y 10 con 450 relatos. Para el entrenamiento se empleó la librería spaCy, siguiendo los pasos indicados en su documentación²⁵:

- Del *Widget* proporcionado por spaCy, seleccionar el lenguaje Español (Spanish), los componentes del *pipeline* (solamente NER), el Hardware (en este caso CPU) y por último “Optimizado para” *efficiency* (eficiencia) o *accuracy* (exactitud). De los 12

²⁵ <https://spacy.io/usage/training#quickstart>

modelos mencionados 7 fueron entrenados con la opción de *efficiency* y 5 con *accuracy*. La diferencia principal radica en el empleo de vectores dentro del *pipeline* (esto lo realiza el componente Tok2Vec). Los modelos configurados con *efficiency* no utilizan dichos vectores, por lo tanto, son más pequeños, consumen menos memoria y realizan inferencias con mayor velocidad, mientras que los configurados con *accuracy* son de un tamaño considerable (superando el Gigabyte) y requieren de más tiempo para entrenarse. Desde luego, los resultados obtenidos con los modelos de *accuracy* son mejores. Al momento de elegir eficiencia o exactitud, spaCy ajusta el tipo de arquitectura, los pesos pre entrenados y los hiper parámetros.

- spaCy genera un archivo de configuración en base a los parámetros elegidos en el apartado anterior. Este archivo suele llamarse *base_config.cfg* y se guarda en el directorio de trabajo. En él deben especificarse las rutas para los archivos de extensión *.spacy* correspondientes a los datos de entrenamiento y validación.
- A partir del archivo *base_config.cfg* se crea otro archivo de configuración completo al ejecutar el comando:

```
python -m spacy init fill-config base_config.cfg config.cfg
```

- Con el archivo de configuración completo, el cual suele recibir el nombre *config.cfg* ya se puede iniciar el entrenamiento:

```
python -m spacy train config.cfg --output ./output
```

- El modelo final se guarda en la carpeta *output* con la estructura que se muestra en la imagen 20.

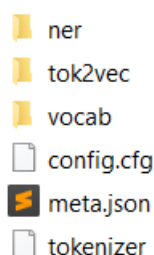


Imagen 20: Estructura del modelo entrenado

- Las carpetas *ner* y *tok2vec* se corresponden a los componentes de spaCy que reciben el mismo nombre. Dentro de ellas encontramos dos archivos más: *cfg* y *model*. El primero contiene la configuración de ese componente, la cual la extrae del archivo *config.cfg*. El archivo *model* contiene los datos binarios del modelo.
- El archivo *config.cfg* es el utilizado para entrenar el modelo.
- El archivo *meta.json* contiene información del modelo una vez entrenado.
- El archivo *tokenizer* se corresponde con otro componente de spaCy. Es el primer componente del pipeline de procesamiento. Es el único componente que recibe como entrada texto crudo, en este caso los relatos, a los cuales divide en los segmentos con significado llamados *tokens*.

- En la carpeta *vocab* encontramos tres archivos relevantes:
 - El archivo *strings.json* consistente en un array con miles de *strings* (cadenas de texto). Cada *string* es una palabra o carácter que el modelo incorporó como parte de su vocabulario.
 - El archivo *vectors* contiene los vectores de palabras utilizados por el modelo.
 - El archivo *key2rows* consiste en un diccionario que mapea los hashes de las palabras a filas dentro del archivo *vectors*.
 -

Los archivos *vectors* y *key2rows* solo son utilizados en los modelos de exactitud (*accuracy*). Esto ocurre porque en el archivo de configuración de estos modelos (*config.cfg*), en el apartado *[initialize]* (imagen 21) se indica cuáles son los vectores que se utilizarán. En los modelos de eficiencia el valor de este parámetro es *null*, mientras que en un modelo de exactitud, por defecto se asigna el valor *es_core_news_lg*.

```
[initialize]
vectors = null
init_tok2vec = ${paths.init_tok2vec}
vocab_data = null
lookups = null
before_init = null
after_init = null
```

Imagen 21: Apartado Initialize del archivo *config.cfg*

El modelo de spaCy que lleva por nombre *es_core_news_lg*²⁶ es un modelo pre entrenado sobre un gran cuerpo de noticias en español. Está preparado para realizar distintos tipos de tareas, entre ellas, reconocimiento de entidades. Si se lo emplea en un contexto distinto al de noticias, como es el caso de las denuncias, el desempeño no es muy bueno. Una particularidad de este modelo es que incluye 500 mil vectores de 300 dimensiones, entrenados sobre el cuerpo mencionado. Cuando en el archivo *config.cfg*, en el apartado *[initialize]*, indicamos el nombre de este modelo pre entrenado, le estamos diciendo a spaCy que utilice los 500 mil vectores de este modelo para entrenar al nuevo. Como resultado del entrenamiento dichos vectores se ajustan al nuevo contenido procesado (relatos de denuncias). Debido a esto, para los modelos de exactitud los archivos *vectors* y *key2rows* tienen un tamaño mucho mayor que en los modelos de eficiencia. El más pesado es el archivo *vectors* alcanzando un tamaño próximo a los 600 MB para estos modelos.

3.7.4.2 Construcción de modelos

En la tabla 4 se resumen las distintas configuraciones de modelos entrenados y validados. Se trata de un total de 12 modelos, 5 optimizados para exactitud y 7 para eficiencia. Se indica la cantidad de relatos utilizados en cada uno y las modificaciones realizadas al archivo *config.cfg*. Respecto a este último punto cabe aclarar que los valores por defecto que ofrece spaCy son los que mejores resultados suelen ofrecer, sin embargo, en cuatro de los modelos se optó por modificar solamente el hiper parámetro “Learning Rate”.

²⁶ https://spaCy.io/models/es#es_core_news_lg

El parámetro *learning-rate* (tasa de aprendizaje), también conocido como *step-size* (tamaño del paso), puede tomar diferentes valores, los cuales afectarán el ajuste de los pesos que se produce en el interior de las redes neuronales durante el proceso de entrenamiento a través de la técnica de *descenso del gradiente*, mencionada en las secciones 2.2.2.2 y 2.2.2.3. Como dicen Deisenroth y otros (2020, pág. 229): elegir un buen *step-size* es importante en el descenso del gradiente. Si el *step-size*, es muy pequeño, el descenso del gradiente puede ser lento. Si, en cambio, se elige un valor muy grande, el descenso del gradiente puede excederse, fallar en converger o incluso diverger.

El informático y experto en aprendizaje automático, Andrew Ng²⁷, cofundador de Coursera²⁸ y deeplearning.ai²⁹, recomienda en los cursos que dicta sobre estas plataformas, la prueba y error con diferentes valores del parámetro *learning-rate*, en búsqueda de la convergencia del modelo, sin esperar excesivamente (a causa de un valor muy pequeño de este parámetro), ni caer en *sobreajuste*, lo cual se observaría si las métricas arrojan resultados próximos al 100% con los datos de entrenamiento y resultados bastante inferiores con los datos de validación y prueba. Andrew, por experiencia propia, recomienda comenzar con un valor de 0.1, e ir aumentándolo siguiendo la siguiente secuencia: 0.3, 0.01, 0.03, 0.001, 0.003, y así sucesivamente hasta encontrar el valor que mejores resultados arroje.

En la fase de evaluación (sección 3.7.5), se incluyen imágenes con los resultados por época del proceso de entrenamiento/validación de los 12 modelos. Si, por ejemplo, tomamos los modelos 10, 11 y 12, que, según la tabla 4, fueron entrenados bajo igualdad de condiciones, con excepción del parámetro *learn-rate* cuyo valor fue de 0.001, 0.003 y 0.0003 respectivamente, es posible observar una diferencia en los valores de la columna *SCORE*, resultando el modelo 12 con el mayor puntaje. Debido a que el valor del *learn-rate* para este modelo fue el menor, se puede observar que el rango de puntajes obtenidos entre una época y otra también es menor comparado a los obtenidos en el proceso de entrenamiento/validación de los otros modelos.

Tanto la recomendación de Andrew Ng, como la tendencia a mejorar el puntaje a la hora de reducir el valor del *learn-rate*, observado en el proceso de prueba y error, fueron los que orientaron el proceso de modelado hacia la reducción de este parámetro.

A partir de esta ligera modificación los resultados variaron notablemente. La última columna “Separación en lotes” indica si al momento de separar los relatos en lotes de entrenamiento, validación y prueba se tuvo en cuenta algún criterio o no:

- “Al azar” se refiere al uso de la semilla 0 para dividir los relatos.
- Balanceo de entidades se refiere a que se buscó hacer la división aproximando la cantidad de cada tipo de entidad en cada lote a las cantidades calculadas para un caso ideal.
- Balanceo de entidades con pesos es igual al anterior solo que se le dio mayor peso a los tipos de entidades con menor presencia como son patente y teléfono. Para este criterio encontramos algunos casos donde se realizaron modificaciones al archivo *config.cfg*.

²⁷ https://en.wikipedia.org/wiki/Andrew_Ng

²⁸ <https://www.coursera.org/>

²⁹ <https://www.deeplearning.ai/>

Optimizado para	Modelo	Cantidad de Relatos	Archivo config.cfg	Separación en lotes
Exactitud (Accuracy)	1	150 (96, 24, 30)	Con valores por defecto	Al azar
	2	450 (350, 50, 50)	Con valores por defecto	Al azar
	3	450 (350, 50, 50)	Con valores por defecto	Balanceo de entidades
	4	450 (350, 50, 50)	Con valores por defecto	Balanceo de entidades con pesos
	5	450 (350, 50, 50)	Con valores por defecto Learn rate: 0.0003	Balanceo de entidades con pesos
Eficiencia (Efficiency)	6	150 (96, 24, 30)	Con valores por defecto	Al azar
	7	450 (350, 50, 50)	Con valores por defecto	Al azar
	8	450 (350, 50, 50)	Con valores por defecto	Balanceo de entidades
	9	450 (350, 50, 50)	Con valores por defecto	Balanceo de entidades con pesos
	10	450 (350, 50, 50)	Con valores por defecto Learn rate: 0.001	Balanceo de entidades con pesos
	11	450 (350, 50, 50)	Con valores por defecto Learn rate: 0.003	Balanceo de entidades con pesos
	12	450 (350, 50, 50)	Con valores por defecto Learn rate: 0.0003	Balanceo de entidades con pesos

Tabla 4: Configuraciones de modelos entrenados

A continuación, se mencionan algunos de los hiper parámetros más relevantes del archivo *config.cfg* junto con los valores que spaCy les asigna por defecto:

Sección *[components.tok2vec.model.encode]*:

@architectures = "spaCy.MaxoutWindowEncoder.v2"

Parámetros del Encoder (Tipo de Red Neuronal) para modelo de Eficiencia

width = 96

depth = 4

window_size = 1

maxout_pieces = 3

Parámetros del Encoder (Tipo de Red Neuronal) para modelo de Exactitud

width = 256

depth = 8

window_size = 1

maxout_pieces = 3

Sección [training]:

```
accumulate_gradient = 1  
patience = 1600  
max_epochs = 0  
max_steps = 20000  
eval_frequency = 200
```

Sección [training.optimizer]:

```
@optimizers = "Adam.v1"  
beta1 = 0.9  
beta2 = 0.999  
L2_is_weight_decay = true  
L2 = 0.01  
grad_clip = 1.0  
use_averages = false  
eps = 0.00000001  
learn_rate = 0.001
```

Este último hiper parámetro del optimizador Adam: *learn_rate*, es el que se modificó en algunos modelos por los valores indicados en la tabla 4. El resto de parámetros no se modificaron ya que requieren de una mayor profundización de su significado para que dicha modificación se realice en base a algún criterio. Además, mientras más parámetros se tomen en cuenta para las pruebas, la cantidad de posibles combinaciones se incrementa rápidamente. Por último, los resultados obtenidos sólo con los 12 modelos probados, se consideraron satisfactorios, dado el tamaño del lote empleado para entrenamiento. Si se utilizara un lote de miles de relatos, las exigencias al modelo podrían aumentarse ya que se supone que habrá visto más posibles casos y debería poder identificar aún más entidades.

3.7.4.3 Evaluación de modelos

Para evaluar los modelos se utilizó la métrica *F1 score* (Sección 2.2.2.4). El archivo de configuración *config.cfg* empleado por spaCy para entrenar el modelo, permite definir las métricas a emplear durante el proceso de entrenamiento y validación. Además, se muestra una métrica final consistente en el promedio ponderado de las otras métricas definidas. Es posible elegir el peso que tendrá cada una de ellas en esta métrica final. Por defecto, al tratarse de un modelo basado en un *pipeline* que sólo contiene un componente *ner*, las métricas usadas son *ents_f* (*F1 score*), *ents_p* (*precision*), *ents_r* (*recall*), donde se asigna el 100% del peso a la métrica *ents_f* para el cálculo de la métrica final. Esto podemos verlo en la imagen 22. Por lo tanto, estas dos son iguales, solo que a la última se la redondea a la segunda cifra decimal.

```
[training.score_weights]  
ents_per_type = null  
ents_f = 1.0  
ents_p = 0.0  
ents_r = 0.0
```

Imagen 22: Configuración de métricas en el archivo *config.cfg*

Para aquellos modelos con mejor desempeño tras la etapa de entrenamiento y validación, se generó una matriz de confusión con los relatos asignados al lote de prueba para obtener información sobre los falsos positivos y falsos negativos del modelo. Esto se hizo al ejecutar el script *matrizDeConfusion.py* (Anexo 4). El script original se obtuvo de un

post de Medium³⁰ y se modificó ligeramente para adaptarlo a este problema. Fue necesario recurrir a una fuente externa debido a que spaCy no incluye un método para evaluar los modelos obtenidos.

3.7.5 Fase 5: Evaluación

3.7.5.1 Evaluación de resultados

Al ejecutar el comando *train* en la línea de comandos, a fin de entrenar un modelo, se van presentando los resultados de las métricas empleadas para cada iteración sobre los relatos de validación. La última columna de cada iteración nos muestra la puntuación (Score), la cual, en este caso, se corresponde a la métrica *F1 score*.

A continuación, se presentan los resultados obtenidos en cada uno de los modelos durante el proceso de entrenamiento y validación. Cada modelo se referencia con el número asignado en la tabla 4.

Modelo 1

Resultados del entrenamiento en la imagen 23.

```
===== Training pipeline =====
[ Pipeline: ['tok2vec', 'ner']
[ Initial learn rate: 0.001
E  #      LOSS TOK2VEC  LOSS NER  ENTS_F  ENTS_P  ENTS_R  SCORE
-----
```

0	0	0.00	52.10	0.82	0.53	1.79	0.01
2	200	330.88	3180.11	35.79	43.59	30.36	0.36
4	400	29.85	309.46	53.45	51.67	55.36	0.53
6	600	48.73	198.62	59.79	70.73	51.79	0.60
8	800	52.42	172.38	53.85	58.33	50.00	0.54
10	1000	40.74	138.48	55.67	65.85	48.21	0.56
12	1200	31.41	82.09	61.05	74.36	51.79	0.61
14	1400	21.03	48.77	53.85	58.33	50.00	0.54
16	1600	34.56	71.79	54.55	55.56	53.57	0.55
18	1800	4.63	10.59	53.93	72.73	42.86	0.54
20	2000	51.43	44.50	47.73	65.62	37.50	0.48
22	2200	27.54	41.17	61.54	66.67	57.14	0.62
25	2400	29.08	41.20	51.11	67.65	41.07	0.51
27	2600	24.66	32.57	52.94	58.70	48.21	0.53
29	2800	29.38	24.83	57.43	64.44	51.79	0.57
31	3000	39.41	33.85	49.54	50.94	48.21	0.50
33	3200	23074.99	166.11	51.61	64.86	42.86	0.52
35	3400	142.61	77.82	56.86	63.04	51.79	0.57
37	3600	46.88	54.78	44.80	40.58	50.00	0.45
39	3800	14.07	14.08	61.36	84.38	48.21	0.61

Imagen 23: Resultados de entrenamiento y validación del modelo 1

³⁰ Consultado en fecha: 14/08/2021 <https://dulaj.medium.com/confusion-matrix-visualization-for-spaCy-ner-9e9d99120ee9>

Modelo 2

Resultados del entrenamiento en la imagen 24.

```
===== Training pipeline =====
Pipeline: ['tok2vec', 'ner']
Initial learn rate: 0.001
```

E	#	LOSS TOK2VEC	LOSS NER	ENTS_F	ENTS_P	ENTS_R	SCORE
0	0	0.00	111.09	0.00	0.00	0.00	0.00
0	200	488.82	2512.49	48.89	50.99	46.95	0.49
1	400	1601.81	918.13	65.53	74.42	58.54	0.66
1	600	61.76	487.21	71.38	75.51	67.68	0.71
2	800	847.48	425.38	73.65	90.27	62.20	0.74
2	1000	80.67	259.42	76.07	76.54	75.61	0.76
3	1200	103.40	257.40	77.65	75.00	80.49	0.78
4	1400	151.40	214.45	79.21	86.33	73.17	0.79
4	1600	72.66	147.22	79.38	80.12	78.66	0.79
5	1800	117.19	169.09	77.48	84.78	71.34	0.77
5	2000	129.92	125.10	82.50	84.62	80.49	0.83
6	2200	222.70	113.22	77.78	78.75	76.83	0.78
6	2400	207.38	132.53	72.41	83.33	64.02	0.72
7	2600	162.89	127.73	75.71	78.43	73.17	0.76
8	2800	153.73	130.86	79.63	80.62	78.66	0.80
9	3000	235.97	139.74	75.88	80.27	71.95	0.76
9	3200	98.79	90.52	79.13	80.89	77.44	0.79
11	3400	221.85	106.15	80.38	83.55	77.44	0.80
12	3600	229.76	120.20	82.72	83.75	81.71	0.83
14	3800	170.24	71.70	80.89	84.67	77.44	0.81
16	4000	144.51	58.08	79.75	82.89	76.83	0.80
18	4200	45.24	19.06	84.37	86.54	82.32	0.84
20	4400	101.58	23.80	82.65	85.62	79.88	0.83
22	4600	124.79	54.05	79.08	85.21	73.78	0.79
24	4800	820.51	74.66	75.43	87.20	66.46	0.75
26	5000	266.27	87.12	83.60	88.44	79.27	0.84
28	5200	7444.33	131.51	80.91	86.21	76.22	0.81
30	5400	101.75	40.83	82.87	84.71	81.10	0.83
32	5600	45.74	17.79	83.60	88.44	79.27	0.84
35	5800	81.73	30.75	78.64	79.87	77.44	0.79

Imagen 24: Resultados de entrenamiento y validación del modelo 2

Modelo 3

Resultados del entrenamiento en la imagen 25.

```
===== Training pipeline =====
Pipeline: ['tok2vec', 'ner']
Initial learn rate: 0.001
```

E	#	LOSS TOK2VEC	LOSS NER	ENTS_F	ENTS_P	ENTS_R	SCORE
0	0	0.00	188.09	0.00	0.00	0.00	0.00
0	200	1381.47	4079.41	26.85	26.49	27.21	0.27
1	400	855.80	1207.83	61.42	72.90	53.06	0.61
1	600	1888.71	668.50	74.91	80.47	70.07	0.75
2	800	58.59	352.25	70.14	71.63	68.71	0.70
2	1000	86.55	290.93	64.71	70.40	59.86	0.65
3	1200	90.12	217.50	69.26	72.06	66.67	0.69
4	1400	139.03	232.04	77.24	78.32	76.19	0.77
4	1600	608.85	241.54	72.58	89.11	61.22	0.73
5	1800	280.72	236.49	74.58	74.32	74.83	0.75
5	2000	121.56	140.29	78.23	78.23	78.23	0.78
6	2200	134.52	129.51	67.16	74.38	61.22	0.67
6	2400	118.47	95.24	75.81	80.77	71.43	0.76
7	2600	244.22	102.18	72.31	69.38	75.51	0.72
8	2800	839.71	128.27	74.37	79.23	70.07	0.74
9	3000	132.23	94.29	73.04	73.29	72.79	0.73
9	3200	170.16	119.45	67.14	69.85	64.63	0.67
11	3400	1064.08	196.88	80.42	82.73	78.23	0.80
12	3600	275.78	96.43	76.09	81.40	71.43	0.76
14	3800	222.26	68.99	73.61	81.15	67.35	0.74
16	4000	880.72	73.91	79.56	85.83	74.15	0.80
18	4200	161.30	67.90	73.72	79.53	68.71	0.74
20	4400	296.19	115.23	75.09	82.79	68.71	0.75
22	4600	344.79	81.75	76.69	85.71	69.39	0.77
24	4800	195.49	46.56	78.57	82.71	74.83	0.79
26	5000	94.71	39.76	74.63	82.64	68.03	0.75

Imagen 25: Resultados de entrenamiento y validación del modelo 3

Modelo 4

Resultados del entrenamiento en la imagen 26.

```
===== Training pipeline =====
Pipeline: ['tok2vec', 'ner']
Initial learn rate: 0.001
```

E	#	LOSS TOK2VEC	LOSS NER	ENTS_F	ENTS_P	ENTS_R	SCORE
0	0	0.00	157.36	0.27	0.17	0.67	0.00
0	200	1514.97	4267.14	37.27	57.75	27.52	0.37
1	400	979.36	774.80	53.21	50.92	55.70	0.53
1	600	45.83	464.48	73.68	83.76	65.77	0.74
2	800	506.61	411.91	75.76	86.96	67.11	0.76
2	1000	91.13	286.98	72.66	78.29	67.79	0.73
3	1200	98.76	306.63	73.27	72.08	74.50	0.73
4	1400	248.88	264.94	80.95	82.07	79.87	0.81
4	1600	505.31	226.67	77.66	79.58	75.84	0.78
5	1800	204.35	274.06	72.79	76.87	69.13	0.73
5	2000	119.56	161.33	80.66	78.85	82.55	0.81
6	2200	113.78	160.77	75.66	74.19	77.18	0.76
6	2400	83.83	102.14	78.62	80.85	76.51	0.79
7	2600	958.56	103.15	73.05	77.44	69.13	0.73
8	2800	183.84	156.57	77.94	86.18	71.14	0.78
9	3000	6166.71	162.75	80.74	90.08	73.15	0.81

Imagen 26: Resultados de entrenamiento y validación del modelo 4

Modelo 5

Resultados del entrenamiento en la imagen 27.

```
===== Training pipeline =====
Pipeline: ['tok2vec', 'ner']
Initial learn rate: 0.00030000000000000003
```

E	#	LOSS TOK2VEC	LOSS NER	ENTS_F	ENTS_P	ENTS_R	SCORE
0	0	0.00	157.36	0.27	0.17	0.67	0.00
0	200	478.90	4950.68	17.57	23.33	14.09	0.18
1	400	35.99	706.25	57.04	60.00	54.36	0.57
1	600	1067.29	669.46	80.58	86.82	75.17	0.81
2	800	23.81	305.65	82.39	86.67	78.52	0.82
2	1000	27.36	238.94	81.12	84.67	77.85	0.81
3	1200	26.14	169.72	83.96	85.42	82.55	0.84
4	1400	4457.00	299.22	86.39	87.59	85.23	0.86
4	1600	20.20	89.57	83.28	84.72	81.88	0.83
5	1800	38.76	141.93	86.99	88.81	85.23	0.87
5	2000	29.14	82.71	83.50	83.78	83.22	0.84
6	2200	27.26	84.27	84.85	85.14	84.56	0.85
6	2400	20.91	51.64	86.39	87.59	85.23	0.86
7	2600	27.82	59.95	85.71	91.60	80.54	0.86
8	2800	37.08	80.98	87.50	90.65	84.56	0.88
9	3000	30.40	67.77	87.63	92.54	83.22	0.88
10	3200	19.21	49.82	84.48	91.41	78.52	0.84
11	3400	35.26	69.11	82.22	91.74	74.50	0.82
13	3600	34.30	43.63	88.50	92.03	85.23	0.89
15	3800	17.53	32.31	86.64	93.75	80.54	0.87
17	4000	21.45	31.89	86.51	89.29	83.89	0.87
19	4200	32.14	49.13	85.51	90.30	81.21	0.86
21	4400	10.84	16.34	85.31	89.05	81.88	0.85
23	4600	26.61	29.00	85.81	88.57	83.22	0.86
25	4800	10.98	13.36	85.82	90.98	81.21	0.86
28	5000	14.85	16.77	81.78	91.67	73.83	0.82
30	5200	8.47	10.95	85.31	89.05	81.88	0.85

Imagen 27: Resultados de entrenamiento y validación del modelo 5

Modelo 6

Resultados del entrenamiento en la imagen 28.

```
===== Training pipeline =====
Pipeline: ['tok2vec', 'ner']
Initial learn rate: 0.001
```

E	#	LOSS TOK2VEC	LOSS NER	ENTS_F	ENTS_P	ENTS_R	SCORE
0	0	0.00	52.10	0.00	0.00	0.00	0.00
2	200	1814.77	3814.64	21.43	32.14	16.07	0.21
4	400	108.22	352.78	36.73	42.86	32.14	0.37
6	600	80.84	113.73	30.43	38.89	25.00	0.30
8	800	65.61	55.24	42.42	48.84	37.50	0.42
10	1000	51.22	33.42	41.30	52.78	33.93	0.41
12	1200	62.58	34.55	32.94	48.28	25.00	0.33
14	1400	50.99	33.43	41.76	54.29	33.93	0.42
16	1600	99.34	41.13	32.50	54.17	23.21	0.33
18	1800	109.68	44.43	51.76	75.86	39.29	0.52
20	2000	95.72	29.84	45.98	64.52	35.71	0.46
22	2200	42.04	18.84	36.36	50.00	28.57	0.36
25	2400	108.11	35.57	43.96	57.14	35.71	0.44
27	2600	93.63	33.97	55.32	68.42	46.43	0.55
29	2800	414.27	22.41	50.51	58.14	44.64	0.51
31	3000	24.92	10.04	38.64	53.12	30.36	0.39
33	3200	54.55	17.94	52.25	52.73	51.79	0.52
35	3400	289.25	52.83	37.36	48.57	30.36	0.37
37	3600	86.74	29.98	40.43	50.00	33.93	0.40
39	3800	307.87	68.29	40.96	62.96	30.36	0.41
42	4000	87.53	21.46	41.30	52.78	33.93	0.41
45	4200	0.00	0.00	39.13	50.00	32.14	0.39

Imagen 28: Resultados de entrenamiento y validación del modelo 6

Modelo 7

Resultados del entrenamiento en la imagen 29.

```
===== Training pipeline =====
Pipeline: ['tok2vec', 'ner']
Initial learn rate: 0.001
```

E	#	LOSS	TOK2VEC	LOSS NER	ENTS_F	ENTS_P	ENTS_R	SCORE
0	0	0.00		111.09	0.00	0.00	0.00	0.00
0	200	217.54		2709.18	24.49	37.04	18.29	0.24
1	400	1963.04		1044.59	59.70	76.92	48.78	0.60
1	600	144.79		486.64	67.14	79.83	57.93	0.67
2	800	9171.09		394.86	63.12	75.42	54.27	0.63
2	1000	161.57		256.61	74.40	84.50	66.46	0.74
3	1200	156.93		203.49	73.52	75.16	71.95	0.74
4	1400	157.63		141.50	73.50	87.39	63.41	0.73
4	1600	173.71		134.68	67.59	77.78	59.76	0.68
5	1800	141.38		110.31	74.17	81.16	68.29	0.74
5	2000	252.84		111.65	73.98	76.13	71.95	0.74
6	2200	162.52		85.26	73.58	81.48	67.07	0.74
6	2400	128.26		72.85	74.68	79.86	70.12	0.75
7	2600	298.83		93.40	71.38	72.05	70.73	0.71
8	2800	155.54		89.57	75.00	81.43	69.51	0.75
9	3000	153.31		67.51	69.53	84.35	59.15	0.70
9	3200	166.22		48.15	74.91	85.83	66.46	0.75
11	3400	355.00		92.78	76.16	77.36	75.00	0.76
12	3600	253.88		93.89	76.82	84.06	70.73	0.77
14	3800	325.74		68.26	73.43	86.07	64.02	0.73
16	4000	169.95		49.88	76.19	86.15	68.29	0.76
18	4200	343.39		42.48	78.81	86.23	72.56	0.79
20	4400	111.98		23.58	75.95	78.95	73.17	0.76
22	4600	155.19		29.19	72.90	77.40	68.90	0.73
24	4800	189.86		29.73	70.14	81.45	61.59	0.70
26	5000	162.87		23.84	76.87	82.52	71.95	0.77
28	5200	97.45		15.67	72.47	84.55	63.41	0.72
30	5400	212.99		28.63	74.21	76.62	71.95	0.74
32	5600	296.87		39.74	76.51	85.07	69.51	0.77
35	5800	193.28		28.60	72.96	78.32	68.29	0.73

Imagen 29: Resultados de entrenamiento y validación del modelo 7

Modelo 8

Resultados del entrenamiento en la imagen 30.

```
===== Training pipeline =====
Pipeline: ['tok2vec', 'ner']
Initial learn rate: 0.001
```

E	#	LOSS	TOK2VEC	LOSS NER	ENTS_F	ENTS_P	ENTS_R	SCORE
0	0	0.00		188.09	0.00	0.00	0.00	0.00
0	200	1919.86		3670.37	13.91	19.28	10.88	0.14
1	400	90.11		714.84	49.01	58.49	42.18	0.49
1	600	198.54		526.63	63.24	75.47	54.42	0.63
2	800	115.63		305.52	66.90	70.15	63.95	0.67
2	1000	158.63		227.40	62.72	64.29	61.22	0.63
3	1200	124.20		159.21	67.16	73.39	61.90	0.67
4	1400	148.58		181.45	70.90	78.51	64.63	0.71
4	1600	188.46		101.21	74.07	81.30	68.03	0.74
5	1800	180.53		117.51	69.89	77.05	63.95	0.70
5	2000	153.11		81.21	63.57	73.87	55.78	0.64
6	2200	147.02		81.46	72.00	77.34	67.35	0.72
6	2400	149.77		65.96	66.20	68.61	63.95	0.66
7	2600	390.09		101.93	67.94	77.39	60.54	0.68
8	2800	238.99		78.50	70.88	73.19	68.71	0.71
9	3000	255.38		86.79	66.16	75.00	59.18	0.66
9	3200	168.89		59.61	71.60	83.64	62.59	0.72

Imagen 30: Resultados de entrenamiento y validación del modelo 8

Modelo 9

Resultados del entrenamiento en la imagen 31.

```
===== Training pipeline =====
Pipeline: ['tok2vec', 'ner']
Initial learn rate: 0.001
```

E	#	LOSS TOK2VEC	LOSS NER	ENTS_F	ENTS_P	ENTS_R	SCORE
0	0	0.00	157.36	0.00	0.00	0.00	0.00
0	200	1835.31	3448.02	35.90	49.41	28.19	0.36
1	400	292.90	884.77	68.44	78.95	60.40	0.68
1	600	459.84	551.07	70.07	76.80	64.43	0.70
2	800	150.85	351.74	79.09	91.23	69.80	0.79
2	1000	160.34	244.40	77.30	81.95	73.15	0.77
3	1200	183.69	219.33	75.49	89.81	65.10	0.75
4	1400	673.99	237.67	74.44	84.62	66.44	0.74
4	1600	128.04	94.18	80.44	89.34	73.15	0.80
5	1800	274.22	153.55	76.49	94.12	64.43	0.76
5	2000	202.63	116.89	76.36	83.33	70.47	0.76
6	2200	231.21	137.52	68.27	85.00	57.05	0.68
6	2400	204.48	83.01	76.92	80.29	73.83	0.77
7	2600	151.88	67.70	75.49	89.81	65.10	0.75
8	2800	316.48	101.52	73.06	81.15	66.44	0.73
9	3000	148.16	62.45	79.53	96.19	67.79	0.80
10	3200	223.55	65.01	81.29	87.60	75.84	0.81
11	3400	365.82	90.12	79.58	83.70	75.84	0.80
13	3600	584.85	101.20	74.81	86.73	65.77	0.75
15	3800	227.75	53.92	78.79	90.43	69.80	0.79
17	4000	558.93	61.79	75.49	89.81	65.10	0.75
19	4200	17663.33	115.57	79.41	87.80	72.48	0.79
21	4400	293.92	48.15	82.27	87.22	77.85	0.82
23	4600	169.15	25.53	80.00	91.38	71.14	0.80
25	4800	174.72	30.14	81.75	89.60	75.17	0.82
28	5000	189.56	29.68	78.16	91.07	68.46	0.78
30	5200	49.38	8.37	74.42	88.07	64.43	0.74
32	5400	175.30	24.57	77.78	86.78	70.47	0.78
34	5600	322.66	47.90	76.70	82.31	71.81	0.77
36	5800	302.08	43.43	80.30	92.17	71.14	0.80
38	6000	217.53	33.01	78.20	88.89	69.80	0.78

Imagen 31: Resultados de entrenamiento y validación del modelo 9

Modelo 10

Resultados del entrenamiento en la imagen 32.

```
===== Training pipeline =====
Pipeline: ['tok2vec', 'ner']
Initial learn rate: 0.01
```

E	#	LOSS TOK2VEC	LOSS NER	ENTS_F	ENTS_P	ENTS_R	SCORE
0	0	0.00	157.36	0.00	0.00	0.00	0.00
0	200	601711.39	5456.69	29.15	36.73	24.16	0.29
1	400	4621029.36	5389.36	22.12	38.98	15.44	0.22
1	600	1210561.30	2990.39	23.23	46.94	15.44	0.23
2	800	93036.66	1392.51	23.85	37.68	17.45	0.24
2	1000	5695600.59	2829.48	12.56	18.92	9.40	0.13
3	1200	4144922.71	2405.64	25.81	41.18	18.79	0.26
4	1400	3424710.47	2353.78	41.56	58.54	32.21	0.42
4	1600	4263363.15	1433.15	28.14	32.46	24.83	0.28
5	1800	976172.72	1282.60	35.29	47.19	28.19	0.35
5	2000	1003558.07	1220.48	36.64	42.48	32.21	0.37
6	2200	563093.33	1021.35	63.64	73.04	56.38	0.64
6	2400	164887.59	644.69	56.69	68.57	48.32	0.57
7	2600	448639.81	860.49	48.10	64.77	38.26	0.48
8	2800	7295059.68	1662.13	56.45	70.71	46.98	0.56
9	3000	7876644.46	1359.04	49.36	67.44	38.93	0.49
10	3200	238989.59	677.14	61.78	72.73	53.69	0.62
11	3400	167241.19	627.75	63.78	77.14	54.36	0.64
13	3600	188407.45	568.38	62.66	86.90	48.99	0.63
15	3800	1433626.57	822.40	71.91	81.36	64.43	0.72
17	4000	663713.59	580.72	64.29	78.64	54.36	0.64
19	4200	1220782.86	630.09	66.14	80.00	56.38	0.66
21	4400	642520.92	478.68	71.16	80.51	63.76	0.71
23	4600	928901.06	516.66	66.93	82.35	56.38	0.67
25	4800	482761.38	353.26	72.39	81.51	65.10	0.72
28	5000	1123009.41	553.55	71.59	79.51	65.10	0.72
30	5200	4151694.62	748.07	66.91	75.00	60.40	0.67
32	5400	307688.30	347.42	65.89	77.98	57.05	0.66
34	5600	1071573.25	444.58	71.43	81.20	63.76	0.71
36	5800	3688590.36	583.49	70.73	89.69	58.39	0.71
38	6000	319202.78	304.18	70.50	82.14	61.74	0.70
40	6200	829816.49	350.00	70.50	82.14	61.74	0.70
42	6400	239527.98	245.51	72.00	78.57	66.44	0.72

Imagen 32: Resultados de entrenamiento y validación del modelo 10

Modelo 11

Resultados del entrenamiento en la imagen 33.

```
===== Training pipeline =====
Pipeline: ['tok2vec', 'ner']
Initial learn rate: 0.003
```

E	#	LOSS TOK2VEC	LOSS NER	ENTS_F	ENTS_P	ENTS_R	SCORE
0	0	0.00	157.36	0.00	0.00	0.00	0.00
0	200	2543.49	2323.34	27.62	47.54	19.46	0.28
1	400	491.83	782.62	59.56	65.85	54.36	0.60
1	600	416.36	583.09	56.82	65.22	50.34	0.57
2	800	1440.76	517.81	62.65	78.00	52.35	0.63
2	1000	1028.99	411.42	74.33	86.61	65.10	0.74
3	1200	993.35	394.17	68.10	73.08	63.76	0.68
4	1400	13518.29	358.10	78.63	91.15	69.13	0.79
4	1600	1546.66	235.26	69.66	78.81	62.42	0.70
5	1800	3213.47	347.24	78.36	88.24	70.47	0.78
5	2000	1844.08	241.08	67.92	77.59	60.40	0.68
6	2200	2271.61	227.53	70.92	75.19	67.11	0.71
6	2400	1817.04	142.95	58.70	59.72	57.72	0.59
7	2600	2913.23	172.01	66.43	70.99	62.42	0.66
8	2800	1836.20	174.85	64.03	62.99	65.10	0.64
9	3000	3298.16	219.43	74.81	83.47	67.79	0.75

Imagen 33: Resultados de entrenamiento y validación del modelo 11

Resultados del entrenamiento en la imagen 34.

Imagen 34: Resultados de entrenamiento y validación del modelo 12

Tabla 5: Resumen de puntajes obtenidos para los 12 modelos entrenados

El modelo con mejor puntaje en la métrica *F1 score* y *recall* es el número 5. Para la métrica *precision* el mejor es el modelo 4, pero el que le sigue es el 5 nuevamente. De aquí podemos concluir que los mejores resultados según estas métricas los obtenemos con el modelo 5.

Para los modelos 4, 5 y 12 se utilizó el lote de 50 relatos de prueba con los cuales se generó una matriz de confusión. A continuación, se presentan las 3 matrices para cada uno de estos modelos. En los ejes de estas matrices se presentan los 5 tipos de entidades más la entidad “O” la cual quiere decir *output* haciendo alusión a que esa palabra está fuera de cualquiera de los 5 tipos de entidades objetivo. Por lo tanto, si a una palabra se la etiqueta como “O” es equivalente a decir que el modelo consideró que esa palabra no pertenece a ninguno de los 5 tipos de entidades analizados.

Modelo 5

Matriz de confusión a partir de 50 relatos en imagen 35

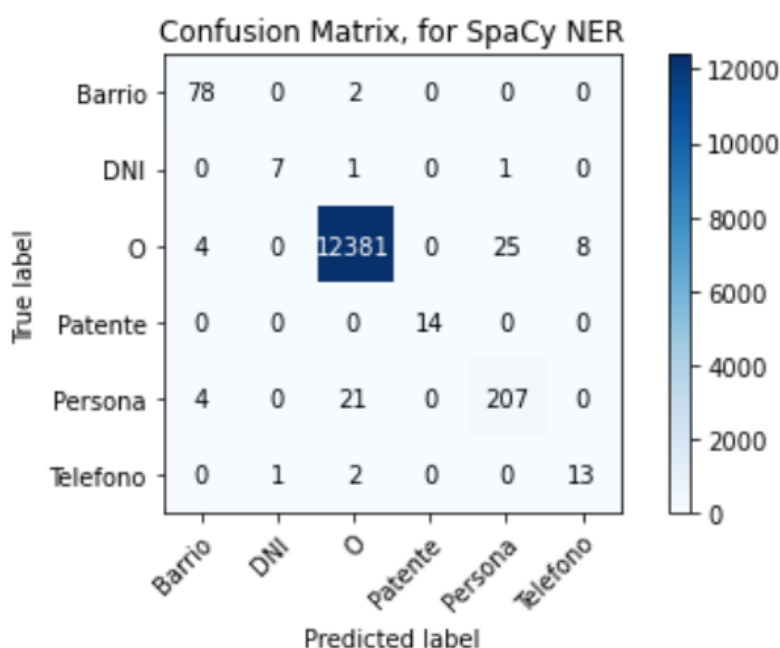


Imagen 35: Matriz de confusión del modelo 5

Al tratarse del modelo con mejores resultados, se calcularon además las métricas *precision*, *recall* y *F1 score* para cada tipo de entidad (tabla 6). Esto se hizo a partir de la matriz de confusión de la imagen 35.

Tipo de entidad	Precision	Recall	F1-Score
Persona	0,888	0,892	0,890
Barrio	0,907	0,975	0,940
DNI	0,875	0,778	0,824
Patente	1,000	1,000	1,000
Teléfono	0,812	0,812	0,812

Tabla 6: Precision, recall y F1 score por tipo de entidad del modelo 5

Modelo 4

Matriz de confusión a partir de 50 relatos en imagen 36

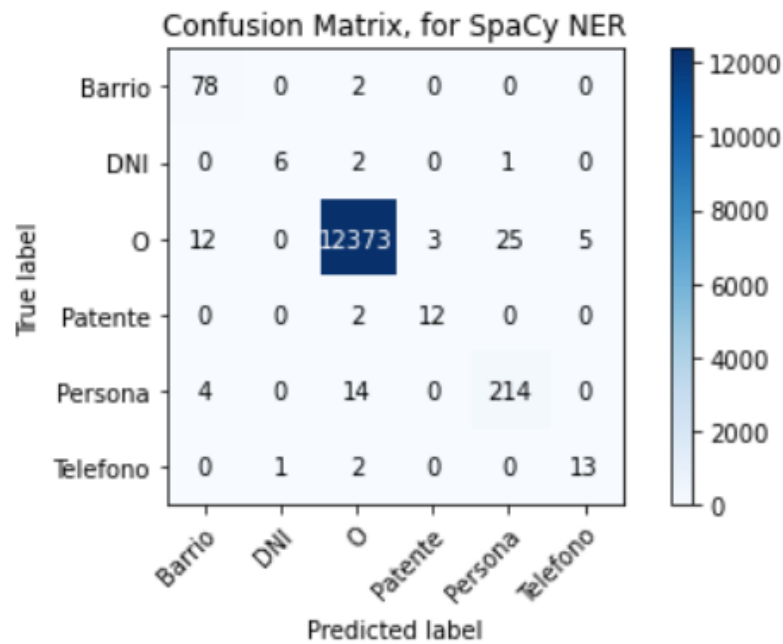


Imagen 36: Matriz de confusión del modelo 4

Modelo 12

Matriz de confusión a partir de 50 relatos en imagen 37

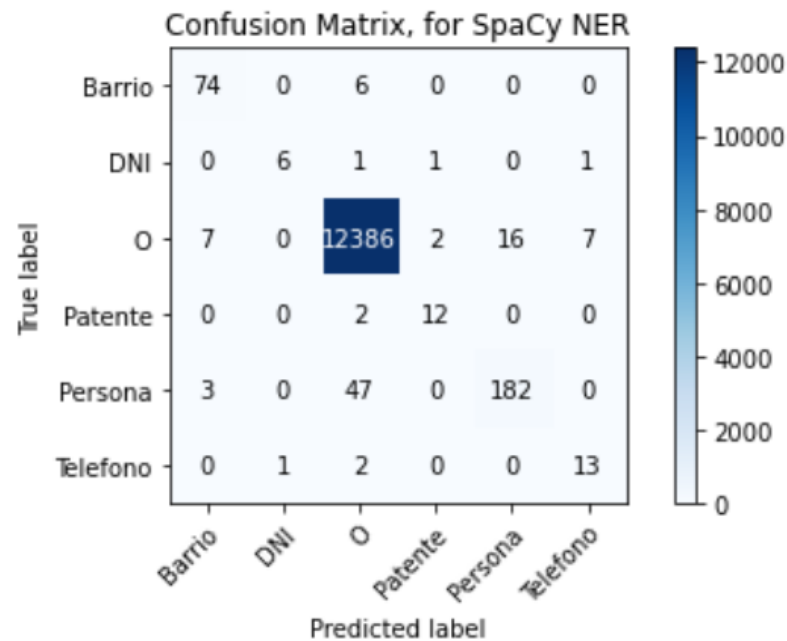


Imagen 37: Matriz de confusión del modelo 12

3.7.5.2 Revisión del proceso

Como era de esperar, el modelo optimizado para exactitud da mejores resultados. Por ello, tanto el modelo 5 como el 4 son los dos mejores. Sin embargo, el modelo 12, de eficiencia, también presenta resultados aceptables.

A la hora de analizar las 3 matrices de confusión presentadas podemos realizar las siguientes observaciones:

1. El modelo 5 identificó correctamente a más entidades del tipo barrio, DNI, patente y teléfono que los otros modelos.
2. El modelo 4 identificó correctamente a más entidades de tipo persona que el resto de modelos.
3. Para los 3 modelos, casi todas las entidades de tipo barrio fueron identificadas correctamente, y aquellas pocas que no, fueron marcadas como “O”.
4. Para los 3 modelos, podemos observar que, para los 5 tipos de entidades, aquellas que no fueron correctamente marcadas, en su mayoría fueron etiquetadas como “O”.
5. Siguiendo el punto anterior y si ignoramos los casos donde se etiquetó equivocadamente como tipo “O” en el modelo 5, encontramos que:
 - a. Hubo 4 entidades que fueron etiquetadas como persona cuando eran del tipo barrio, por lo tanto, vemos que existe una confusión muy pequeña en el modelo al diferenciar estos dos tipos de entidades entre sí.
 - b. Para la entidad de tipo DNI encontramos dos pequeñas confusiones del modelo: en un caso se etiquetó a un DNI como si fuera persona y en otro caso se etiquetó a un teléfono como si fuera DNI. Evidentemente el primer caso es el que más llama la atención. Si bien parecen pocos errores, al tratarse de un lote tan pequeño de entidades de tipo DNI, representan un porcentaje medianamente relevante.
 - c. Para la entidad de tipo teléfono encontramos una conexión con el caso anterior, existe una confusión con el tipo DNI.
 - d. En este modelo la entidad de tipo patente fue la que presentó mejor resultado ya que las 14 entidades fueron correctamente etiquetadas como patente y no hubo otro tipo de entidad que se haya confundido con esta.
 - e. Por último, para el tipo de entidad persona nos encontramos con el mismo caso extraño donde se etiquetó como persona a un DNI, además de la pequeña confusión al etiquetar como barrio a 4 de las entidades persona.
 - f. Como conclusión de los puntos anteriores podemos ver que el modelo debería aprender a distinguir mejor las entidades de tipo teléfono con las de tipo DNI y las de tipo barrio con las de tipo persona.

Para mejorar los resultados obtenidos es importante atacar los problemas planteados en los puntos 3 y 5f. Para ello se podría realizar lo siguiente:

1. Aumentar la cantidad de relatos anotados para el lote de entrenamiento y validación. Con ello entrenar un nuevo modelo.
2. Aplicar técnicas de Aumentación de Datos para conseguir nuevos relatos que contengan en su mayoría entidades de los tipos minoritarios, como son patente, teléfono y DNI, en otras palabras, balancear mejor los tipos de entidades.
3. Otra opción para el punto anterior es filtrar los relatos anotados, a fin de incorporar aquellos que presenten entidades de esos 3 tipos y descartar los que posean mayoría de entidades de tipo Barrio y/o Persona.

4. Utilizar alguna librería como Gensim³¹ para entrenar vectores de palabras específicos para este problema, ya que los empleados actualmente son tomados del modelo pre entrenado de spaCy: *es_core_news_lg*. De esta manera se podrían obtener mejores resultados.

3.7.6 Fase 6: Implementación

3.7.6.1 Desarrollo del sistema

En la sección 3.4 se presentaron las herramientas utilizadas durante todo el proyecto. Python con spaCy, Brat y WSL fueron empleados en las fases anteriores de la metodología, desde la comprensión de los datos hasta la evaluación. El resto de herramientas fueron utilizadas para desarrollar la base de datos, servidor y cliente del sistema web.

En la sección siguiente, 3.8, se presentan capturas de pantalla del sistema web desarrollado. A través de esa interfaz gráfica el usuario podrá analizar nuevos relatos de denuncias para identificar sus entidades y visualizarlas. Estos relatos cargados son enviados desde React a la API de Flask. Allí la API se comunica con el script de spaCy, se aplica el modelo entrenado al texto y se obtienen las entidades identificadas, con su tipo y ubicación. Estos datos se almacenan en la base de datos MongoDB. Finalmente, la API responde al cliente React con un relato anotado (con los datos de las entidades) para que éste las muestre al usuario y genere o actualice las visualizaciones a través de Recharts.

3.7.6.2 Anonimización de relatos

Como se mencionó en la fase de Preparación de los datos, un total de 70 relatos fueron reservados y anonimizados para ser cargados en el sistema web desarrollado.

La anonimización fue necesaria debido a la sensibilidad de los datos manejados en los relatos, además de tratarse de un requisito en el acuerdo de confidencialidad realizado para proteger los datos de los denunciantes, denunciados, testigos y de todos los involucrados o mencionados en una denuncia.

El proceso consistió en leer los 70 relatos y reemplazar los datos sensibles de la siguiente manera:

- Las entidades de tipo persona fueron reemplazadas por nombres de personajes de distintas series como los Simpsons y Juego de Tronos. En algunos casos también se utilizaron nombres inventados o genéricos.
- Las entidades de tipo barrio y nombres de calles, localidades, plazas, negocios, etc. se reemplazaron por nombres de ciudades de Buenos Aires, estados de Estados Unidos, entre otros.
- Las entidades de tipo DNI, patente y teléfono se reemplazaron por otros números (y letras para el caso de las patentes) al azar, pero conservando el formato original.
- Similar al punto anterior se hizo para las edades de las personas, los números de calles, de blocks departamentales, casas, etc.

³¹ <https://radimrehurek.com/gensim/>

3.7.6.3 Informe final del proyecto

Como se explicó en la fase anterior el mejor modelo, según la métrica *F1 score*, fue el modelo optimizado para exactitud, el número 5 con un puntaje de 85.31. El mismo se entrenó con un lote de 350 relatos anotados, se validó con un lote de 50 y se probó con otros 50 al generar una matriz de confusión. En el anexo 5 se encuentra toda la configuración del archivo *config.cfg* para el modelo 5.

Debido a un inconveniente, desarrollado en la siguiente sección, 3.8, se implementó el modelo 12 en el sistema desarrollado. Además, se emplearon las denuncias anonimizadas para realizar una prueba final de su correcto funcionamiento.

3.8 Sistema desarrollado

A continuación, se presentan capturas de pantalla del sistema desarrollado. Los datos que allí figuren provienen de relatos de denuncias anonimizadas.

3.8.1 Pestaña INICIO

En las imágenes 38, 39 y 40, vemos la pestaña inicio la cual presenta el proyecto en cuestión, las funcionalidades que incluye y algunos datos del alumno que lo realizó.

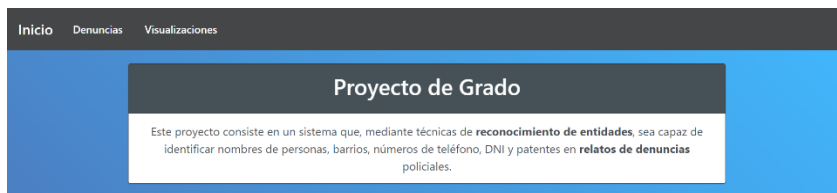


Imagen 38: Sistema web - Pestaña Inicio - Presentación



Imagen 40: Sistema web - Pestaña Inicio - Funcionalidades

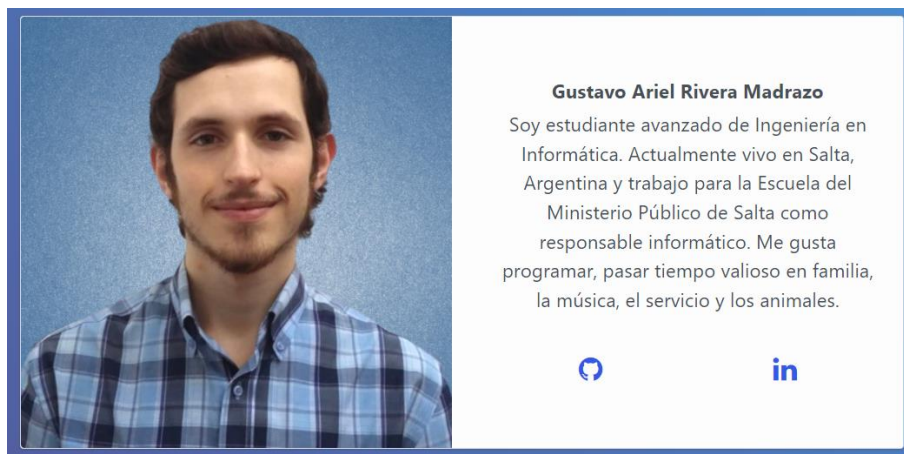


Imagen 39: Sistema web - Pestaña Inicio - Datos del Alumno

3.8.2 Pestaña DENUNCIAS

En la imagen 41 vemos el listado de todos los relatos de denuncias cargados. Por cada relato se indica el título o identificador de la denuncia a la que pertenece y el cuerpo del relato. Además, se dispone de un buscador para filtrar las denuncias por palabras clave, un desplegable para seleccionar la cantidad de denuncias que se desean mostrar por página y 5 botones de color correspondientes a los 5 tipos de entidades analizados. A través de estos botones podemos cambiar el color con el cual se resaltan las entidades identificadas en la pestaña de *Entidades*.

Gustavo Ariel Rivera Madrazo

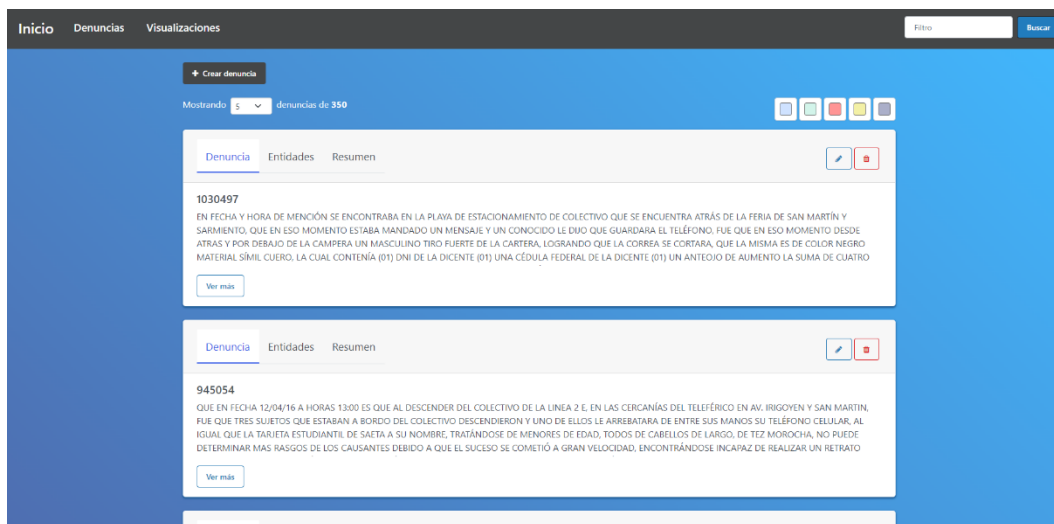


Imagen 41: Sistema web - Pestaña Denuncias – Listado de denuncias

En la imagen 42, se muestra el formulario para cargar los datos de una nueva denuncia, en este caso el título o identificador y el cuerpo correspondiente al relato.

Imagen 42: Sistema web - Pestaña Denuncias - Cargar denuncia

En la imagen 43 vemos el identificador y relato de una denuncia cargada. Para cada denuncia se nos presentan las opciones de desplegar el total del texto del relato con el botón *Ver más*, editar el título o relato de una denuncia (botón azul con el ícono de un lápiz) y eliminar la denuncia (botón rojo con el ícono de un tacho de basura).

Imagen 43: Sistema web - Pestaña Denuncias - Texto de una denuncia

En la imagen 44 se muestra el apartado *Entidades* de una denuncia. Inmediatamente luego de cargar la denuncia esta es analizada, se identifican sus entidades automáticamente y se marcan en esta pestaña. Como se observa en la imagen para cada tipo de entidad se emplea un color diferente el cual se puede modificar desde los botones de colores. La imagen está con las entidades corregidas, pero al momento de marcar automáticamente el modelo se equivocó al marcar como teléfono al número de licencia (1242) de un chofer de remis. En ese caso, al hacer click en el número de licencia se mostraba la opción para quitarle la etiqueta de teléfono. A su vez es posible marcar manualmente entidades que el modelo pasó por alto. Esto se haría eligiendo en el desplegable el tipo de entidad a marcar, seleccionando con el mouse el fragmento de texto a marcar y dando click al botón *Marcar*.

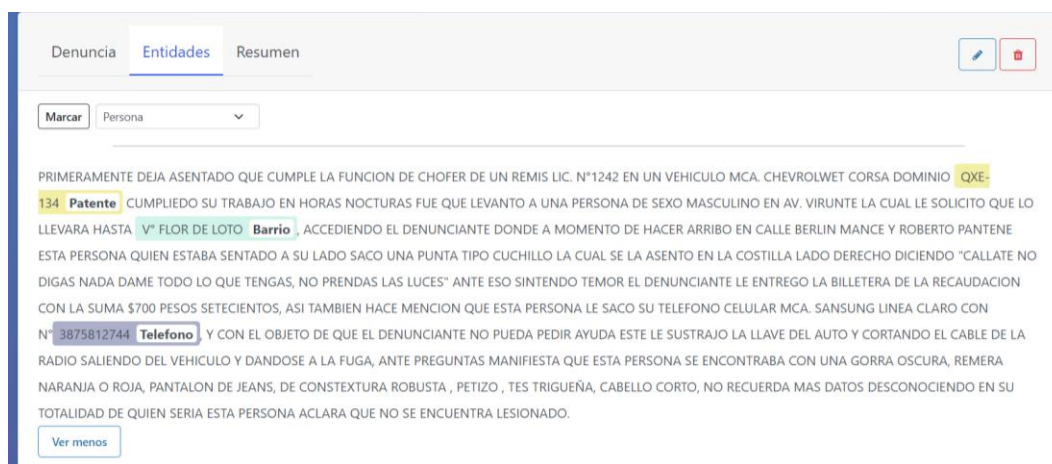


Imagen 44: Sistema web - Pestaña Denuncias - Entidades de una denuncia

En la imagen 45 se muestra el apartado *Resumen* de una denuncia. Nos indica la cantidad de denuncias de cada tipo encontradas. Aquellas para las que no hubo resultados no se muestran. Al hacer click en alguno de los tipos, se despliegan las entidades marcadas con ese tipo.

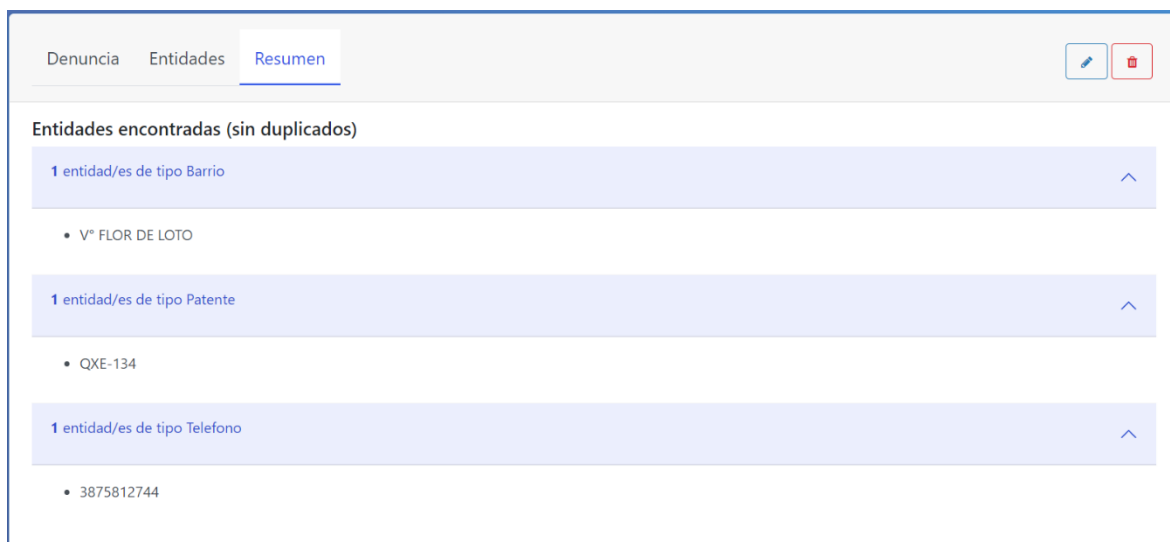


Imagen 45: Sistema web - Pestaña Denuncias - Resumen de una denuncia

3.8.3 Pestaña VISUALIZACIONES

En la imagen 46 vemos una imagen para cada uno de los 4 tipos de visualizaciones que presenta el sistema. Al colocar el mouse encima de estas tarjetas se muestra una breve explicación sobre la visualización y al hacer click podemos acceder a ella. Los 4 tipos de gráficos son: gráfico de torta (imagen 47), gráfico de barra (imagen 48), gráfico de línea (imagen 49) y gráfico mapa de árbol (imagen 50). Todos los gráficos, excepto el de torta, permiten seleccionar desde el desplegable el tipo de entidad que se desea visualizar. Por defecto está activado el conteo para los 5 tipos de entidades (Opción: “Todas las entidades”). Además, los gráficos de barra y línea presentan una barra deslizante en la parte inferior para filtrar los datos que se deseen visualizar simultáneamente.

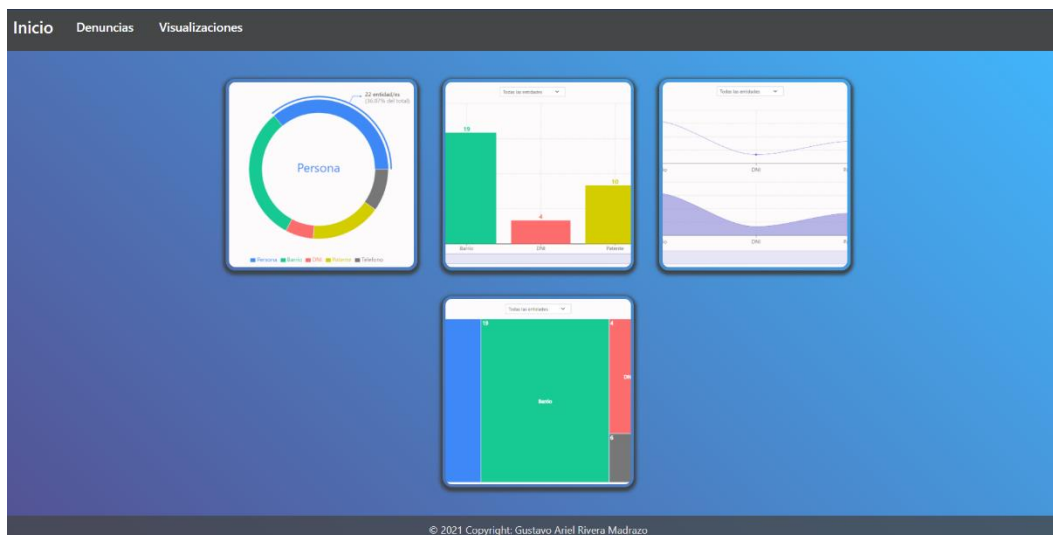


Imagen 47: Sistema web - Pestaña Visualizaciones – Visualizaciones disponibles

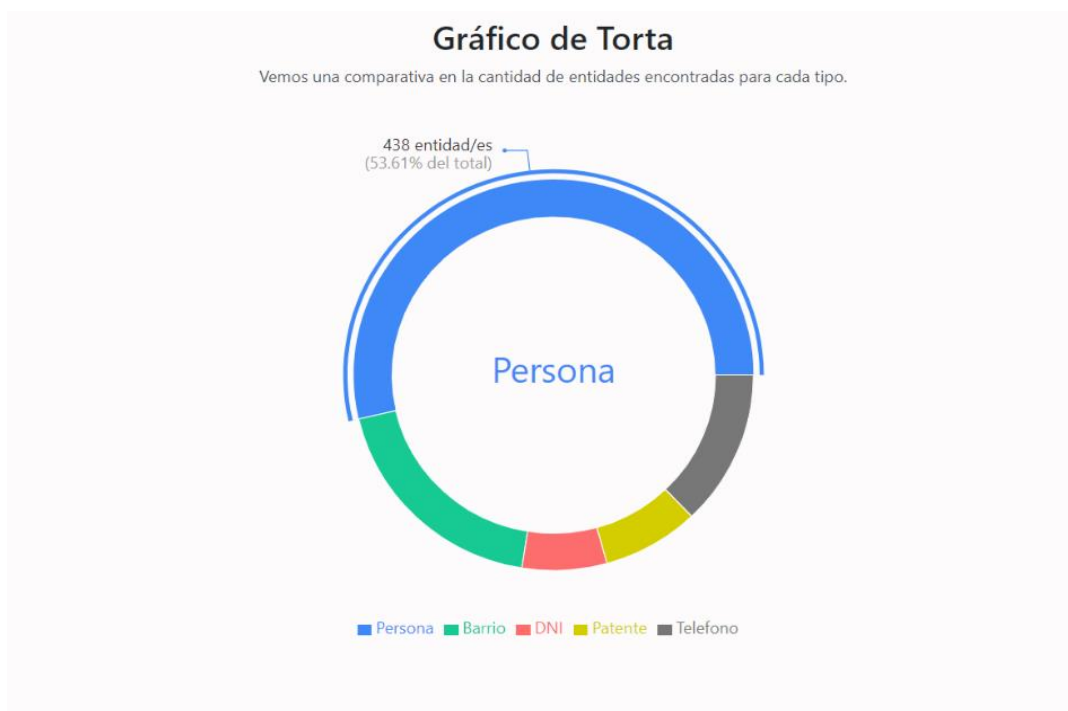


Imagen 46: Sistema web - Pestaña Visualizaciones – Gráfico de Torta

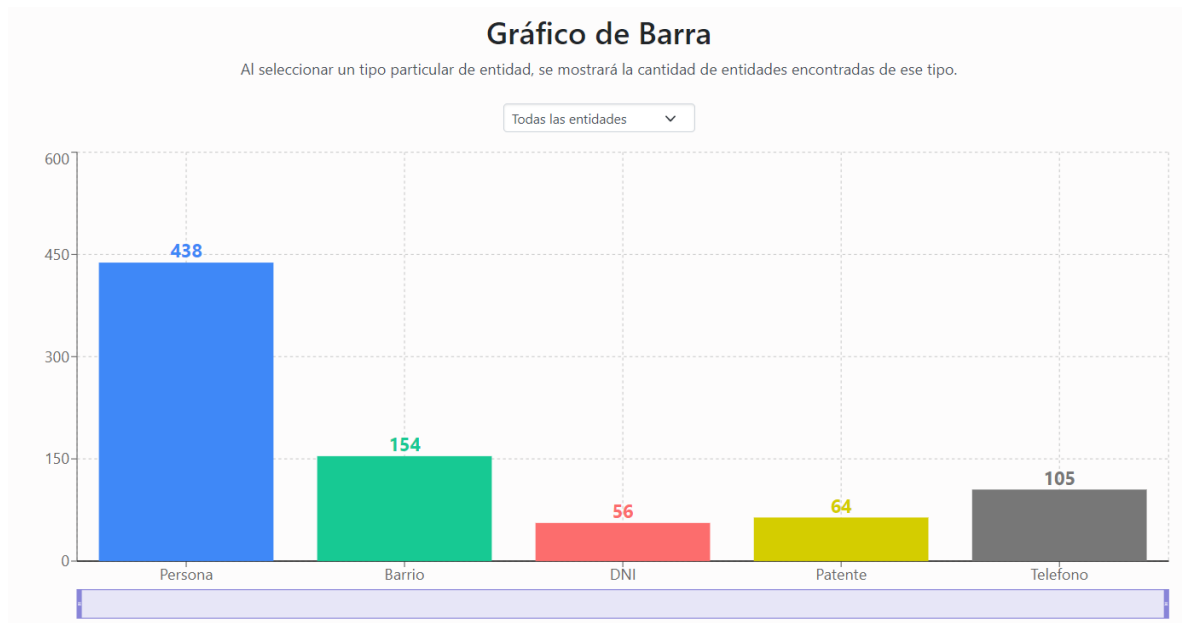


Imagen 48: Sistema web - Pestaña Visualizaciones – Gráfico de Barra



Imagen 49: Sistema web - Pestaña Visualizaciones – Gráfico de Línea

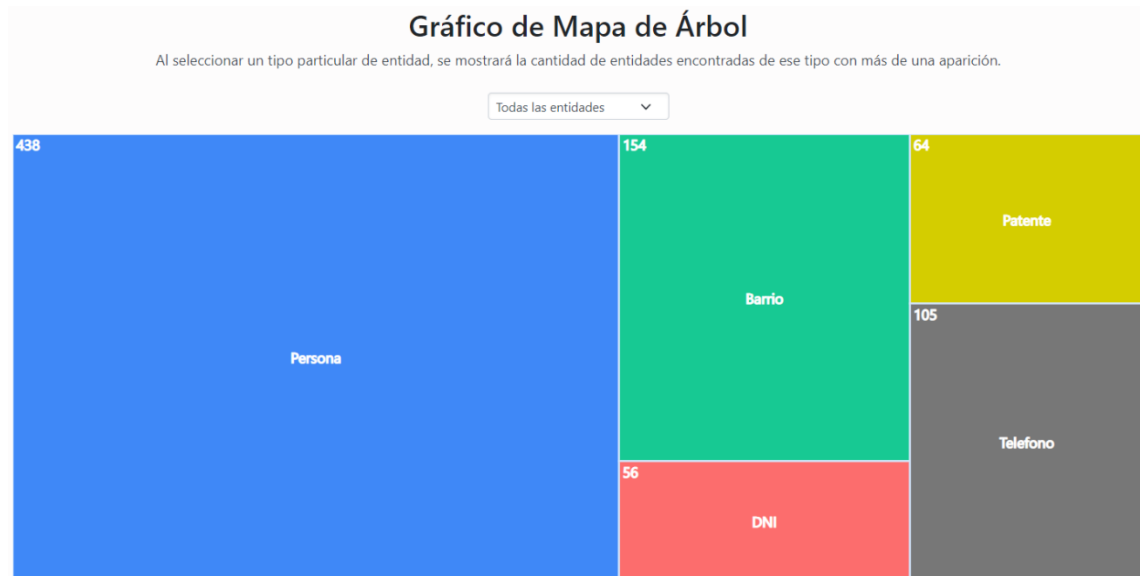


Imagen 50: Sistema web - Pestaña Visualizaciones – Gráfico Mapa de Árbol

Los datos presentados en las visualizaciones cumplen el propósito de mostrar visualmente las entidades que el modelo fue capaz de reconocer en los relatos. Estos datos pueden resultar de interés a un analista, permitiéndole con un simple vistazo obtener información relevante, por ejemplo: saber cuáles son los barrios que más aparecen en las denuncias, o los nombres de personas denunciantes o denunciadas que más se mencionan. Esto puede permitir detectar los casos, tristemente bastante comunes, de personas que son denunciadas reiteradas veces por delitos menores como hurto, consumo de narcóticos, etc. y que en algún momento terminan cometiendo un delito grave, como un homicidio, debido a la inacción o falta de rigurosidad en el accionar de la justicia.

Como se planteó en el alcance del proyecto (sección 3.1.3), un sistema más completo implicaría el reconocimiento de relaciones entre entidades, con el cual se podrían generar visualizaciones de grafos, mucho más interesantes para un analista. Con este tipo de visualizaciones sería posible identificar que entidades suelen mencionarse en conjunto y encontrar patrones útiles. Por ejemplo, ver que cierto apodo de un delincuente está muy relacionado a un barrio en particular donde suele delinquir frecuentemente.

3.8.4 Modelo Utilizado

A la hora de desplegar el sistema de manera gratuita se presenta un inconveniente: la plataforma Heroku donde se desplegó el servidor Flask y el modelo entrenado de spaCy presenta una limitación de 500 MB en su versión gratis. Este tamaño era superado en su totalidad con los archivos del modelo 5, sin incluir el resto de archivos, el cual, como se mencionó en la Fase de Modelado de la sección 3.7, solamente su archivo *vectors* presenta un tamaño próximo a los 600 MB. Por este motivo se optó por desplegar el sistema, pero con el modelo 12, el cual, al ser optimizado para eficiencia, su archivo *vectors* pesa casi 0MB. Desde luego los resultados del modelo 12 son inferiores a los del modelo 5 pero aún así se encuentra entre los 3 mejores modelos obtenidos.

Es posible acceder al código fuente del sistema y al sistema final a través de los siguientes enlaces a los repositorios de GitHub:

- Servidor – *Backend* y modelo:
<https://github.com/GustavoRiveraM/Analizador-Denuncias-Backend>
- Cliente - *Frontend*:
<https://github.com/GustavoRiveraM/Analizador-Denuncias-Frontend>
- Sistema: <https://analizadordedenuncias.netlify.app/>

3.9 Análisis FODA

Esta técnica examina el proyecto desde cada uno de los aspectos FODA (fortalezas, oportunidades, debilidades y amenazas) para aumentar el espectro de riesgos identificados, incluidos los riesgos generados internamente. El análisis FODA identifica luego cualquier oportunidad para el proyecto con origen en las fortalezas de la organización y cualquier amenaza con origen en las debilidades de la organización. El análisis también examina el grado en el que las fortalezas de la organización contrarrestan las amenazas, e identifica las oportunidades que pueden servir para superar las debilidades (Project Managment Institute, 2013, pág. 326)

Análisis Interno

- Fortalezas:
 - Experiencia académica en tareas de análisis de datos.
 - Experiencia académica en tareas de gestión de proyectos.
 - Experiencia académica en tareas de desarrollo web.
 - Capacidad de ofrecer una herramienta para automatizar procesos manuales.
 - Disponibilidad de expertos asesores en los diversos temas tratados en el proyecto.
 - Disponibilidad de relatos de denuncias que ya han sido transcritos y preparados para su análisis.
- Debilidades:
 - Equipamiento informático obsoleto con el tiempo.
 - Falta de infraestructura para soportar el escalamiento del sistema.
 - Falta de experiencia con las últimas técnicas empleadas para tareas de reconocimiento de entidades.
 - Existe un desbalance en los tipos de entidades que se encuentran en los relatos.
 - Debido a la sensibilidad de los datos manejados en las denuncias se restringe el número de personas que podría asistir en la tarea de anotación.

Análisis Externo

- Oportunidades:
 - Gran disponibilidad de herramientas de software libre para el desarrollo del sistema.
 - Existencia de herramientas con planes gratuitos para el despliegue del sistema.
 - Herramientas software bien documentadas.
 - Se cuenta con una gran fuente de relatos de denuncias digitalizados y muchos más por digitalizar.
 - Falta de personal en el MPF que pueda ser destinado a analizar todas las denuncias que ingresan por día.
 - Falta de un sistema en el MPF que aporte lo propuesto en este trabajo.
- Amenazas:
 - Las herramientas utilizadas se actualizan rápidamente aumentando el riesgo de incompatibilidades.
 - Inconstancia en la digitalización de ciertos relatos de denuncias necesarios para la actualización del sistema.
 - Cambios permanentes en el estado del arte de herramientas de procesamiento del lenguaje natural, pudiendo surgir competidores con mejores ofertas.
 - Desinterés o poco uso del sistema por parte de quienes lo implementen.
 - Incremento en la tasa de delitos que exija la capacidad del sistema.

3.9.1 Estrategias

Una vez identificados los cuatro aspectos FODA se definen las siguientes estrategias:

Estrategias FO: emplean las Fortalezas para maximizar las Oportunidades:

- Es posible hacer uso de herramientas de software libre y destinar el dinero ahorrado en licencias para la contratación de nuevos programadores o científicos de datos que permitan escalar el proyecto.
- Debido al interés del MPF en el proyecto y los buenos resultados que se esperan del sistema es posible cerrar un acuerdo para implementar el sistema o algunas de sus funcionalidades junto a los sistemas actuales de denuncias.
- Gracias a que se poseen relatos anotados suficientes para este proyecto y se tiene el conocimiento necesario para realizar nuevas anotaciones, es posible aprovechar el interés del MPF para que asignen personal que colabore en esta tarea de anotación sobre nuevos relatos.

Estrategias DO: minimizan las Debilidades aprovechando las Oportunidades:

- Aprovechar la gran cantidad de denuncias digitalizadas para buscar un lote de relatos que presente un mejor balanceo de entidades.
- Aprovechar las versiones gratuitas de las herramientas de despliegue para elegir aquella que provea mejor infraestructura en un comienzo, a modo de prueba, para luego pagar por mayores recursos en caso de ser necesario.
- Convencer a los encargados actuales de analizar las denuncias para que colaboren en la anotación de los relatos ya que están autorizados para leer su contenido y además serán beneficiarios del resultado obtenido.

Estrategias FA: emplean las Fortalezas para minimizar las Amenazas:

- Gracias al lote de relatos digitalizados con que se cuenta, es posible emplear técnicas de aumentación de datos para generar más relatos en caso que la disponibilidad de denuncias digitales decaiga.
- En caso de incompatibilidades de software debido al avance en las versiones de algunas librerías y al estancamiento de otras, es posible solventarlo con la gran comunidad que existe en torno a las herramientas libres utilizadas. Además, existen otras herramientas libres que podrían llegar a emplearse en una eventual necesidad de migrar a otra tecnología.
- Aprovechar la experiencia presente con las herramientas actuales de procesamiento del lenguaje natural, el acceso a personas expertas en el tema y la documentación online, buscando el aprendizaje constante sobre el estado del arte en materia de desarrollo, aprendizaje automático y procesamiento del lenguaje natural.
- Aplicar la experiencia propia y de asesores a fin de generar interés en los usuarios finales del sistema para que realicen el uso esperado del mismo.

Estrategias DA: minimizan las Debilidades evitando las Amenazas:

- Realizar seguimiento de la infraestructura del sistema a fin de identificar un eventual aumento en la demanda de recursos que deban adquirirse para mantener al sistema operando correctamente.
- Buscar rápidamente ganar experiencia y mantenerse informado sobre el estado del arte en materia de desarrollo, aprendizaje automático y procesamiento del lenguaje natural para poder actualizar el sistema con las nuevas tecnologías en caso de considerarse posible y necesario.
- Mantenerse al día con la digitalización y anotación de relatos que pueda realizar personal autorizado para evitar llegar al escenario donde no se disponga de la cantidad suficiente de los mismos y no se pueda recurrir a otros por la sensibilidad de los datos.
- Perseverar en la digitalización de las denuncias a fin de tener siempre, del total de relatos digitales, un lote con un número balanceado de cada tipo de entidad.

3.10 Plan de Proyecto

En esta sección se presentan las actividades necesarias para la realización del proyecto y su planificación. Tanto la Estructura de Desglose de Trabajo como la Lista de Actividades, Recursos Requeridos y Plan del Proyecto sustentan la sección anterior en donde se desarrolla la Propuesta.

3.10.1 Estructura de Desglose de Trabajo

Según La Guía del PMBOK (Project Management Institute, 2013) se define que, “la Estructura de Desglose del Trabajo (EDT) es una descomposición jerárquica, orientada al producto entregable del trabajo que será ejecutado por el equipo del proyecto, para lograr los objetivos del proyecto y crear los productos entregables requeridos”

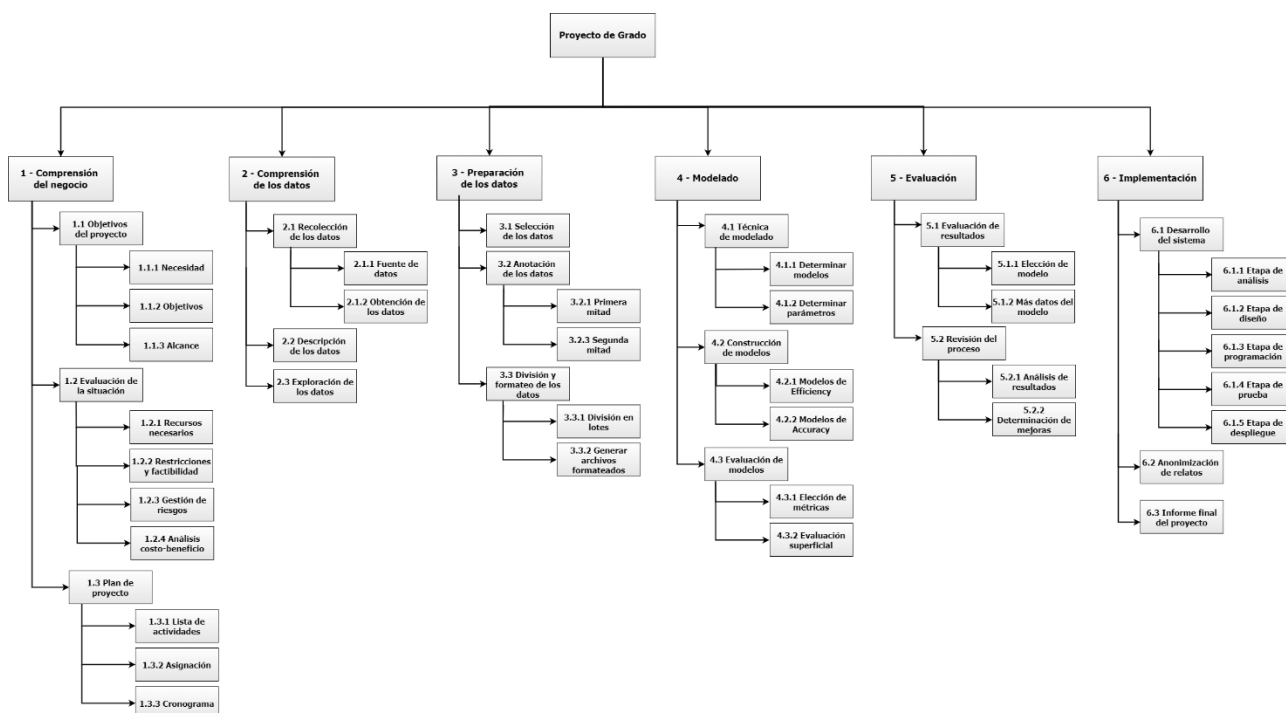


Imagen 51: Estructura de Desglose del Trabajo para el proyecto

Para este proyecto la EDT se definió a partir de cada una de las fases de la metodología CRISP-DM, como se indica en la Figura 51. Además, se definió su correspondiente diccionario.

Diccionario de la EDT.

1. Comprensión del negocio

1.1 Objetivos del proyecto. Se determina el objetivo general del proyecto y sus correspondientes objetivos específicos. También se define el alcance del proyecto y los criterios de éxito del mismo. Todo esto, partiendo previamente, de la identificación de un problema o necesidad. Las subtarefas serían:

- 1.1.1 *Necesidad*: se identifica y formaliza el problema o necesidad.
- 1.1.2 *Objetivos*: se definen los objetivos del proyecto.
- 1.1.3 *Alcance*: se define el alcance del proyecto, lo que se pretende realizar y lo que no.

1.2 *Evaluación de la situación*. Se identifican los recursos necesarios, las restricciones que puedan afectar al proyecto, suposiciones que deban tenerse en cuenta y la factibilidad del proyecto. Riesgos y contingencias. Costos y beneficio. Las subtarefas serían:

- 1.2.1 *Recursos necesarios*: se identifican los recursos de hardware, software y humanos (capital humano) necesarios.
- 1.2.2 *Restricciones y factibilidad*: se determinan las restricciones del proyecto y las factibilidades técnica, operativa y legal.
- 1.2.3 *Gestión de riesgos*: se identifican los riesgos, se les asigna un valor de impacto y se proponen medidas para mitigarlos.
- 1.2.4 *Análisis costo-beneficio*: se estiman los costos del proyecto, se cuantifican los beneficios y se define el tiempo estimado de recupero de la inversión.

1.3 *Plan de proyecto*. Se elabora el plan del proyecto con sus correspondientes actividades, más los tiempos y recursos que éstas requieran. Las subtarefas serían:

- 1.3.1 *Lista de actividades*: se define la lista de actividades que se deberán realizar.
- 1.3.2 *Asignación*: se asignan los distintos tipos de recursos a las actividades identificadas.
- 1.3.3 *Cronograma*: se realiza el cronograma del proyecto (Diagrama de Gantt).

2. *Comprensión de los datos*

2.1 *Recolección de los datos*. Se recolectan los textos disponibles, relacionados con la temática objetivo del proyecto, potencialmente útiles para el entrenamiento, validación o prueba del modelo a construir. Las subtarefas serían:

- 2.1.1 *Fuentes de datos*: se determina los posibles lugares de donde obtener los datos y la manera de hacerlo.
- 2.1.2 *Obtención de datos*: se realizan los acuerdos necesarios y se aplican las técnicas adecuadas para obtener los datos.

2.2 *Descripción de los datos*. Se examinan el contexto al cual pertenecen los textos, buscando identificar propiedades comunes a todos ellos y sus principales atributos incluyendo: formato y tipos.

2.3 *Exploración de los datos*. Se realiza un análisis sobre los datos encontrados a fin de conocerlos a profundidad, evaluando diferentes propiedades de los mismos para determinar la cantidad, valores que toman y su calidad.

3. Preparación de los datos

3.1 *Selección de los datos.* Se definen cuáles serán los textos con los cuales se trabajará y aquellos con los que no. Esta decisión debe fundamentarse en los objetivos, restricciones técnicas, calidad de los datos o tipo de los datos identificados.

3.2 *Anotación de los datos.* Sobre los textos seleccionados se inicia el proceso de anotación consistente en identificar las entidades planteadas en los objetivos del proyecto y generar los archivos que contengan esa información. Las subtarefas serían:

3.2.1 *Primera mitad:* se anota la primera mitad del total de textos que se pretende anotar, obteniendo los correspondientes archivos .ann.

3.2.2 *Segunda mitad:* se anota la segunda mitad del total de textos que se pretende anotar, obteniendo los correspondientes archivos .ann.

3.3 *División y formateo de los datos.* Los textos anotados se dividen en lotes para las tareas de entrenamiento, validación y prueba de la fase de modelado. Se modifica el formato de los archivos que contienen la información resultante del proceso de anotación a fin de prepararlos para la fase de modelado. Las subtarefas serían:

3.3.1 *División en lotes:* se divide el total de textos anotados en conjunto de entrenamiento, validación y prueba.

3.3.2 *Generar archivos formateados:* se utilizan los lotes de textos para generar tres archivos .json y dos archivos .spacy a ser utilizados durante las fases de modelado y evaluación.

4. Modelado

4.1 *Técnica de modelado.* Se define la técnica de modelado a utilizar: cuantos datos y qué datos se utilizarán, el criterio de selección de datos y los pasos a seguir para entrenar los modelos. Las subtarefas serían:

4.1.1 *Determinar modelos:* se define la cantidad de modelos que se van a entrenar, validar y probar, además de los parámetros del widget de spaCy para cada uno de ellos.

4.1.2 *Determinar parámetros:* se define el resto de parámetros del archivo config.cfg y la cantidad de textos con los cuales se entrenarán y validarán los modelos.

4.2 *Construcción de modelos.* Se entrenan y validan los distintos modelos definidos en el punto anterior. Las subtarefas serían:

4.2.1 *Modelos de Efficiency:* se entrenan y validan los modelos contruidos con el widget de Efficiency.

4.2.2 *Modelos de Accuracy:* se entrenan y validan los modelos contruidos con el widget de Accuracy.

4.3 *Evaluación de modelos.* Se evalúan los modelos entrenados con el lote de prueba preparado y las métricas escogidas. Las subtarefas serían:

- 4.3.1 *Elección de métricas:* se definen las métricas de evaluación de que se utilizaran para puntuar los modelos.
- 4.3.2 *Evaluación superficial:* se revisan los puntajes obtenidos para cada modelo, resumiendo los resultados en una tabla y generando matrices de confusión para aquellos con mayor puntaje haciendo uso de los datos de prueba.

5. Evaluación

5.1 *Evaluación de resultados.* Se evalúa el cumplimiento de los objetivos definidos en función del desempeño del modelo en los casos de prueba. Además, en caso que el modelo sea deficiente, se evalúa la existencia de alguna razón de negocio que provoque esto. Las subtarefas serían:

- 5.1.1 *Elección de modelo:* se determina el mejor modelo en base a las métricas consideradas.
- 5.1.2 *Más datos del modelo:* se busca obtener más información sobre el mejor modelo obtenido calculando las métricas elegidas, pero para cada tipo de entidad.

5.2 *Revisión del proceso.* Se revisa como fue el desempeño del modelo y se identifican aspectos a mejorar. Las subtarefas serían:

- 5.2.1 *Análisis de resultados:* se profundiza el análisis sobre los modelos que demostraron un mejor desempeño (aquellos con matrices de confusión).
- 5.2.2 *Determinación de mejoras:* se identifican posibles puntos de mejora a tener en cuenta para modelos más robustos y puntajes más altos.

6. Implementación

6.1 *Desarrollo del sistema.* Comprende el diseño y programación del sistema que hará uso del modelo entrenado a fin de permitir a los usuarios utilizarlo como una interfaz gráfica para trabajar con dicho modelo.

- 6.1.1 *Etapas de análisis:* se determinan las funcionalidades mínimas con las cuales contará el sistema.
- 6.1.2 *Etapas de diseño:* se define las secciones con las que contará el sistema y la estructura de la base de datos, además del flujo de datos entre el cliente, servidor y base de datos.
- 6.1.3 *Etapas de programación:* se desarrolla el sistema con las tecnologías elegidas.
- 6.1.4 *Etapas de prueba:* se cargan datos al sistema y se prueban las funcionalidades en búsqueda de errores y mejoras.
- 6.1.5 *Etapas de despliegue:* se despliega el sistema en plataformas escogidas para tal propósito.

6.2 *Anonimización de relatos.* Refiere a la tarea de reemplazar los datos sensibles de un lote de relatos por datos ficticios. Dicho lote se emplea para realizar la prueba final del sistema y publicar resultados.

6.3 *Informe final del proyecto.* Se presentan los resultados finales de todo el proyecto.

3.10.2 Lista de Actividades

- **Fase 1: Comprensión del negocio**
 - Definir los objetivos del proyecto.
 - Identificar y formalizar el problema o necesidad.
 - Definir el objetivo general y los específicos.
 - Definir el alcance.
 - Evaluar la situación actual.
 - Determinar los recursos necesarios.
 - Determinar las restricciones y factibilidad del proyecto.
 - Gestionar los riesgos del proyecto.
 - Realizar el análisis costo-beneficio.
 - Desarrollar el plan del proyecto.
 - Realizar la lista de actividades.
 - Asignar los recursos a las actividades.
 - Realizar el cronograma del proyecto.
- **Fase 2: Comprensión de los datos**
 - Recolectar los datos.
 - Determinar las fuentes de datos.
 - Obtener los datos.
 - Describir los datos.
 - Explorar los datos.
- **Fase 3: Preparación de los datos**
 - Seleccionar los datos.
 - Anotar los datos.
 - Anotar la primera mitad.
 - Anotar la segunda mitad.
 - Formatear y dividir los datos.
 - Dividir los datos en lotes.
 - Generar archivos formateados.
- **Fase 4: Modelado**
 - Definir técnica de modelado
 - Determinar modelos a estudiar.
 - Determinar parámetros de los modelos.
 - Construir los modelos
 - Entrenar y validar modelos de Efficiency.
 - Entrenar y validar modelos de Accuracy.
 - Evaluar los modelos
 - Seleccionar métricas de evaluación.
 - Realizar primera evaluación superficial.

- **Fase 5: Evaluación**
 - Evaluar los resultados.
 - Elegir el mejor modelo.
 - Obtener más datos del mejor modelo.
 - Revisar el proceso.
 - Analizar los resultados de los mejores modelos.
 - Determinar posibles mejoras.
- **Fase 6: Implementación**
 - Desarrollar el sistema
 - Realizar etapa de análisis.
 - Realizar etapa de diseño.
 - Realizar etapa de programación.
 - Realizar etapa de prueba.
 - Realizar etapa de despliegue.
 - Anonimizar los relatos
 - Desarrollar informe final del proyecto

3.10.3 Recursos Requeridos

Para el desarrollo de las actividades del proyecto se requerirán los recursos humanos indicados en la tabla 7. Debido a que los recursos indicados para una tarea, son los mismos que se requieren para sus respectivas subtareas, se optó por no incluir las subtareas en la tabla y simplificar la visualización de la información.

Actividades	Recurso	Cantidad
Fase 1: Comprensión del negocio		
Definir los objetivos del proyecto.	Líder del proyecto	1
Evaluar la situación actual.	Líder del proyecto	1
Desarrollar el plan del proyecto.	Líder del proyecto	1
Fase 2: Comprensión de los datos		
Recolectar los datos.	Científico de datos	1
Describir los datos.	Científico de datos	1
Explorar los datos.	Científico de datos	1
Fase 3: Preparación de los datos		
Seleccionar los datos	Científico de datos	1
Anotar los datos	Científico de datos y Líder del Proyecto	1

Formatear y dividir los datos	Científico de datos	1
Fase 4: Modelado		
Definir técnica de modelado	Científico de datos	1
Construir los modelos	Científico de datos	1
Evaluar los modelos	Científico de datos	1
Fase 5: Evaluación		
Evaluar los resultados	Científico de datos y Líder del proyecto	1
Revisar proceso	Científico de datos y Líder del proyecto	1
Fase 6: Implementación		
Desarrollar el sistema	Programador	1
Anonimizar los relatos	Científico de datos y Líder del Proyecto	1
Desarrollar informe final del proyecto	Científico de datos y Líder del Proyecto	1

Tabla 7: Recursos Humanos requeridos para el proyecto

Al definir el perfil encargado de las tareas: “Anotar los datos” y “Anonimizar los relatos” existen distintas alternativas:

- Puede realizarlo un futuro usuario del sistema, quien es conocedor del contexto y sabrá mejor cuál palabra es una entidad de determinado tipo y cuál no lo es. Para esto el superior de este usuario debería autorizarle que dedique parte de su tiempo a realizar estas tareas o podría pagársele un extra para que lo haga en horario no laboral.
- También puede realizarlo una persona que se contrate exclusivamente para esta tarea. En diversas tareas de anotación, tanto en el campo del PLN como en Visión Computacional se suelen contratar anotadores para este tipo de tareas.
- La tercera alternativa y la que se presenta en la tabla anterior consiste en que el mismo Científico de Datos realice la tarea. Debido a que en este proyecto existe además un Líder de Proyecto, sería este último quien se relaciona con los usuarios finales y manejaría información del contexto, motivo por el cual debería estar presente, aunque sólo sea supervisando, en las tareas de anotación y anonimización.

Los recursos hardware requeridos se corresponden con las computadoras utilizadas por el científico de datos y el líder del proyecto, además del servidor empleado para configurar el entorno de ejecución de la plataforma web. Dichos recursos se encuentran disponibles para iniciar el proyecto y presentan las siguientes características:

PC del científico de datos:

- Laptop HP Pavilion 15-CB001LA.
- Procesador: I5 7ma.
- RAM: 12 GB.
- Almacenamiento:
 - 1TB HDD.
 - 480GB NVME SSD
- Gráfica: NVIDIA GeForce GTX 1050
- Pantalla: 15.6’’.

PC del líder del proyecto:

- Laptop HP Pavilion 15-bs022la
- Procesador: Intel Core I7 7ma.
- RAM: 12 GB
- Almacenamiento: 1TB HDD
- Pantalla: 15.6’’.

En cuanto al software, se utilizarán los especificados en la sección 3.4 sobre herramientas utilizadas.

3.10.4 Cronograma del Proyecto

A partir de la lista de actividades de la sección 3.10.2 y de los recursos humanos de la sección 3.10.3 se realizó el cronograma dividido en las imágenes 52 y 53. Allí podemos observar la lista de tareas organizadas en las 6 fases de la metodología empleada y el diagrama de Gantt. En la parte del diagrama, al lado de cada tarea se muestran las iniciales de los recursos humanos empleados, correspondiendo:

- LP con Líder del Proyecto
- CD con Científico de Datos
- P con Programador

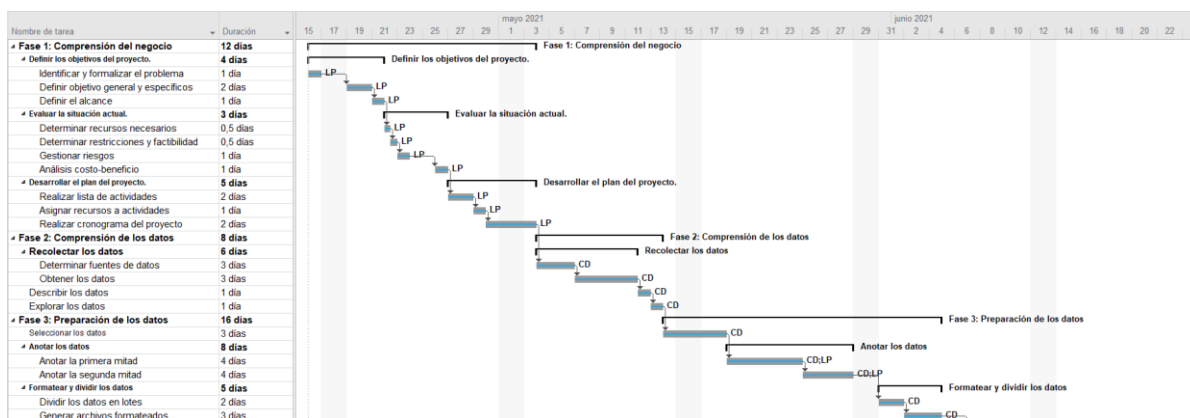


Imagen 52: Listado de tareas con sus duraciones y diagrama de Gantt – Fases 1, 2 y 3

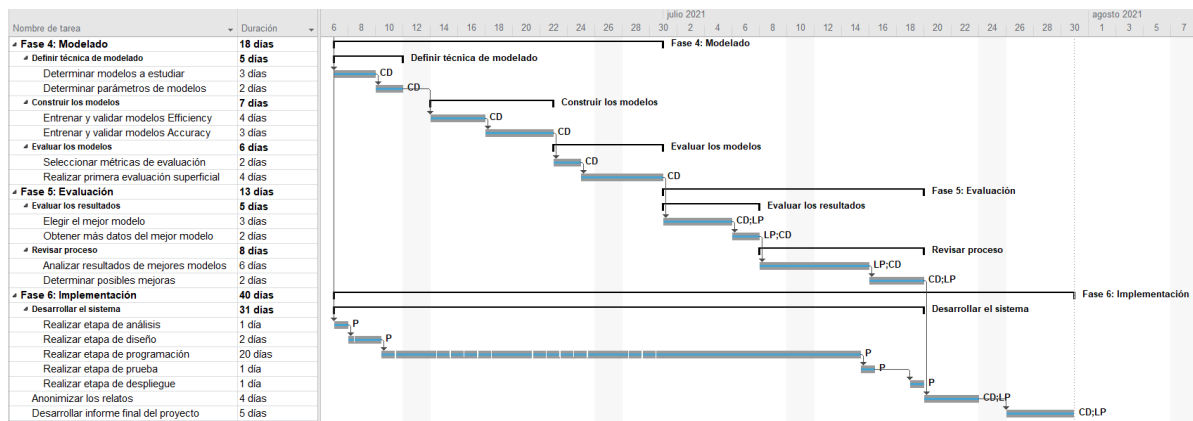


Imagen 53: Listado de tareas con sus duraciones y diagrama de Gantt – Fases 4, 5 y 6

Si bien en el diagrama de Gantt gran parte de las tareas se presentan de manera secuencial, en la práctica fue necesario regresar a pasos anteriores en algunos momentos para realizar correcciones y probar distintas alternativas. Esto queda reflejado en la imagen de CRISP-DM (sección 2.2.1) donde se observa la naturaleza cíclica de la metodología.

En las imágenes 54 y 55 podemos ver con mayor detalle la lista de tareas, su relación de dependencia y recursos asignados a las mismas.

	Nombre de tarea	Duración	Modo de tarea	Comienzo	Fin	EDT	Predecesoras	Nombres de los recursos	% completado
1	Fase 1: Comprensión del negocio	12 días		vie 16/4/21	lun 3/5/21	1			100%
2	Definir los objetivos del proyecto.	4 días		vie 16/4/21	mié 21/4/21	1.1			100%
3	Identificar y formalizar el problema	1 día		vie 16/4/21	vie 16/4/21	1.1.1		Lider del Proyecto	100%
4	Definir objetivo general y específicos	2 días		lun 19/4/21	mar 20/4/21	1.1.2	3	Lider del Proyecto	100%
5	Definir el alcance	1 día		mié 21/4/21	mié 21/4/21	1.1.3	4	Lider del Proyecto	100%
6	Evaluar la situación actual.	3 días		jue 22/4/21	lun 26/4/21	1.2			100%
7	Determinar recursos necesarios	0,5 días		jue 22/4/21	jue 22/4/21	1.2.1	5	Lider del Proyecto	100%
8	Determinar restricciones y factibilidad	0,5 días		jue 22/4/21	jue 22/4/21	1.2.2	7	Lider del Proyecto	100%
9	Gestionar riesgos	1 día		vie 23/4/21	vie 23/4/21	1.2.3	8	Lider del Proyecto	100%
10	Análisis costo-beneficio	1 día		lun 26/4/21	lun 26/4/21	1.2.4	9	Lider del Proyecto	100%
11	Desarrollar el plan del proyecto.	5 días		mar 27/4/21	lun 3/5/21	1.3			100%
12	Realizar lista de actividades	2 días		mar 27/4/21	mié 28/4/21	1.3.1	10	Lider del Proyecto	100%
13	Asignar recursos a actividades	1 día		jue 29/4/21	jue 29/4/21	1.3.2	12	Lider del Proyecto	100%
14	Realizar cronograma del proyecto	2 días		vie 30/4/21	lun 3/5/21	1.3.3	13	Lider del Proyecto	100%
15	Fase 2: Comprensión de los datos	8 días		jue 4/5/21	jue 13/5/21	2			100%
16	Recolectar los datos	6 días		mar 4/5/21	mar 11/5/21	2.1			100%
17	Determinar fuentes de datos	3 días		mar 4/5/21	jue 6/5/21	2.1.1	14	Científico de Datos	100%
18	Obtener los datos	3 días		vie 7/5/21	mar 11/5/21	2.1.2	17	Científico de Datos	100%
19	Describir los datos	1 día		mié 12/5/21	mié 12/5/21	2.2	18	Científico de Datos	100%
20	Explorar los datos	1 día		jue 13/5/21	jue 13/5/21	2.3	19	Científico de Datos	100%
21	Fase 3: Preparación de los datos	16 días		vie 14/5/21	vie 4/6/21	3			100%
22	Seleccionar los datos	3 días		vie 14/5/21	mar 18/5/21	3.1	20	Científico de Datos	100%
23	Anotar los datos	8 días		mié 19/5/21	mar 28/5/21	3.2			100%
24	Anotar la primera mitad	4 días		mié 19/5/21	lun 24/5/21	3.2.1	22	Científico de Datos; Lider del Proyecto	100%
25	Anotar la segunda mitad	4 días		mar 25/5/21	jue 28/5/21	3.2.2	24	Científico de Datos; Lider del Proyecto	100%
26	Formatear y dividir los datos	5 días		lun 31/5/21	vie 4/6/21	3.3			100%
27	Dividir los datos en lotes	2 días		lun 31/5/21	mar 1/6/21	3.3.1	25	Científico de Datos	100%
28	Generar archivos formateados	3 días		mié 2/6/21	vie 4/6/21	3.3.2	27	Científico de Datos	100%

Imagen 54: Listado de tareas con sus duraciones, fechas, relaciones y recursos – Fases 1, 2 y 3

	Nombre de tarea	Duración	Modo de tarea	Comienzo	Fin	EDT	Predecesoras	Nombres de los recursos	% completado
29	Fase 4: Modelado	18 días		lun 7/6/21	mié 30/6/21	4			100%
30	Definir técnica de modelado	5 días		lun 7/6/21	vie 11/6/21	4.1			100%
31	Determinar modelos a estudiar	3 días		lun 7/6/21	mié 9/6/21	4.1.1	28	Científico de Datos	100%
32	Determinar parámetros de modelos	2 días		jue 10/6/21	vie 11/6/21	4.1.2	31	Científico de Datos	100%
33	Construir los modelos	7 días		lun 14/6/21	mar 22/6/21	4.2			100%
34	Entrenar y validar modelos Efficiency	4 días		lun 14/6/21	jue 17/6/21	4.2.1	32	Científico de Datos	100%
35	Entrenar y validar modelos Accuracy	3 días		vie 18/6/21	mar 22/6/21	4.2.2	34	Científico de Datos	100%
36	Evaluar los modelos	6 días		mié 23/6/21	mié 30/6/21	4.3			100%
37	Seleccionar métricas de evaluación	2 días		mié 23/6/21	jue 24/6/21	4.3.1	35	Científico de Datos	100%
38	Realizar primera evaluación superficial	4 días		vie 25/6/21	mié 30/6/21	4.3.2	37	Científico de Datos	100%
39	Fase 5: Evaluación	13 días		jue 1/7/21	lun 19/7/21	5			100%
40	Evaluar los resultados	5 días		jue 1/7/21	mié 7/7/21	5.1			100%
41	Elegir el mejor modelo	3 días		jue 1/7/21	lun 5/7/21	5.1.1	38	Científico de Datos; Lider del Proyecto	100%
42	Obtener más datos del mejor modelo	2 días		mar 6/7/21	mié 7/7/21	5.1.2	41	Lider del Proyecto; Científico de Datos	100%
43	Revisar proceso	8 días		jue 8/7/21	lun 19/7/21	5.2			100%
44	Analizar resultados de mejores modelos	6 días		jue 8/7/21	jue 15/7/21	5.2.1	42	Lider del Proyecto; Científico de Datos	100%
45	Determinar posibles mejoras	2 días		vie 16/7/21	lun 19/7/21	5.2.2	44	Científico de Datos; Lider del Proyecto	100%
46	Fase 6: Implementación	40 días		lun 7/6/21	vie 30/7/21	6			100%
47	Desarrollar el sistema	31 días		lun 7/6/21	lun 19/7/21	6.1			100%
48	Realizar etapa de análisis	1 día		lun 7/6/21	lun 7/6/21	6.1.1	28	Programador	100%
49	Realizar etapa de diseño	2 días		mar 8/6/21	jue 10/6/21	6.1.2	48	Programador	100%
50	Realizar etapa de programación	20 días		jue 10/6/21	jue 15/7/21	6.1.3	49	Programador	100%
51	Realizar etapa de prueba	1 día		jue 15/7/21	vie 16/7/21	6.1.4	50	Programador	100%
52	Realizar etapa de despliegue	1 día		lun 19/7/21	lun 19/7/21	6.1.5	51	Programador	100%
53	Anonimizar los relatos	4 días		mar 20/7/21	vie 23/7/21	6.2	45	Científico de Datos; Lider del Proyecto	100%
54	Desarrollar informe final del proyecto	5 días		lun 26/7/21	vie 30/7/21	6.3	53	Científico de Datos; Lider del Proyecto	100%

Imagen 55: Listado de tareas con sus duraciones, fechas, relaciones y recursos – Fases 1, 2 y 3

3.11 Gestión de Costos y Adquisiciones

La gestión de costos y adquisiciones comprende los costos relacionados a los recursos humanos intervinientes en el proyecto, costos de hardware y equipamiento, costos de software, y otros gastos varios que sean necesarios.

3.11.1 Costos de Recursos Humanos

La estimación de costos de recursos humanos se encuentra en la Tabla 8. En la misma se detalla a qué recurso se hace referencia, así como las etapas en las cuales se necesita. La columna Cantidad indica el número de horas necesarias para completar una actividad, este número se calculó considerando una jornada laboral de 6 horas diarias. El costo unitario por hora de cada recurso se estimó basado en la tabla de honorarios indicativos del Consejo Profesional de Agrimensores, Ingenieros y Profesiones Afines (COPAIPA, 2021). Los mismos corresponden a los valores consultados en fecha 14/08/2021.

Debido a que en la página de COPAIPA no figura honorario para el científico de datos o, en su defecto, analista de datos, se optó por asignarle el mismo honorario que a un analista funcional. De este modo sería superior al del programador, pero inferior al del líder de proyecto. Esto último debido a que, al momento de realizarse este proyecto, en general suele pagarse mejor a quien desempeña tareas de análisis de datos que a un programador web.

Actividad	Recurso	Cantidad [hs]	Costo Unitario [\$]	Costo Total [\$]
Comprensión del negocio				
Definir los objetivos del proyecto.	Líder del proyecto	24	1080	25.920
Evaluar la situación actual.	Líder del proyecto	18	1080	19.440
Desarrollar el plan del proyecto.	Líder del proyecto	30	1080	32.400
Comprensión de los datos				
Recolectar los datos.	Científico de datos	36	720	25.920
Describir los datos.	Científico de datos	6	720	4.320
Explorar los datos.	Científico de datos	6	720	4.320
Preparación de los datos				
Seleccionar los datos	Científico de datos	18	720	12.960
Anotar los datos	Científico de datos	36	720	25.920
	Líder del Proyecto	12	1080	12.960

Formatear y dividir los datos	Científico de datos	30	720	21.600
Modelado				
Definir técnica de modelado	Científico de datos	30	720	21.600
Construir los modelos	Científico de datos	42	720	30.240
Evaluar los modelos	Científico de datos	36	720	25.920
Evaluación				
Evaluar los resultados	Científico de datos	18	720	12.960
	Líder del proyecto	12	1080	12.960
Revisión el proyecto	Científico de datos	30	720	21.600
	Líder del proyecto	18	1080	19.440
Implementación				
Desarrollar el sistema	Programador	150	600	90.000
Anonimizar los relatos	Científico de datos	24	720	17.280
	Líder del proyecto	6	1080	6.480
Desarrollar informe final del proyecto	Científico de datos	12	720	8.640
	Líder del Proyecto	18	1080	19.440
TOTAL				471.960

Tabla 8: Costo estimado de Recursos Humanos

3.11.2 Costos de Hardware

Los costos de hardware hacen referencia al equipamiento necesario para desarrollar el proyecto. Debido a que dicho equipamiento es el disponible al inicio del proyecto y no se considera necesaria ninguna otra adquisición, no se incurrieron en costos de hardware. De todos modos, se indica en la tabla 9 el costo estimado, en fecha 12/08/2021, del hardware empleado. Las referencias a pie de página dirigen a publicaciones de Mercado Libre de productos similares a los utilizados debido a que no fue posible encontrar exactamente los mismos al momento de redactarse este texto.

Actividad	Recurso	Costo Unitario [\$]	Costo Total [\$]
Laptop HP Pavilion 15-CB001la. Procesador: Intel Core I5 7ma. RAM: 12GB. Almacenamiento: 1TB HDD. 480GB NVME SSD Gráfica: NVIDIA GeForce GTX 1050 Pantalla: 15.6’’	1	120.000 ³²	120.000
Laptop HP Pavilion 15-bs022la Procesador: Intel Core I7 7ma. RAM: 12 GB Almacenamiento: 1TB HDD Pantalla: 15.6’’	1	118.000 ³³	118.000
TOTAL			238.000

Tabla 9: Costo estimado de Hardware

3.11.3 Costos de Software

Los costos de software hacen referencia a licencias y permisos de utilización de las herramientas que se necesitarán en el proyecto.

Todas las herramientas software empleadas en el proyecto son gratuitas. Además, los laptops empleadas con sistema operativo Windows, cuentan con la licencia correspondiente de Windows 10 Home. Debido a esto, no se incurre en costos de software.

3.11.4 Línea Base de Costos

Finalmente, en la Tabla 10 se indican los costos del proyecto.

³² Valor tomado de MercadoLibre en fecha: 12/08/2021. https://articulo.mercadolibre.com.ar/MLA-878253376-notebook-hp-gaming-pavilio-amd-ryzen-5-8gb-256gb-gtx-1050-JM?searchVariation=63846730788#searchVariation=63846730788&position=7&search_layout=stack&type=item&tracking_id=f2900092-f688-435e-8264-6ebe24968a1f

³³ Valor tomado de MercadoLibre en fecha: 12/08/2021. https://articulo.mercadolibre.com.ar/MLA-928077233-notebook-hp-pavilion-i7-11va-quad-8gb-ssd256-optane-156-hd-JM?searchVariation=89804411983#searchVariation=89804411983&position=7&search_layout=stack&type=item&tracking_id=5fdc2ae6-9d79-4b17-8f8d-25545396af6c

Concepto	Costo [\$]
Costo de Recursos Humanos	471.960
Costo de Hardware	238.000
Costo de Software	0
TOTAL	709.960

Tabla 10: Costos del Proyecto en pesos argentinos

El costo de cada uno de los recursos expresados en pesos argentinos puede variar con el avance del proyecto debido al crecimiento inflacionario de Argentina, lo que llevaría a la pérdida de beneficios económicos. A fin de mitigar este riesgo, el costo total del proyecto se presentará en dólares, teniendo en cuenta la cotización en fecha 14/08/2021, con un valor por dólar de \$96.00³⁴. Los costos en dólares pueden observarse en la Tabla 11.

Concepto	Costo [USD]
Costo de Recursos Humanos	4.916,25
Costo de Hardware	2.479,17
Costo de Software	0
TOTAL	7.395,42

Tabla 11: Costos del Proyecto en dólares

3.11.4.1 Análisis Costo-Beneficio

Este sistema brinda importantes beneficios de tiempo, dinero y calidad del trabajo, que justifican el costo de su implementación. Estos puntos se desarrollan a continuación.

1. *Longitud de relatos*: en la sección 3.7.2.3, cuando se realizó la exploración de los datos, se mencionó que aproximadamente el 80% de los relatos tiene una longitud entre 400 y 1800 caracteres, es decir, que en promedio dicha longitud es de 1100 caracteres.
2. *Velocidad estimada de lectura*: existe un estudio titulado “¿Cuántas palabras leemos por minuto? Una revisión y meta-análisis de la velocidad de lectura” (Brysbaert, 2019), realizado en el departamento de Psicología Experimental de la Universidad Ghent, Bélgica. En él se encuentran los resultados de seis estudios que miden la velocidad de lectura silenciosa y en español, que una persona adulta tiene en promedio. Dicha velocidad es de 278 PPM (Palabras Por Minuto). Análogamente, la velocidad de lectura en voz alta es de 191 palabras por minuto. Este estudio también indica que la velocidad de lectura en textos de no ficción, es menor que en textos de ficción. Por lo tanto, para

³⁴ Extraído de la tabla “Cotización billetes” de <https://www.bna.com.ar/Personas> en fecha: 12/08/2021

textos de no ficción, el promedio sería menor a 278 PPM. Si, además, la lectura es en voz alta sería aún menor.

3. *Tiempo estimado ahorrado por relato*: con los puntos anteriores es posible calcular el tiempo promedio, que un adulto requiere para leer un relato de denuncia. Con 1100 palabras por relato y una velocidad de lectura de 278 PPM, obtenemos un tiempo de lectura de 3,96 minutos por relato. Si consideramos que la lectura no solo implica leer comprensivamente, sino que además requiere identificar las entidades involucradas (tarea que el sistema automatiza), podríamos redondear los 3,96 minutos a 4 minutos.
4. *Tiempo estimado ahorrado por día*: el sistema de denuncias y sistema de denuncias web, ambos pertenecientes al Ministerio Público Fiscal de Salta, registran entre los dos un promedio de 350 denuncias por día. Debido a la gran cantidad de relatos, actualmente no se realiza un análisis de todos ellos, pero sería lo ideal. De este modo, si en promedio se requieren 4 minutos para leer el relato e identificar sus entidades, y si se registran 350 relatos por día, podemos aproximar el tiempo que el sistema ahorraría a 140 minutos diarios.
5. *Dinero estimado ahorrado por día*: la escala salarial del Ministerio Público sitúa el sueldo de un personal administrativo, a mediados del 2021, en un monto promedio aproximado de \$65.000. Tomando los 22 días hábiles de un mes, serían alrededor de \$3.000 por día (6 horas), es decir, \$500 por hora. El equivalente a los 140 minutos, mencionados en el punto 4, sería alrededor de \$1.165. Éste último sería el dinero estimado ahorrado por día.

Tras este análisis observamos tres tipos de beneficios:

- Beneficio en tiempo: se ahorrarían alrededor de 140 minutos diarios, lo equivalente a casi un 40% de la jornada laboral (suponiendo que una sola persona se encargue de este trabajo). Además, si la persona deseara obtener estadísticas de las entidades encontradas, el ahorro en tiempo sería aún mayor, pudiendo con pocos clicks consultar gráficos que resuman dichos datos. Este tiempo ahorrado podría emplearse para realizar otras tareas.
- Beneficio en dinero: con un ahorro aproximado de \$1.165 por día, estaríamos hablando de alrededor de \$25.630 mensuales y \$307.560 al año.
 - Recupero de la inversión: Tomando la cotización de 1 dólar a \$96 (sección 3.11.4), serían 266,98 dólares mensuales o 3.203,75 dólares al año. El costo estimado del proyecto fue de 7.395,42 dólares.
 - Al consultar el diagrama del proyecto en la sección 3.10.4, vemos que el tiempo total estimado fue de 107 días (aproximadamente 4 meses).
 - Para iniciar el proyecto se necesita el hardware especificado como “Costos de Hardware” y el dinero destinado mensualmente al recurso humano.
 - En la tabla 12 se observa un Flujo de Fondos del proyecto. Allí vemos que en los 4 meses que dura el proyecto, no se aprecian beneficios, sino que estos se observan a partir del quinto mes, es decir, durante los 8 meses restantes del año.
 - El recupero de la inversión se daría a los 2,31 años, es decir, a los 28 meses, de implementado el proyecto. Como el flujo de fondos se expresó en años, el recupero se daría en el tercer año.

FLUJO DE CAJA / FONDOS			
	INGRESOS (USD)	EGRESOS (USD)	SALDO (USD)
Primer año	8 meses de ahorro en salarios: 2135,94	Costos de Hardware: 2.479,17 Costo de Recursos Humanos: 4.916,25	- 5.259,48
Segundo año	Ahorro en salarios: 3.203,75		- 2.055,73
Tercer año	Ahorro en salarios: 3.203,75		1.148,02

Tabla 12: Flujo de fondos del proyecto

- Beneficio en la calidad del trabajo: el analista puede consultar las entidades específicas de una denuncia, en la pestaña entidades, puede consultar el número total de entidades de un determinado tipo, en la página de visualizaciones, o puede ver cuáles son dichas entidades en la misma página de visualizaciones. Es decir, puede consultar mayor cantidad de información, en menos tiempo, consiguiendo así análisis más eficientes. Además, considerando que actualmente no se analizan todas las denuncias que ingresan por día, al tener un sistema que analiza las denuncias automáticamente, sería posible atender el caudal de 350 denuncias diarias.

3.12 Gestión de Riesgos

Todo proyecto lleva asociado un riesgo, un problema potencial que puede ocurrir o no. En la tabla 12, se mencionan y analizan los riesgos identificados en este proyecto, junto con las acciones propuestas para mitigarlos.

La probabilidad de los riesgos se define en el rango [0,1]. El impacto de los mismos se define en una escala del 1 al 3, siendo 1 impacto bajo y 3 impacto alto. El producto de la probabilidad y el impacto da como resultado la valoración del riesgo. Por último, se define una acción para mitigarlo, es decir, reducir su probabilidad y/o impacto.

Riesgo	Probabilidad	Impacto	Valoración	Acción para mitigar
Acceso mal intencionado	0.4	2	0.8	- Agregar al sistema la funcionalidad de autenticación: usuario, clave y algún captcha para evitar bots.
Infraestructura insuficiente	0.3	2	0.6	- Contratar más espacio de almacenamiento, memoria RAM y núcleos virtuales del servicio de hosting. - Optimizar el código del sistema para que gestione mejor la concurrencia de usuarios y carga masiva de datos.

Resultados de entrenamiento deficientes.	0.3	2	0.6	- Asegurarse que los relatos sean variados en su contenido sin cantidades predominantes de un mismo tipo de entidad para evitar sesgos. - Reentrenar el modelo variando parámetros. - Ampliar el plazo de entrenamiento.
Fallos en los equipos.	0.2	3	0.6	- Realizar respaldos con los avances del proyecto. - Considerar el uso de herramientas gratuitas para el hosting del sistema. - De ser posible, contar con un segundo equipo de contingencia.
Restricciones en el uso de información.	0.5	1	0.5	- Definir textos de pruebas para validar el modelo, cuyas entidades hayan sido anonimizadas.
Inflación.	0.7	1	0.7	- Adquirir los recursos necesarios con la mayor anticipación posible.
Fin de gratuidad de alguna herramienta.	0.1	1	0.1	- Trabajar con la última versión gratuita. - Identificar alternativas gratuitas. - Destinar una cantidad de dinero para reserva que permita mitigar el impacto de este riesgo en caso de materializarse.

Tabla 13: Análisis de riesgos del Proyecto

3.13 Análisis de Factibilidad

Factibilidad Técnica

El proyecto es técnicamente factible de realizarse debido a la disponibilidad del hardware necesario y a la gratuidad de las herramientas software utilizadas. No se requiere de un equipo avanzado para llevar a cabo las diversas tareas en este proyecto, aunque para reducir los tiempos de espera en las fases de entrenamiento y evaluación toda mejora en el hardware es bienvenida. Además, se cuenta con el conocimiento mínimo requerido y la suficiente documentación para la concreción del proyecto.

Factibilidad Operativa

Debido a la naturaleza web del sistema, solo se requerirá de conexión a internet y un navegador para que los usuarios puedan hacer uso del sistema a través de una interfaz intuitiva y amigable, garantizando así la factibilidad operativa.

Factibilidad Legal

En el desarrollo se utilizarán herramientas de software libre, por lo que no se necesitarán licencias ni permisos de uso.

En cuanto al uso de datos personales, previo al trabajo con estos, se realizó un acuerdo de protección de los datos personales presentes en los textos judiciales utilizados, a fin de salvaguardar la identidad de las personas allí mencionadas como así también todo lo referente a direcciones, fechas, números telefónicos y demás datos personales contemplados en la Ley 25.326³⁵ de Protección de Datos Personales.

³⁵ <https://www.argentina.gob.ar/normativa/nacional/ley-25326-64790/texto>

4 CONCLUSIONES Y FUTURAS LÍNEAS DE INVESTIGACIÓN

Los grandes avances en investigaciones de Aprendizaje Automático han causado que cada día más aplicaciones y, software en general, implementen funcionalidades “inteligentes”. Una de ellas, el Reconocimiento automático de Entidades, se implementó en el sistema desarrollado en este proyecto.

Si bien los resultados pueden mejorarse, los aquí obtenidos gracias a la utilización de la avanzada librería para tareas de Procesamiento de Lenguaje Natural, spaCy, son satisfactorios.

De esta manera fue posible cumplir con los objetivos definidos en la sección 3.1, al desarrollar un sistema que implemente uno de los modelos de Reconocimiento de Entidades conseguidos. Dicho modelo, como se planteó en los criterios de éxito de la sección 1.4, tuvo un rendimiento superior al 70% de la métrica F1 score. Además, todo el trabajo se realizó siguiendo los pasos propuestos por la metodología elegida, CRISP-DM.

La investigación es fundamental para conocer nuevas técnicas y perfeccionar las ya conocidas, a fin de obtener resultados cada vez mejores. Es por ello, que a continuación se presentan algunas de las futuras líneas de investigación que podrían abordarse para llevar este proyecto a un siguiente nivel y formar una base sólida que permita abordar nuevos proyectos más ambiciosos:

- Está probado que la mayoría de las veces, lo que más contribuye a que un modelo de aprendizaje profundo consiga mejores resultados, es la obtención de un lote de datos cada vez mayor sobre el cual se entrene. Por ello un primer paso muy importante sería aumentar la cantidad de relatos anotados. Esto es posible dado a que el lote original recibido con denuncias sin anotar supera las 60 mil. En caso de necesitarse aún más datos, se podría consultar al Ministerio Público Fiscal para la obtención de los mismos.
- Es posible aplicar técnicas de aumentación de datos para incrementar el lote de entrenamiento y validación, incluso el de prueba.
- Implementar técnicas de Desambiguación de entidades. Como se mencionó en la sección 2.2.4.2, al combinar la desambiguación con el reconocimiento estamos ante un sistema de vínculo o enlace de entidades. Esto solucionaría un problema actual del sistema el cual es considerar a dos entidades que podrían referirse a los mismo, como si fueran distintas. Por ejemplo: Victoria y Lic. Victoria, en el sistema actual se considerarían dos entidades diferentes, cuando según el contexto en que se encuentre puede que traten de la misma persona.
- Utilizar otras librerías de aprendizaje automático, como lo son Scikit-Learn³⁶, TensorFlow³⁷ o Pytorch³⁸ en lugar de spaCy. Se trata de librerías más flexibles que

³⁶ <https://scikit-learn.org/stable/index.html>

³⁷ <https://www.tensorflow.org/>

³⁸ <https://pytorch.org/>

spaCy, cuya curva de aprendizaje es un poco mayor, pero cuentan con una comunidad más grande y permiten realizar tareas de aprendizaje automático que escapen del área del PLN.

- Investigar sobre Reconocimiento y Extracción de Relaciones, junto con una manera práctica de implementarlo sobre el sistema realizado. Esta tarea permitiría obtener mayor información de las entidades reconocidas mediante la identificación de vínculos entre ellas, de esta manera es posible descubrir patrones y visualizarlos en nuevos tipos de gráficos, preferiblemente grafos.
- Implementar las visualizaciones con la librería D3JS. De esta manera se podrían conseguir gráficos completamente flexibles y modificables. Se trata de una librería muy complicada de utilizar pero que ofrece los mejores resultados. Con ella es posible realizar una gran variedad de visualizaciones con diversos tipos de interacciones del usuario a fin de mejorar su experiencia en el análisis de entidades y relaciones.
- Profundizar en el conocimiento de la librería spaCy. Esta ofrece una gran cantidad de herramientas para realizar tareas de PLN que en este proyecto no han sido exploradas. Una de ellas es el uso de Transformers, los cuales son uno de los mejores modelos de aprendizaje profundo que existen para tareas de PLN en la actualidad. Algunos modelos de Transformers que podemos encontrar son: BERT, XLNet y GPT-2. Su particularidad es que han sido entrenados en conjuntos inmensos de datos y han sido puestos a disposición para emplearlos a modo de transferencia de aprendizaje (transfer Learning). Como dice (Vajjala, Majumder, Gupta, & Surana, 2020) “el aprendizaje de transferencia es una técnica en IA donde el conocimiento adquirido al resolver un problema se aplica a un problema relacionado”. Estos 3 modelos han sido implementados en los pipelines de spaCy que trabajan con Transformers.
- Investigar nuevas formas de visualizar entidades y sus relaciones.
- Agregar la funcionalidad de autenticación al sistema para que sólo los usuarios autorizados puedan acceder con sus credenciales.

Bibliografía

- Brat. (14 de 08 de 2021). *Acerca de Brat*. Obtenido de <https://brat.nlplab.org/about.html>
- Brysbaert, M. (2019). *How many words do we read per minute? A review and meta-analysis of reading rate*.
- Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). *CRISP-DM 1.0*. SPSS.
- COPAIPA. (14 de 08 de 2021). *Consejo Profesional de Agrimensores, Ingenieros y Profesionales Afines*. Obtenido de Honorarios de Informática: <http://www.copaipa.org.ar/informatica/>
- Deisenroth, M. P., Faisal, A. A., & Ong, C. S. (2020). *MATHEMATICS FOR MACHINE LEARNING*. Cambridge University Press.
- Explosion. (14 de 08 de 2021). *Documentación de spaCy*. Obtenido de ¿Qué es spaCy?: <https://spacy.io/usage/spacy-101#whats-spacy>
- Facebook. (14 de 08 de 2021). *Documentación de Create React App*. Obtenido de Empezando: <https://create-react-app.dev/docs/getting-started>
- Hoffman, M. D., Blei, D. M., & Back, F. (2010). Online Learning for Latent Dirichlet Allocation. *Advances in Neural Information Processing Systems*, 1-9.
- Howard, J., & Gugger, S. (2020). *Deep Learning for Coders with fastai & PyTorch AI Applications Without a PhD*. Sebastopol, CA: O'Reilly Media.
- Loewen, C., & olprod. (14 de 08 de 2021). *Microsoft*. Obtenido de ¿Qué es el Subsistema de Windows para Linux?: <https://docs.microsoft.com/es-es/windows/wsl/about>
- Microsoft. (14 de 08 de 2021). *Documentación de VS Code*. Obtenido de Empezando: <https://code.visualstudio.com/docs>
- Mitchell, T. (1997). *Machine Learning*. : McGraw-Hill.
- mongoDB. (14 de 08 de 2021). *Documentación de MongoDB*. Obtenido de ¿Qué es MongoDB?: <https://www.mongodb.com/es/what-is-mongodb>
- Narechania, A., Srinivasan, A., & Stasko, J. (2020). NL4DV: A toolkit for generating analytic specifications for data visualization from natural language queries. *IEEE Transactions on Visualization and Computer Graphics*, 1.
- Netlify. (14 de 08 de 2021). *Documentación de Netlify*. Obtenido de Bienvenido a Netlify: https://docs.netlify.com/?_ga=2.64554342.812467170.1628976148-11289759.1628976148
- Project Managment Institute. (2013). *Guía de los FUNDAMENTOS PARA LA DIRECCIÓN DE PROYECTOS (GUÍA DEL PMBOK)* (Quinta ed.). Newtown Square, Pensilvania: Project Management Institute.

Python Software Foundation. (14 de 08 de 2021). *Documentación de Python*. Obtenido de El tutorial de Python: <https://docs.python.org/3/tutorial/index.html>

Recharts Group. (14 de 08 de 2021). *Recharts*. Obtenido de <https://recharts.org/en-US>

Salesforce. (14 de 08 de 2021). *Heroku*. Obtenido de ¿Qué es Heroku?: <https://www.heroku.com/what>

Simeone, O. (2018). A Very Brief Introduction to Machine Learning With Applications to Communication Systems. *IEEE*, 1-20.

The Pallets Projects. (14 de 08 de 2021). *Flask*. Obtenido de <https://palletsprojects.com/p/flask/>

Torvalds, L. (14 de 08 de 2021). *Git*. Obtenido de <https://git-scm.com/>

Vajjala, S., Majumder, B., Gupta, A., & Surana, H. (2020). *Practical Natural Language Processing A Comprehensive Guide to Building Real-World NLP Systems*. Sebastopol, CA: O'Reilly Media.

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2017). *DATA MINING Practical Machine Learning Tools and Techniques* (Cuarta ed.). Cambridge: Morgan Kaufmann.

ANEXO 1 – TERMINOLOGÍA

API: abreviatura de *Application Programming Interface*, en español interfaz de programación de aplicaciones se trata del conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones, permitiendo la comunicación entre dos aplicaciones. En el proyecto se desarrolló una API con Flask la cual permite la comunicación entre el cliente desarrollado en React y el servidor de Python. A través de una serie de reglas (protocolo HTTP) se comunican cliente y servidor para realizar las diferentes operaciones con relatos y entidades que se han programado.

EI: Extracción de Información

ER: Extracción de Relaciones

RI: Recuperación de Información

PLN: Procesamiento del Lenguaje Natural

REN: Reconocimiento de Entidades Nombradas

DEN: Desambiguación de Entidad Nombrada

EEN: Enlace de Entidad Nombrada

Frontend: se refiere a la parte del sistema que ven los usuarios. Para este proyecto comprende todo lo desarrollado en React y Recharts, ya que con ellos se realizó la interfaz gráfica del sistema, la parte con la cual interactúan los usuarios.

Backend: se refiere a la parte del sistema a la cual los usuarios no tienen acceso, aquello que no pueden ver desde el navegador. Aquí entraría el servidor realizado en Flask, los distintos scripts de Python y spaCy y la base de datos MongoDB.

Pipeline: en Aprendizaje Automático se refiere al flujo o proceso que debe seguirse para producir un modelo de aprendizaje automático. Un pipeline consiste en múltiples pasos secuenciales que involucran entre otras cosas la extracción de datos, entrenamiento y validación del modelo. Cuando se construyó el archivo *config.cfg* en la fase 4 de la sección 4.5, se eligieron los elementos que irían presentes en el pipeline: qué datos se utilizarían, de dónde se los extraería, que componentes se entrenarían y validarían (tok2vec y NER) y en qué orden.

ANEXO 2 – INSTALACIÓN Y EMPLEO DE BRAT

Brat sólo puede ser instalado en un sistema UNIX. Existen 3 opciones diferentes:

- Instalarlo en un sistema Linux, ya sea en una PC Linux o una PC con dos sistemas instalados (dual boot): Windows y Linux.
- Utilizar una máquina virtual con Linux instalado.
- Utilizar el sistema WSL (Linux sobre Windows 10). Ésta última es la alternativa escogida para este proyecto.

Instalar WSL Ubuntu 20.04

1. Entrar en configuración de Windows (Windows + I), ir a “Para programadores” y habilitar el modo de programador.
2. Luego ir al panel de control, programas, activar o desactivar características de Windows, habilitar “Subsistema de Windows para Linux”, aceptar. Se reiniciará la PC.
3. Buscar WSL Ubuntu 20.04 en la Microsoft Store e instalarlo.
4. Abrir Ubuntu, se comienza a instalar. Elegir un nombre de usuario y contraseña.
5. Una vez instalado ejecutar el WSL. Se abrirá la línea de comandos con Ubuntu.
6. Podemos ver que Python ya viene instalado con:

```
python3 --version
```

Brat necesita alguna versión desde la 2.5 en adelante pero dentro de la serie 2.x. Por ello instalamos Python con el comando:

```
sudo apt install python
```

7. Comprobar si ya está en la versión 2.x ejecutando:

```
python --versin
```

Instalar Brat

1. Instalar *wget* con

```
sudo apt install wget
```

2. Ubicarse en el directorio donde se desee instalar Brat
3. Ejecutar:

```
sudo wget http://weaver.nlpplab.org/~brat/releases/brat-v1.3\_Crunchy\_Frog.tar.gz
```

(revisar si funciona el link sino obtener Brat desde el repositorio de Github³⁹).

³⁹ <https://github.com/nlpplab/brat>

4. Extraer el archivo con:

```
tar -xvzf <nombre del archivo>
```

5. Entrar a la carpeta extraída y ejecutar:

```
./install.sh
```

6. Completar los datos solicitados

7. Ejecutar:

```
python standalone.py
```

8. Abrir en el navegador de Windows la dirección indicada en el WSL (por defecto `http://127.0.0.1:8001/`)

Agregar relatos para anotar

En la carpeta de *data* ubicada en el directorio de instalación de Brat es donde se deben agregar los archivos txt para anotar. Allí se recomienda crear una carpeta, en este caso para los relatos de las denuncias. Para este proyecto se creó la carpeta *denuncias* ejecutando:

```
mkdir denuncias
```

y se le asignó permiso de acceso total con:

```
chmod 777 denuncias
```

Luego se copiaron los relatos de denuncias desde la carpeta en Windows hasta la carpeta *denuncias* en Ubuntu con el comando copy:

```
cp /mnt/disco/.../denuncias_txts -r ./
```

Por cada archivo .txt, Brat necesita un archivo .ann. Esto se puede crear, ubicado en la carpeta *denuncias*, con:

```
find denuncias_txts -name '*.txt' | sed -e 's|\.txt|.ann|g' | xargs touch
```

Anotar relatos

Una vez cargados los relatos en la carpeta *data*, se debe ingresar a la dirección de Brat desde el navegador. En segundo lugar, iniciar sesión con el usuario y clave definidos en la instalación. Con esto ya es posible iniciar la anotación. El mismo Brat incluye una serie de tutoriales para realizar dicha tarea.

Una manera de ir supervisando cuantos archivos hemos anotado consiste en contar los archivos .ann de la carpeta *denuncias* con tamaño mayor a 0 kb. Aquellos sin entidades anotadas pesan 0Kb (ya sea porque no presentaban los tipos de entidades buscados o porque aún no se analizaron). Esto se puede hacer ejecutando:

```
find . -name '*.ann' -size +0 -ls | wc -l
```

Luego de anotar los relatos

Tras anotar los relatos interesa quedarse sólo con los archivos txt y ann que presenten entidades. Para ello es recomendable hacer una copia de la carpeta con denuncias ejecutando:

```
cp ./denuncias denunciasAnotadas
```

Luego dentro de *denunciasAnotadas* borrar en primer lugar todos los archivos .ann vacíos (de peso 0Kb) ejecutando:

```
- find -name '*.ann' -size 0 -delete
```

Luego, a todos los archivos resultantes se los copia al directorio de trabajo en Windows con:

```
cp * /mnt/d/.../denuncias_annotadas
```

Por último, borrar los archivos .txt sobrantes ejecutando el script de Python: *eliminar_txts_sin_archivo_ann.py* (Anexo 4)

Cambiar tipos de entidades

Si se desea cambiar los tipos de entidades que Brat permite anotar por defecto, ya sea quitando las existentes o agregando nuevas, esto es posible hacerlo si se modifica el archivo *annotation.conf* ubicado en la carpeta de instalación de Brat. Una vez abierto el archivo, en la sección *[entities]* es donde se realiza la modificación.

ANEXO 3 – PAQUETES UTILIZADOS

Frontend – Lado del Cliente

Para comenzar a trabajar desde el lado del cliente es necesario tener instalado Node.js⁴⁰. Es posible verificar esto escribiendo en un terminal:

```
node -v
```

Si como resultado obtenemos una versión válida de Node.js ya está listo. Caso contrario deberemos instalarlo. En el proyecto se utilizó Windows y la manera de instalarlo en este sistema operativo es a través del instalador provisto en la página de Node.js⁴¹.

A la hora de trabajar con React se utilizó el paquete *create-react-app* el cual está creado por su equipo. Incluye todas las dependencias configuradas y genera una estructura básica de proyecto para comenzar a trabajar.

Para utilizar create-react-app es necesario tener instalada la herramienta de ejecución de paquetes de npm llamada *npx*. Esta se instala automáticamente junto con Node.js. De esta manera podremos iniciar un proyecto nuevo ejecutando:

```
npx create-react-app nombreDelProyecto.
```

Una vez creada la plantilla inicial del proyecto es posible instalar las diferentes librerías que fueron necesarias en el proyecto. Estas son:

- React-router-dom⁴²: librería que permite definir las rutas internas dentro de la aplicación. Se instala con:

```
npm i react-router-dom
```

- Bootswatch⁴³: librería que ofrece una variedad de temas realizados en Bootstrap para modificar el diseño del sitio web. Se instala con

```
npm i bootswatch
```

- Font Awesome⁴⁴: kit de herramientas que incluye fuentes e íconos basados en CSS y Less. Se instala con

```
npm i font-awesome
```

- Recharts: presentado en la sección 4.2 “Herramientas utilizadas”. Se instala con

```
npm i recharts
```

⁴⁰ <https://nodejs.org/es/>

⁴¹ <https://nodejs.org/es/download/>

⁴² <https://github.com/ReactTraining/react-router#readme>

⁴³ <https://bootswatch.com/>

⁴⁴ <https://fontawesome.com/>

- Sweetalert 2⁴⁵: librería que sirve para reemplazar las ventanas emergentes de JavasCript por otras personalizables y con mejor diseño. Se instala con

```
npm i sweetalert2
```

- React-paginate⁴⁶: componente de React que permite agregar paginación personalizable al sitio web. Se instala con

```
npm i react-paginate
```

Backend – Lado del Servidor

Para el lado del servidor se trabajó con Miniconda⁴⁷, una versión minimalista de Anaconda. Miniconda incluye tanto Python como Conda⁴⁸, la herramienta para manejar dependencias de Python y crear ambientes virtuales.

Una vez instalado Miniconda, se creó un ambiente virtual con el comando:

```
conda create -n nombreDelAmbiente
```

Luego se activó el ambiente creado con:

```
conda activate nombreDelAmbiente
```

Una vez situados en el ambiente se instalaron las siguientes librerías:

- Flask: presentado en la sección 4.2 “Herramientas utilizadas”. Se instala con:

```
conda install -c anaconda flask
```

- Flask-pymongo: presentado en la sección 4.2 “Herramientas utilizadas”. Se instala con:

```
pip install Flask-PyMongo
```

- Flask-cors⁴⁹: extensión de Flask que permite manejar el intercambio de recursos de origen cruzado (CORS), lo que hace posible AJAX de origen cruzado. Se instala con:

```
pip install flask-cors
```

- spaCy: presentado en la sección 4.2 “Herramientas utilizadas”. Se instala con:

```
conda install -c conda-forge spaCy
```

⁴⁵ <https://sweetalert2.github.io/>

⁴⁶ <https://www.npmjs.com/package/react-paginate>

⁴⁷ <https://docs.conda.io/en/latest/miniconda.html>

⁴⁸ <https://docs.conda.io/projects/conda/en/latest/>

⁴⁹ <https://flask-cors.readthedocs.io/en/latest/>

- `es_core_news_lg`: modelo pre entrenado de spaCy que se utilizó en el proyecto. Se instala con:

```
python -m spacy download es_core_news_lg
```

- `Dnspython`⁵⁰: conjunto de herramientas de DNS para Python. Se puede utilizar para consultas, transferencias de zona, actualizaciones dinámicas, pruebas de servidores DNS, etc. Es necesario instalarlo para desplegar la aplicación en *Heroku*. Se instala con:

```
conda install -c anaconda dnspython
```

- `Gunicorn`⁵¹: es un servidor HTTP WSGI de Python para UNIX. Es recomendable instalarlo a la hora de desplegar la aplicación en *Heroku*. Se instala con:

```
pip install gunicorn.
```

Además, para los scripts de Python del Anexo 4 se requirió instalar `Pandas`⁵², una librería de código abierto para análisis y manipulación de datos para Python, en el mismo ambiente de conda.

⁵⁰ <https://www.dnspython.org/>

⁵¹ <https://gunicorn.org/>

⁵² <https://pandas.pydata.org/>

ANEXO 4 – SCRIPTS DE PYTHON

Script generarTxts.py

```
import pandas as pd

df_raw = pd.read_csv("./denuncias_2016-limpio.csv",
                    sep=";", index_col="denunciaId")

# Excluir las denuncias separadas para Anonimización y carga en el sistema final
excluir = [908246, 910965, 931013, 945316, 950241, 982263, 959138, 105079
6, 1029958, 1084262, 925934, 972770, 1070947, 1083880, 943053, 958682, 970033,
1018783, 1020529, 965104, 1033326, 1066940, 927911, 1085601, 1046761, 1064298,
1036205, 981993, 1071168, 1047844, 920695, 928376, 953429, 1048035, 1051907, 949
706, 957806, 979010, 941446, 916003, 982584, 970167, 1058597, 1037701, 1094797,
1084959, 1083252, 965712, 930049, 936266, 1066343, 979029, 960014, 937908, 105
3553, 1021580, 1076363, 908302, 943788, 911197, 910369, 923930, 966932, 931011,
935943, 1038237, 973934, 922679, 1036157, 975210]

# Eliminar columnas de atributos no utilizados
col_to_drop = ['Anio', 'idModalidad', 'Modalidad', 'tipoLugarId', 'TipoLu
gar']

df = df_raw.drop(excluir, axis=0)
df.drop(col_to_drop, axis=1, inplace=True)

# Mezclar datos
df = df.sample(frac=1, random_state=0)

# Genera archivos txt por cada fila del DataFrame
def generar_txts(ruta, data):
    for index, fila in data.iterrows():
        with open(f"{ruta}/{index}.txt", "w", encoding="utf-8") as f:
            f.write(fila["relato"])

# Generar archivos txts para los 1000 primeros relatos
data = df.iloc[:1000]

generar_txts("./1000DenunciasParaAnotar", data)
```

Script eliminar_txts_sin_archivo_ann.py

```
from pathlib import Path
import os

nombre_carpeta_denuncias = '450DenunciasAnotadas'
archivos_ann = []
archivos_txt = []

for p in Path(f'./{nombre_carpeta_denuncias}').glob('**/*.ann'):
    archivos_ann.append(p.name.split('.')[0])

for p in Path(f'./{nombre_carpeta_denuncias}').glob('**/*.txt'):
    archivos_txt.append(p.name.split('.')[0])

# Obtener los txt sobrantes
for f in archivos_ann:
    if f in archivos_txt:
        archivos_txt.remove(f)

for f in archivos_txt:
    os.remove(f'./{nombre_carpeta_denuncias}/{f}.txt')
```

Script contarTotalDeEntidades.py

```
from pathlib import Path
import re

nombre_carpeta_denuncias = '450DenunciasAnotadas - balanceadas - original'

totalEntrenamiento = 350
totalValidacion = 50
totalPrueba = 50

cant_entidades = {"Persona": 0, "Barrio": 0,
                  "DNI": 0, "Patente": 0, "Telefono": 0}

for p in Path(f'./{nombre_carpeta_denuncias}/').glob('**/*.ann'):
    idDenuncia = p.name.split('.')[0]
    entidades_raw = []
```

```

entidades = {}
entidades_raw = p.read_text(
    encoding='utf-8').replace("\t", " ")
entidades_raw = re.sub("T\d+", "", entidades_raw).split("\n")
# Por defecto cada lista tiene un elemento vacío. Debajo se remueve.
entidades_raw = list(filter(None, entidades_raw))
for ent in entidades_raw:
    ent = ent.strip()
    ent_partes = ent.split(" ")
    label = str(ent_partes[0])
    cant_entidades[f"{label}"] += 1
print(cant_entidades)

```

Script determinarMejorSeedParaBalanceo.py

```

from pathlib import Path
import random
import re

nombre_carpeta_denuncias = '450DenunciasAnotadas - balanceadas - con pesos'

train_ideal = {'Persona': 626, 'Barrio': 143,
               'DNI': 62, 'Patente': 101, 'Telefono': 104}
valid_ideal = {'Persona': 89, 'Barrio': 20,
               'DNI': 9, 'Patente': 15, 'Telefono': 15}
test_ideal = {'Persona': 88, 'Barrio': 20,
               'DNI': 8, 'Patente': 14, 'Telefono': 14}

totalEntrenamiento = 350
totalValidacion = 50
totalPrueba = 50
menor_diferencia = 10000
mejor_semilla = 1
denuncias = []

for den in Path(f'./{nombre_carpeta_denuncias}').glob('**/*.ann'):
    denuncias.append(den)

```

```

for sem in range(10000):
    train_seed = {'Persona': 0, 'Barrio': 0,
                  'DNI': 0, 'Patente': 0, 'Telefono': 0}
    valid_seed = {'Persona': 0, 'Barrio': 0,
                  'DNI': 0, 'Patente': 0, 'Telefono': 0}
    test_seed = {'Persona': 0, 'Barrio': 0,
                 'DNI': 0, 'Patente': 0, 'Telefono': 0}
    diferencia = {'entrenamiento': 0,
                  'validacion': 0, 'prueba': 0, 'promedio': 0}
    random.seed(sem)
    denuncias_mezcladas = random.sample(denuncias, len(denuncias))

    for i in range(totalEntrenamiento):
        entidades_raw = []
        entidades_raw = denuncias_mezcladas[i].read_text(
            encoding='utf-8').replace("\t", " ")
        entidades_raw = re.sub("T\d+", "", entidades_raw).split("\n")
        # Por defecto cada lista tiene un elemento vacío. Debajo se lo re
mueve.

        entidades_raw = list(filter(None, entidades_raw))
        for ent in entidades_raw:
            ent = ent.strip()
            ent_partes = ent.split(" ")
            label = str(ent_partes[0])
            train_seed[f"{label}"] += 1

    for i in range(totalEntrenamiento, totalEntrenamiento + totalValidaci
on):
        entidades_raw = []
        entidades_raw = denuncias_mezcladas[i].read_text(
            encoding='utf-8').replace("\t", " ")
        entidades_raw = re.sub("T\d+", "", entidades_raw).split("\n")
        # Por defecto cada lista tiene un elemento vacío. Debajo se lo re
mueve.

        entidades_raw = list(filter(None, entidades_raw))

```

```
        for ent in entidades_raw:
            ent = ent.strip()
            ent_partes = ent.split(" ")
            label = str(ent_partes[0])
            valid_seed[f"{label}"] += 1

    for i in range(totalEntrenamiento + totalValidacion, totalEntrenamien
to + totalValidacion + totalPrueba):
        entidades_raw = []
        entidades_raw = denuncias_mezcladas[i].read_text(
            encoding='utf-8').replace("\t", " ")
        entidades_raw = re.sub("T\d+", "", entidades_raw).split("\n")
        # Por defecto cada lista tiene un elemento vacío. Debajo se lo re
mueve.

        entidades_raw = list(filter(None, entidades_raw))
        for ent in entidades_raw:
            ent = ent.strip()
            ent_partes = ent.split(" ")
            label = str(ent_partes[0])
            test_seed[f"{label}"] += 1

# PESOS:
diferencia["entrenamiento"] += (abs(
    train_ideal["Persona"] - train_seed["Persona"]) * 0.3953)
diferencia["entrenamiento"] += (abs(
    train_ideal["Barrio"] - train_seed["Barrio"]) * 0.8622)
diferencia["entrenamiento"] += (abs(
    train_ideal["DNI"] - train_seed["DNI"]) * 0.9405)
diferencia["entrenamiento"] += (abs(
    train_ideal["Patente"] - train_seed["Patente"]) * 0.9021)
diferencia["entrenamiento"] += (abs(
    train_ideal["Telefono"] - train_seed["Telefono"]) * 0.8999)

diferencia["validacion"] += (abs(
    valid_ideal["Persona"] - valid_seed["Persona"]) * 0.3953)
```

```
diferencia["validacion"] += (abs(
    valid_ideal["Barrio"] - valid_seed["Barrio"]) * 0.8622)
diferencia["validacion"] += (abs(
    valid_ideal["DNI"] - valid_seed["DNI"]) * 0.9405)
diferencia["validacion"] += (abs(
    valid_ideal["Patente"] - valid_seed["Patente"]) * 0.9021)
diferencia["validacion"] += (abs(
    valid_ideal["Telefono"] - valid_seed["Telefono"]) * 0.8999)

diferencia["prueba"] += (abs(
    test_ideal["Persona"] - test_seed["Persona"]) * 0.3953)
diferencia["prueba"] += (abs(
    test_ideal["Barrio"] - test_seed["Barrio"]) * 0.8622)
diferencia["prueba"] += (abs(
    test_ideal["DNI"] - test_seed["DNI"]) * 0.9405)
diferencia["prueba"] += (abs(
    test_ideal["Patente"] - test_seed["Patente"]) * 0.9021)
diferencia["prueba"] += (abs(
    test_ideal["Telefono"] - test_seed["Telefono"]) * 0.8999)

diferencia["promedio"] = (
    diferencia["entrenamiento"] + diferencia["validacion"] + diferencia["prueba"])/3

if (diferencia["promedio"] < menor_diferencia):
    menor_diferencia_dict = diferencia
    menor_diferencia = diferencia["promedio"]
    mejor_semilla = sem

print(f"Mejor semilla: {mejor_semilla} - Diferencia: {menor_diferencia_dict}")
```

Script dividirRelatosTrainValidTest.py

```
from pathlib import Path
import os
import random

nombre_carpeta_denuncias = '450DenunciasAnotadas - balanceadas - con pesos'

SEMILLA = 7448

totalEntrenamiento = 350
totalValidacion = 50
totalPrueba = 50

denuncias = []

for den in Path(f'./{nombre_carpeta_denuncias}').glob('**/*.txt'):
    denuncias.append(den.name.split('.')[0])

# Se define la semilla para que siempre se mezcle de la misma forma
random.seed(SEMILLA)

denuncias_mezcladas = random.sample(denuncias, len(denuncias))

os.mkdir(f'{nombre_carpeta_denuncias}/train_data')
os.mkdir(f'{nombre_carpeta_denuncias}/valid_data')
os.mkdir(f'{nombre_carpeta_denuncias}/test_data')

for i in range(totalEntrenamiento):
    denuncia = denuncias_mezcladas.pop(0)
    os.replace(f'{nombre_carpeta_denuncias}/{denuncia}.txt',
              f'{nombre_carpeta_denuncias}/train_data/{denuncia}.txt')
    os.replace(f'{nombre_carpeta_denuncias}/{denuncia}.ann',
              f'{nombre_carpeta_denuncias}/train_data/{denuncia}.ann')

for i in range(totalValidacion):
    denuncia = denuncias_mezcladas.pop(0)
    os.replace(f'{nombre_carpeta_denuncias}/{denuncia}.txt',
              f'{nombre_carpeta_denuncias}/valid_data/{denuncia}.txt')
    os.replace(f'{nombre_carpeta_denuncias}/{denuncia}.ann',
              f'{nombre_carpeta_denuncias}/valid_data/{denuncia}.ann')
```

```

for i in range(totalPrueba):
    denuncia = denuncias_mezcladas.pop(0)
    os.replace(f'{nombre_carpeta_denuncias}/{denuncia}.txt',
              f'{nombre_carpeta_denuncias}/test_data/{denuncia}.txt')
    os.replace(f'{nombre_carpeta_denuncias}/{denuncia}.ann',
              f'{nombre_carpeta_denuncias}/test_data/{denuncia}.ann')

```

Script bratAnnotationTospaCyAnnotation.py

```

import re
import json
from pathlib import Path

nombre_carpeta_denuncias = '450DenunciasAnotadas - balanceadas - con pesos'

carpetas = ["train_data", "valid_data", "test_data"]
for carpeta in carpetas:
    anotaciones = []
    for p in Path(f'./{nombre_carpeta_denuncias}/{carpeta}').glob('**/*.ann'):
        idDenuncia = p.name.split('.')[0]
        texto = Path(
            f'./{nombre_carpeta_denuncias}/{carpeta}/{idDenuncia}.txt').read_text(encoding='utf-8')

        entidades_raw = []
        entidades = {}
        entidades_raw = p.read_text(
            encoding='utf-8').replace("\t", " ")
        entidades_raw = re.sub("T\d+", "", entidades_raw).split("\n")
        # Por defecto cada lista tiene un elemento vacío. Debajo se lo re
        # mueve.

        entidades_raw = list(filter(None, entidades_raw))
        entidades_procesado = []
        for ent in entidades_raw:

```

```
        ent = ent.strip()
        ent_partes = ent.split(" ")
        label = str(ent_partes[0])
        inicio = int(ent_partes[1])
        final = int(ent_partes[2])
        entidades_procesado.append([inicio, final, label])

    if len(entidades_raw) > 0:
        entidades['entities'] = (entidades_procesado)
        anotaciones.append([texto, entidades])

    with open(f"./{nombre_carpeta_denuncias}/{carpeta}.json", "w", encoding="utf-8") as f:
        json.dump(anotaciones, f, indent=4)
```

Script generarArchivospaCy.py

```
from tqdm import tqdm
from spacy.tokens import DocBin
import json
import spacy

# FUNCIONES AUXILIARES
def cargar_datos(archivo):
    with open(archivo, "r", encoding="utf-8") as f:
        datos = json.load(f)
    return (datos)

def escribir_datos(archivo, datos):
    with open(archivo, "w", encoding="utf-8") as f:
        json.dump(datos, f, indent=4)

def transformar_datos(datos):
    db = DocBin()
```

```
for texto, anotaciones in tqdm(datos):
    doc = nlp.make_doc(texto)
    ents = []
    for inicio, final, label in anotaciones["entities"]:
        span = doc.char_span(inicio, final, label=label, alignment_model="contract")

        if span is None:
            print("Entidad omitida")
        else:
            ents.append(span)
    doc.ents = ents
    db.add(doc)
return db

train_data = cargar_datos("./datos/train_data.json")
valid_data = cargar_datos("./datos/valid_data.json")

train_data = transformar_datos(train_data)
valid_data = transformar_datos(valid_data)

train_data.to_disk("./datos/train_data.spacy")
valid_data.to_disk("./datos/valid_data.spacy")
```

Script matrizDeConfusion.py

```
import spacy
import json

from spacy.training import offsets_to_biluo_tags
from sklearn.metrics import confusion_matrix
from matplotlib import pyplot
import numpy

# FUNCIONES AUXILIARES
def cargar_datos(archivo):
```

```
with open(archivo, "r", encoding="utf-8") as f:
    datos = json.load(f)
    return (datos)

def escribir_datos(archivo, datos):
    with open(archivo, "w", encoding="utf-8") as f:
        json.dump(datos, f, indent=4)

def get_cleaned_label(label: str):
    if "-" in label:
        label = label.split("-")[1]
        if label == '':
            label = "O"
        return label
    else:
        return label

def create_total_target_vector(docs):
    target_vector = []
    for doc in docs:
        new = nlp.make_doc(doc[0])
        entities = doc[1]["entities"]
        bilou_entities = offsets_to_biluo_tags(new, entities)
        final = []
        for item in bilou_entities:
            final.append(get_cleaned_label(item))
        target_vector.extend(final)
    return target_vector

def create_prediction_vector(text):
    return [get_cleaned_label(prediction) for prediction in get_all_ner_predictions(text)]
```

```
def create_total_prediction_vector(docs: list):
    prediction_vector = []
    for doc in docs:
        prediction_vector.extend(create_prediction_vector(doc[0]))
    return prediction_vector

def get_all_ner_predictions(text):
    doc = nlp(text)
    entities = [(e.start_char, e.end_char, e.label_) for e in doc.ents]
    bilou_entities = offsets_to_biluo_tags(doc, entities)
    return bilou_entities

def get_model_labels():
    labels = list(nlp.get_pipe("ner").labels)
    labels.append("O")
    return sorted(labels)

def get_dataset_labels():
    return sorted(set(create_total_target_vector(docs)))

def generate_confusion_matrix(docs):
    classes = sorted(set(create_total_target_vector(docs)))
    y_true = create_total_target_vector(docs)
    y_pred = create_total_prediction_vector(docs)
    return confusion_matrix(y_true, y_pred, classes)

def plot_confusion_matrix(docs, classes, normalize=False, cmap=pyplot.cm.
Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    title = 'Confusion Matrix, for spaCy NER'
```

```
# Compute confusion matrix

cm = generate_confusion_matrix(docs)

if normalize:
    cm = cm.astype('float') / cm.sum(axis=1)[:, numpy.newaxis]

fig, ax = pyplot.subplots()
im = ax.imshow(cm, interpolation='nearest', cmap=cmap)
ax.figure.colorbar(im, ax=ax)
# We want to show all ticks...
ax.set(xticks=numpy.arange(cm.shape[1]),
       yticks=numpy.arange(cm.shape[0]),
       # ... and label them with the respective list entries
       xticklabels=classes, yticklabels=classes,
       title=title,
       ylabel='True label',
       xlabel='Predicted label')

# Rotate the tick labels and set their alignment.
pyplot.setp(ax.get_xticklabels(), rotation=45, ha="right", rotation_m
ode="anchor")

# Loop over data dimensions and create text annotations.
fmt = '.2f' if normalize else 'd'
thresh = cm.max() / 2.
for i in range(cm.shape[0]):
    for j in range(cm.shape[1]):
        ax.text(j, i, format(cm[i, j], fmt),
                ha="center", va="center",
                color="white" if cm[i, j] > thresh else "black")
fig.tight_layout()
return cm, ax, pyplot

plot_confusion_matrix(docs, classes=get_dataset_labels(), normalize=False)
```

ANEXO 5 – ARCHIVO CONFIG.CFG MODELO 5

```
[paths]
train = "./datos/train_data.spacy"
dev = "./datos/valid_data.spacy"
vectors = null
init_tok2vec = null

[system]
gpu_allocator = null
seed = 0

[nlp]
lang = "es"
pipeline = ["tok2vec","ner"]
batch_size = 1000
disabled = []
before_creation = null
after_creation = null
after_pipeline_creation = null
tokenizer = {"@tokenizers":"spaCy.Tokenizer.v1"}

[components]

[components.ner]
factory = "ner"
moves = null
update_with_oracle_cut_size = 100

[components.ner.model]
@architectures = "spaCy.TransitionBasedParser.v2"
state_type = "ner"
extra_state_tokens = false
hidden_width = 64
maxout_pieces = 2
use_upper = true
nO = null

[components.ner.model.tok2vec]
@architectures = "spaCy.Tok2VecListener.v1"
width = ${components.tok2vec.model.encode.width}
upstream = "*"

[components.tok2vec]
factory = "tok2vec"

[components.tok2vec.model]
@architectures = "spaCy.Tok2Vec.v2"
```

```
[components.tok2vec.model.embed]
@architectures = "spaCy.MultiHashEmbed.v2"
width = ${components.tok2vec.model.encode.width}
attrs = ["ORTH","SHAPE"]
rows = [5000,2500]
include_static_vectors = true

[components.tok2vec.model.encode]
@architectures = "spaCy.MaxoutWindowEncoder.v2"
width = 256
depth = 8
window_size = 1
maxout_pieces = 3

[corpora]

[corpora.dev]
@readers = "spaCy.Corpus.v1"
path = ${paths.dev}
max_length = 0
gold_preproc = false
limit = 0
augmenter = null

[corpora.train]
@readers = "spaCy.Corpus.v1"
path = ${paths.train}
max_length = 2000
gold_preproc = false
limit = 0
augmenter = null

[training]
dev_corpus = "corpora.dev"
train_corpus = "corpora.train"
seed = ${system.seed}
gpu_allocator = ${system.gpu_allocator}
dropout = 0.1
accumulate_gradient = 1
patience = 1600
max_epochs = 0
max_steps = 20000
eval_frequency = 200
frozen_components = []
before_to_disk = null

[training.batcher]
@batchers = "spaCy.batch_by_words.v1"
discard_oversize = false
tolerance = 0.2
get_length = null
```

```
[training.batcher.size]
@schedules = "compounding.v1"
start = 100
stop = 1000
compound = 1.001
t = 0.0

[training.logger]
@loggers = "spaCy.ConsoleLogger.v1"
progress_bar = false

[training.optimizer]
@optimizers = "Adam.v1"
beta1 = 0.9
beta2 = 0.999
L2_is_weight_decay = true
L2 = 0.01
grad_clip = 1.0
use_averages = false
eps = 0.00000001
learn_rate = 0.0003

[training.score_weights]
ents_per_type = null
ents_f = 1.0
ents_p = 0.0
ents_r = 0.0

[pretraining]

[initialize]
vectors = "es_core_news_lg"
init_tok2vec = ${paths.init_tok2vec}
vocab_data = null
lookups = null
before_init = null
after_init = null

[initialize.components]

[initialize.tokenizer]
```