Universidad de la República

Facultad de Ingeniería

Aprendizaje profundo para la anonimización de textos legales

Informe de Proyecto de Grado presentado por

María Victoria Cruces, Nicolás Ferraro

en cumplimiento parcial de los requerimientos para la graduación de la carrera de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de la República

Supervisor

Diego Garat

Montevideo, 20 de octubre de 2023

Agradecimientos

Agradecemos a nuestras familias y amigos por el cariño y apoyo constante, por su paciencia en los momentos de frustración, no dejarnos bajar los brazos y empujarnos hacia adelante.

A nuestro tutor Diego Garat por su excelente disposición, su ayuda y paciencia a lo largo de todo el proyecto.

Resumen

Las sentencias judiciales contienen datos sensibles sobre los actores involucrados que, dado su carácter público, deben ser anonimizadas para proteger la identidad de las personas. Hoy en día la anonimización de sentencias judiciales se hace de forma manual, proceso que es costoso y propenso a errores.

En este contexto, se aplican técnicas de aprendizaje profundo con intención de automatizar el proceso. Para llevar esto a cabo, se trabaja sobre el corpus de sentencias de la Base de Jurisprudencia Nacional recopilado por Garat y Wonserver en "Towards De-identification of Legal Texts"[1], en donde las sentencias se encuentran anotadas con etiquetas que indican los participantes involucrados. Con el fin de poder diferenciarlos de otras personas como ministros, abogados, doctrinos y ubicaciones que llevan el nombre de personas, se agregan nuevas etiquetas para poder mejorar la capacidad de detección de participantes a anonimizar. Partiendo del conjunto, se implementan dos soluciones: un reconocedor de entidades nombradas basado en *Transformers* y como prueba de concepto, una red neuronal siamesa acompañada de *clustering* para vincular las entidades encontradas.

El reconocedor alcanza una F_1 de 95.3 % al identificar personas involucradas en los hechos de las sentencias (las de interés al anonimizar), representando una mejoría frente a soluciones previas del 3,54 %. En el caso de la red se obtiene un V-measure de 93.3 % y un ARI de 88.2 %, inferiores a resultados existentes [2].

Palabras clave: Anonimización, Sentencias Judiciales, Reconocimiento de entidades, Vinculación de entidades, Resolución de Coreferencias, Transformers, Redes Siamesas, Clusterización

Índice general

Capítulo 1

Introducción

Con el fin de democratizar el acceso a las sentencias judiciales del Uruguay se forma la Base de Jurisprudencia Nacional (o BJN), mantenida por el poder Judicial. La base recopila las sentencias judiciales expedidas por Tribunales de Apelaciones, Juzgados de Primera Instancia y la Suprema Corte de Justicia y se expone a través de un portal web[1] de acceso público. Las sentencias contienen datos de carácter personal y, por lo tanto, se debe tener en cuenta la conservación de la privacidad de las personas involucradas; la publicación de datos sensibles se encuentra regulada y puede acarrear sanciones legales. Para ello, se ofuscan aquellos datos que permitan identificar personas o instituciones.

Este procedimiento de desidentificación en la BJN es manual; se requiere al menos una persona que lea la sentencia e identifique las personas involucradas en los hechos —los participantes— y reemplace sus nombres por otros. El formato de reemplazo elegido en la BJN consiste de dos letras iguales AA, BB, .., ZZ. Cabe destacar un requisito de la anonimización es que la sentencia anonimizada no pierda su sentido y, por tanto, se debe mantener una coherencia en la forma que se ofuscan los nombres sin alterar la redacción original: todas las variaciones del nombre de un participante se reemplazan de la misma forma. Es una tarea costosa, tanto en tiempo como en recursos humanos y, además, propensa a errores. Se cuenta con ejemplos de sentencias con errores en su anonimización, en los que se expone información personal o la identidad de los participantes.

En la figura 1.1 se muestra un ejemplo de una sentencia previo a ser anonimizada con etiquetas que indican el texto por el que los nombres serán reemplazados. La figura 1.2 ya cuenta con los nombres reemplazados. Este es el proceso que se lleva a cabo manualmente.

Los procesos automáticos permitirían asistir los procesos de anonimización

[1] https://bjn.poderjudicial.gub.uy

... Ahora bien, **AA** Julio admite expresamente en sus declaraciones que realizó esa negociación a pesar de no contar con ningún tipo de documento que lo habilitara a efectuarla, por lo que en buen romance no existe prueba alguna de la alegada permuta con el denunciante.- B) En segundo término, no hay duda que la camioneta era propiedad de **BB** Fariña Martinez ya que ambos así lo expresan, pero además coinciden en que con ella **AA** Julio Villar realizó una mudanza a la ciudad de San Carlos, precisamente el día en que **BB** Martinez expresa se la prestó la camioneta con dicha finalidad...

Figura 1.1: Extracto de sentencia con dos participantes, ambos son referenciados de dos formas distintas. Las etiquetas representan los nombres con los que serán anonimizados. Fragmento de la sentencia número 284/2011[3].

... Ahora bien, AA admite expresamente en sus declaraciones que realizó esa negociación a pesar de no contar con ningún tipo de documento que lo habilitara a efectuarla, por lo que en buen romance no existe prueba alguna de la alegada permuta con el denunciante.- B) En segundo término, no hay duda que la camioneta era propiedad de BB ya que ambos así lo expresan, pero además coinciden en que con ella AA realizó una mudanza a la ciudad de San Carlos, precisamente el día en que BB expresa se la prestó la camioneta con dicha finalidad...

Figura 1.2: Sentencia luego de anonimizar los participantes.

manuales, disminuyendo la probabilidad de errores y sus costos. Mediante el reconocimiento y vinculación de entidades es posible crear herramientas que dada una sentencia, identifiquen a todos sus participantes y sugieran cómo anonimizarlos. La participación humana se vería reducida únicamente a verificar la solución y corregir lo que sea necesario.

Existen dos trabajos previos llevados a cabo por Garat y Wonserver que dieron origen a este proyecto. El primero “Towards De-identification of Legal Texts”[1] recopila un corpus de sentencias de la BJN agregando etiquetas para identificar a sus participantes y explora el desempeño de herramientas de procesamiento de lenguaje natural en tareas como el reconocimiento de entidades nombradas (NER) y la desidentificación de información confidencial. En el segundo trabajo, “Automatic Curation of Court Documents: Anonymizing Personal Data”[2], se reentrena el modulo de NER de Spacy obteniendo un F1

micro de 90.21 % y se utilizan técnicas de agrupamiento para la vinculación de entidades que resultan en un ARI de 95.95 %.

Para automatizar el proceso de anonimización es necesario capturar características semánticas propias de las sentencias, que se diferencian de otros textos dado su dominio y el uso de una jerga jurídica específica. Una herramienta que permite la extracción de las características de forma automática es el aprendizaje profundo, un campo dentro del aprendizaje automático que utiliza redes neuronales.

El objetivo de este proyecto es aplicar aprendizaje profundo basado en *Transformers* para el reconocimiento de entidades y explorar el uso de redes siamesas para la vinculación de entidades, y así evaluar mediante el desarrollo de una solución cómo se comparan los resultados con los trabajos previos.

En particular, en este proyecto se propone:

- Analizar el conjunto de sentencias anotadas en trabajos previos y determinar su composición.
- Estudiar la literatura existente para conocer aproximaciones a los problemas de desidentificación de datos.
- Estudiar la literatura sobre arquitecturas basadas en redes neuronales profundas para el problema.
- Implementar una solución utilizando lo estudiado.
- Analizar los resultados y compararlos con soluciones previas.

Para la aplicación de aprendizaje profundo, se cuenta con un corpus proveniente de la BJN que cuenta con sentencias en sus versiones originales y anonimizadas previamente anotadas[1]. En este trabajo se analiza el corpus, observando las etiquetas, su distribución e identificando su utilidad para la generación de un proceso automático. Para esta finalidad se determinan nuevas etiquetas que facilitan la identificación de participantes, lo cuál requiere un reetiquetado del conjunto.

Para distinguir soluciones posibles basadas en aprendizaje profundo se estudia la literatura sobre arquitecturas de redes neuronales profundas aplicadas a problemas de anonimización de datos personales. Esto deriva en el estudio de reconocimiento de entidades y su vinculación.

1.1. Organización del documento

En el marco teórico (capítulo 2) se introduce a la Base de Jurisprudencia Nacional, y se describen los problemas de reconocimiento de entidades nombradas y vinculación de entidades. Luego se mencionan arquitecturas basadas en

redes neuronales y herramientas frecuentemente utilizadas.

En el capítulo 3 se analizan los datos contenidos en el corpus de la BJN y se describe el proceso seguido para adecuar las anotaciones existentes al contexto de este proyecto, también se presentan las características principales de este.

El capítulo 4 introduce la solución implementada para el reconocimiento de entidades nombradas. Se evalúan distintas alternativas usando *Transformers* con el fin de mejorar los resultados obtenidos, y se comparan a resultados de soluciones existentes. Además se agregan ejemplos resultantes de la ejecución de la solución.

En vinculación de entidades (capítulo 5) se presenta una prueba de concepto utilizando redes siamesas y aglomeración jerárquica que vincula las entidades reconocidas para mantener la coherencia una vez que la sentencia sea anonimizada. Al igual que en el capítulo anterior se exponen resultados y ejemplos de su ejecución.

Por último, el capítulo 6 concluye el proyecto y presenta posibles puntos para continuar el trabajo realizado.

Capítulo 2

Marco teórico

En este capítulo se establecen los antecedentes del proyecto. Se introducen a la Base de Jurisprudencia Nacional (BJN), y a los dos temas principales de este proyecto: reconocimiento de entidades nombradas y vinculación de entidades, proporcionando una idea general del estado del arte de ambos. Además se incluye una breve sección introductoria a las herramientas utilizadas para resolver los temas mencionados.

2.1. Base de Jurisprudencia Nacional

La BJN es una base de datos mantenida por el Poder Judicial uruguayo que recopila sentencias judiciales dictadas anteriormente por Tribunales de Apelaciones, Juzgados de Primera Instancia y la Suprema Corte de Justicia. Esta base es creada como parte del proceso de modernización del Poder Judicial con el objetivo de democratizar el acceso a información judicial. Los usuarios pueden acceder a los fallos judiciales a través de su portal[1] y utilizar filtros avanzados para consultarlos.

Existen dos versiones de la BJN, la pública y la privada. La versión privada, de uso interno del Poder Judicial, permite acceso a la versiones originales de las sentencias, mientras que la versión pública es de libre acceso a través de la web, y contiene solo versiones editadas de las sentencias. Los documentos son editados con el fin de no revelar información de caracter privado o sensible, proceso al que se lo denomina anonimización.

Todas las sentencias accesibles de la BJN pública se exponen siguiendo la misma estructura: cuentan con su texto anonimizado y además información accesoria a la sentencia que resulta de interés, como la fecha, su número y el tipo

[1] https://bjn.poderjudicial.gub.uy

entre otras. Esta información adicional (figura 2.1) no se utiliza en el contexto de este proyecto pero podría ser de gran utilidad para trabajo a futuro, por ejemplo, al construir una lista de referencia con todos los ministros participantes en sentencias legales.

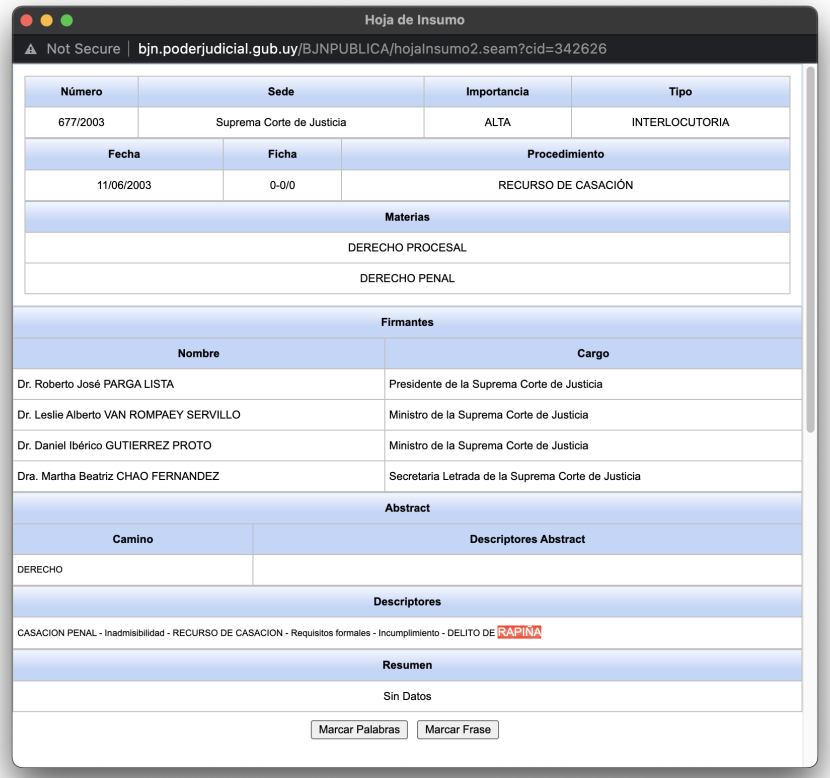


Figura 2.1: BJN: Información adicional disponible en una sentencia

La forma de acceder a las sentencias almacenadas en la BJN es a través de consultas. Los usuarios pueden elegir hacer una consulta simple u otra más avanzada denominada selectiva. La consulta simple permite hallar sentencias mediante una búsqueda de texto. La búsqueda puede ser exacta, aproximada, o es posible requerir que alguna o todas las palabras de la consulta coincidan en la sentencia. Las consultas avanzadas en cambio permiten un mayor control sobre los parámetros de la búsqueda. Además de filtrar por el texto contenido dentro de la sentencia, es posible filtrar por su información accesoria, como lo son el número de sentencia y los ministros firmantes y discordes. La figura 2.2 ejemplifica todos los filtros disponibles.

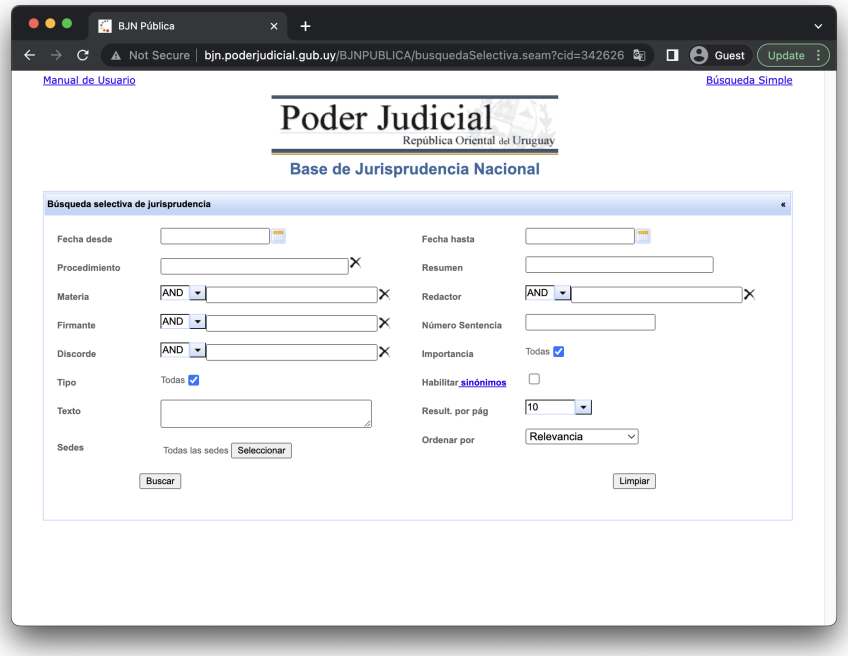


Figura 2.2: BJN: Filtros disponibles en la consulta selectiva del portal

Se cuenta con un corpus proveniente de la BJN formado por 79.000 sentencias legales de las cuales 17.000 se encuentran manualmente anonimizadas. No solo es necesario modificar los nombres reemplazándolos, también en algunas ocasiones se requiere editar la narrativa. Dado que el proceso es manual, no siempre es exacto y pueden cometerse errores.

En el contexto de este proyecto se trabaja con un subconjunto de 999 sentencias previamente etiquetadas y analizadas por Diego Garat y Dina Wonsever[1]. En su trabajo se centran en los actores de los eventos, ignorando ministros y abogados. Tampoco se marcaron otro tipo de entidades como, por ejemplo, ubicaciones y organizaciones.

Las sentencias se encuentran almacenadas en archivos de texto plano utilizando un archivo para la versión anonimizada y otro para la versión sin anonimizar. En el capítulo 3 se analiza en detalle este subconjunto de sentencias y se presenta el proceso seguido para generar el corpus.

2.2. Redes Neuronales

En esta sección se presentan distintos tipos de redes neuronales mencionadas o utilizadas en este trabajo. Se hace énfasis en las redes basadas en *Transformers* ya que son utilizadas para resolver el problema de reconocimiento de entidades.

2.2.1. Redes Neuronales Recurrentes

Las **redes neuronales recurrentes** (*RNN*)[4] contienen un ciclo en sus conexiones, lo que permite la retroalimentación entre las neuronas dentro de las capas. Esto hace que puedan utilizar sus propias salidas como entradas, utilizando sus estados previos, además de recibir la entrada desde la capa anterior.

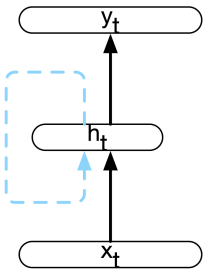


Figura 2.3: Red neuronal recurrente simple. Extraída de "Speech and Language Processing"[4].

Las redes **LSTM** tienen la ventaja de contar con la capacidad de recordar información de estados previos durante largos períodos de tiempo, resultando en un mejor procesamiento de secuencias de datos. Durante su entrenamiento, se gestiona su «memoria» eliminando información que ya no se necesite y agregando aquella que puede que sea necesaria más adelante.

Los modelos propuestos por los autores mencionados incluyen redes *LSTM*, redes *LSTM* bidireccionales (*BI-LSTM*), redes *LSTM* con una capa *CRF* (*LSTM-CRF*) y redes *LSTM* bidireccionales con una capa *CRF* (*BILSTM-CRF*). Huang et al. explican que las redes *LSTM* son similares a las *RNN*, excepto que se reemplazan las actualizaciones de la capa oculta por celdas de memoria especialmente diseñadas. Como resultado, pueden ser mejores para encontrar y explotar dependencias de largo alcance en los datos.

Las redes *LSTM* bidireccionales son una extensión de las *LSTM*. Permiten hacer uso eficiente de las características pasadas (a través de estados hacia adelante) y características futuras (a través de estados hacia atrás) por un período de tiempo específico. Se duplican las capas, una recibe la secuencia original, y la otra la secuencia inversa.

En el reconocimiento de entidades se utilizan los **CRF** para calcular la

probabilidad condicional de valores en nodos de salida seleccionados, dados los valores en otros nodos de entrada seleccionados. Se basan en la idea es que el contexto que rodea al nombre se convierte en una buena evidencia al etiquetar otra aparición del mismo nombre en un contexto ambiguo diferente[5].

Se demostró que con el modelo LSTM bidireccional con una capa CRF se obtiene una mejor precisión de etiquetado que un solo modelo CRF con conjuntos de características idénticos.

2.2.2. Transformers

Los **mecanismos de atención** (*Attention Mechanisms* en inglés) son una tendencia reciente en aprendizaje profundo. Se basan en la atención visual de los humanos que centran el foco en una imagen con «alta resolución», mientras se percibe la imagen de alrededor en «baja resolución». En función de la entrada y de lo que se generó hasta el momento, el modelo "aprende" a qué otorgar su atención. Se basan en la estructura codificador-decodificador[6].

Transformers es un modelo basado únicamente en mecanismos de atención, más precisamente en autoatención (o *self-attention* en inglés). El modelo prescinde de las redes neuronales convolutivas o recurrentes dado su caracter secuencial, el cual acarrea problemas a la hora de paralelizar el entrenamiento. Este enfoque puro en atención permite que el modelo alcance resultados superiores a las alternativas descriptas anteriormente, con un menor costo de entrenamiento. *Self-attention* refiere a la capacidad del modelo de ver otras palabras en la secuencia de entrada mientras se codifica una palabra específica. Esto permite incluir en la codificación de una palabra parte de la representación de otras en la secuencia que resulten de interés. Se denomina *self-attention* dado que las palabras que componen la entrada interactúan entre sí[7].

Jakob Uszkoreit[8] explica como funciona el modelo *Transformers*: comienza generando representaciones iniciales, o *embeddings*, para cada palabra. Luego, usando *self-attention*, agrega información de todas las otras palabras, generando una nueva representación por palabra, informada por todo el contexto. Este paso se repite varias veces en paralelo para todas las palabras de la secuencia, generando sucesivamente nuevas representaciones. La figura 2.4 muestra su arquitectura.

Algunas de sus ventajas incluyen: las salidas de las capas se pueden calcular en paralelo, en lugar de serialmente como una *RNN*; los elementos distantes pueden afectar la salida de los demás sin tener que pasar por muchos pasos en *RNN* o capas de convolución; y puede aprender dependencias de largo alcance.[2]

En el 2018 Google presenta *BERT* (*Bidirectional Encoder Representations from Transformers*) al público. Es un modelo de representación de lenguajes

[2]Extraído de: *Tensorflow: Transformer model for language understanding* (`https://www.tensorflow.org/text/tutorials/transformer`).

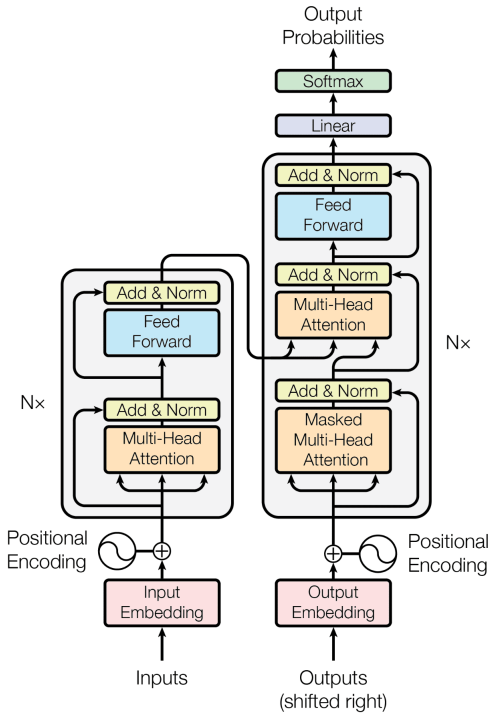


Figura 2.4: Arquitectura de un *transformer*. Extraído de "Attention is all you need"[7].

diseñado para ser preentrenado en texto sin etiquetar y luego fácilmente ajustado a una gran cantidad de tareas mediante el uso de una capa adicional de salida.

El modelo es preentrenado utilizando 3.300 millones de palabras obtenidas de texto en inglés proveniente de Wikipedia y BookCorpus. Su arquitectura consiste de bloques *transformers*, cabezales de auto atención y una capa oculta. Para representar la entrada se utilizan *embeddings* subpalabra Wordpiece[9] con un vocabulario de 30.000 tokens.

Estos *embeddings* dividen secuencias de caracteres en unidades subpalabra, buscando ser un punto medio entre los tokenizadores a nivel de palabra y los de carácter. Utiliza, entonces, un vocabulario de menor tamaño que los primeros y a su vez secuencias más cortas con mayor significado que los últimos. Los tokens utilizados dependen fuertemente del corpus con el que se entrena, ya que se busca generar una cantidad mínima de unidades al segmentar las entradas del modelo. Durante el proceso de tokenización, aquellas palabras que no se encuentren en el vocabulario serán divididas en subpalabras, buscando los prefijos más largos que sí lo estén[9].

Tomando el siguiente ejemplo extraído de una sentencia: *Como lo indicó Vescovi*, el tokenizador WordPiece que utiliza el modelo BETO produce la siguiente salida: *'Como', 'lo', 'indic', '##o', 'Ves', '##co', '##vi'*. En este caso tanto *indicó* como *Vescovi* no forman parte del vocabulario y por lo tanto son divididas en subpalabras. *indic* es un prefijo de varias palabras del español — índice, indicio, indicar y sus conjugaciones — y *ves* una palabra, por ende se encuentran en el vocabulario y no son divididas. El resto de los tokens representan la secuencia más larga presente en el vocabulario.

El proceso no supervisado de preentrenado de BERT consiste en enmascarar de forma aleatoria el 15 % de los tokens wordpiece y predecir los tokens ocultos haciendo uso del resto, obteniendo un modelo de lenguaje bidireccional. Además, se utiliza una segunda tarea donde el modelo predice si una oración le sigue a otra con el fin de aprender relaciones entre sentencias.

Se presentan resultados utilizando dos arquitecturas: $BERT_{Base}$ y $BERT_{Large}$ [10][11]. En el cuadro 2.1 se compara su composición:

Arquitectura	B. *transformers*	Capa oculta	C. de autoatención	# parámetros
$BERT_{Base}$	12	768	12	110M
$BERT_{Large}$	24	1024	16	340M

Cuadro 2.1: Comparación de arquitecturas de BERT

Integrantes de *Google AI Language*[11] aplicaron *BERT* ajustado (*fine-tuned* en inglés) a tareas de PLN, donde se agrega una capa de clasificación simple al modelo previamente entrenado, y todos los parámetros se ajustan o afinan conjuntamente en una tarea posterior.

En el marco de reconocimiento de entidades nombradas, compararon el enfoque basado en características junto con *BERT* con el enfoque nombrado previamente (*BERT fine-tuned*). El enfoque basado en características tiene ciertas ventajas: no todas las tareas se pueden representar fácilmente mediante una arquitectura de codificador *Transformer*, y computacionalmente puede ser mejor precalcular una vez una representación costosa de los datos de entrenamiento, y luego ejecutar múltiples experimentos con modelos que demanden menos recursos computacionales además de esta representación. Para el enfoque basado en características no se ajustó ningún parámetro de *BERT*. Los *embeddings* contextuales se utilizan como entrada para un *BI-LSTM* antes de la capa de clasificación.

Sus resultados mostraron que *BERT* se desempeña de manera competitiva con otros métodos del estado del arte y es eficaz tanto para el ajuste fino como para los enfoques basados en características.

Tarea	Tipo de Tarea	F_1 BERT	Mejor previo
SQUAD[12]	Contestado de preguntas	**0.931**	0.912
SWAG[13]	Inferencia de sentido común	**0.863**	0.78
GLUE[14]	Entendimiento lenguaje natural	**0.796**	0.751
CoNLL-EN 2003[15]	Reconocimiento de entidades	0.924	**0.931**

Cuadro 2.2: Resultados alcanzados por BERT en distintas tareas[3].

Desde su lanzamiento, han surgido distintos modelos basados en BERT; que modifican los hiperparámetros y métodos de entrenamiento usados por el modelo *BERT* original buscando mejorar su desempeño para determinada tarea, enfocándose en métricas y otras áreas cómo lo son el costo computacional o el tiempo de entrenamiento.

Las cuatro variedades expuestas a continuación resultan de particular interés para el reconocimiento de entidades nombradas en documentos en Español y además toman distintos enfoques a la hora de mejorar los resultados ofrecidos por *BERT*.

BERT Multilingual fue introducido por los autores originales de *BERT*, se basa en la arquitectura $BERT_{BASE}$ y es entrenado utilizando la misma cantidad de parámetros —110 millones— pero con datos provenientes de 104 lenguajes distintos. El corpus de entrenamiento de cada lenguaje se construyó traduciendo el conjunto original de entrenamiento de forma automática (corpus XNLI), sin personas involucradas. Cabe destacar que es imposible saber qué tanto afecta al desempeño la calidad del traductor automático.

BETO[16] es un modelo derivado de $BERT_{BASE}$ entrenado sobre una compilación de corpus en español[16] obtenidos de distintas fuentes —ocupando 4GB en total— utilizando, al igual que *BERT*, la técnica de enmascaramiento de palabras enteras, y obtiene resultados superiores a *BERT Multilingual*. Utiliza la misma arquitectura que $BERT_{BASE}$ y aumenta la cantidad de tokens del vocabulario a 31.002. Se cuenta con dos versiones del modelo: una distingue entre mayúsculas mientras que la otra no.

TinyBERT[17] es un modelo creado para acelerar el tiempo de ejecución y reducir el tamaño del modelo BERT original. Para lograr esto Xiaoqi Jiao et al. proponen un método de destilación de conocimiento para modelos basados en *Transformers*, logrando un desempeño similar con un menor costo computacional. El modelo utiliza una arquitectura estudiante-profesor donde una red de menor tamaño —el estudiante, TinyBERT— aprende a comportarse como una red de mayor tamaño —el profesor, $BERT_{BASE}$.

[3]Extraído de: `https://huggingface.co/blog/bert-101`

TinyBERT cuenta con cuatro bloques *Transformers*, una capa oculta de tamaño 312 y doce cabezales de atención. Resultando en 14,5 millones de parámetros, una reducción del 86 % respecto a $BERT_{BASE}$. Utilizando esta arquitectura el tiempo de inferencia original se reduce en un $\sim 90\,\%$.

XLM-RoBERTa[18].*RoBERTa*[19] es una reimplementación de *BERT* que modifica hiperparámetros clave del modelo y utiliza una mayor cantidad de datos en su entrenamiento para obtener mejores resultados, manteniendo la arquitectura original. El artículo sugiere que *BERT* no fue entrenado correctamente por los autores, produciendo un desempeño menor a su potencial. Los cambios propuestos por RoBERTa son los siguientes:

- Enmascarado dinámico de tokens. En vez de definir un enmascarado estático de la entrada durante la etapa de preprocesamiento, se enmascaran dinámicamente las secuencias utilizadas como entrada del modelo.
- Eliminación de la tarea secundaria en la que el modelo predice si dos oraciones son contiguas.
- Entrenamiento con lotes de mayor tamaño. Aumentar los lotes de 256 a 2048 oraciones permite mejorar la perplejidad del conjunto de datos y la paralelización del entrenamiento.
- Mayor tamaño de vocabulario, utilizando una representación más eficiente de las subpalabras.
- Aumento de la cantidad de datos de entrenamiento de 16GB a 160GB. Además del corpus original compuesto por *Wikipedia* y *Book Corpus*, se suman: *CC-News*, *OpenWebText* y *STORIES*.

XLM-RoBERTa en particular es un modelo multilenguaje introducido posteriormente de 550 millones de parámetros entrenado sobre un total de 2.5TB de datos que cuenta con la siguiente arquitectura: 24 bloques transformers, capa oculta de largo 1.024, 16 cabezales de atención y un vocabulario de doscientos cincuenta mil tokens. Este introduce el uso de *Sentencepiece*[20], un tokenizador agnóstico al lenguaje de entrada y que ofrece mejor soporte a lenguajes que no utilizan espacios para separar palabras. Esto permite que el modelo logre una mejoría general sobre los resultados obtenidos por BERT multilingual (en particular, $+2{,}4\,\%$ F_1 en NER).

Existen además otras arquitecturas para *NER* basadas en *Transformers* como por ejemplo: *TENER*[21] que adopta un codificador *Transformer* adaptado para modelar las características a nivel de carácter y a nivel de palabra. Los autores experimentaron con seis conjuntos de datos de *NER* y sus resultados muestran que *TENER* incluso logra un rendimiento superior a los modelos basados en *BI-LSTM* predominantes (al 10 de noviembre del 2019).

El artículo que propone a *TENER* identifica una mejora sobre la arquitectura

Transformer. En la arquitectura original se utilizan embeddings de posición que solo almacenan la distancia entre tokens pero no su direccionalidad. Los embeddings saben que tan lejos un token se encuentra de otro, pero no si aparece antes o después. Ambas propiedades son es muy importantes en *NER*. Dado que una entidad es un tramo continuo de palabras, conocer la clasificación del token anterior o siguiente puede ayudar a identificar la etiqueta del token a clasificar.

Entonces, TENER propone la codificación posicional relativa, en lugar de la codificación posicional absoluta.

TENER alcanza resultados de 91,45 y 88,25 en ConLL2003 y OntoNotes 5.0, mientras que otros modelos BERT obtienen valores de 93,74 y 92,07 respectivamente[22][23].

Como los modelos basados en *BERT* logran mejores resultados —según su F_1—, se enfoca la investigación en ellos.

2.2.3. Redes Neuronales Siamesas

Se les llama redes siamesas a aquellas formadas por dos o más subredes idénticas —que cuentan con la misma arquitectura y pesos— que se unen en su salida. Fueron introducidas por Bromley et al en 1993[24] y dada su composición, son comúnmente utilizadas para medir la distancia o similitud entre sus entradas[25]. Esto se logra proveyendo cada subred con una de las entradas para su procesamiento, comparando luego los vectores de características resultantes mediante el uso de una función distancia o métrica (como por ejemplo: Manhattan[4], coseno[4] o tripletas[26]). La figura 2.5 ejemplifica este proceso.

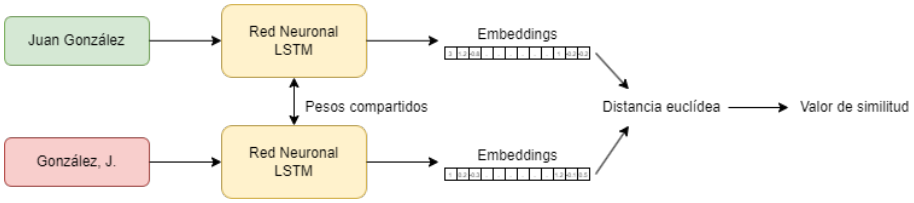


Figura 2.5: Diagrama de una red siamesa con subredes LSTM y distancia euclídea.

Mediante el proceso de entrenamiento, las redes aprenden a representar las entradas en un espacio de embeddings que logra capturar sus características principales y las relaciones semánticas entre ellas[27]. La métrica de similitud se obtiene de calcular distancias o diferencias entre vectores de este espacio. Generalizar la red a nuevas entradas es sencillo ya que basta con proyectar las entradas en el espacio aprendido, no es necesario reentrenar la red para que se adapte a nuevas clases de datos[28].

Es importante notar que los pesos pueden compartirse o no entre las subredes

dependiendo de las propiedades buscadas. En el contexto de este informe es de interés que las subredes sean capaces de priorizar las mismas características relevantes de las entradas y que la métrica aprendida sea simétrica, es decir, que el resultado de similitud para un par de nombres sea el mismo si se invierte su orden. Por lo tanto los pesos se actualizan de igual manera en ambas redes.

Estas redes son ampliamente usadas en tareas que requieran una comparación de elementos cómo la comparación de imágenes, el reconocimiento de objetos en imágenes o la detección de fraude.

2.3. Agrupamiento jerárquico

El agrupamiento jerárquico (o *Agglomerative Clustering* en inglés) es un algoritmo que mediante el uso de una métrica agrupa datos en conjuntos —denominados clústers— similares. La similitud entre datos se mide con una distancia de enlace (o *linkage distance*). El algoritmo funciona de forma recursiva construyendo un nuevo grupo en cada paso, hasta que todos los datos pertenecen al mismo grupo o se llega a una condición de parada. Este sigue un enfoque de abajo hacia arriba, cada observación comienza en su propio grupo y luego, los grupos se unen sucesivamente dependiendo de la distancia de enlace utilizada.

Las posibles condiciones de parada del algoritmo son: se alcanza la cantidad de grupos buscada o la distancia entre los grupos supera cierto umbral α, definido por los usuarios.

El código 2.1 ilustra el funcionamiento del algoritmo cuando se busca una cantidad fija de grupos N y se cuenta con una distancia de enlazado *distanciaEntreClusters* que permite medir la distancia entre ellos.

2.4. Reconocimiento de entidades nombradas

Uno de los objetivos de este proyecto consiste en la desidentificación de entidades nombradas. Para lograrlo, primero es necesario identificar las entidades mencionadas en los textos. Esto nos permitirá luego removerlas u ofuscarlas, dificultando o evitando la identificación de las entidades.

Todo aquello que se pueda referenciar con un nombre propio —una persona, una ubicación, una organización— se le denomina **entidad nombrada**. Aunque muchas veces a algunas expresiones temporales o numéricas también se las considera entidades nombradas dependiendo de la definición que se utilice[4][29].

La tarea de identificar y clasificar entidades nombradas en un texto se denomina reconocimiento de entidades nombradas, o *named entity recognition* (NER) en inglés. Se buscan expresiones lingüísticas que constituyan nombres propios, cada una de ellas es clasificada en una categoría previamente definida y se le coloca la etiqueta correspondiente a la categoría. Las cuatro etiquetas más común-

```
encontrarClustersCercanos(clusters) {
  distancia_minima = -1
  par = null, null

  for i in 0..largo(clusters):
    for j in i+1..largo(clusters):
      distancia =
        distanciaEntreClusters(clusters[i], clusters[j])
      if distancia < distancia_minima:
        distancia_minima = distancia
        par = clusters[i], clusters[j]

  return par
}

clustering_aglomerativo(datos) {
  clusters = []
  for dato in datos:
    clusters.insertar([dato])

  while largo(clusters) > N:
    cluster_x, cluster_y = encontrarClustersCercanos(clusters)
    nuevo_cluster = unir(cluster_x, cluster_y)

    clusters.remover(cluster_x)
    clusters.remover(cluster_y)
    clusters.insertar(nuevo_cluster)

  return clusters
}
```

Código 2.1: Pseudocódigo del algoritmo de agrupamiento jerárquico.

mente utilizadas son: PER (persona), LOC (ubicación), ORG (organización) y GPE (entidad geopolítica)[4]. En general las entidades con nombre son expresiones multipalabra y se encuentran en espacios contiguos de texto, convirtiendo *NER* en un problema secuencial. El reconocimiento de entidades nombradas es muy utilizado como el primer paso en la extracción de información[30].

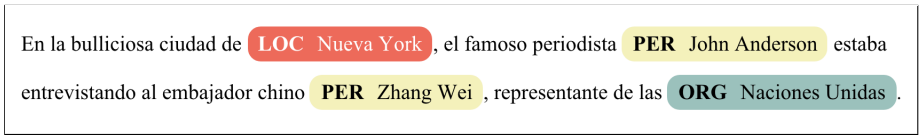


Figura 2.6: Ejemplo de aplicación de reconocimiento de entidades nombradas. Texto ficticio.

En la figura 2.6 se puede observar el resultado luego de aplicar las etiquetas PER, LOC y ORG. Para el caso de estudio además de identificar a las personas participantes se pretende clasificarlas según su rol dentro de la sentencia analizada.

A lo largo de esta sección se hace una introducción general a métodos utilizados para reconocer entidades, incluyendo aquellos que forman parte del estado del arte y son utilizados en las soluciones presentadas en este proyecto.

2.4.1. Enfoques tradicionales

Dentro de los enfoques tradicionales NER se encuentran aquellos: basados en reglas, aprendizaje no supervisado y aprendizaje supervisado basado en características (*features*).

Los sistemas NER basados en reglas se apoyan en reglas escritas a mano. Funcionan muy bien cuando el léxico es exhaustivo. En gran parte de los sistemas de este tipo, las reglas se definen específicamente para el dominio sobre el que se trabaja, por lo que no pueden ser transferidas a otro dominio. Además, suelen incluir análisis de estructuras propias del lenguaje (por ejemplo reglas gramaticales), pero no siempre resuelven correctamente todas las ambigüedades.

Los métodos de aprendizaje no supervisados parten de datos no etiquetados, por lo que no necesitan esfuerzo manual y se pueden aplicar a grandes cantidades de datos. Dentro de este enfoque se utilizan los sistemas basados en *clusters* ya que infieren las entidades nombradas en función de la similitud del contexto, para ello se utilizan recursos léxicos, los patrones léxicos y estadísticas calculadas. Consiste en la agrupación automática de datos, pudiendo ser en diferentes grados de granularidad (documentos, párrafos, oraciones, términos). Algunos de los algoritmos más conocidos de *clustering* son: *clustering* jerárquico, *clustering K-means*, y *clustering* probabilístico con modelos temáticos[31][32].

El *clustering* jerárquico intenta construir una jerarquía de grupos similar a la estructura de árbol. El *clustering k-means* agrupa objetos en k grupos, donde cada objeto de los datos es asignado al grupo que tenga el valor medio más cercano. En los modelos temáticos cada tema es una distribución de probabilidad sobre las palabras y el corpus es una mezcla de temas.

En el aprendizaje supervisado basado en características, dado un corpus anotado se utilizan algoritmos de aprendizaje automático para enseñar a un modelo a reconocer patrones similares en datos nuevos. La entrada implica una matriz donde cada columna es una característica diferente y cada fila es una observación. Algunos de los algoritmos de aprendizaje automático basados en características son: *Hidden Markov Models*, árboles de decisión, *Maximum Entropy Models*, *Support Vector Machines* y *Conditional Random Fields* (CRF). Una desventaja de estos métodos es que el etiquetado de datos no es sencillo y requiere tiempo. Para que funcione eficientemente es muy importante la correcta selección del corpus y su anotación, y normalmente se necesita de cierta experiencia en el dominio.

2.4.2. Reconocedores basados en redes neuronales profundas

El aprendizaje profundo (*deep learning* en inglés) es un campo dentro del aprendizaje automático basado en redes neuronales artificiales (también llamadas redes neuronales, en inglés *neural networks - NN*). Los modelos de aprendizaje profundo parten de datos sin procesar y capturan varias características semánticas complejas; generar reglas para capturarlas es difícil incluso para expertos.

Estas redes se componen de múltiples capas de procesamiento para aprender representaciones de datos con múltiples niveles de abstracción, de ahí el término profundo. Tal como exponen Yann LeCun, Yoshua Bengio y Geoffrey Hinton[33], cada capa transforma la representación en un nivel, llevándola a una representación más abstracta. Si es grande la composición de transformaciones, y por lo tanto la cantidad de capas, es posible aprender funciones muy complejas. En las tareas de clasificación, las capas superiores de representación amplifican los aspectos de la entrada que son importantes para la discriminación y suprimen las variaciones irrelevantes.

Zhiheng Huang, Wei Xu y Kai Yu[34] proponen una variedad de modelos basados en redes neuronales para la tarea de etiquetado secuencial, especialmente se basan en un modelo especial de redes recurrentes: *Long Short Term Memory* (LSTM).

Hyejin Cho y Hyunju Lee[35] proponen un sistema *NER* para el reconocimiento de entidades nombradas biomédicas mediante la incorporación de n-gramas con *BI-LSTM* y *CRF*, y lo comparan con varios enfoques de aprendizaje profundo. Su método mejora significativamente el rendimiento en tareas *NER*

biomédicas. Se ha demostrado que la información contextual conduce a mejoras significativas en el aprendizaje automático, es por este motivo que el sistema *NER* propuesto fue diseñado para tratar de manera más explícita la información contextual en el texto.

El hecho de que el enfoque previamente mencionado sea exitoso para el reconocimiento de entidades biomédicas es prometedor para el dominio de este proyecto, porque más allá de ser muy diferentes, cuentan con la característica en común de ser muy específicos.

Una arquitectura reciente de uso popular para el reconocimiento de entidades es *Transformers*. En particular, modelos basados en BERT. En la sección 2.2.2 se explica su funcionamiento.

Existen estudios que muestran que la aplicación de aprendizaje profundo para el reconocimiento de entidades nombradas produce buenos resultados.

Vikas Yadav y Steven Bethard[36] presentan un estudio de arquitecturas de redes neuronales profundas para NER y lo comparan con enfoques basados en ingeniería de características (*feature engineering* en inglés) y otros algoritmos de aprendizaje supervisado o semi-supervisado. Los sistemas NER considerados son: basados en conocimiento, aprendizaje débilmente supervisado (*bootstrapping*), supervisados basados en características y redes neuronales que infieren características. Estos últimos pueden ser categorizados según su representación de las palabras en una oración: arquitecturas a nivel de palabra, carácter, subpalabra, o combinados.

Los autores estudiaron que los sistemas con redes neuronales que infieren características superan a los sistemas basados en ingeniería de características. Además, los modelos híbridos palabra + carácter son generalmente superiores a los modelos basados en palabras y que los basados en caracteres. Esto significa que sus resultados utilizando aprendizaje profundo fueron mejores. Los modelos de redes neuronales generalmente superan a los modelos basados en ingeniería de características, incluso afirman que los resultados son mejores si se consideran los afijos (modelo palabra + carácter + afijo), ya que capturan información complementaria.

2.5. Vinculación de entidades

La sección anterior se corresponde con la primer parte del problema, la identificación de las entidades nombradas en el texto. Esta sección en cambio tratará con la parte restante, asignarle a cada entidad identificada la persona a la que hace referencia.

A la hora de anonimizar las sentencias es necesario reemplazar todas las menciones que se hacen de una persona en el texto por un nuevo nombre, de forma de mantener la historia presentada por la sentencia coherente sin simul-

táneamente divulgar información que permita identificar al actor original.

A modo de ejemplo, se tiene el siguiente texto (figura 2.7) en donde se identifican tres entidades: Pedro Pérez, Raúl Gomez y Pedro.

Se declara a Pedro Pérez como víctima del robo llevado a cabo por Raúl Goméz. Pedro se encontraba...

Figura 2.7: Ejemplo de sentencia posible con dos participantes.

Para anonimizar la sentencia correctamente, el modelo propuesto para agrupar referencias debe ser capaz de identificar que *Pedro Pérez* y *Pedro* hacen referencia a la misma persona, mientras que *Raúl Gomez* representa a una persona distinta (figura 2.8).

Se declara a Pedro Pérez como víctima del robo llevado a cabo por Raúl Goméz. Pedro se encontraba...

Figura 2.8: El modelo detecta dos participantes: Pedro Pérez y Raúl Gomez.

Para lograr esto, es necesario contar con un método o herramienta que permita clasificar las entidades por cercanía o similitud. Del ejemplo, el método debe identificar que Pedro Pérez y Pedro refieren a la misma entidad mientras que Raúl Gómez refiere a otra. Existen varias alternativas que resuelven este problema; en este trabajo se hace foco en aglomeración jerárquica (o clusterización jerárquica) en línea con el trabajo de Garat y Wonsever[2].

La aglomeración jerárquica es un algoritmo que agrupa instancias similares en clústers. En este caso, cada clúster representa una persona e instancias dentro de él son las distintas formas de referirse a esa persona. En la sección 2.6.4 se introduce la implementación del algoritmo a utilizar y una herramienta para su visualización.

2.6. Herramientas

A continuación se provee una introducción a herramientas utilizadas para resolver los problemas descritos anteriormente. La primer subsección se enfoca en utilidades para la extracción de información mientras que el resto en la implementación de redes neuronales.

2.6.1. Herramientas para recuperación de información

Existen herramientas para el procesamiento de lenguaje natural enfocadas en la extracción de información, comprensión y análisis del lenguaje natural. Algunas de las más utilizadas son Freeling[37], CoreNLP[38], Nltk[39] y Spacy[40].

Freeling para NER provee tres tipos de módulos: "basic", "bio" y "crf". Establece que "basic" es simple y rápido, y fácil de adaptar para su uso en nuevos idiomas, siempre que las mayúsculas sean la pista básica para la detección de entidades nombradas en el idioma de destino. El módulo "bio" se basa en algoritmos de aprendizaje automático, es más preciso pero más lento. "crf" se basa en el algoritmo de aprendizaje automático CRF, es más preciso, más rápido que "bio", pero no tanto como "basic".

CoreNLP usa modelos de secuencia de aprendizaje automático para etiquetar entidades, pero también puede utilizar componentes especializados basados en reglas. Usa una combinación de tres marcadores de secuencia CRF.

Nltk como técnica básica utiliza *Chunking*, es un proceso de extracción de frases del texto mediante la utilización de expresiones regulares y etiquetado *POS*.

Spacy presenta una estrategia de *word embedding* que utiliza características de subpalabras y *"Bloom"* embeddings [4][41], y una red neuronal convolucional profunda con conexiones residuales[42].

Aplicar estas herramientas preentrenadas es una opción para el reconocimiento de entidades nombradas[1], aunque no siempre logran adaptarse a nuevos dominios. En estos casos es necesario reentrenar las herramientas para mejorar su funcionamiento. Garat y Wonsever utilizan las herramientas mencionadas para ejecutar tareas como segmentación, tokenización[5] y reconocimiento de entidades nombradas a textos extraídos del corpus de la Base de Jurisprudencia Nacional.

Todas las alternativas descriptas anteriormente muestran problemas, se estima que se debe a que el corpus presenta características diferentes a los usualmente empleados para el entrenamiento de las herramientas probadas (periodísticos o científicos). En el caso del reconocimiento de entidades nombradas, muchos segmentos reconocidos están incompletos y los tipos asignados a las entidades son erróneos.

[4]Los embeddings "Bloom" hacen uso de una estructura de datos probabilística —"Bloom" filters— para representar los vectores de forma eficiente en memoria.

[5]**Tokenización** es el proceso mediante el cual se divide el texto original en unidades de menor tamaño llamadas *tokens*. Estos tokens luego son utilizados por los modelos para construir su vocabulario y procesar el texto.

2.6.2. HuggingFace

En 2019, luego de la explosión en popularidad del modelo *transformers*, en gran parte gracias a BERT, la compañía Huggingface lanzó oficialmente la librería de código abierto *Transformers*. **Huggingface Transformers** tiene como foco principal soportar modelos basados en arquitecturas *transformer* así como facilitar la distribución de corpus y modelos preentrenados[43]. Hoy en día es la librería más popular en lo que respecta a modelos *transfomer*, cuenta con 99.400 estrellas en Github[44] y más de 1.800 contribuidores.

La librería es mantenida por empleados de HuggingFace y acepta contribuciones de la comunidad a través de un repositorio en Github. Su código está compuesto principalmente de Python, con ciertos módulos de mayor carga computacional implementados en Rust para obtener mayor desempeño —como por ejemplo la librería tokenizers[6].

Además, la compañía cuenta con un sitio web centralizado (figura 2.9) que permite el fácil acceso a modelos preentrenados y conjuntos de datos para ser utilizados en la librería *transformers*. Es posible filtrar las opciones disponibles por nombre, lenguaje, cantidad de descargas y tarea de destino (NER, detección de objetos, clasificación de imágenes, etc). Los modelos y corpus disponibles son aportados tanto por contribuidores individuales como empresariales, contando con la presencia de empresas como OpenAI, Meta y Microsoft.

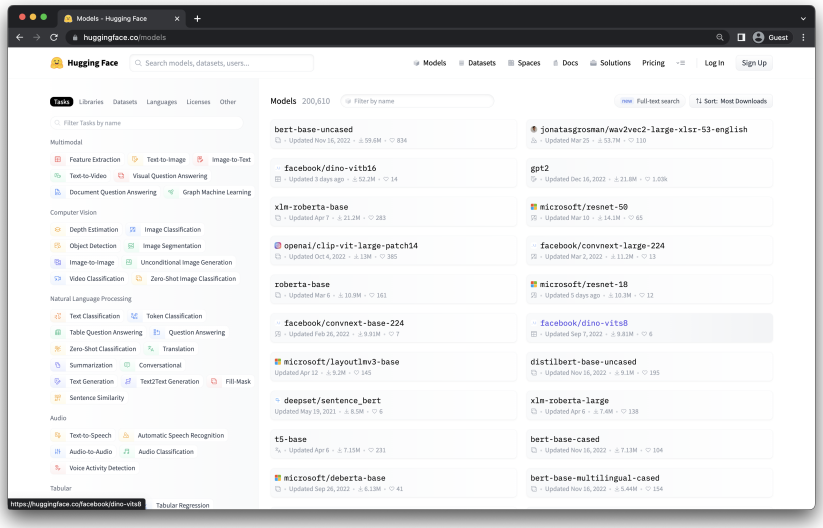


Figura 2.9: Hub de modelos de Huggingface[7]

[6] `https://github.com/huggingface/tokenizers`
[7] `https://huggingface.co/models`

En conjunto, esto permite instanciar un modelo, seleccionar un corpus y ajustar el modelo sobre este con gran facilidad. *Transformers* incluye abstracciones de alto nivel para el proceso de entrenamiento mediante su clase *Trainer*, la clase permite controlar hasta el más fino detalle de cómo se ejecuta el entrenamiento, y facilita tareas como: evaluar el modelo, pausar y resumir el entrenamiento, realizar búsquedas de hiperparámetros y otras.[8]

2.6.3. Keras

Keras[45] es una librería de código abierto orientada al desarrollo de modelos de aprendizaje profundo creada por François Chollet. Su misión principal es proveer a los usuarios con una API de alto nivel para el desarrollo de redes neuronales que resulte sencilla de usar, priorizando la experiencia de usuario sobre todo lo demás.

Actualmente se basa exclusivamente en la librería Tensorflow para compilar los modelos y realizar las computaciones necesarias; el soporte para otras librerías populares como PyTorch y JAX se encuentra en desarrollo. Estas librerías le permiten a Keras ejecutar modelos con muy alto desempeño, haciendo posible el uso de tarjetas de vídeo, entrenamiento distribuido y más. Es extremadamente popular y cuenta con una multitud de recursos de aprendizaje de gran calidad disponibles.

La API de Keras es extensa, se cuentan con clases y módulos que implementan la gran mayoría de elementos usados actualmente: redes neuronales, optimizadores, funciones de pérdida, métricas y demás.

Los modelos se pueden definir utilizando dos modelos distintos, el secuencial y el funcional. El primero permite definir modelos formados por una pila lineal de capas con exactamente un tensor de entrada y uno de salida. En todos los otros casos se recomienda utilizar el modelo funcional. Este permite más de una entrada y salida y soporta definiciones de modelos más complejos, mientras sean representables como un grafo acíclico dirigido. El modelo funcional es más flexible pero complejo de utilizar. En el cuadro 2.2 se incluye un ejemplo de la definición de un modelo usando la API secuencial.

Si bien la librería provee una API de alto nivel, no cuenta con modelos predefinidos y entrenados como es el caso de HuggingFace Transformers. Para entrenar un modelo es necesario definir su arquitectura —tipos de red a utilizar, funciones de activación, número de capas ocultas, etc— luego compilarlo, para así construir el objeto Python y funciones necesarias para correr el modelo, y por último entrenarlo para ajustar sus parámetros al corpus disponible.

[8] `https://huggingface.co/docs/transformers/main_classes/trainer`
[9] Ejemplo obtenido de Keras: https://keras.io/examples/vision/mnist_convnet/

```
model = keras.Sequential(
  [
    keras.Input(shape=input_shape),
    layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
    layers.MaxPooling2D(pool_size=(2, 2)),
    layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
    layers.MaxPooling2D(pool_size=(2, 2)),
    layers.Flatten(),
    layers.Dropout(0.5),
    layers.Dense(num_classes, activation="softmax"),
  ]
)
```

Código 2.2: Definición de red convolucional para reconocer dígitos en el conjunto MNIST.[9]

Keras Tuner

La librería **KerasTuner**[10][46] —parte del proyecto Keras— permite realizar búsquedas de hiperparámetros en modelos implementados con Keras. Esta sigue la misma filosofía de desarrollo que su librería padre buscando ser fácil de utilizar y a su vez proveer una herramienta muy versátil. Permite definir hiperparámetros, sus valores posibles (o un rango de valores); y por defecto cuenta con varios algoritmos de búsqueda como Hyperband[47], optimización Bayesiana y búsqueda aleatoria. Cabe destacar que busca ser una herramienta general y su uso no se limta a Keras, puede ser utilizada para ajustar modelos implementados con otras librerías como Scikit-Learn.

La API provista resulta sencilla para definir hiperparámetros y sus rangos de valores posibles. Cada uno de ellos debe ser provisto de un nombre, y existen funciones específicas para los distintos tipos primitivos —*Float*, *Int*, *Boolean*— y *Choice* para elegir un valor dentro de una lista predefinida (código 2.3).

2.6.4. Scikit-Learn

Scikit-Learn[48] es una librería de código abierto desarrollada en Python que alberga implementaciones de funciones, algoritmos y modelos relacionados al aprendizaje automático. Es una de las librerías más populares del área, cuenta con 55.600 estrellas en Github.

Se originó en 2007 como un proyecto de Google Summer of Code, desarrollada por David Cournapeau. En el 2010 tuvo su primer lanzamiento al público y se ha trabajado sobre ella desde ese momento. Actualmente es mantenida por un equipo de voluntarios y cuenta con compañías de alto renombre como sus donantes (Google, Microsoft, Hugging Face y más).

[10] `https://keras.io/keras_tuner/`

```
def model():
  ...
  epochs = hp.Choice("epochs", [100, 200, 300, 500])
  lstm_units = hp.Int(
    "lstm_units",
    min_value=128,
    max_value=1024,
    step=128
  )
  dropout_rate = hp.Float(
    "dropout_rate",
    min_value=0.1,
    max_value=0.5,
    step=0.1
  )
  ...
```

Código 2.3: Definición de hiperparámetros con KerasTuner.

La librería cuenta con una variedad de modelos y algoritmos implementados, así como herramientas para producir visualizaciones y manejo de conjuntos de datos. Las guías de usuarios son extensas y cuentan con un sinfín de ejemplos mostrando como usar la librería. En el contexto de la vinculación de entidades el algoritmo *AgglomerativeClustering* y la función *dendrogram* resultan de mayor interés. El primero permite agrupar los datos mientras que el otro permite visualizar el proceso de agrupación.

AgglomerativeClustering[11] implementa el algoritmo de agrupamiento jerárquico. El comportamiento del algoritmo es altamente configurable, entre las opciones más importantes, es posible elegir si se busca un número fijo de clústers o si el algoritmo debe encontrarlos. En caso de que el número de grupos a encontrar no sea fijo, se debe proveer un valor límite de distancia para poder formarlos. También, se pueden utilizar distintas estrategias de enlace:

- Enlazado máximo o completo: Se minimiza la distancia máxima entre los pares de observaciones de los grupos.
- Enlazado Promedio: Se minimiza la distancia promedio de los pares de observaciones de los grupos.
- Enlazado singular: Se minimiza la distancia mínima entre los pares de observaciones de los grupos.
- Ward: Se minimiza la suma entre las diferencias cuadradas de todos los grupos.

[11] https://scikit-learn.org/stable/modules/clustering.html#hierarchical-clustering

Un dendograma es un tipo de diagrama que permite ver la relación jerárquica entre las observaciones disponibles. En el caso de ser usado con algoritmos de agrupamiento, muestra el proceso de formado de los grupos y su estado final. Además, la altura del diagrama será representada por la distancia de enlazado entre los elementos.

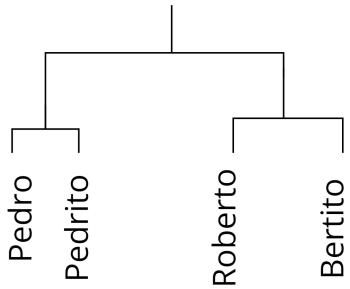


Figura 2.10: Ejemplo de agrupación de nombres usando un algoritmo de agrupamiento jerárquico

En el ejemplo (figura 2.10) se puede ver el funcionamiento del algoritmo, primero las observaciones —nombres en este caso— comienzan separadas. Luego, dado que la distancia de enlazado entre Pedro y Pedrito es la menor, se forma un grupo con los dos elementos. El próximo paso agrupa a Roberto y Bertito, ya que la distancia entre esos nombres es menor que la distancia con los otros. Por último, al alcanzar cierta distancia los grupos se unen formando un solo grupo. El umbral α mencionado anteriormente decide cuántos grupos se consideran.

Su valor traza una línea horizontal sobre el diagrama, la cantidad de líneas verticales que lo intersecten serán la cantidad de grupos finales. En este ejemplo pueden obtenerse cuatro variaciones finales dependiendo de como se defina el umbral. Si se le asigna un valor bajo se obtienen cuatro grupos, uno con cada nombre. Existe otro valor en el que se retornan tres grupos, {Pedro, Pedrito}, {Roberto} y {Bertito}. Un valor más alto obtiene dos grupos {Pedro, Pedrito} y {Roberto, Bertito}. Mientras que el valor más alto obtiene un grupo solo con los cuatro elementos.

Capítulo 3

Conjunto de datos

En este capítulo se introduce el corpus utilizado en este proyecto, se explica la razón detrás de las etiquetas definidas y se provee una breve introducción a la herramienta utilizada para anotar. Por último, se analiza la distribución de las distintas etiquetas definidas.

Se parte de 999 sentencias judiciales previamente anotadas en el artículo “Towards De-identification of Legal Texts"[1], de las cuales se toman 353 y se las etiqueta manualmente utilizando BRAT. Se obtiene así un corpus especialmente anotado para los intereses de este informe.

Nota: Para mantener en el anonimato la identidad de los participantes, todos los fragmentos de sentencias visibles en el informe fueron modificados reemplazando los nombres originales por otros generados aleatoriamente. Por más información en el anexo A se detalla cómo se realiza este procedimiento.

3.1. Anotaciones

En general, las sentencias judiciales siguen un mismo patrón de redacción: comienzan con la fecha y el juzgado o tribunal a cargo de la sentencia, se listan los ministros redactores y luego se hace una recolección de los hechos según los distintos participantes. Siguen con una interpretación del precedente judicial existente, y por último, se incluyen tanto la sentencia dictada, como los ministros firmantes y discordes.

Hay, entonces, varias personas presentes con distintos grados de participación. Los ministros, abogados, o personas citadas para establecer precedente son de carácter público y por lo tanto no se anonimizan; las únicas que resultan de interés anonimizar son los participantes y testigos afectados por la sentencia.

Las sentencias con las que se construyó el corpus original fueron anotadas utilizando Spacy para generar las etiquetas de forma automática y posteriormente se corroboraron y editaron manualmente (en caso de ser necesario). Solo se etiquetan participantes (de interés a anonimizar), con el siguiente formato: AA, BB, ..., YY y ZZ. Cada etiqueta representa un participante distinto dentro de una sentencia y a las múltiples referencias a un mismo participante se les asigna la misma etiqueta[1].

Debido al tiempo que insume el etiquetado manual, de las 999 sentencias revisadas solo 353 sentencias se volvieron a anotar para la realización de este proyecto, adaptándolas al nuevo problema a resolver. Se verificaron las anotaciones existentes y se agregaron nuevas etiquetas mencionadas en el cuadro 3.1 para facilitar el futuro entrenamiento del modelo.

Etiqueta	**Objetivo**	**Afectados**
P	Persona participante	Delincuentes, testigos, víctimas
NP	Persona no participante	Abogados, jueces, ministros
REF	Doctrino referenciado	Autores
LOC	Ubicación	Calles, ciudades, localidades, países

Cuadro 3.1: Etiquetas utilizadas por el conjunto.

Se considera participante a una persona involucrada directamente en los hechos de la sentencia, estos pueden ser delincuentes, testigos o víctimas y son las entidades a anonimizar. En cambio las no participantes son aquellas involucradas de forma indirecta: ministros y abogados. Las etiquetas P, NP y REF corresponden a personas, es decir, en un sistema de etiquetado clásico la etiqueta PER les sería asignada. Las últimas dos no fueron etiquetadas en el corpus original.

A los efectos de este proyecto agregar nuevas etiquetas es de utilidad para que el modelo sea capaz de distinguir entre no participantes, doctrinos, ubicaciones con nombres de personas y participantes, ya que solo los últimos deben ser anonimizados. La figura 3.1 muestra un ejemplo del etiquetado de participantes en un fragmento de sentencia.

3.2. Criterios de etiquetado

Durante la anotación se debieron tomar decisiones respecto al alcance de cada etiqueta sobre todo en casos donde, por ejemplo, es difícil clasificar una instancia porque podría pertencer a varias clases distintas.

Respecto a las etiquetas PER, se decide no anotar organizaciones y títulos honoríficos (Dr., Prof., Sra.) de las personas involucradas porque no es algo que

Figura 3.1: Fragmento de la sentencia número 640/2012[3].

sea de interés anonimizar (ejemplo en la figura 3.2).

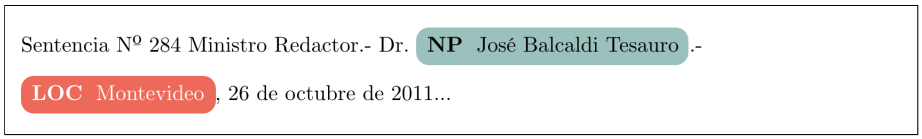


Figura 3.2: Ejemplo con anotación de no participante. Fragmento de la sentencia número 284/2011[3].

Es muy frecuente que en las sentencias se mencionen a los juzgados, y para identificarlos se utiliza el nombre de la ciudad en el que se encuentran. En algunos ejemplos como el de la figura 3.3 se observa que no forman parte de una anotación. Esto se debe a que «Juzgado Letrado de Primera Instancia de Paso de los Toros» corresponde a una organización, y como se mencionó anteriormente, las organizaciones no son de interés en el problema a resolver. Se optó por que la red no identifique «Paso de los Toros» como LOC ya que al no haber sido anotado en ninguna de las ocurrencias de similares características como tal, se espera que bajo contextos similares no lo sea. Lo mismo sucede con los jueces no participantes, dado que normalmente se indica a qué juzgado representan (ejemplo en la figura 3.4).

Se agregó la etiqueta LOC porque muchas ubicaciones tienen nombre de personas lo que podría ser confuso a la hora de entrenar. Un ejemplo de este caso es el fragmento de sentencia de la figura 3.5 que menciona a la calle "Santiago Vázquez".

En el caso de intersección de calles, cada calle fue etiquetada como LOC (ejemplo en la figura 3.5), y en caso de direcciones con número de puerta, el nombre de la calle y el número de puerta juntos corresponden a una LOC (ejemplo en la figura 3.6).

... RESULTANDO QUE: I. En los autos que por pensión alimenticia promoviera **AA** Julio Villar ante el Juzgado Letrado de Primera Instancia de Paso de los Toros, al evacuar el traslado conferido el demandado, interpuso excepción de inconstitucionalidad del Art. 183 inc. 1 del Código Civil, por considerar que el mismo vulneraba el principio de igualdad consagrado en el artículo 8 de la Constitución...

Figura 3.3: Ejemplo con juzgado. Fragmento de la sentencia número 4.632/2011[3].

... dictada por la Sra. Juez Letrado de Primera Instancia en lo Penal de San Carlos Dra. **NP** María Helena Mainard, y;...

Figura 3.4: Ejemplo con juez. Fragmento de la sentencia número 313/2011[3].

Muchos de los textos anotados manualmente generaron dudas respecto a cuáles etiquetas utilizar, debido a que se utilizan términos técnicos o poco usuales para personas ajenas al rubro judicial.

Entre las sentencias recibidas se cuenta con versiones anonimizadas manualmente por la BJN. Debido a que los no participantes se encuentran sin anonimizar, ante la duda de etiquetar una persona como participante o no participante se utilizaron estas sentencias de guía. Un ejemplo para el que fue de utilidad consultar la versión anonimizada de la BJN es la sentencia a la que corresponde el fragmento de la figura 3.7. Dos empresas («NEWSPAPER»y «COMPANY», nombres ficticios para no revelar su identidad) fueron etiquetadas como participantes porque a pesar de no ser personas, son participantes de la sentencia que de no haber sido anonimizados darían información sobre otros participantes involucrados, información que al ser pública podría revelar su identidad.

En la figura 3.8 hay otro ejemplo que generó confusión y podría también afectar los resultados del modelo; un abogado tiene asignado el rol de participante en lugar de no participante como suele suceder. Este caso también debió ser evaluado utilizando la sentencia previamente anonimizada.

Las sentencias judiciales son escritas manualmente, por lo que el error humano existe y es otra causante de dudas al anotar. En la figura 3.9 se observa como en tres fragmentos diferentes de una sentencia se hace referencia a un participante de cuatro formas diferentes. Por ejemplo, en la primera y en la tercer anotación BB el nombre finaliza con una letra diferente y en la primera el nombre «Maria» está abreviado.

...fue abordado por dos personas en la calle **LOC** Graña y **LOC** Santiago Vázquez (fs. 1) y mientras que uno de los atacantes...

Figura 3.5: Ejemplo de anotación LOC con nombre de persona. Fragmento de la sentencia número 233/2011[3].

... RESULTANDO QUE: 1.- El 9 de julio de 2011, fue incautada en un ex parador ubicado en Cno. Vecinal (**LOC** Delta del Tigre – **LOC** San José) la moto Marca Winner, Matrícula ARI 741; birrodado que era propiedad del Sr. **CC** Elias Fariña y le había sido hurtado en su casa en calle **LOC** Atira 5567 (**LOC** Montevideo) el día 24 de agosto de 2009...

Figura 3.6: Ejemplo de anotación de LOC con calle y número de puerta. Fragmento de la sentencia número 1.510/2012[3].

En caso de no contar con el archivo anonimizado por la BJN, se tomó la decisión de etiquetar como participante para de esta forma no se revelar la identidad de alguien indebido, por seguridad. Al dudar entre NP y REF, se anota como REF por ser menos relevante en cuanto a su grado de participación en la sentencia.

3.3. Herramienta de anotación

La herramienta utilizada para realizar las anotaciones es BRAT[49]. Es gratuita, simple de instalar y se utiliza desde el navegador.

Por cada archivo `.txt` recibido como entrada, BRAT genera un archivo `.ann` que contiene las anotaciones: etiqueta, inicio y fin. Los valores de inicio y fin son medidos en cantidad de caracteres. A continuación se muestra un ejemplo del contenido de un archivo `.ann`:

```
T1      AA 85 98        Villar , Julio
T2      AA 4508 4520    JULIO VILLAR
T3      AA 897 910      Julio Villar ,
T4      NP 414 438      Gabriel Ohanián Hagopián
T5      NP 522 537      Carlos A. Reyes
T6      NP 576 594      María Eugenia Elso
T7      BB 2889 2901    Gisela Segui
T8      CC 2985 2997    Elias Fariña
```

Sentencia Nº 172 Ministro Redactor: Dr. **NP** William Corujo Guardia .

LOC Montevideo , 27 de junio de 2012.- VISTA: Para Sentencia Definitiva de Segunda Instancia esta Causa " **AA** Julio Villar CON DIARIO " **BB** NEWSPAPER ."

...

3. Conferido legal traslado a la parte actora manifestó en prieta síntesis: a) " **BB** NEWSPAPER " publicó los dichos de **AA** Julio Villar en los mismos momentos y en las mismas notas en que lo vinculaban con la " **YY** COMPANY "; ...

Figura 3.7: Ejemplo de anonimización de dos organizaciones. Fragmento de la sentencia número 172/2012[3].

... 4) A fs. 135/138, el Dr. **FF** Nahia Grau , en calidad de curador -defensor de la niña **CC** Elias Fariña evacua el traslado del recurso interpuesto por la parte demandada, manifestando en síntesis que: la limitación de la patria potestad es consecuencia de conductas inapropiadas por parte de quienes la desempeñan e implica un juicio disvalioso para el padre imputado en ellas...

Figura 3.8: Ejemplo abogado etiquetado como participante. Fragmento de la sentencia número 168/2012[3].

Utilizando el archivo de anotaciones, BRAT sabe qué etiquetas marcar en el texto dado por el archivo `.txt` tal como se muestra en la figura 3.10.

Las etiquetas se definen en un archivo `annotation.conf`; y los estilos, tales como el color de fondo o color de borde, se definen en el archivo `visual.conf`.

Para etiquetar se debe seleccionar el texto y se abre una ventana como la de la figura 3.11 donde se listan las etiquetas definidas. Una vez escogida la etiqueta, se muestra del color correspondiente y automáticamente BRAT agrega al archivo `.ann` una nueva linea.

Algunas ventajas de utilizar una herramienta como BRAT son que: simplifica el proceso de etiquetado manual por no tener que considerar valores como los números de caracteres, previene errores al escribir el nombre de la etiqueta (por estar listadas), es cómoda para identificar visualmente etiquetas ya anotadas o verificar que hayan sido marcadas correctamente, los cambios se guardan automáticamente y al etiquetar alerta pintando de rojo etiquetas superpuestas.

... Para resolución en estos autos caratulados: "**BB** SEGUI, MA. GISELE en autos caratulados: "**AA** VILLAR, JULIO C/ **BB** SEGUI, MARIA – DIV. POR CAUSAL R/D Fa: 2-31656/2011" C/ SRA. JUEZA LETRADA DE FAMILIA DE 26º TURNO RECURSO DE QUEJA POR DENEGACION DE APELACION - Nº de Expediente 67-12/2012"

...

2.- A fs. 23 a 26 compareció la representante de la Sra. **BB** Gisela Segui, interponiendo recurso de queja contra la sentencia Nº 419/2012 por denegación del efecto suspensivo de recurso de apelación contra la providencia Nº 5413/2011,

...

II.- En la especie, en el proceso que por divorcio iniciara el Sr. **AA** Villar contra la Sra. **BB** Segui, ...

Figura 3.9: Ejemplo con cuatro formas diferentes de nombrar a una misma persona. Fragmento de la sentencia número 174/2012[3].

Como desventaja se puede mencionar que es muy sensible para seleccionar texto y agrega un salto de línea por cada punto seguido de un espacio, lo cual ocurre con mucha frecuencia en sentencias judiciales por las abreviaciones (ejemplos: Sr., Dra.).

3.4. Distribución de etiquetas

La figura 3.12 y cuadro 3.2 presentan la distribución de las etiquetas obtenidas como resultado del proceso manual de marcado. Se puede observar que P es la etiqueta con mayor cantidad de ocurrencias, seguida por NP, mientras que se cuenta con una muy baja cantidad de etiquetas REF y LOC —este desbalance será tratado posteriormente en la sección 4.4—. Parte de la diferencia se debe a que solo se volvió a anotar el 35.34 % del total de las sentencias. El resto de las sentencias no cuentan con estas etiquetas. Cabe destacar que la distribución se obtiene a partir del conjunto total de sentencias.

En el cuadro 3.2, la columna "# sentencias" indica la cantidad de sentencias con al menos una ocurrencia de la etiqueta de cada fila. Como era previsible, todas las sentencias contienen al menos una etiqueta P.

Las gráficas circulares de las figuras 3.13a y 3.13b permiten observar que la cantidad de etiquetas P es más de tres veces mayor a la suma de NP, REF y

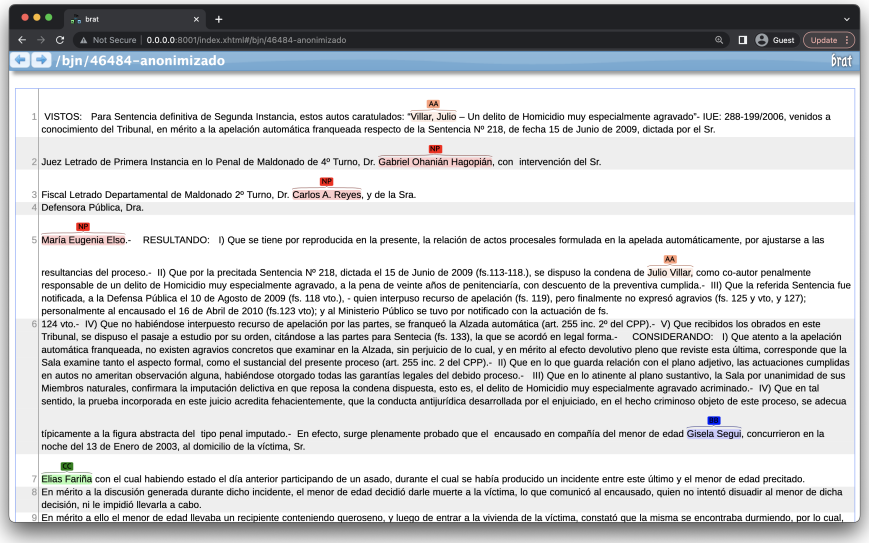


Figura 3.10: Interfaz de BRAT con una sentencia etiquetada.

LOC. Esto puede ser de ayuda a la resolución del problema de este proyecto por ser la etiqueta de interés a anonimizar, aunque también puede dificultar deducir las otras etiquetas (este problema se estudiará en la sección 4.4). La figura 3.13b permite observar la distribución de las etiquetas menos representadas, sin considerar P.

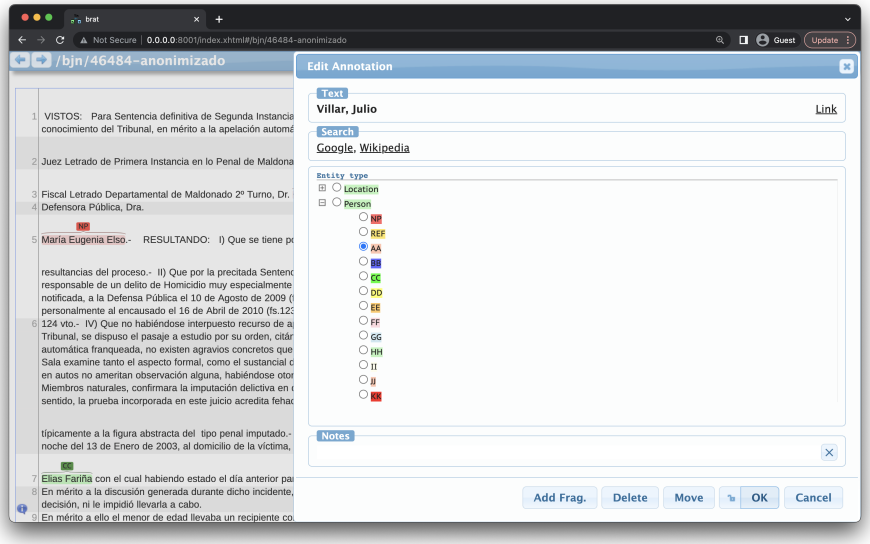


Figura 3.11: Ventana para elegir etiqueta en BRAT al seleccionar un texto.

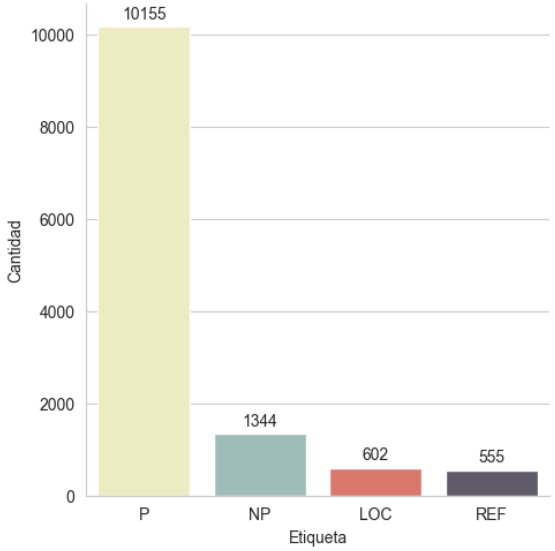
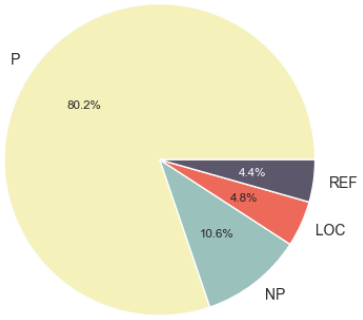


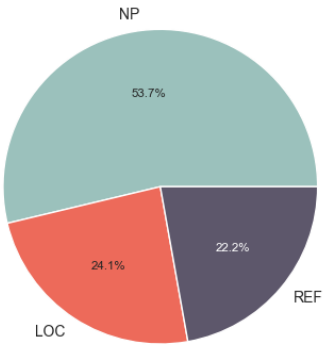
Figura 3.12: Distribución de etiquetas en el total de sentencias.

Etiqueta	**# ocurrencias**	**% ocurrencias**	**# sentencias**	**% sentencias**
P	10.155	80,23 %	999	100 %
NP	1.344	10,62 %	339	33,93 %
LOC	602	4,76 %	284	28,42 %
REF	555	4,39 %	129	12,91 %
Total	**12.656**	**100 %**	999	100 %

Cuadro 3.2: Distribución de etiquetas en el total de sentencias.



(a) Distribución de etiquetas dentro de las sentencias.



(b) Distribución de las etiquetas sin considerar P.

Capítulo 4

Reconocimiento de entidades nombradas

Este capítulo presenta la solución implementada para el reconocimiento de entidades nombradas realizada durante este proyecto. Se expone el diseño e implementación de la solución mediante el uso de *Transformers* y la librería *Hugging Face*, así como las pruebas ejecutadas con los distintos modelos existentes y las distintas estrategias empleadas para el procesamiento de datos. Al final, se presentan y se realiza un análisis de los resultados.

Nota: Las pruebas descritas en este capítulo se ejecutaron en un entorno Google Colab PRO[50] con las siguientes especificaciones:

- GPU: Tarjeta NVIDIA A100-SXM4 con 40GB vRAM.
- RAM: 90GB.

  Además, para aquellos resultados obtenidos al entrenar y evaluar el modelo, el resultado que se muestra es el mejor obtenido en un período de cinco épocas.

4.1. Esqueleto de solución

Como se menciona en la subsección 2.6.2, la librería *Hugging Face* provee, entre otras cosas, implementaciones para los distintos modelos existentes basados en *Transformers*, así como facilidades relacionadas al manejo de conjuntos de datos y el proceso de entrenamiento.

En pos de aprender a utilizar la librería y evaluar sus beneficios, se comienza por la implementación de una solución básica que utilice *Transformers* y sea ca-

```
pipeline() {
  modelo = inicializar_modelo_y_tokenizer();
  training_dataset, test_dataset = cargar_datasets()
  mejor_modelo = modelo.entrenar(training_dataset)
  mejor_modelo.guardar()
  imprimir_resultados(mejor_modelo)
}
```

Código 4.1: Algoritmo que ejecuta la solución.

paz de ser reajustada a los datos del proyecto. Para esto es necesario seleccionar un modelo pre entrenado en *HuggingFace* que cumpla con estas condiciones.

Los modelos considerados son BETO —la arquitectura de $BERT_{BASE}$ preentrenado con un corpus español— ajustado a la tarea de reconocimiento de entidades[51], TinyBERT — modelo de menor tamaño que hereda conocimiento de BERT —y XLM-RoBERTa— que incorpora los cambios de RoBERTa y además utiliza una mayor cantidad de datos y parámetros, descriptos en la comparación de modelos derivados de BERT (subsección 2.2.2). Se elige BETO ya que, dentro de los modelos basados en BERT, se encuentra en el medio tanto de desempeño como costo computacional; la cantidad de parámetros es muy superior a TinyBERT (14.5 millones contra 110) pero menor a XLM-RoBERTa (110 millones contra 550) y se cuenta con más ejemplos de uso. Además, XLM-RoBERTa requiere mayor configuración ya que los tokens sentencepiece en los que se basa requieren el uso de una librería especial. Y por otro lado, en caso obtener un desempeño pobre con TinyBERT, del cuál se esperan los peores resultados dado su tamaño, no se tendría una solución con la cual corroborar que no se deban a problemas de implementación.

Para validar que la solución se comporta de la forma esperada, se entrenó y evaluó el modelo utilizando los sets de datos de CoNLL2002 en español[52]. Verificando que los resultados obtenidos fueran similares a los reportados por el modelo preentrenado, que utilizó el mismo set para el *fine-tuning*. El modelo preentrenado obtiene un resultado f_1 de 0.901 mientras que el esqueleto 0.947. Las diferencias pueden deberse a varias razones: una división distinta del conjunto de entrenamiento, distinta semilla aleatoria, distintos parámetros de entrenamiento o el hecho de que se parte de un modelo ya ajustado al conjunto y se vuelve a entrenar sobre el mismo conjunto. De cualquier manera, esto valida que la implementación de la solución es correcta.

Se obtuvo, entonces un prototipo de solución que se sabía capaz de ajustarse al conjunto de datos correspondiente y predecir etiquetas de forma correcta. Posteriormente, el ajuste de la solución a las sentencias de la BJN y el uso de distintos modelos preentrenados requerirá pocos cambios a la implementación de la solución original.

Las secciones a continuación describen el proceso seguido para llegar al modelo con mejor desempeño.

4.2. Épocas

Una época representa un ciclo entero en el cual todos los datos de entrenamiento pasan por el algoritmo o modelo. El número de épocas a utilizar en el entrenamiento es, por lo tanto, un hiperparámetro —un valor de configuración que no se deduce de los datos— que se fija antes de que el entrenamiento comience y depende fuertemente del problema a resolver y del algoritmo a utilizar. Es importante fijar un número correcto, ya que una baja cantidad de épocas puede implicar la pérdida de convergencia del modelo. A su vez, un número muy alto puede llevar a un sobreajuste del modelo respecto a los datos de entrenamiento y un malgaste de recursos de cómputo. Para asegurar la utilización de un número de épocas correcto y evitar estos problemas se observa el comportamiento de la pérdida de evaluación, evitando entrenar el modelo una vez que comienza a subir después de haber alcanzado su valor mínimo.

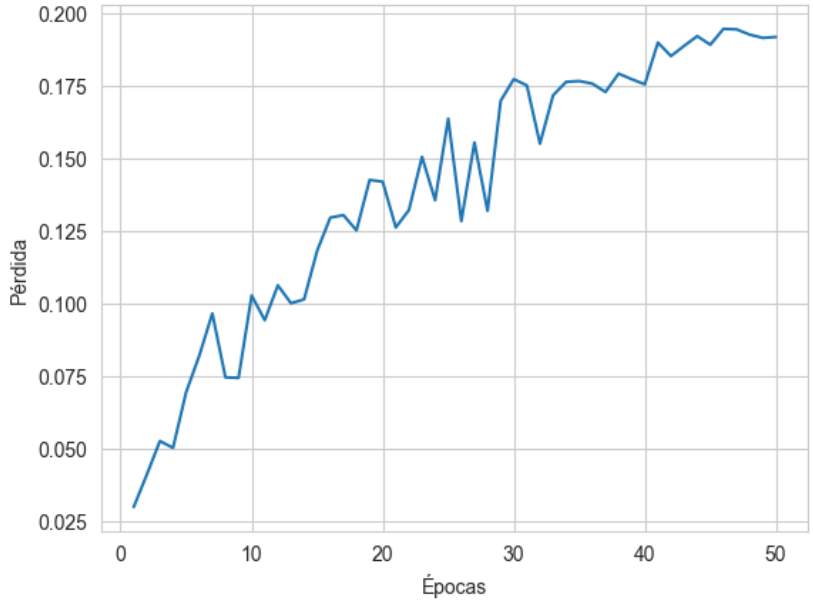


Figura 4.1: Evolución de la pérdida de validación al ajustar BETO con el conjunto de entrenamiento del corpus de la BJN durante cincuenta épocas.

Los creadores de BERT recomiendan fijar el número de épocas a un valor de entre dos y cuatro épocas[11], lo cual se ve reflejado en la figura 4.1. Los modelos basados en BERT tienden a requerir pocas épocas para ser ajustados

y un número mayor puede llevar a un sobreajuste al corpus de entrenamiento o perder conocimiento obtenido en la etapa de preentrenamiento, un fenómeno conocido como olvido catastrófico[1][54][55].

Si bien el mínimo de pérdida de evaluación se alcanza en dos épocas para la mayoría de las pruebas realizadas en este informe, parte alcanza su mínimo en tres o cuatro. Es por esto que se decide fijar el número a cinco épocas, y es este el valor usado a través de todas las pruebas a continuación.

4.3. Generación de partición de datos

La librería *HuggingFace* utiliza el concepto de *datasets* para cargar datos a sus modelos; estos proveen una interfaz común para su acceso y simplifican la aplicación de transformaciones. Además, son fácilmente exportables y por lo tanto se pueden reutilizar en distintos modelos sin requerir cambios adicionales.

Como se menciona en el capítulo 3, se construyó un conjunto de datos con las sentencias judiciales de la BJN. El conjunto se divide en tres particiones distintas: **entrenamiento**, **validación** y **evaluación**. Que contienen datos independientes. Los conjuntos de **entrenamiento** y **validación** se utilizan a la hora de entrenar, mientras que el conjunto de **evaluación** se utiliza una vez finalizado el entrenamiento para estimar el desempeño de la solución obtenida.

Cada ejemplo de estos conjuntos contiene la porción de texto. Así como una lista de anotaciones en donde se especifica su etiqueta y su número de caracter de inicio y fin en el texto.

En la versión inicial, cada ejemplo corresponde al texto completo de una sentencia. Como se explica en las siguientes secciones, esto fue posteriormente modificado para mejorar el rendimiento del modelo.

4.3.1. Preprocesamiento de datos

Para comenzar, es necesario adaptar la solución existente a que sea capaz de predecir las etiquetas utilizadas por el proyecto, presentadas en el capítulo 3. Para lograr esto, basta con sobrescribir la configuración del modelo para que use cinco etiquetas en total: *LOC, P, NP, REF y Empty*, donde *Empty* será asignada a aquellas palabras que no pertenezcan a ninguna de las categorías anteriores. Se modifica la última capa del modelo —el clasificador lineal— para que el tamaño del vector de salida (cinco) coincida con las nuevas etiquetas.

Los modelos basados en *Transformers* son capaces de procesar secuencias de texto de hasta cierto tamaño máximo. En caso de encontrarse con una secuencia de mayor largo, el modelo ignora la entrada sobrante. Para el caso de *BERT* este

[1] El fenómeno llamado olvido catastrófico ocurre cuando conocimiento adquirido previamente se pierde al incorporar información nueva[53].

límite es de 512 tokens[2], el cual es sobrepasado con facilidad por las sentencias judiciales.

Al entrenar, la solución original utiliza cada sentencia entera como entrada del modelo, por lo cual gran parte termina siendo descartada en la mayoría de los casos —esto es equivalente a utilizar los primeros 512 tokens de cada sentencia. Al hacer esto, se ignoran muchas entidades del documento, llevando a un desempeño menor dado que no son aprovechadas para reforzar el aprendizaje del modelo. Buscando mejorar los resultados obtenidos, se evaluaron más alternativas a lo descrito anteriormente:

- dividir cada sentencia en *chunks* de 512 tokens para utilizarlos como entrada del modelo, aprovechando de esta forma el contenido total de los documentos.
- utilizar el primer y último *chunk* —de 512 tokens— de cada sentencia como entrada, ya que las personas involucradas tienden a aparecer al principio, como se mencionó anteriormente; o al final, cuando se dicta la sentencia.
- utilizar los últimos 512 tokens de cada sentencia, basándose en que las sentencias incluyen un resumen de los actores afectados al final.

Con el propósito de tener una mayor organización de los *chunks*, y poder seleccionar subconjuntos específicos de estos para pruebas futuras, se almacenan en una base de datos PostgreSQL[3] bajo una tabla llamada "*chunks*" dónde cada fila contiene las columnas descritas en el cuadro 4.1.

La tabla resultante contiene 2920 *chunks* distintos.

Se evalúa utilizar un archivo de extensión `.csv` en lugar de una tabla de base de datos, pero la base de datos es más eficiente de consultar, filtrar, procesar y obtener métricas.

4.3.2. Comparación de particiones

En adelante, los resultados y métricas expuestas serán respecto a las nuevas etiquetas. Los resultados se expresarán en términos de precisión (en inglés *precision*), exhaustividad (en inglés *recall*) y su media armónica F_1 —de ahora en adelante: P, R y F_1 respectivamente— tal como se definen en *Speech and language processing* por Dan Jurafsky[4]. Para calcular los resultados se utiliza la métrica Seqeval[56] que considera como correctas predicciones exactas de la entidad, predicciones parciales solo cuentan para la exactitud. El corpus no sigue un esquema de etiquetado por lo que para poder utilizar Seqeval los tokens sin etiquetas se marcan como vacíos —etiqueta O— y al resto se les agrega el prefijo I-, simulando el esquema IOB1. Se consideran todos los *chunks* contengan

[2]Se recuerda que un token puede ser incluso menor a una palabra, ver sección 2.2.2.

[3]PostgreSQL es un gestor de bases de datos relacionales gratis y de código abierto, la versión utilizada es 14.8.

Columna	**Tipo**	**Descripción**
id	Número	Identificador del *chunk* en la base de datos.
code	Número	Código del nombre del archivo al que pertenece el *chunk*.
chunk_start	Número	Caracter de inicio del *chunk* dentro de la sentencia.
chunk_end	Número	Caracter de fin del *chunk* dentro de la sentencia.
chunk	Texto	Texto del *chunk*.
annotations	Arreglo	Arreglo de anotaciones que contiene la etiqueta, el inicio de la etiqueta y el fin.
p_count	Número	Cantidad de etiquetas P dentro del *chunk*.
np_count	Número	Cantidad de etiquetas NP dentro del *chunk*.
ref_count	Número	Cantidad de etiquetas REF dentro del *chunk*.
loc_count	Número	Cantidad de etiquetas LOC dentro del *chunk*.

Cuadro 4.1: Descripción de la tabla "*chunks*".

o no etiquetas (posteriormente se evaluará este caso). Además, de los resultados obtenidos de las cinco épocas, se selecciona aquella con la menor pérdida de evaluación.

Nota: cabe destacar que las cuatro pruebas utilizan el mismo conjunto de validación para que los resultados sean comparables.

Solución	**P**	**R**	**F1**
Todos los *chunks* de c/sentencia	0.888	**0.562**	**0.688**
Primer *chunk* (sin dividir en *chunks*)	0.824	0.535	0.649
Último *chunk*	0.695	0.522	0.596
Primer y *último* chunk	**0.906**	0.547	0.683

Cuadro 4.2: Resultados obtenidos luego de subdividir las sentencias en *chunks* de 512 tokens.

Como se observa en el cuadro 4.2, la subdivisión de las sentencias en *chunks* mejora el aprendizaje del modelo en las tres métricas. Se modifica entonces el conjunto de datos generado anteriormente de forma que cada entrada es un *chunk* de sentencia de largo 512 tokens. Se hace notar que, al dividir las sentencias de a 512 tokens, pueden perderse anotaciones compuestas por varios tokens con una parte perteneciente a un *chunk* y otra parte perteneciente al siguiente,

es decir, etiquetas que se encuentran en el límite de una partición. Esto es así por simplicidad de la implementación; se podría utilizar una ventana deslizante o inclusive dividir la sentencia en bloques de tamaño menor para evitar cortar alguna anotación.

4.3.3. Mecanismo de división

Las particiones o conjuntos utilizadas hasta el momento no son estrictamente representativas de la realidad. Las etiquetas no están divididas manteniendo una correcta proporción en los conjuntos de entrenamiento, validación y evaluación; lo cual puede generar sesgos en el modelo. Es por esto que se construyó un conjunto de evaluación y validación fijos que mantengan las proporciones de entidades presentes en la realidad.

Junto a la construcción del conjunto fijo de evaluación se construyeron los conjuntos de entrenamiento y validación. Si se evalúa con *chunks* que intentan respetar la realidad, se estima que dará mejores resultados entrenar al modelo de la misma forma.

La totalidad de los *chunks* se divide en tres conjuntos: los que solo contienen etiquetas P, los que contienen al menos una etiqueta que no sea P (NP, REF o LOC) y los que no contienen ninguna etiqueta.

Para los *chunks* que solo contienen etiquetas P se obtiene la suma total de etiquetas. La suma se utiliza para posteriormente dividir el conjunto de *chunks* en los conjuntos de entrenamiento, validación y evaluación intentando mantener la proporción de etiquetas: 80 %, 10 % y 10 % respectivamente.

Con el conjunto de *chunks* con al menos una etiqueta que no sea P se realiza la división mencionada para el conjunto anterior, pero en lugar de utilizar la suma de etiquetas P, se obtiene el total de etiquetas NP. Se tomó la decisión de usar NP en lugar de REF o LOC pues NP tiene mayor tendencia a generar confusión en el modelo con la etiqueta P (etiqueta que debe ser anonimizada); esto se puede observar en la matriz de confusión de la figura 4.2 generada sobre *tokens*, no sobre entidades. Se consideró que sería de mayor utilidad mantener la proporción de ocurrencias de la etiqueta que genera mayor confusión. Además, se intentó utilizar REF y LOC, pero los conjuntos de entrenamiento, validación y evaluación perdían balance.

En el caso de los *chunks* sin etiquetas, solo se toman en consideración aquellas que fueron anotadas para este proyecto. Se recuerda que las sentencias inicialmente no contenían etiquetas de tipo NP, REF y LOC, por lo que el hecho de que un *chunk* no contenga etiquetas puede significar que no fue adaptado o que verdaderamente no contiene ninguna entidad relevante al proyecto. El total de *chunks* sin etiquetas, al igual que los otros conjuntos, se divide manteniendo la proporción aproximada de 80 % en entrenamiento, 10 % para validación y el restante 10 % para entrenamiento.

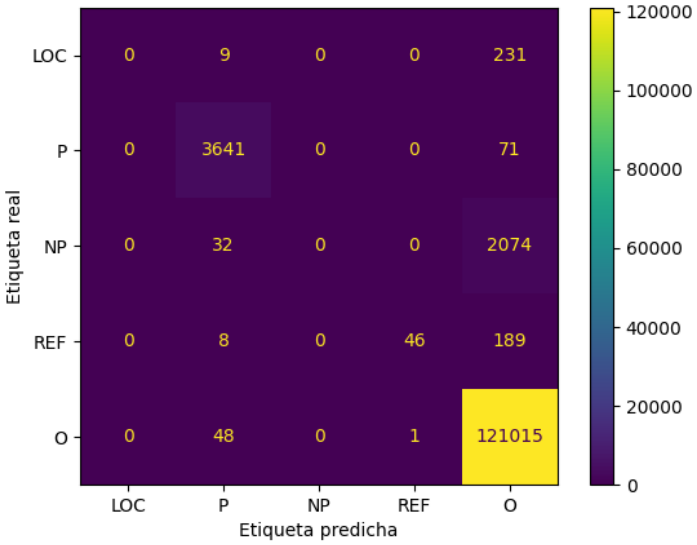


Figura 4.2: Matriz de confusión.

La proporción resultante de la partición se muestra en la figura 4.3 y en el cuadro 4.3.

Se realizaron tres pruebas adicionales manteniendo el conjunto de validación definido en la prueba anterior. Consisten en modificar el conjunto de entrenamiento utilizando de cada sentencia no perteneciente al conjunto de validación y evaluación: únicamente el primer *chunk* (cuadro 4.4), únicamente el último *chunk* (cuadro 4.5), y el primer y último *chunk* (cuadro 4.6). En el cuadro 4.7 se muestran los resultados obtenidos (cuatro primeras pruebas).

Utilizar el primer y último *chunk* de cada sentencia da mejores resultados. Esto puede deberse a que el modelo cuanta con más etiquetas que al usar solo el primer *chunk* o el último *chunk*, es decir, cuenta con más ejemplos para entrenar. Aunque cuenta con menos ejemplos que al utilizar todos los *chunks*, la concentración de etiquetas es mayor.

Dado que utilizar primer y último chunk, y todos los chunks obtienen resultados casi comparables se decide continuar realizando pruebas para ambos casos hipotetizando que se debería poder extraer mejores resultados de esta última al contar con una mayor cantidad de etiquetas para su entrenamiento.

Luego se probó no utilizar *chunks* sin etiquetas en el conjunto de entrenamiento, porque por más que considerarlos respeta la realidad, podrían quitarle importancia a las etiquetas de interés a anonimizar. Los resultados se muestran

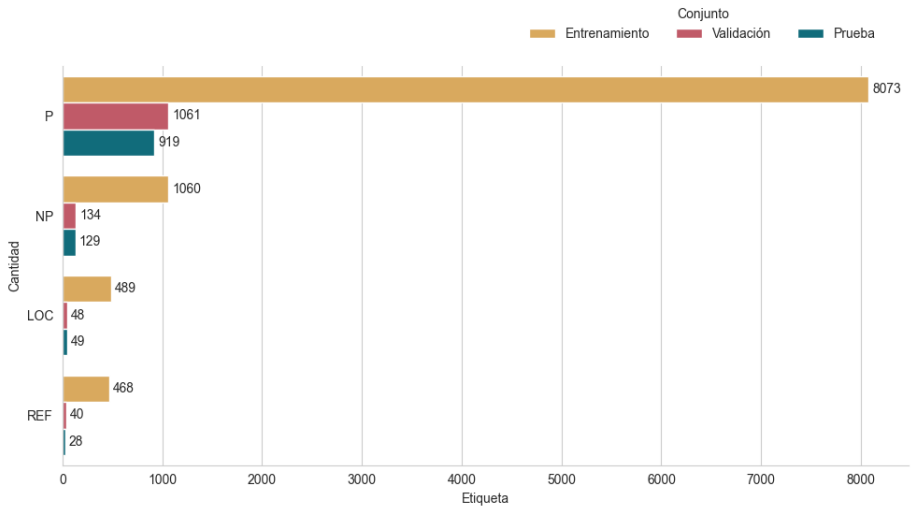


Figura 4.3: Cantidad de etiquetas por conjunto de datos utilizando sentencias completas divididas en *chunks*.

en las 2 últimas líneas del cuadro 4.7. F1 para todos los *chunks* y para primer y último *chunk* mantuvieron su pequeña diferencia (0.01), reforzando la idea de continuar haciendo pruebas para los dos casos; pero F1 bajó en comparación a entrenar con *chunks* sin etiquetas, por lo que se continuará entrenando con ellos. Además se encontraron casos en los que se anotaron textos que no corresponden a entidades nombradas; en la figura 4.4 se ejemplifica un caso donde el delito es confundido por una persona (segunda anotación).

Solución	**P**	**R**	**F1**
Primer *chunk*	0.871	0.739	0.800
Último *chunk*	0.837	0.759	0.796
Primer y último *chunk*	0.854	**0.834**	**0.844**
Todos los *chunks* de c/sentencia	**0.896**	0.781	0.834
Primer y último *chunk*	0.831	**0.823**	0.827
Todos los *chunks* de c/sentencia	**0.897**	0.785	**0.837**

Cuadro 4.7: Resultados obtenidos utilizando un conjunto de validación representativo de la realidad. En la primer sección se utilizan *chunks* sin etiquetas, y en la segunda no.

Etiqueta	**Entrenamiento**		**Validación**		**Evaluación**	
	Cant.	**%**	**Cant.**	**%**	**Cant.**	**%**
P	8.073	80,31 %	1.061	10,55 %	919	9,14 %
NP	1.060	80,12 %	134	10,13 %	129	9,75 %
LOC	489	83,45 %	48	8,19 %	49	8,36 %
REF	468	87,31 %	40	7,46 %	28	5,22 %
Chunks	1.968	82,06 %	211	8,8 %	219	9,14 %

Cuadro 4.3: Distribución de etiquetas por conjunto de datos utilizando sentencias completas divididas en *chunks*.

Etiqueta	**Entrenamiento**		**Validación**		**Evaluación**	
	Cant.	**%**	**Cant.**	**%**	**Cant.**	**%**
P	3.639	64,77 %	1.061	18,88 %	919	16,35 %
NP	729	73,49 %	134	13,51 %	129	13,0 %
LOC	289	74,87 %	48	12,44 %	49	12,69 %
REF	20	22,73 %	40	45,45 %	28	31,82 %
Chunks	865	66,80 %	211	16,29 %	219	16,91 %

Cuadro 4.4: Distribución de etiquetas por conjunto de datos utilizando solo el **primer *chunk*** de las sentencias en el conjunto de entrenamiento.

... Por los fundamentos expuestos, EL TRIBUNAL, FALLA: CONFÍRMASE LA SENTENCIA DE PRIMERA INSTANCIA, SALVO EN CUANTO A QUE **P** GISELA SEGUI DEBE RESPONDER COMO AUTORA DEL DELITO DE **P** RAPIÑA ATRIBUIDO ...

Figura 4.4: Ejemplo donde se anota texto que no se corresponde con la etiqueta asignada. Fragmento de la sentencia número 80/2011[3].

4.4. Desbalance de clases

Las etiquetas LOC y REF presentan un bajo desempeño cuando se las compara con el resto. Esto se debe en parte al gran desbalance que hay cuando se compara su cantidad con la de otras etiquetas (en especial P), tal como se mues-

Etiqueta	**Entrenamiento**		**Validación**		**Evaluación**	
	Cant.	**%**	**Cant.**	**%**	**Cant.**	**%**
P	1.240	38,51 %	1.061	32,95 %	919	28,54 %
NP	332	55,8 %	134	22,52 %	129	21,68 %
LOC	58	37,42 %	48	30,97 %	49	31,61 %
REF	69	50,36 %	40	29,2 %	28	20,44 %
Chunks	521	56,17 %	211	21,51 %	219	22,32 %

Cuadro 4.5: Distribución de etiquetas por conjunto de datos utilizando solo el **último *chunk*** de las sentencias en el conjunto de entrenamiento.

Etiqueta	**Entrenamiento**		**Validación**		**Evaluación**	
	Cant.	**%**	**Cant.**	**%**	**Cant.**	**%**
P	4.580	69,82 %	1.061	16,17 %	919	14,01 %
NP	994	79,08 %	134	10,66 %	129	10,26 %
LOC	322	76,85 %	48	11,46 %	49	11,69 %
REF	89	56,69 %	40	25,48 %	28	17,83 %
Chunks	1.301	75,16 %	211	12,19 %	219	12,65 %

Cuadro 4.6: Distribución de etiquetas por conjunto de datos utilizando solo el **primer y último *chunk*** de las sentencias en el conjunto de entrenamiento.

tra en la figura 3.12. REF representa un 4.29 % del total de etiquetas y LOC 4.68 %, y en el caso de primer y último chunk un 1.87 % y 4.99 % respectivamente. Esta falta de etiquetas introduce un sesgo en nuestro modelo llevando a que termine priorizando las clases con más ejemplos disponibles: NP y en particular P, disminuyendo el desempeño de las clases subrepresentadas.

Solución	LOC			REF		
	P	R	F1	P	R	F1
Primer y último *chunk*	**0.444**	**0.216**	**0.291**	0.273	0.207	0.235
Todos los *chunks*	0.0	0.0	0.0	**0.412**	**0.241**	**0.304**

Cuadro 4.8: Desempeño de LOC y REF.

Solución	P			NP		
	P	R	F1	P	R	F1
Primer y último *chunk*	0.916	0.921	0.919	0.602	**0.592**	**0.597**
Todos los *chunks*	**0.927**	**0.928**	**0.928**	**0.618**	0.168	0.264

Cuadro 4.9: Desempeño de P y NP.

Si bien la predicción errónea de una entidad cualquiera como P no necesariamente resulta un problema a la hora de anonimizar, es posible aplicar métodos que ayuden a balancear los datos disponibles para obtener un mejor comportamiento del modelo. A continuación se describen los dos considerados en este proyecto para mejorar el desempeño de las clases subrepresentadas.

Sobremuestreo (o submuestreo) de etiquetas - Una de las formas más sencillas de aumentar la proporción de datos relevantes, consiste en añadir instancias de las clases subrepresentadas de forma artificial. El submuestreo es el proceso equivalente contrario, se remueven instancias de las clases sobrerepresentadas de forma de disminuir su proporción.

Existen distintas alternativas para añadir instancias nuevas de una clase al conjunto de datos, una de ellas es la duplicación de instancias ya existentes. Otra es generar datos nuevos a partir de las ya existentes.

Tanto el uso de submuestreo como sobremuestreo introducen nuevos problemas al conjunto de datos. El primero resulta en pérdida de información al descartar instancias reales mientras que el segundo introduce ruido al duplicar datos ya existentes o introducir datos artificiales. Por estas razones se descartan ambos métodos y se procede a utilizar el balanceo a través de pesos que no modifica el conjunto de datos, si no la importancia que se le asigna a cada ejemplo.

Pesos de etiquetas - Este método consiste en proveer al algoritmo con distintos pesos para cada etiqueta. En el entrenamiento se penaliza a la red cuando comete un error de clasificación en una de las clases con menor representación, mientras que las clases sobrerepresentadas incurren en una penalización menor. De esta forma, se introduce un sesgo artificial a la red para disminuir el error en las clases con menor cantidad de instancias. Para lograr esto se modifica la función de pérdida del modelo, en cada paso se multiplican los valores obtenidos para cada clase por los pesos respectivos.

Estos pesos son valores numéricos y si bien no existe una fórmula para obtener los valores óptimos, se cuenta con varios métodos para calcularlos. En particular, se identifican los dos siguientes métodos:

- El peso de cada clase es representado inversamente proporcional a su fre-

cuencia utilizando la fórmula:

$$w_i = \frac{\#instancias}{\#etiquetas \cdot \#instancias_i}$$

donde w_i representa el peso de la etiqueta i.

Utilizando este método se obtienen los siguientes pesos y resultados:

Solución	w_{LOC}	w_P	w_{NP}	w_{REF}	P	R	F_1
Primer y último *chunk*	3.718	0.261	1.204	13.450	**0.820**	0.788	**0.804**
Todos los *chunks*	4.225	0.249	1.878	4.544	0.714	**0.890**	0.792

Cuadro 4.10: Pesos obtenidos según la frecuencia de cada etiqueta y los resultados que logran.

- El peso de cada etiqueta es representado proporcionalmente respecto a una etiqueta base. En el contexto de este proyecto se decidió utilizar P como la etiqueta de referencia, obteniendo la siguiente fórmula:

$$w_i = \frac{\#etiquetas_P}{\#etiquetas_i}$$

Obteniendo los siguientes pesos y resultados:

Solución	w_{LOC}	w_P	w_{NP}	w_{REF}	P	R	F_1
Primer y último *chunk*	14.229	1.0	4.609	51.84	0.650	0.893	0.752
Todos los *chunks*	10.0	1.0	7.5	18.0	**0.698**	**0.930**	**0.797**

Cuadro 4.11: Resultados obtenidos aplicando pesos según la proporción de cada etiqueta.

Ambos resultados no logran una mejoría respecto a aquellos obtenidos anteriormente. Con el mejor caso representando una pérdida de 5 % de F_1, aunque se destaca que el *recall* mejora.

Buscando obtener mejores resultados se procede a utilizar una **búsqueda de hiperparámetros** en la que se perturban los pesos obtenidos un ± 25 % de su valor original. Se toma este porcentaje de variación ya que no es lo suficientemente grande como para aumentar el tiempo de búsqueda a niveles imprácticos, pero a su vez permite probar con un amplio rango de valores.

Para los pesos obtenidos según su frecuencia se obtienen los siguientes resultados:

Solución	w_{LOC}	w_P	w_{NP}	w_{REF}	P	R	F_1
Primer y último *chunk*	3.381	0.276	0.935	10.638	**0.805**	0.769	0.787
Todos los *chunks*	4.996	0.256	1.799	4.346	0.714	**0.890**	**0.793**

Cuadro 4.12: Resultados obtenidos aplicando pesos según la frecuencia de cada etiqueta (hiperparámetros).

En el caso de los pesos proporcionales, los resultados obtenidos son:

Solución	w_{LOC}	w_P	w_{NP}	w_{REF}	P	R	F_1
Primer y último *chunk*	13.334	1.046	3.762	50.591	0.693	0.911	0.787
Todos los *chunks*	12.242	0.817	8.710	20.812	**0.709**	**0.912**	**0.798**

Cuadro 4.13: Resultados obtenidos aplicando pesos según la proporción de cada etiqueta (hiperparámetros).

Dado que las pruebas anteriores no superaron los resultados de F1 sin aplicar balance de clases, se optó por hacer una última prueba con búsqueda de hiperparámetros utilizando valores enteros del 1 al 12 para los pesos. Se tomó como base el peso de la etiqueta P asignándole el valor 1, y se hicieron 40 intentos variando los pesos de NP, REF y LOC. En el cuadro 4.14 se exponen los pesos y resultados con mejor solución para las pruebas utilizando todos los *chunks*, y primer y último *chunk*.

Solución	w_{LOC}	w_P	w_{NP}	w_{REF}	P	R	F_1
Primer y último *chunk*	8.0	1.0	9.0	9.0	**0.723**	0.906	0.805
Todos los *chunks*	11.0	1.0	12.0	8.0	0.714	**0.928**	**0.807**

Cuadro 4.14: Mejores resultados obtenidos y pesos de etiquetas hallados con hiperparámetros con valores enteros entre 1 y 12.

Las últimas pruebas son las que dieron mejores resultados, aunque no superan los resultados previos a esta sección (sin balancear las clases). Con el objetivo de buscar una explicación ante el descenso del valor de F1, se observa el desempeño de cada etiqueta de forma independiente (cuadros 4.15 y 4.16).

En las secciones siguientes se ahondará en el uso de búsquedas de hiperparámetros aplicadas a los distintos parámetros del modelo.

Solución	LOC			REF		
	P	R	F1	P	R	F1
Primer y último *chunk*	**0.252**	0.784	**0.382**	0.378	0.483	0.424
Todos los *chunks*	0.235	**0.865**	0.370	**0.431**	**0.759**	**0.550**

Cuadro 4.15: Desempeño de LOC y REF aplicando balance de clases.

Solución	P			NP		
	P	R	F1	P	R	F1
Primer y último *chunk*	**0.910**	0.918	0.914	0.418	0.960	0.582
Todos los *chunks*	0.907	**0.931**	**0.919**	**0.422**	**0.968**	**0.587**

Cuadro 4.16: Desempeño de P y NP aplicando balance de clases.

4.5. Comparación de modelos

Desde su lanzamiento han surgido distintos modelos basados en BERT. Estos modelos modifican los hiperparámetros y métodos de entrenamiento usados por el modelo original buscando mejorar su desempeño en métricas y otras áreas como lo es el costo computacional. Para obtener el mejor resultado posible, se realizó una comparación de las métricas —precisión, exhaustividad y media armónica— obtenidas al ajustar los distintos modelos.

Cómo se expuso en el marco teórico, centramos nuestra búsqueda en aquellas variaciones entrenadas utilizando conjuntos de datos en español con un foco en NER, lo cual redujo la cantidad de opciones disponibles de forma considerable. De las variaciones mencionadas, se consideran todas a excepción de *BERT Multilingual* dado su similitud con *BETO* y el hecho de que este último obtiene mejores resultados en el idioma español, como se puede ver en el cuadro 4.17. Cada modelo utiliza los chunks descriptos anteriormente así como la mejor combinación de pesos.

Modelo	Primer y último chunk			Todos los chunks		
	P	R	F1	P	R	F1
RoBERTa	**0.892**	0.748	0.814	**0.903**	0.761	0.826
BETO	0.854	**0.834**	**0.844**	0.896	**0.781**	**0.834**
TinyBERT	0.541	0.169	0.257	0.504	0.185	0.271

Cuadro 4.17: Resultados obtenidos al ejecutar la solución con cada variación de modelo listada.

Si bien *TinyBERT* obtiene resultados pobres, se destaca que el tamaño del modelo es un orden de magnitud menor que el resto. $55MB$ contra los $420MB$ del siguiente con menor tamaño. Si el tiempo de cómputo y el tamaño de almacenaje fueran variables a considerar, no debería descartarse esta alternativa.

Del cuadro anterior se observa que BETO obtiene los mejores resultados de F1 en ambos escenarios. Aún así se procede a realizar una búsqueda de hiperparámetros sobre todas las variantes para hallar la combinación de modelo y parámetros que mejor se desempeña.

4.6. Búsqueda de hiperparámetros

Los hiperparámetros son valores fijos que sirven de entrada al algoritmo y se utilizan para configurar su comportamiento, por ejemplo, entre ellos se encuentran el tamaño de lote de entrenamiento o el número de épocas. Estos no se modifican durante el entrenamiento y por lo tanto no son aprendidos por el modelo.

Dada su capacidad de controlar el comportamiento del algoritmo, es importante elegir una combinación de valores que produzca los mejores resultados posibles. Para esto se ejecuta una búsqueda de hiperparámetros exhaustiva (*Grid Search*) en la que se prueban distintas combinaciones de valores buscando optimizar un objetivo, en este caso, maximizar los resultados del modelo.

HuggingFace provee una *API* para facilitar la ejecución de búsquedas que permite seleccionar una librería de optimización de hiperparámetros y especificar rangos o distintos valores posibles para cada uno de ellos, las opciones disponibles actualmente son: Optuna, Sigopt, Raytune y Wandb. En el contexto de este proyecto se optó por utilizar Optuna[57], una librería de código abierto escrita en Python enfocada en la optimización de hiperparámetros, ya que se cuenta con más ejemplos de cómo utilizarla.

El tiempo requerido para ejecutar las búsquedas aumenta exponencialmente con cada variación de hiperparámetro posible. Dado el interés de mantener el tiempo y costo de ejecución limitados, se centró la búsqueda en los parámetros expuestos a continuación siguiendo las recomendaciones que proponen Chi Sun, Xipeng Qui, et al[58].

1. *Epochs*: Dado lo mencionado en el capítulo de épocas (capítulo 4.2), se prueba con un rango de valores de entre dos y cinco épocas.

2. *Learning rate*: cómo se menciona en el artículo, es importante mantener un valor pequeño para modelos basados en BERT, de forma de evitar «el olvido catastrófico», en un rango de valores de $2e^{-5}$ a $5e^{-5}$.

3. *Batch size*: dado el largo de secuencia utilizado por el modelo (512) es

necesario mantener el valor menor a 16, ya que en caso contrario el bache no cabe en las *GPU*s disponibles. Los valores utilizados son: 1, 4, 8, 16.

4. *Weight decay*: partiendo del valor original de 0.01 se considera un rango de variación del 10 % (0.009 a 0.011).

Para el caso de primer y último chunk:

Modelo	*Learning rate*	*Batch size*	*Weight decay*	P	R	F_1
RoBERTa	4e-05	1	0.00988	0.754	**0.913**	**0.826**
BETO	4e-05	8	0.00994	**0.874**	0.773	0.820
TinyBERT	4e-05	8	0.00940	0.523	0.172	0.259

Para el caso de todos los chunks:

Modelo	*Learning rate*	*Batch size*	*Weight decay*	P	R	F_1
RoBERTa	4e-05	1	0.00973	**0.891**	0.785	0.835
BETO	5e-05	8	0.00913	0.885	**0.802**	**0.841**
TinyBERT	4e-05	8	0.00903	0.511	0.171	0.256

Cuadro 4.18: Resultados obtenidos al ejecutar una búsqueda de hiperparámetros para cada modelo.

Se observa que algunos resultados son mejores que resultados de secciones anteriores, mientras que otros son más bajos.

Los resultados que son más bajos respecto a resultados anteriores indican que los parámetros usados inicialmente eran los más adecuados. Esto tiene sentido porque los valores iniciales ya fueron ajustados por los autores del los modelos, y por los tanto son los recomendados.

4.7. Resultados

En esta sección se detalla la solución obtenida con mejor desempeño. Además se presentan los resultados obtenidos al evaluar dicha solución con el conjunto de evaluación — cuyos *chunks* no fueron utilizados para entrenar ni validar — profundizando a nivel de cada etiqueta. También se incluyen ejemplos del comportamiento del modelo así como comparaciones con otras soluciones existentes.

4.7.1. Solución

Recapitulando las secciones anteriores, el mejor F_1 global se obtuvo con BETO al utilizar solo el primer y último *chunk* de cada sentencia para el entrenamiento, previo a las pruebas con hiperparámetros y balance de clases (cuadro 4.7).

Luego se realizaron pruebas con el objetivo de intentar balancear los resultados de las diferentes etiquetas (sección 4.4). Los mejores resultados se obtuvieron al realizar una búsqueda de hiperparámetros con valores enteros del uno al doce, utilizando BETO, y con todos los *chunks* de cada sentencia en el entrenamiento. El F_1 de las etiquetas LOC y REF mejoró, pero el F_1 de P y NP bajó (cuadros 4.15 y 4.16). Dado que P y NP son las etiquetas con más ocurrencias, el descenso del valor de F_1 global también se vió afectado, siendo menor que el F_1 global previo a la aplicación de balance de clases (diferencia de 0,037).

Si se comparan los resultados de las etiquetas P y NP con los mejores hasta el momento, se observa que la precisión disminuyó, pero el *recall* aumentó. Dado que el objetivo de este proyecto es anonimizar nombres de participantes, se considera mejor anonimizar de más, aunque sea menos preciso, que ser muy preciso en los aciertos pero dejar nombres de participantes expuestos. Teniendo esto en cuenta y considerando que la diferencia del F_1 global no es muy grande, se tomarán los resultados de BETO utilizando todos los *chunks* con balance de clases como los mejores hasta el momento.

En la sección 4.5 se realizó la comparación de tres modelos diferentes, y los mejores resultados se obtuvieron utilizando BETO. El entrenamiento de BETO es en español, mientras que XLM-RoBERTa utiliza más datos en su entrenamiento pero es multilenguaje. Esto pudo haber impactado en el entrenamiento con sentencias judiciales.

Para las últimas pruebas se realizó una búsqueda de hiperparámetros variando *learning rate*, *batch size* y *weight decay*. Los mejores resultados se obtuvieron con el modelo BETO, utilizando todos los *chunks*. Respecto a los resultados del balance de clases, el F_1 global aumentó quedando por debajo, pero muy cercano a los resultados iniciales (diferencia de 0,003).

Al aplicar a BETO utilizando todos los *chunks*, con los valores de los pe-

sos y parámetros que dieron mejores resultados, evaluando con el conjunto de **evaluación**, se obtienen los siguientes resultados:

Etiqueta	**P**	**R**	F_1
P	0.951	0.955	0.953
NP	0.798	0.934	0.861
LOC	0.555	0.761	0.642
REF	0.405	1.0	0.577

Cuadro 4.19: Resultados obtenidos al correr BETO sobre el conjunto que incluye todos los *chunks*.

El desempeño resulta bajo para aquellas etiquetas que no son P. Esto puede deberse a múltiples factores y, principalmente, a la diferencia en el número de instancias comparadas con las de P.

Cabe recordar que al momento de agregar las nuevas etiquetas al conjunto (NP, REF y LOC) no fue posible anotar todas las sentencias, por lo que algunas de las sentencias utilizadas en entrenamiento no cuentan con estas etiquetas.

Otro factor de gran importancia es el español utilizado en las sentencias judiciales que resulta muy particular al dominio. Esto incide en las componentes semánticas y sintácticas del modelo, ya que el español utilizado para entrenarlo es distinto. Ciertas palabras reciben un significado completamente distinto cuando son utilizadas dentro del mundo judicial. La palabra «autos» resulta un claro ejemplo de esto, normalmente es utilizada para referirse a vehículos; mientras que en el contexto judicial significa una resolución del tribunal o juez y generalmente en la sentencias de la BJN, precede el veredicto de la sentencia, de gran importancia semántica.

Un dominio diferente impacta el desempeño del tokenizador y por ende del modelo. Si bien los tokenizadores no deben ser entrenados como los modelos, deben ajustar su vocabulario al corpus utilizado. Esto se realiza mediante un proceso estadístico que genera el vocabulario adecuado para un conjunto de datos dado. Ya que el proceso es determinístico, existe un solo vocabulario por corpus. Contar con un vocabulario bien ajustado que tenga en cuenta las particularidades del corpus, permite lograr una mayor resiliencia a faltas ortográficas e incluir términos específicos del dominio. Esto resulta de gran importancia ya que palabras desconocidas serán tratadas como fuera de vocabulario — out of vocabulary en inglés (OOV) — las cuales llevan a pérdidas de desempeño. Las palabras OOV utilizan una mayor cantidad de tokens al ser tokenizadas, esto acarrea un mayor costo computacional al procesar las entradas y disminuye el contenido semántico que el modelo puede procesar en una secuencia. Además, como muestran Nayak et al.[59] existe una pérdida general del sentido semántico

en palabras *OOV*.

Llevado al dominio de las sentencias judiciales, encontramos un ejemplo en la palabra *vernácula*, la cual es utilizada ocasionalmente. Con el tokenizador preentrenado con BETO y removiendo el acento, obtenemos múltiples tokens. Un tokenizador alternativo entrenado con las mil sentencias disponibles, detecta solo un token (ver figura 4.5).

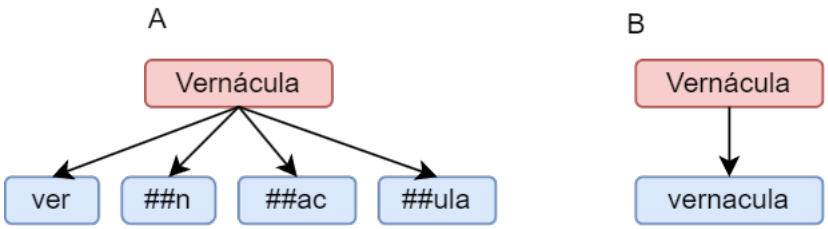


Figura 4.5: (A) Tokens generados por BETO. (B) Tokens generados por un tokenizador entrenado con sentencias judiciales.

Dado que el dominio legal difiere en gran parte con el del corpus utilizado para entrenar el modelo, el cual se basa en contenido de noticias periodísticas, es necesario entrenarlo desde cero con los nuevos tokens, tarea que requiere un gran costo computacional y un amplio corpus del cual no se dispone.

4.7.2. Comparación de resultados

Dado el uso de etiquetas ideadas específicamente para el contexto de este proyecto, no es posible una comparación directa con los resultados obtenidos con otras soluciones. Se podría considerar utilizar los modelos existentes — en particular BETO, que registra el mejor desempeño — con su clasificador reemplazado, de forma que pueda estimar P, NP, REF y LOC. Esto se descarta rápidamente dado que para obtener algún resultado valioso es necesario entrenar los pesos de dicho clasificador. BETO en tareas NER para el idioma español reporta un F_1 global de 0.884[16], si se reemplaza el clasificador y evalúa utilizando las etiquetas definidas anteriormente se obtiene un F_1 de *0.027*.

Es posible basarse en los resultados del modelo sin ajustar. Las etiquetas P, NP y REF son comprendidas por la etiqueta estándar PER y por lo tanto pueden ser comparadas de forma directa. Se consideran dos alternativas: calcular el promedio de estas etiquetas y compararlo con PER; o manteniendo el enfoque en anonimización, comparar P y PER directamente.

Además, haciendo referencia a los resultados presentados por Garat y Wonsever en “Automatic Curation of Court Documents: Anonymizing Personal Data"[2], donde ejecuta la tarea de reconocimiento sobre el mismo corpus utilizando un modelo basado en redes convolucionales provisto por la herramienta

spaCY; se centran en la etiqueta que el artículo define como PER equivalente a P, alcanzando un máximo F_1 de 0.902.

En el cuadro 4.20 se presentan las opciones mencionadas. Se incluye a modo de referencia los resultados del estado del arte en NER español alcanzados por el modelo *ACE + document context*[60], evaluados sobre el dataset CoNLL2002.

Modelo	F_{1PER}
Estado del arte	0.959
Este proyecto (solo P)	**0.953**
Este proyecto (P, NP, REF y LOC)	0.911
Garat y Wonsever	0.902
XLM-RoBERTa	0.897
BETO	0.884

Cuadro 4.20: Desempeño de PER en distintos modelos.

El modelo utilizado en este trabajo logra una mejoría del 4.4 % sobre el segundo mejor modelo en lo que respecta al reconocimiento de participantes en sentencias legales. Si bien se evalúan sobre conjuntos distintos, este resultado se encuentra a 1.7 % del estado del arte en reconocimiento general de personas.

Tiempo de entrenamiento

Cabe destacar que dado que el objetivo del proyecto es anonimizar de forma correcta las sentencias legales, se prioriza el desempeño obtenido por el modelo por sobre el costo computacional de entrenarlo. A su vez, el modelo no es ejecutado sobre dispositivos portables o de bajo rendimiento, por lo que la variable tiempo de ejecución tampoco es optimizada. A modo de referencia, el entrenamiento de BETO en una GPU con 40GB de vRAM toma en promedio seis minutos mientras que evaluarlo cinco segundos. En cambio, si se utiliza solo el procesador para entrenar el tiempo de entrenamiento aumenta a horas.

4.7.3. Ejemplos de funcionamiento

Al revisar la solución con el conjunto de evaluación se observan casos de éxito y casos de falla. A continuación se ejemplifican algunos de esos casos, siempre intercambiando los nombres reales por los de la tabla del anexo A para preservar la identidad de los participantes.

Casos de éxito

Como se mencionó anteriormente, algunas sentencias no fueron anotadas para este proyecto, por lo que no cuentan con etiquetas NP, REF o LOC, aunque si deberían. La figura 4.6 corresponde a una partición de un *chunk* cuyas

etiquetas no fueron anotadas previamente. Fueron arrojadas como errores en la solución, pero en realidad el error fue humano, la solución es correcta.

... dictada por el Sr. Juez Letrado de Primera Instancia en lo Penal de 5to. turno Dr. **NP** Daniel Tapié Santarelli .- Intervinieron en éstas actuaciones, en representación del Ministerio Público la Sra. Fiscal Letrada Nacional en lo Penal de 10mo. turno Dra. **NP** Diana Salvo y el defensor de confianza Dr. **NP** Carlos Abdala Souto .- I. U. E. 99 - 10435/2.004...

Figura 4.6: Ejemplo de sentencia no anotada para este proyecto donde la solución identificó etiquetas nuevas. Fragmento de la sentencia número i332/2007[3].

Lo mismo ocurre con sentencias que fueron anotadas para este proyecto que debido al error humano carecen de algunas etiquetas. En las figuras 4.7 y 4.8 la solución identifica las etiquetas LOC, en la figura 4.9 las etiquetas REF (la primera había sido anotada y es un acierto en la solución), en la figura 4.10 la etiqueta P, y en la figura 4.11 la etiqueta NP. Se identifican casos de buen funcionamiento para todos los tipos de etiquetas considerados.

... y vio a la pareja que por la calle **LOC** Arenal Grande cruzaron la avenida y...

Figura 4.7: Ejemplo de etiqueta LOC identificada en la solución. Fragmento de la sentencia número 368/2007[3].

... con frente a las calles **LOC** Reconquista y **LOC** Brecha, abonando por dicha transacción la suma de...

Figura 4.8: Ejemplo de etiquetas LOC identificadas en la solución. Fragmento de la sentencia número 275/2011[3].

En el anexo C se presenta una sentencia completa que fue anotada sin errores.

Casos de falla

Existen casos en los que las predicciones son completamente erróneas y otras que presentan una coincidencia parcial con la etiqueta esperada.

En las figuras 4.12 y 4.13 se muestran casos de predicciones erróneas por etiquetarse como participantes cadenas de caracteres que no se corresponden

... Y añaden: ¿Qué debe entenderse por delito mayor? Para determinarlo, expresa **REF** Crivellani no debe tomarse por base la penalidad abstractamente conminada en las varias figuras delictivas a cargo del agente, sino la pena que el Juez determine en concreto para cada uno de los delitos concurrentes; cita en su apoyo las opiniones de **REF** Impallomeni, presunto creador del sistema...

Figura 4.9: Ejemplo de etiquetas REF identificadas en la solución. Fragmento de la sentencia número 286/2011[3].

... Agrega, que dada la patología que sufre **P** Julio, el tiempo de internación que ya ha sufrido superior a los 60 días...

Figura 4.10: Ejemplo de etiqueta P identificada en la solución. Fragmento de la sentencia número 287/2011[3].

con personas ("Jueces" y "Juzgado"). Esto puede deberse a la redacción de las sentencias, que en ocasiones personifican estos términos.

La figura 4.14 contiene un ejemplo de error parcial, ya que se utilizó la etiqueta correcta (NP), pero en lugar de ser una sola anotación continua, fueron dos, una para "Ma" y otra para "Jacqueline Enrique Toledo". La sentencia cuenta con un salto de línea luego de "Ma." y esto es posible que haya contribuido a la equivocación. Esto también sucedió en otra sentencia con las mismas características (sentencia número 323/2011).

Se encuentran casos similares al anterior en los que solo se etiqueta una parte de la entidad; las figuras 4.15 y 4.16 son ejemplos de ello.

En otros casos es probable que haya influido el error humano, pero no al anotar las sentencias, si no al escribirlas. En la figura 4.17 el primer nombre del participante quedó unido al token anterior, y es probable que por esto no se haya marcado el primer nombre como parte del nombre completo del participante.

... El Señor Ministro Dr. **NP** Alfredo Gómez Tedeschi, señaló al respecto en su voto, qué...

Figura 4.11: Ejemplo de etiqueta NP identificada en la solución. Fragmento de la sentencia número 92/2011[3].

... Sostiene, asimismo, que la resistencia contumaz observada por **P** Villar en relación al requerimiento judicial que le fuera comunicado, se encuentra más cerca de la desidia o de la molestia, o de un actuar a su aire, que de la desobediencia abierta, entendida como medio típico para menoscabar el prestigio o la honra de **P** Jueces...

Figura 4.12: Ejemplo de etiqueta P identificada incorrectamente ("Jueces"). Fragmento de la sentencia número 670/2011[3].

... véase que luego el mismo médico admite el error de no haberlo especificado; no obstante debe también señalarse que el propio **P** Juzgado tomó el certificado en la forma de tiempo de inhabilitación y procesó por lesiones graves...

Figura 4.13: Ejemplo de etiqueta P identificada incorrectamente ("Juzgado"). Fragmento de la sentencia número 273/2011[3].

... Por el contrario, surge fehacientemente probado que dicha información fue correctamente proporcionada a la entonces indagada, no habiendo existido afectación alguna en las garantías del debido proceso, como surge del Informe rendido al respecto por quién era la Magistrado Titular de la Sede de origen, en esa etapa procesal, Dra. **NP** Ma.
NP Jacqueline Enrique Toledo...

Figura 4.14: Ejemplo de error parcial con etiqueta NP. Fragmento de la sentencia número 92/2011[3].

... Asiste razón al Sr. defensor de que en la audiencia de fs. 345 de los autos principales caratulados " **P** Villar, Julio ; **P** Segui, Ma . Gisela - Un delito de Falsificación o alteración de certificados"...

Figura 4.15: Ejemplo de error parcial con etiqueta P (segunda etiqueta identificada). Fragmento de la sentencia número i328/2007[3].

... dice que estando de recorrida ven correr por **LOC** Blanes un NN "...y toma por **LOC** Chaná , saliendo atrás de él..." lo logran detener en Joaquín **LOC** de Salterain y **LOC** Guan á...

Figura 4.16: Ejemplo de errores parciales con etiquetas LOC (últimas dos etiquetas). Fragmento de la sentencia número 368/2007[3].

... Concluye la Sra. Fiscal que resultó probado que el acusadoLucas **P** Gabriel Nievas Padilla debe responder penalmente como autor de un delito de abigeato...

Figura 4.17: Ejemplo de error parcial con etiquetas P. Fragmento de la sentencia número 416/2007[3].

Capítulo 5

Vinculación de entidades

En el proceso de anonimización de una sentencia se deben reemplazar todos los nombres utilizados para referir a una persona con el mismo nombre ficticio para su anonimización. Es de interés, una vez aplicado el reconocimiento de entidades, establecer cuántos participantes distintos hay y qué instancias de la etiqueta PER pertenecen a cada uno.

En el siguiente ejemplo se aplica NER sobre la figura 3.9, obteniendo:

... Para resolución en estos autos caratulados: “ **PER** SEGUI, MA. GISELE en autos caratulados: “ **PER** VILLAR, JULIO C/ **PER** SEGUI, MARIA - DIV. POR CAUSAL R/D Fa: 2-31656/2011” C/ SRA. JUEZA LETRADA DE FAMILIA DE 26º TURNO RECURSO DE QUEJA POR DENEGACION DE APELACION - Nº de Expediente 67-12/2012” ... 2.- A fs. 23 a 26 compareció la representante de la Sra. **PER** Gisela Segui , interponiendo recurso de queja contra la sentencia Nº 419/2012 por denegación del efecto suspensivo de recurso de apelación contra la providencia Nº 5413/2011, ... II.- En la especie, en el proceso que por divorcio iniciara el Sr. **PER** Villar contra la Sra. **PER** Segui , ...

Figura 5.1: Resultado de aplicar NER. Fragmento de la sentencia número 174/2012[3].

En esta sentencia (figura 5.1) se identifican dos participantes: María Gisela Segui y Julio Villar. La sentencia alude a ambos participantes de distintas maneras:

- **María Gisela Segui**: {SEGUI, MA. GISELE; SEGUI, MARIA; Gisela Segui; Segui}
- **Julio Villar**: {VILLAR, JULIO; Villar}

Este requerimiento se aborda en dos etapas. Primero, se genera una métrica de similitud entre nombres. Luego siguiendo el mismo proceso que Garat y Wonsever[2], se generan grupos —también conocidos como clusters— de nombres para vincular las entidades entre sí. Como prueba de concepto se implementa la distancia mediante una red neuronal siamesa.

Una métrica de similaridad definida para este contexto debería asignar valores cercanos a los nombres de una misma persona y considerar distantes a los nombres cruzados. Luego, el algoritmo de agrupamiento hace uso de esta métrica para identificar que hay dos personas (o grupos) distintos.

Este capítulo describe la implementación de la red neuronal utilizada para vincular las entidades participantes dentro de las sentencias judiciales almacenadas en la BJN. Se provee una breve descripción del conjunto de datos utilizados para entrenarla, también se analiza su implementación a través de la librería Keras y se presentan las distintas pruebas ejecutadas con fin de mejorar su desempeño al máximo posible. Por último se presentan los resultados obtenidos y se los compara con aquellos obtenidos por Garat y Wonsever en el artículo "*Automatic Curation of Court Documents*"[2].

Nota: Las pruebas descritas a lo largo de este capítulo se ejecutaron en una computadora personal con las siguientes especificaciones:

- GPU: Tarjeta NVIDIA RTX 2070 CON 8GB vRAM.
- RAM: 16GB DDR4 3000 MHZ.
- CPU: Intel Core i5-8600K 3.60GHZ.

5.1. Procesamiento de datos

Con el objetivo de poder agrupar nombres que identifican a una misma persona, se construye un nuevo conjunto de datos compuesto por pares de nombres de participantes obtenidos de las sentencias judiciales. Este conjunto hace uso de las anotaciones generadas para el problema de reconocimiento de entidades nombradas.

La forma del conjunto generado resulta muy sencilla, cada entrada es compuesta por dos nombres y una etiqueta 1 o 0 que especifica si los nombres pertenecen a la misma persona o no, respectivamente (ejemplo en el cuadro 5.1).

Para armar el conjunto se reutilizan las etiquetas definidas para anotar sen-

Nombre A	Nombre B	Misma Persona
Juan Pablo	Juan P.	1
Juan Pablo	Pedro Gomez	0
...	...	...

Cuadro 5.1: Ejemplo de entradas del dataset.

tencias. Por su construcción, cada etiqueta de participante dentro de una sentencia en particular —AA, BB, CC, etc.— refiere a una persona única. Entonces, para cada sentencia se genera una entrada del *dataset* por cada combinación posible de una etiqueta indicando que pertenecen a la misma persona. Luego por cada instancia de una etiqueta, se genera una entrada por cada combinación con el resto de las instancias de otras etiquetas de la sentencia, indicando que son personas distintas.

Las referencias repetidas a una persona se descartan para evitar generar pares iguales que reduzcan la calidad del conjunto al agregar ruido, los nombres iguales generan vectores iguales al pasar por la red siamesa y su distancia siempre será cero. Además, por la misma razón se descartan aquellas entradas para las cuales su opuesto simétrico ya se haya considerado. La red implementada para aprender la métrica es del tipo siamesa y las subredes comparten pesos, por lo cuál a entradas simétricas es asignada la misma distancia.

A modo de ejemplo, se considera una sentencia que solo incluye dos participantes: Juan Pablo y Micaela. El primero es referenciado de dos formas distintas, mientras que la última es referida solo por su nombre:

AA: {Juan Pablo, Juan} ; **BB:** {Micaela}

Se obtienen las siguientes entradas al aplicar estas reglas sobre la sentencia (cuadro 5.2):

Nombre A	Nombre B	Misma Persona
Juan Pablo	Juan	Sí
Juan Pablo	Micaela	No
Juan	Micaela	No

Cuadro 5.2: Conjunto de datos resultante a partir de la sentencia de ejemplo.

Cabe destacar que las etiquetas NP y REF se ignoran ya que no se puede distinguir a qué persona pertenece cada instancia. Por ejemplo, dos ministros diferentes serán etiquetados como NP.

Las variantes resultantes por falta de tildes, distinto uso de mayúsculas, o aquellas que incluyen puntuación dentro del nombre, no aportan valor al entrenamiento de la red; al contrario, puede considerarse que agregan ruido al conjunto. Para remover este ruido, se normalizan los nombres convirtiéndolos a minúscula, removiendo caracteres de puntuación y acentos.

El conjunto generado a partir de las sentencias judiciales disponibles cuenta con 8.033 entradas. De ellas, 1.450 ($18,05\,\%$) de los pares pertenecen a la misma persona y 6.583 ($81,95\,\%$) a personas diferentes (figura 5.2).

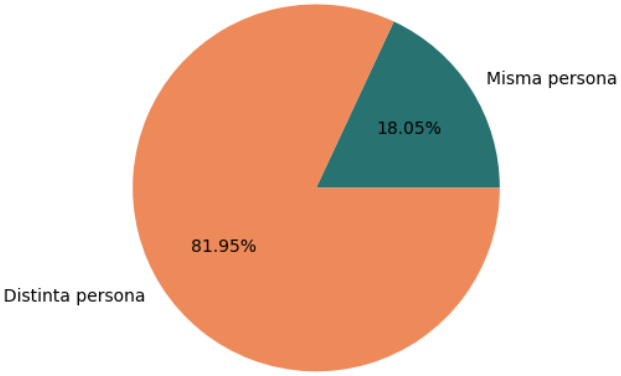


Figura 5.2: Visualización del conjunto de nombres

5.2. Distancia entre nombres

Dada su arquitectura, las redes siamesas son muy usadas para comparar objetos. Estás permiten proyectar los objetos en un espacio vectorial, el cuál es ajustado durante el entrenamiento para que capture las cualidades semánticas más interesantes. Una vez que los objetos están representados como vectores, cualquier medida de distancia vectorial es aplicable.

Existen una multitud de arquitecturas de redes siamesas para comparar similitud entre textos, la mayoría —y las que mejores resultados obtienen— utilizan embeddings y redes LSTM o BI-LSTM para construirlas[25][61][62]. Teniendo esto en cuenta, la red propuesta implementa una arquitectura similar que resulta sencilla y efectiva.

La red cuenta con dos entradas de tipo texto —donde cada una contiene un nombre— que pasan a ser vectorizadas y luego convertidas a embeddings. Una vez obtenidos los embeddings, se alimenta con ellos una red BI-LSTM que los codifica según sus pesos. Por último los vectores resultantes pasan por una capa de *dropout* (o retiro) para evitar sobreajustar la red al conjunto de datos. Se calcula su distancia utilizando una función ya existente —del coseno, por

ejemplo— y para ajustar; se compara la distancia a cero o uno, dependiendo de si los nombres pertenecen a la misma persona o no.

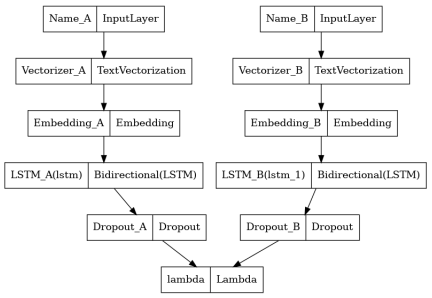


Figura 5.3: Visualización de la red siamesa. Imagen generada utilizando Keras.

La vectorización de los nombres se lleva a cabo con la capa de preprocesamiento *TextVectorization*[1] que toma los strings originales y los convierte a un vector de índices enteros por cada token (figura 5.4). Los índices apuntan a un vocabulario que es aprendido del conjunto de datos. En este caso, la división en tokens se hace a nivel de cáracter unicode, por lo que el vocabulario termina siendo el alfabeto, el cáracter espacio y alguno más proveniente de anotaciones que contengan números o similares. El vector resultante tiene el mismo largo que el nombre de entrada.

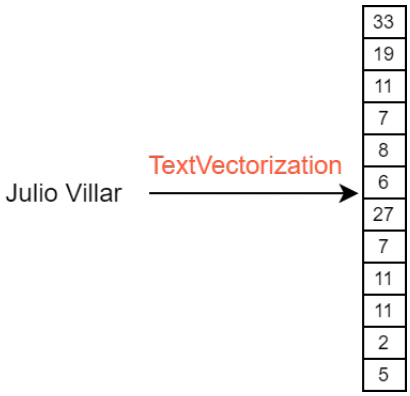


Figura 5.4: Vectorización del nombre Julio Villar

Luego, a partir de los vectores se obtienen embeddings mediante la capa *Embedding*[2], que convierte los vectores de índices a vectores de tamaño fijo. Los valores contenidos dejan de ser enteros y pasan a ser números reales. El tamaño

[1] https://keras.io/api/layers/preprocessing_layers/text/text_vectorization/
[2] https://keras.io/api/layers/core_layers/embedding/

del vector resultante es un hiperparámetro que se ajusta posteriormente con una búsqueda de hiperparámetros.

Los embeddings son la entrada a la capa *BI-LSTM*[3] que se encarga de codificarlos. También se aplica una capa de *Dropout*[4] que aleatoriamente cambia el valor de entradas a cero dependiendo de la tasa utilizada, para así introducir ruido y prevenir sobreajuste[63].

Los vectores obtenidos luego de pasar por todas estas capas son comparados mediante una función de distancia vectorial. Existen varias distancias que pueden ser utilizadas pero para mantener los experimentos acotados se seleccionan dos ampliamente utilizadas: coseno y Manhattan.

$$d_{coseno} = \frac{\sum_{n=1}^{n} A_i B_i}{\sqrt{\sum_{n=1}^{n} A_i^2 \sum_{n=1}^{n} B_i^2}} \quad ; \quad d_{Manhattan} = \sum n = 1^n |A_i - B_i|$$

Por último se compara la distancia obtenida con el valor 0 o 1 de la etiqueta que indica si los nombres pertenecen a la misma persona o no. Considerando que la distancia se encuentre dentro de un valor α del valor de la etiqueta. El valor α es un hiperparámetro que es ajustado posteriormente en una búsqueda.

Cómo es común, en caso de que la red prediga un valor incorrecto se ajustan los pesos usando una función de pérdida. Dado que en este caso se busca aprender una métrica, también se evalúa el uso de una pérdida de margen. La pérdida de margen además de considerar la predicción como correcta o no, tiene en cuenta la distancia relativa entre las entradas a la hora de ajustar la red.

La pérdida **contrastiva** se encarga de acercar o distanciar pares de valores similares o distintos (respectivamente) dentro del espacio vectorial. Se define un margen que actúa como la distancia mínima que deben tener dos puntos para ser considerados similares o no. Sea **L** la pérdida contrastiva, se define como:

$$L = (1 - E) * ||x - y||^2 + E * max(0, m - ||x - y||^2)$$
$$\text{con } E = \begin{cases} 0 & \text{si } x \text{ es similar a } y \\ 1 & \text{si } x \text{ es distinto a } y \end{cases}$$

La métrica aprendida por el modelo se ajusta utilizando la etiqueta "son similares" del dataset de nombres. Una distancia igual o cercana a cero indica que los nombres pertenecen a la misma persona, mientras que el equivalente para uno indica que las personas son distintas.

La métrica *BinaryAccuracy* de Keras calcula el porcentaje de valores predichos que se corresponden con las etiquetas verdaderas, dado que la métrica se corresponde a una distancia y no un resultado binario —uno o cero— se fija un umbral de $0,5$ para considerar las predicciones similares o no.

[3] https://keras.io/api/layers/recurrent_layers/lstm/
[4] https://keras.io/api/layers/regularization_layers/dropout/

Para comprobar el comportamiento correcto del modelo se procede a ajustarlo. Como con el modelo anterior, el conjunto de datos se divide en tres subconjuntos: entrenamiento, validación, evaluación; compuestos por el 80 %, 10 % y 10 % de los datos disponibles respectivamente. A modo de prueba y sin utilizar un criterio en particular, los hiperparámetros se fijan a los siguientes valores:

Hiperparámetro	**Valor**	**Descripción**
lstm_units	128	Cantidad de unidades LSTM dentro de cada subred
distance_name	cosine_distance	Fun. de distancia a utilizar para comparar los vectores obtenidos
loss_name	contrastive_loss	Fun. de pérdida a utilizar para ajustar la red
learning_rate	1e-2	Tasa de ajuste de la subred
accuracy_threshold	0,5	Umbral de precisión al comparar predicciones y etiquetas
dropout_rate	0,1	Tasa de dropout aplicado a la subred
embedding_dim	256	Dimensión de los vectores de embeddings
max_features	500	Tamaño máximo del vocabulario usado para vectorizar nombres
epochs	100	Número de épocas
batch_size	16	Tamaño de bache

Cuadro 5.3: Valores elegidos para evaluar el desempeño del modelo.

El entrenamiento del modelo toma un minuto y veintidós segundos en completar —para esta combinación de hiperparámetros— y alcanza una precisión binaria en el conjunto de validación de $0,875$ (figura 5.5). Si bien se fijó el número de épocas a cien, el modelo detiene su entrenamiento antes —en la época trece— al no mejorar la pérdida de validación en las últimas tres épocas (figura 5.6).

A modo de referencia, se calcula la precisión binaria para las cuatro mejores distancias listadas por el trabajo de Garat y Wonsever con el conjunto de datos utilizado para entrenar la red y se muestran los resultados en el cuadro 5.4.

5.3. Búsqueda de hiperparámetros

Al igual que en el capítulo anterior, se realiza una búsqueda de hiperparámetros para optimizar el desempeño del modelo. En este caso, dada la cantidad

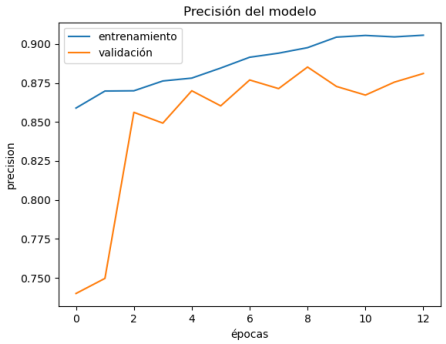


Figura 5.5: Precisión en entrenamiento y validación en función del número de época.

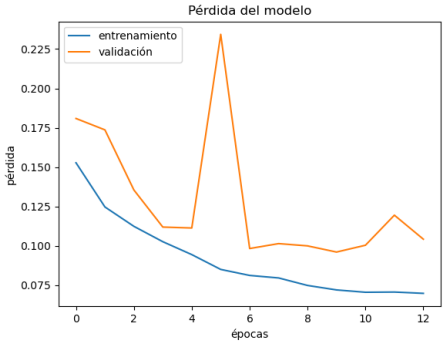


Figura 5.6: Pérdida en entrenamiento y validación en función del número de época.

de combinaciones posibles, la búsqueda exhaustiva se reemplaza por un optimizador Bayesiano. En este tipo de búsqueda se tienen en cuenta los resultados obtenidos anteriormente para formar un modelo probabilístico capaz de predecir el desempeño de distintas combinaciones de parámetros. La búsqueda luego prioriza las combinaciones que el modelo estime son las mejores. Con cada prueba ejecutada por la búsqueda se agregan datos que permiten refinar el modelo, y de forma iterativa, alcanzar mayor desempeño.

La principal diferencia con la búsqueda exhaustiva u aleatoria es que el optimizador busca mejores combinaciones tomando decisiones informadas. Esto lleva generalmente a una mejor exploración del espacio de búsqueda ya que se profundizan las áreas que llevan a mejores resultados y a un menor número de ejecuciones necesarias.

De forma de acotar el tiempo de entrenamiento del modelo, se agrega un

Distancia	Precisión
Red Siamesa	**0.875**
Jaccard	0.847
Bag	0.823
Overlap	0.319
Sorensen	0.681

Cuadro 5.4: Precisión binaria obtenida por las distintas distancias al calcular la similitud entre nombres.

criterio de finalización temprana donde cada ejecución se da por terminada si su pérdida de validación no mejora en tres épocas.

Dado que el tiempo de ejecución de cada prueba es menor que para el capítulo anterior, se pueden llevar a cabo en un ordenador personal, se consideran más hiperparámetros y más variaciones de valores. Aún así, considerando que las combinaciones resultan bastante extensas, se limita el número de combinaciones probadas a 250. La búsqueda completa en un día, 16 horas y 8 minutos. Los hiperparámetros considerados y sus rangos de valores son los siguientes:

Nombre	Valores
lstm_units	min=128, máx=1024, step=128
distance_name	{cosine_distance, manhattan_distance}
loss_name	{contrastive_loss, binary_crossentropy, mean_absolute_error}
learning_rate	{1e-2, 1e-3, 1e-4}
accuracy_threshold	min=0.1, máx=0.5, step=0.1
dropout_rate	min=0.1, máx=0.5, step=0.1
embedding_dim	{128, 256, 512, 1024}
max_features	{100, 500, 1000}
epochs	{100, 200, 300}
batch_size	min=4, máx=64, step=4

La mejor combinación de hiperparámetros hallada por la búsqueda alcanza un valor de precisión binaria en validación de 0,942, lo cual constituye una mejora del 7,64 % sobre lo obtenido anteriormente. En el cuadro 5.5 se listan las diez mejores ejecuciones.

Entrenando la red con los mejores parámetros alcanzados se obtiene una precisión sobre el conjunto de evaluación de 0,916. El comportamiento de la

Hiperparámetro	**Prueba 84**	**Prueba 107**	**Prueba 242**	**Prueba 126**	**Prueba 93**	**Prueba 170**	**Prueba 80**	**Prueba 83**	**Prueba 91**	**Prueba 166**
lstm_units	**1024**	1024	896	1024	1024	1024	1024	1024	1024	896
distance_function	**cosine_distance**	cosine_distance	cosine_distance	cosine_distance	cosine_distance	cosine_distance	cosine_distance	cosine_distance	cosine_distance	cosine_distance
loss_function	**contrastive_loss**	contrastive_loss	contrastive_loss	contrastive_loss	contrastive_loss	contrastive_loss	contrastive_loss	contrastive_loss	contrastive_loss	contrastive_loss
learning_rate	**1e-05**	1e-05	1e-05	1e-05	1e-05	1e-05	1e-05	1e-05	1e-05	1e-05
accuracy_threshold	**0.5**	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
dropout_rate	**0.1**	0.1	0.2	0.3	0.1	0.1	0.3	0.1	0.1	0.2
embedding_dim	**256**	256	512	256	256	512	256	256	256	512
max_features	**1000**	500	500	1000	1000	500	1000	1000	1000	500
batch_size	**4**	4	4	4	4	4	4	4	4	4
epochs	**100**	100	200	100	100	100	100	100	100	200
Precisión	**0.9419**	0.9419	0.9391	0.9378	0.9364	0.9364	0.9350	0.9350	0.9350	0.9350

Cuadro 5.5: Hiperparámetros y precisión de las diez mejores pruebas.

pérdida se puede apreciar en la figura 5.7 y la precisión en la figura 5.8.

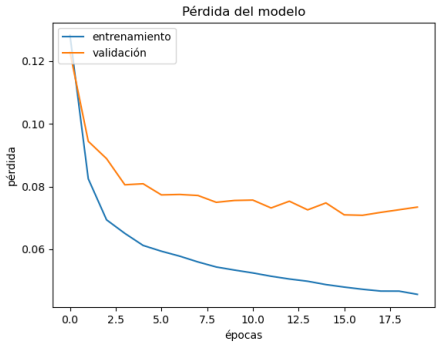


Figura 5.7: Pérdida en entrenamiento y validación en función del número de época.

5.4. Clusterización de nombres

Hasta ahora se cuenta con una red neuronal entrenada que provista de dos nombres es capaz de producir un valor de similaridad —medido como distancia— entre ellos. Para poder vincular las entidades a cada participante de una sentencia es necesario poder identificar cuántos participantes hay en la sentencia, agrupar conjuntos de nombres similares y asignarlos a los participantes que correspondan. Se resuelve este problema de la misma manera que Garat y Wonsever en "*Automatic Curation of Court Documents: Anonymizing Personal Data*"[2], mediante el uso de agrupamiento jerárquico no supervisado (*Unsupervised hierarchical clustering* en inglés).

En particular se implementa un agrupamiento jerárquico aglomerativo, el cual tiene un enfoque constructivo ascendente y funciona de forma iterativa. Inicialmente cada nombre disponible es asignado a un grupo distinto, luego se calcula la distancia entre cada grupo y se unen los dos más cercanos, removiendo

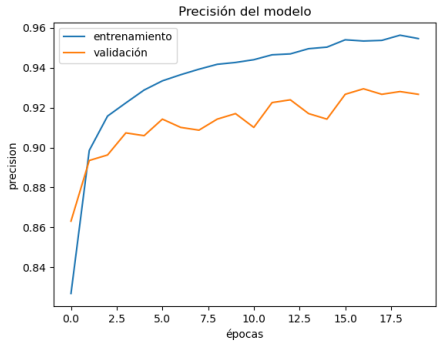


Figura 5.8: Precisión en entrenamiento y validación en función del número de época.

los grupos individuales obsoletos. Esto se repite hasta llegar al criterio de fin, que puede ser basado en el número de grupos finales buscados o en un umbral de distancia máxima en el que se considera a los grupos como demasiado lejanos para ser unidos.

En este caso, para cada sentencia judicial disponible se toma su lista de entidades, para las cuales se computa su matriz de distancia utilizando la red neuronal obtenida en la sección anterior. Esta matriz junto con las entidades constituye la entrada al algoritmo de agrupamiento. Por otro lado, como el número de participantes varía de sentencia a sentencia, se fija el criterio de fin a un umbral α (que será ajustado en pruebas posteriores). El algoritmo se detiene una vez que no existan grupos a una distancia menor que la del umbral α.

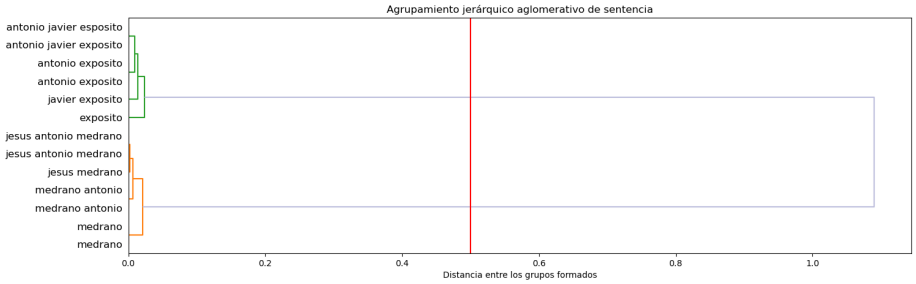


Figura 5.9: Dendrograma resultante de agrupar los participantes de una sentencia, nombres anonimizados.

Los diagramas de la figuras 5.9 y 5.10 ejemplifican el proceso de agrupamiento de las entidades de una sentencia judicial. Cada unión entre dos valores, o cada línea vertical, representa la formación de un grupo. La línea vertical roja en la figura 5.9 representa el punto de corte del algoritmo —pasado ese umbral

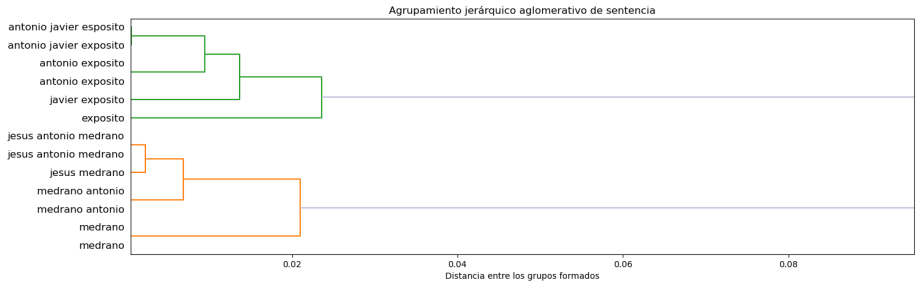


Figura 5.10: Acercamiento al principio del dendrograma.

no se siguen uniendo grupos— en este caso fijando α a un valor de 0,45. La cantidad de grupos identificados está dada por la cantidad de líneas horizontales con las que intersecta el umbral, en este caso dos.

El algoritmo utilizado en la implementación es **AgglomerativeClustering** de la librería Scikit-Learn. Al igual que en el artículo citado, se elige la opción promedio como criterio de enlace entre grupos; esto hace que el algoritmo considere la distancia entre dos grupos como el promedio de las distancias entre cada par de elementos. De igual manera, a modo de comparar los resultados obtenidos por las otras distancias mencionadas en el artículo, se calculan las siguientes métricas de uso común en clustering:

- **Completitud**[5]. Calcula si los grupos obtenidos contienen todos los elementos de su clase correspondiente.
- **Homogeneidad**[6]. Calcula si los grupos obtenidos contienen elementos de una sola clase.
- **Métrica V**[7]. Calcula la medida armónica de la completitud y homogeneidad.
- **Índice Rand ajustado**[8]. El Índice Rand calcula la similitud entre dos grupos, el Índice Rand ajustado además tiene en cuenta la aleatoriedad como un factor al agrupar.

El hiperparámetro del algoritmo, α, es evaluado con distintos valores para encontrar aquel que maximiza el desempeño. Se evalúa un rango partiendo de $0,05$ hasta $0,95$ considerando cada incremento de $0,05$. Análogo al proceso utilizado por Garat y Wonsever. En el cuadro 5.6 se presentan los resultados obtenidos para cada valor de α.

[5]https://scikit-learn.org/stable/modules/generated/sklearn.metrics.completeness_score.html
[6]https://scikit-learn.org/stable/modules/generated/sklearn.metrics.homogeneity_score.html
[7]https://scikit-learn.org/stable/modules/generated/sklearn.metrics.v_measure_score.html
[8]https://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_rand_score.html

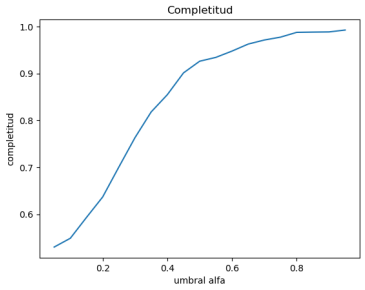
α	**comp**	**homo**	**vmeasure**	**ari**
0.05	0.530014	0.998994	0.622019	0.311490
0.10	0.548770	0.998633	0.638589	0.342288
0.15	0.593259	0.998398	0.679677	0.403228
0.20	0.636817	0.997224	0.714575	0.481822
0.25	0.700641	0.996543	0.765475	0.592941
0.30	0.763446	0.994152	0.817208	0.677467
0.35	0.818235	0.991998	0.861245	0.749144
0.40	0.855034	0.985174	0.884129	0.798995
0.45	0.901534	0.980639	0.919225	0.851036
0.50	0.926440	0.970342	0.933604	0.879049
0.55	**0.934598**	**0.959802**	**0.933048**	**0.882487**
0.60	0.948027	0.941933	0.933874	0.880866
0.65	0.962982	0.923045	0.932714	0.876806
0.70	0.971671	0.894127	0.913730	0.853514
0.75	0.977789	0.864505	0.897611	0.829250
0.80	0.987990	0.809870	0.855171	0.774250
0.85	0.988548	0.754346	0.811890	0.721032
0.90	0.988945	0.707333	0.773326	0.675528
0.95	0.992928	0.633105	0.700100	0.604386

Cuadro 5.6: Resultados obtenidos al agrupar variando α

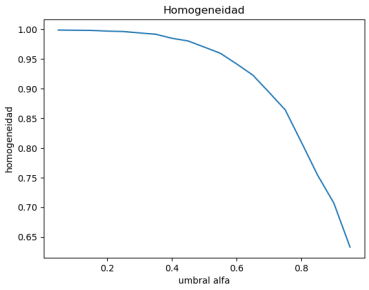
Se puede ver que $\alpha = 0{,}55$ obtiene los mejores resultados (ver figuras 5.11a, 5.11b, 5.11c y 5.11d).

5.5. Resultados

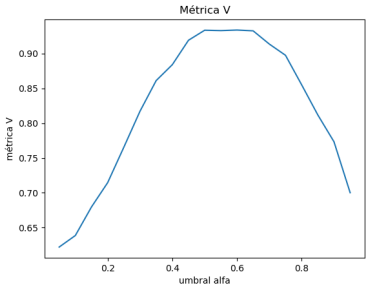
Resumiendo las secciones anteriores, se obtiene una red neuronal siamesa capaz de distinguir si dos nombres pertenecen a la misma persona o no basada en su distancia. Cuenta con una precisión relativamente alta de 0,916 por lo que parece ser una buena candidata para ser utilizada como métrica en algoritmos de agrupamiento que busquen vincular entidades. Debido a restricciones de tiempo no se pudo experimentar con otras arquitecturas distintas, reemplazando por ejemplo las unidades LSTM por GRU o agregando una capa de atención, estas opciones se mencionan como posible trabajo a futuro al final del informe.



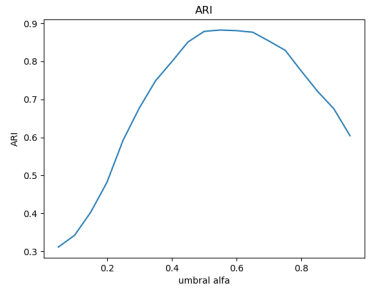
(a) Completitud en función de α



(b) Homogeneidad en función de α



(c) Métrica V en función de α



(d) ARI en función de α

Buscando medir el desempeño de la red cuando es usada para la vinculación de entidades, se siguen los experimentos de "Automatic Curation of Court Documents"[2]. Se utiliza un algoritmo de agrupación aglomerativa que utiliza la red para calcular la distancia entre cada par de nombres. El algoritmo se comporta mejor cuando su umbral de distancia α se fija en 0,55, resultando en un ARI de 0,882 que se encuentra por debajo de los obtenidos por Garat y Wonsever (en particular, su mejor valor fue de 0,959). El cuadro (5.7) compara la red siamesa con los cuatro mejores resultados obtenidos por ellos. Los nombres de los experimentos son compuestos por la distancia utilizada y el umbral α fijado. Exactitud representa el porcentaje de documentos que son agrupados de forma correcta.

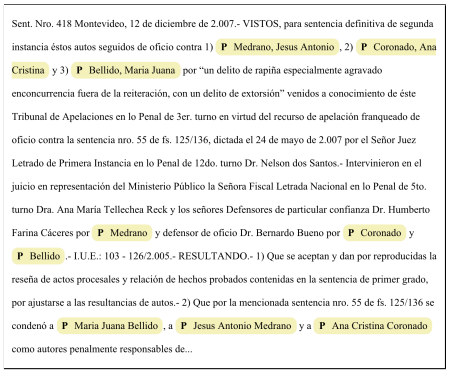
Experimento	Comp	Homo	V-Measure	ARI	Exactitud
Overlap 0.45	**0.967**	**0.986**	**0.970**	**0.950**	**0.810**
Jaccard 0.75	0.964	0.978	0.961	0.937	0.790
Sorensen 0.60	0.963	0.978	0.960	0.936	0.786
Bag 0.85	0.972	0.953	0.949	0.920	0.758
Red Siamesa 0.55	0.935	0.960	0.933	0.882	0.669

Cuadro 5.7: Comparación de los cuatro mejores resultados obtenidos en Automatic Curation of Court Documents y la red siamesa presentada en este capítulo.

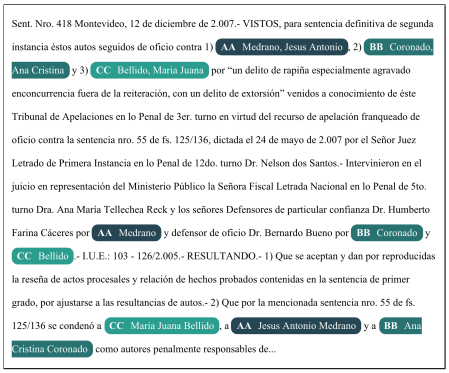
Si bien el desempeño es inferior al de pruebas anteriores, no debe descartarse al aprendizaje profundo como alternativa a las distancias tradicionales usadas para agrupar los nombres. Es posible que una mayor cantidad de datos de entrenamiento o una arquitectura similar aumenten el desempeño de la solución. Las redes siamesas no necesitan reentrenarse para comparar nuevos elementos y permiten ajustar la función de distancia al dominio de sentencias judiciales, capturando heurísticas y patrones presentes en estas. Esto no es posible con métricas ya definidas.

A continuación se presentan dos ejemplos del funcionamiento de la solución, primero se agrupan las entidades en una sentencia judicial (reemplazando los nombres por los del anexo A) y luego se demuestra el comportamiento de la solución cuando es utilizada sobre un dominio diferente al que fue entrenada.

Para el primer ejemplo, se etiqueta un extracto de la sentencia 418/2007 de forma manual; solo se etiquetan aquellas entidades que se ven involucradas en los hechos de la sentencia, al agruparlas con la solución se obtiene un resultado perfecto con $\alpha = 0{,}85$.


Sent. Nro. 418 Montevideo, 12 de diciembre de 2.007.- VISTOS, para sentencia definitiva de segunda instancia éstos autos seguidos de oficio contra 1) P Medrano, Jesus Antonio , 2) P Coronado, Ana Cristina y 3) P Bellido, Maria Juana por "un delito de rapiña especialmente agravado enconcurrencia fuera de la reiteración, con un delito de extorsión" venidos a conocimiento de éste Tribunal de Apelaciones en lo Penal de 3er. turno en virtud del recurso de apelación franqueado de oficio contra la sentencia nro. 55 de fs. 125/136, dictada el 24 de mayo de 2.007 por el Señor Juez Letrado de Primera Instancia en lo Penal de 12do. turno Dr. Nelson dos Santos.- Intervinieron en el juicio en representación del Ministerio Público la Señora Fiscal Letrada Nacional en lo Penal de 5to. turno Dra. Ana María Tellechea Reck y los señores Defensores de particular confianza Dr. Humberto Farina Cáceres por P Medrano y defensor de oficio Dr. Bernardo Bueno por P Coronado y P Bellido .- I.U.E.: 103 - 126/2.005.- RESULTANDO.- 1) Que se aceptan y dan por reproducidas la reseña de actos procesales y relación de hechos probados contenidas en la sentencia de primer grado, por ajustarse a las resultancias de autos.- 2) Que por la mencionada sentencia nro. 55 de fs. 125/136 se condenó a P Maria Juana Bellido , a P Jesus Antonio Medrano y a P Ana Cristina Coronado como autores penalmente responsables de...

(a) Resumen de la sentencia 418/2007 con las personas etiquetadas


Sent. Nro. 418 Montevideo, 12 de diciembre de 2.007.- VISTOS, para sentencia definitiva de segunda instancia éstos autos seguidos de oficio contra 1) AA Medrano, Jesus Antonio , 2) BB Coronado, Ana Cristina y 3) CC Bellido, Maria Juana por "un delito de rapiña especialmente agravado enconcurrencia fuera de la reiteración, con un delito de extorsión" venidos a conocimiento de éste Tribunal de Apelaciones en lo Penal de 3er. turno en virtud del recurso de apelación franqueado de oficio contra la sentencia nro. 55 de fs. 125/136, dictada el 24 de mayo de 2.007 por el Señor Juez Letrado de Primera Instancia en lo Penal de 12do. turno Dr. Nelson dos Santos.- Intervinieron en el juicio en representación del Ministerio Público la Señora Fiscal Letrada Nacional en lo Penal de 5to. turno Dra. Ana María Tellechea Reck y los señores Defensores de particular confianza Dr. Humberto Farina Cáceres por AA Medrano y defensor de oficio Dr. Bernardo Bueno por BB Coronado y CC Bellido .- I.U.E.: 103 - 126/2.005.- RESULTANDO.- 1) Que se aceptan y dan por reproducidas la reseña de actos procesales y relación de hechos probados contenidas en la sentencia de primer grado, por ajustarse a las resultancias de autos.- 2) Que por la mencionada sentencia nro. 55 de fs. 125/136 se condenó a CC Maria Juana Bellido , a AA Jesus Antonio Medrano y a BB Ana Cristina Coronado como autores penalmente responsables de...

(b) Resultado de agrupar con $\alpha = 0{,}85$

Para el segundo ejemplo, se muestra un artículo de un portal de noticias que menciona varios individuos y se refiere a ellos de distintas formas[9]. El español utilizado resulta más común que la jerga judicial. Los nombres completos se sustituyen en general por el apellido, en lugar de usar transposiciones separando nombres y apellidos con comas, como es usual en las sentencias. El ejemplo es un texto compuesto por distintos extractos del texto original, este se incluye en el anexo debido a su largo.

Tras la muerte de la actriz **P** Silvina Luna, de quien este jueves se confirmó que falleció luego de un largo proceso de deterioro de su salud, el nombre de **P** Aníbal Lotocki volvió a ser centro de polémica en Argentina... En julio de este año, por pedido del fiscal **P** Abraldes, la Sala 3 de la Cámara Nacional de Casación en lo Criminal y Correccional ordenó la inhabilitación cautelar de **P** Lotocki... En junio de este año el juez **P** Luis Schelgel procesó sin prisión preventiva a **P** Lotocki por el homicidio simple con dolo eventual del paciente **P** Rodolfo Christian Zárate, que falleció en abril de 2021... **P** Ileana Lombardo, abogada de **P** Lotocki, dijo a CNN que la muerte de **P** Luna "no debería" afectar la situación legal de su defendido...

Figura 5.13: Resumen del artículo con las personas etiquetadas

Aplicando reconocimiento de entidades de forma manual sobre el texto (figura 5.13) se identifican diez entidades pertenecientes a la categoría de personas. Utilizando estas entidades, se aplica el algoritmo de aglomeración jerárquica descrito anteriormente para todos los umbrales α desde 0.05 a 0.95 inclusive.

[9] `https://cnnespanol.cnn.com/2023/09/05/quien-es-anibal-lotocki-medico-que-opero-silvina-luna-orix-arg/`

α	comp	homo	vmeasure	ari
...	...	...	...	...
0.15	0.815731	1.000000	0.898515	0.558824
0.20	0.920696	1.000000	0.958711	0.910180
0.25	0.920696	1.000000	0.958711	0.910180
0.30	0.920696	1.000000	0.958711	0.910180
0.35	0.913865	0.913865	0.913865	0.830827
...	...	...	...	...
0.55	1.000000	0.758406	0.862606	0.672489
...	...	...	...	...

Cuadro 5.8: Resultados obtenidos al agrupar el extracto variando α

Se hace foco en dos valores de umbral particulares: 0.55 y 0.30, el primero es el umbral que mejor agrupa el conjunto de sentencias judiciales mientras que el segundo es el umbral que mejor agrupa las entidades de este extracto. $\alpha = 0{,}55$ empeora el ARI obtenido de 0.882 a 0.672 y el vmeasure de 0.933 a 0.862, indicando que textos de otros contextos pueden requerir recalibrar el umbral a utilizar. En cambio, $\alpha = 0{,}30$ mejora tanto el ARI elevándolo a 0.910, como el vmeasure a 0.958. Es posible entonces que la medida de distancia aprendida por la red se comporte bien frente a otros conjuntos de datos, ya que en este caso basta con cambiar α para obtener aún mejores resultados. Las figuras siguientes muestran cómo fueron agrupadas las entidades para cada valor de alfa:

Tras la muerte de la actriz **AA** Silvina Luna, de quien este jueves se confirmó que falleció luego de un largo proceso de deterioro de su salud, el nombre de **BB** Aníbal Lotocki volvió a ser centro de polémica en Argentina... En julio de este año, por pedido del fiscal **CC** Abraldes, la Sala 3 de la Cámara Nacional de Casación en lo Criminal y Correccional ordenó la inhabilitación cautelar de **BB** Lotocki... En junio de este año el juez **DD** Luis Schelgel procesó sin prisión preventiva a **BB** Lotocki por el homicidio simple con dolo eventual del paciente **BB** Rodolfo Christian Zárate, que falleció en abril de 2021... **CC** Ileana Lombardo, abogada de **BB** Lotocki, dijo a CNN que la muerte de **AA** Luna "no debería" afectar la situación legal de su defendido...

Figura 5.14: Agrupación con $\alpha = 0{,}55$

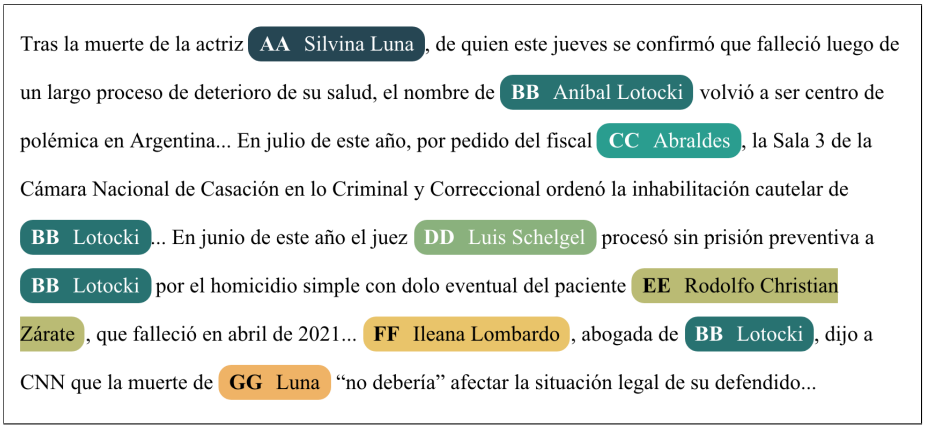


Figura 5.15: Agrupación con $\alpha = 0{,}30$

Para el caso de $\alpha = 0{,}55$ (figura 5.14) tanto la etiqueta BB y CC se asignan a más de una persona, mientras que $\alpha = 0{,}30$ (figura 5.15) comete el error de asignar dos etiquetas distintas a "Silvina Luna" y "Luna".

Capítulo 6

Conclusiones y trabajo futuro

El proceso de anonimización de sentencias legales es engorroso, largo y propenso a errores. Dado la componente narrativa que la anonimización debe mantener el esfuerzo no es trivial y complica el uso de herramientas automáticas preentrenadas; también lo hace el uso de una jerga judicial muy distinto al español de uso cotidiano. Dado este contexto, se buscan alternativas automáticas o semiautomáticas que asistan al proceso de anonimizado.

En particular, este trabajo introduce dos herramientas basadas en aprendizaje profundo: un reconocedor de entidades y un agrupador de entidades. Estas herramientas son entrenadas con un subconjunto del corpus BJN manualmente anotado por Garat y Wonsever[2], al que, como parte del proceso de entrenamiento, se revisan y agregan etiquetas a 353 de las sentencias. Se agregan ubicaciones a los tipos de entidades consideradas y se profundiza en el etiquetado de personas, además de considerar a los participantes se distingue también a referentes doctrinos y ministros participantes.

Durante este proceso se experimenta con varios reconocedores de entidades basados en BERT y la arquitectura Transformers. BETO, el mejor modelo, obtiene un F_1 de 95.3 % al identificar participantes. Este valor se encuentra por encima del promedio reportado en el modelo BETO original (90.2 %) que fue ajustado sobre documentos que utilizan un español más común. También representa una mejora del 3.54 % frente al introducido en *Automatic Curation of Court Documents*. Los resultados empeoran cuando se tienen en cuenta las ubicaciones y las personas no participantes, en este caso el F_1 micro es de 0,911. Gran parte de esto se debe al desbalance existente entre las clases del conjunto, donde locaciones y no participantes cuentan con una representación mucho menor.

Para la vinculación de entidades se introduce una red neuronal siamesa compuesta por unidades BI-LSTM entrenada para aprender una métrica de

distancia entre nombres de participantes. La red alcanza una precisión binaria de 91.6 % usando un valor de umbral de 0,5 al identificar si un par de nombres se corresponde a la misma persona o no.

Dado que no se sabe el número de participantes de una sentencia judicial a priori, se utiliza un algoritmo de agrupamiento jerárquico para identificar los participantes. Siguiendo el mismo proceso implementado por Garat y Wonsever se utiliza el corpus para identificar el mejor umbral de distancia para la formación de grupos; aunque en este caso, la distancia utilizada para formar los grupos es provista por la red siamesa. Esta solución no logra igualar o mejorar los resultados ya existentes, el mejor umbral resulta ser 0.55 —contra 0.45 en su artículo— y presenta un ARI de 88.2 % (contra 95.9 %).

Para concluir, el trabajo demuestra la efectividad de ajustar modelos preentrenados en contextos más generales a un dominio específico. Con una cantidad de datos relativamente reducida se alcanzan resultados de reconocimiento de entidades cercanos al estado del arte en Español. El uso de una métrica de similitud ajustada al contexto del problema permite obtener un buen desempeño pero no suficiente en la vinculación de entidades. Las redes siamesas cuentan con propiedades que resultan de interés —se ajustan bien a nuevos ejemplos, permiten capturar relaciones semánticas entre los nombres, y otras— por lo que su uso es prometedor. Es necesario seguir explorando alternativas dentro de este espacio, en la siguiente sección se listan algunas.

Se logró cumplir el objetivo inicial del proyecto: utilizando el corpus de la BJN se explora la utilización de técnicas de aprendizaje profundo para la implementación de herramientas con el fin de asistir la anonimización de sentencias judiciales.

Para la aplicación de las técnicas mencionadas, fue necesario extender el corpus agregando nuevas etiquetas tales como NP, REF y LOC. Además, para la vinculación de entidades se genera un nuevo corpus que parte del extendido, formado por pares de nombres y un indicador de correspondencia con la misma entidad. Estos conjuntos quedan disponibles para su uso en futuros proyectos.

Es importante destacar que las herramientas utilizadas: HuggingFace y Keras, cuentan con documentación muy extensa y de buena calidad. HuggingFace ofrece soluciones y ejemplos a más alto nivel, lo cual dificulta su aplicación en usos atípicos. Keras en cambio, requiere una mayor curva de aprendizaje, pero es muy versátil.

Tanto el reconocedor de entidades nombradas como la red vinculadora de entidades demuestran el potencial de estas técnicas e invitan a que sean exploradas con mayor profundidad. En el proceso se mejora el corpus existente de sentencias al verificar y refinar las anotaciones existentes, y además se introducen nuevas tipos de entidades que expanden la información cubierta originalmente. Expandiendo la utilidad del conjunto para posibles trabajos futuros.

6.1. Trabajo Futuro

Existe una gran cantidad de variaciones, mejoras y pruebas a ejecutar que no fue posible cubrir en este informe. A continuación se listan algunas de ellas para continuar esta línea de trabajo y que posiblemente tengan un impacto considerable en las soluciones halladas.

Quizás la mejora principal es continuar con el etiquetado de las sentencias judiciales con el nuevo esquema de anotación. Se logran buenos resultados con un conjunto bastante reducido de sentencias —solo mil— por lo que es esperable que el desempeño mejore si se cuentan con más ejemplos. Se destaca que es posible usar el reconocedor de entidades desarrollado para asistir con el trabajo de etiquetado y, dado el buen desempeño del algoritmo de agrupación, usarlo para obtener etiquetas específicas para cada participante.

Las sentencias utilizadas cuentan con varias faltas de ortografía y su estructura es inconsistente (a modo de ejemplo, en la figura 4.14 se incluye un salto de línea que divide un nombre), intentar corregirlas o llevarlas a una forma normalizada antes de entrenar el modelo podría mejorar los resultados.

Como se mencionó anteriormente la división en *chunks* puede causar pérdida de etiquetas y contexto. Se podría buscar una mejor forma de dividirlas o evitar la división con modelos que permitan un mayor largo de secuencia.

Además, ciertas entidades incluidas pertenecen a un universo finito: las ubicaciones, doctrinos y ministros. La construcción de diccionarios para referenciar durante el entrenamiento facilitaría la tarea de aprendizaje ya que sería posible preetiquetar las ocurrencias de estas entidades. Las ubicaciones de Uruguay son de carácter público y fáciles de acceder. Los ministros participantes en las sentencias se encuentran listados en la BJN como muestra la figura 2.1, es posible pedir estos datos a la BJN o recurrir al uso de web scraping para obtenerlos. Por último los doctrinos referenciados son un conjunto muy pequeño de personas y se puede construir una lista al etiquetar sentencias.

Respecto al reconocimiento de entidades nombradas, se identifican dos ejes principales de trabajo a futuro. En primer lugar, dado el gran tamaño de los modelos y el tiempo que insume su entrenamiento, no fue posible realizar búsquedas de hiperparámetros extensas. Es posible que otras combinaciones de hiperparámetros que no hayan sido consideradas o rangos de valores más amplios mejoren el desempeño del reconocedor.

Por otra parte, existen otros modelos basados en Transformers que buscan optimizar el costo de entrenamiento y por ende mejorar el largo máximo de secuencia que puede ser utilizado. En particular se destacan Longformer[64] y Reformer[65] que modifican la arquitectura original de Transformers para mejorar su eficiencia. Ambos reemplazan el mecanismo de atención utilizado dado su costo cuadrático sobre el largo de secuencia. Longformer alcanza resulta-

dos del arte —al punto de Mayo 2020— en WikiHop (81.9) y TriviaQA (77.3), superando a modelos Transformer como RoBERTa.

La red siamesa utilizada para calcular la similaridad entre nombres para la vinculación de entidades obtiene resultados promisorios que pueden ser mejorados. Si bien el enfoque fue en unidades LSTM y BI-LSTM, existen otras arquitecturas por evaluar. Varios de los artículos citados anteriormente mencionan el uso de unidades GRU, otros consideran el uso de atención y de distancia tripleta.

Hasta ahora se ha explorado un solo método de clusterización para agrupar participantes, si bien este obtiene buenos resultados pueden existir otras alternativas mejores. Dentro del mismo algoritmo se deberían evaluar distintos criterios de enlazado de datos, ya que solo se ha usado el criterio promedio.

Finalmente, se puede cambiar el enfoque y probar el uso de modelos de lenguaje (o por sus siglas en inglés LLMs)[66] para la resolución de esta tarea dada la gran cantidad de datos con los que estos se entrenan. Actualmente, existen distintas alternativas tales como: ChatGPT de OpenAI[67], Llama 2 de Meta[68], Palm2 de Google[69] y otros de código abierto cuyos datos de entrenamiento son públicos. Las instrucciones provistas al modelo o *prompts* afectan de gran manera la calidad de la respuesta del modelo. Como se menciona anteriormente el dominio de las sentencias judiciales es muy específico por lo que la ingeniería de *prompting* podría ser de gran ayuda[66].

Referencias

[1] Diego Garat y Dina Wonsever. "Towards De-identification of Legal Texts". En: *CoRR* abs/1910.03739 (2019). arXiv: 1910.03739. URL: http://arxiv.org/abs/1910.03739.

[2] Diego Garat y Dina Wonsever. "Automatic Curation of Court Documents: Anonymizing Personal Data". En: *Information* 13 (ene. de 2022), pág. 27. DOI: 10.3390/info13010027.

[3] Poder Judicial de la República Oriental del Uruguay. *Base de Jurisprudencia Nacional Pública*. URL: http://bjn.poderjudicial.gub.uy/. Último acceso: 16/10/2023.

[4] Dan Jurafsky y James H. Martin. "Speech and Language Processing (3rd ed. draft)". En: 2020. URL: https://web.stanford.edu/~jurafsky/slp3/.

[5] Nita Patil, Ajay Patil y B.V. Pawar. "Named Entity Recognition using Conditional Random Fields". En: *Procedia Computer Science* 167 (2020). International Conference on Computational Intelligence and Data Science, págs. 1181-1188. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2020.03.431. URL: https://www.sciencedirect.com/science/article/pii/S1877050920308978.

[6] *Attention and Memory in Deep Learning and NLP*. URL: http://www.wildml.com/2016/01/attention-and-memory-in-deep-learning-and-nlp/. Último acceso: 16/10/2023.

[7] Ashish Vaswani et al. "Attention Is All You Need". En: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[8] Jakob Uszkoreit. "Transformer: A novel neural network architecture for language understanding". En: *Google AI Blog* 31 (2017). URL: https://blog.research.google/2017/08/transformer-novel-neural-network.html.

[9] Yonghui Wu et al. *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. 2016. arXiv: 1609.08144 [cs.CL].

[10] Jacob Devlin y Ming-Wei Chang. "Open sourcing BERT: state-of-the-art pre-training for natural language processing". En: *Google AI Blog* 2 (2018). URL: https://blog.research.google/2018/11/open-sourcing-bert-state-of-art-pre.html.

[11] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". En: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

[12] Pranav Rajpurkar et al. *SQuAD: 100,000+ Questions for Machine Comprehension of Text*. 2016. arXiv: 1606.05250 [cs.CL].

[13] Rowan Zellers et al. *SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference*. 2018. arXiv: 1808.05326 [cs.CL].

[14] Alex Wang et al. *GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding*. 2019. arXiv: 1804.07461 [cs.CL].

[15] Erik F. Tjong Kim Sang y Fien De Meulder. *Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition*. 2003. arXiv: cs/0306050 [cs.CL].

[16] José Cañete et al. "Spanish Pre-Trained BERT Model and Evaluation Data". En: *PML4DC at ICLR 2020*. 2020.

[17] Xiaoqi Jiao et al. *TinyBERT: Distilling BERT for Natural Language Understanding*. 2019. DOI: 10.48550/ARXIV.1909.10351. URL: https://arxiv.org/abs/1909.10351.

[18] Alexis Conneau et al. *Unsupervised Cross-lingual Representation Learning at Scale*. 2020. arXiv: 1911.02116 [cs.CL].

[19] Yinhan Liu et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. DOI: 10.48550/ARXIV.1907.11692. URL: https://arxiv.org/abs/1907.11692.

[20] Taku Kudo y John Richardson. *SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing*. 2018. arXiv: 1808.06226 [cs.CL].

[21] Hang Yan et al. "TENER: Adapting Transformer Encoder for Named Entity Recognition". En: *CoRR* abs/1911.04474 (2019). arXiv: 1911.04474. URL: http://arxiv.org/abs/1911.04474.

[22] Xiaoya Li et al. *Dice Loss for Data-imbalanced NLP Tasks*. 2019. DOI: 10.48550/ARXIV.1911.02855. URL: https://arxiv.org/abs/1911.02855.

[23] Jouni Luoma y Sampo Pyysalo. "Exploring Cross-sentence Contexts for Named Entity Recognition with BERT". En: *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, dic. de 2020, págs. 904-914. DOI: 10.18653/v1/2020.coling-main.78. URL: https://aclanthology.org/2020.coling-main.78.

[24] Jane Bromley et al. "Signature Verification using a "Siamese"Time Delay Neural Network". En: *Advances in Neural Information Processing Systems*. Ed. por J. Cowan, G. Tesauro y J. Alspector. Vol. 6. Morgan-Kaufmann, 1993. URL: https://proceedings.neurips.cc/paper_files/paper/1993/file/288cc0ff022877bd3df94bc9360b9c5d-Paper.pdf.

[25] Paul Neculoiu, Maarten Versteegh y Mihai Rotaru. "Learning Text Similarity with Siamese Recurrent Networks". En: ene. de 2016. DOI: 10.18653/v1/W16-1617.

[26] Elad Hoffer y Nir Ailon. *Deep metric learning using Triplet network*. 2018. arXiv: 1412.6622 [cs.LG].

[27] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL].

[28] Gregory R. Koch. "Siamese Neural Networks for One-Shot Image Recognition". En: 2015. URL: https://api.semanticscholar.org/CorpusID:13874643.

[29] *Recognising and Interpreting Named Temporal Expressions*. URL: https://aclanthology.org/R13-1015/. Último acceso: 16/10/2023.

[30] Vikas Yadav y Steven Bethard. *A Survey on Recent Advances in Named Entity Recognition from Deep Learning models*. 2019. arXiv: 1910.11470 [cs.CL].

[31] Jing Li et al. "A Survey on Deep Learning for Named Entity Recognition". En: *CoRR* abs/1812.09449 (2018). arXiv: 1812.09449. URL: http://arxiv.org/abs/1812.09449.

[32] Mehdi Allahyari et al. *A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques*. 2017. arXiv: 1707.02919 [cs.CL].

[33] Yann LeCun, Y. Bengio y Geoffrey Hinton. "Deep Learning". En: *Nature* 521 (mayo de 2015), págs. 436-44. DOI: 10.1038/nature14539.

[34] Zhiheng Huang, Wei Xu y Kai Yu. "Bidirectional LSTM-CRF Models for Sequence Tagging". En: *CoRR* abs/1508.01991 (2015). arXiv: 1508.01991. URL: http://arxiv.org/abs/1508.01991.

[35] Hyejin Cho y Hyunju Lee. "Biomedical named entity recognition using deep neural networks with contextual information". En: *BMC Bioinformatics* 20 (dic. de 2019). DOI: 10.1186/s12859-019-3321-4.

[36] Vikas Yadav y Steven Bethard. "A Survey on Recent Advances in Named Entity Recognition from Deep Learning models". En: *CoRR* abs/1910.11470 (2019). arXiv: 1910.11470. URL: http://arxiv.org/abs/1910.11470.

[37] Lluís Padró y Evgeny Stanilovsky. "FreeLing 3.0: Towards Wider Multilinguality". En: *Proceedings of the Language Resources and Evaluation Conference (LREC 2012)*. ELRA. Istanbul, Turkey, mayo de 2012. URL: https://aclanthology.org/L12-1224/.

[38] Christopher Manning et al. "The Stanford CoreNLP Natural Language Processing Toolkit". En: *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Baltimore, Maryland: Association for Computational Linguistics, jun. de 2014, págs. 55-60. DOI: 10.3115/v1/P14-5010. URL: https://www.aclweb.org/anthology/P14-5010.

[39] Steven Bird, Ewan Klein y Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc., 2009.

[40] Matthew Honnibal e Ines Montani. *spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing*. https://spacy.io/. 2017. Último acceso: 16/10/2023.

[41] Joan Serrà y Alexandros Karatzoglou. "Getting deep recommenders fit: Bloom embeddings for sparse binary input/output networks". En: *CoRR* abs/1706.03993 (2017). arXiv: 1706.03993. URL: http://arxiv.org/abs/1706.03993.

[42] Matthew Honnibal. *spaCy's NER model*. URL: https://spacy.io/universe/project/video-spacys-ner-model. Último acceso: 16/10/2023.

[43] Thomas Wolf et al. *HuggingFace's Transformers: State-of-the-art Natural Language Processing*. 2020. arXiv: 1910.03771 [cs.CL].

[44] *GitHub*. URL: https://github.com/. Último acceso: 16/10/2023.

[45] François Chollet et al. *Keras*. https://keras.io. 2015. Último acceso: 16/10/2023.

[46] Tom O'Malley et al. *KerasTuner*. https://github.com/keras-team/keras-tuner. 2019. Último acceso: 16/10/2023.

[47] Lisha Li et al. *Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization*. 2018. arXiv: 1603.06560 [cs.LG].

[48] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". En: *Journal of Machine Learning Research* 12 (2011), págs. 2825-2830.

[49] Pontus Stenetorp et al. "brat: a Web-based Tool for NLP-Assisted Text Annotation". En: *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Avignon, France: Association for Computational Linguistics, abr. de 2012, págs. 102-107. URL: https://aclanthology.org/E12-2021.

[50] *Google Colab*. URL: https://colab.research.google.com/. Último acceso: 16/10/2023.

[51] *Spanish BERT (BETO) + NER*. URL: https://huggingface.co/mrm8488/bert-spanish-cased-finetuned-ner. Último acceso: 16/10/2023.

[52] *Dataset CoNLL2002 es*. URL: https://huggingface.co/datasets/conll2002. Último acceso: 16/10/2023.

[53] James Kirkpatrick et al. "Overcoming catastrophic forgetting in neural networks". En: *Proceedings of the National Academy of Sciences* 114.13 (mar. de 2017), págs. 3521-3526. DOI: 10.1073/pnas.1611835114. URL: https://doi.org/10.1073%2Fpnas.1611835114.

[54] Jonas Wallat, Jaspreet Singh y Avishek Anand. *BERTnesia: Investigating the capture and forgetting of knowledge in BERT*. 2021. arXiv: 2106.02902 [cs.CL].

[55] Y. Xu et al. *Forget Me Not: Reducing Catastrophic Forgetting for Domain Adaptation in Reading Comprehension*. 2020. arXiv: 1911.00202 [cs.CL].

[56] Hiroki Nakayama. *seqeval: A Python framework for sequence labeling evaluation*. Software available from https://github.com/chakki-works/seqeval. 2018. URL: https://github.com/chakki-works/seqeval. Último acceso: 16/10/2023.

[57] Takuya Akiba et al. "Optuna: A Next-generation Hyperparameter Optimization Framework". En: *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2019. URL: https://arxiv.org/abs/1907.10902.

[58] Chi Sun et al. *How to Fine-Tune BERT for Text Classification?* 2019. DOI: 10.48550/ARXIV.1905.05583. URL: https://arxiv.org/abs/1905.05583.

[59] Anmol Nayak et al. "Domain adaptation challenges of BERT in tokenization and sub-word representations of Out-of-Vocabulary words". En: *Proceedings of the First Workshop on Insights from Negative Results in NLP*. Online: Association for Computational Linguistics, nov. de 2020, págs. 1-5. DOI: 10.18653/v1/2020.insights-1.1. URL: https://aclanthology.org/2020.insights-1.1.

[60] Xinyu Wang et al. *Automated Concatenation of Embeddings for Structured Prediction*. 2021. arXiv: 2010.05006 [cs.CL].

[61] Tharindu Ranasinghe, Constantin Orasan y Ruslan Mitkov. "Semantic Textual Similarity with Siamese Neural Networks". En: *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*. Varna, Bulgaria: INCOMA Ltd., sep. de 2019, págs. 1004-1011. DOI: 10.26615/978-954-452-056-4_116. URL: https://aclanthology.org/R19-1116.

[62] Aditya Thyagarajan. "Siamese Recurrent Architectures for Learning Sentence Similarity". En: nov. de 2015. URL: https://ojs.aaai.org/index.php/AAAI/article/view/10350.

[63] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". En: *The journal of machine learning research* 15.1 (2014), págs. 1929-1958. URL: https://jmlr.org/papers/v15/srivastava14a.html.

[64] Iz Beltagy, Matthew E. Peters y Arman Cohan. *Longformer: The Long-Document Transformer*. 2020. arXiv: 2004.05150 [cs.CL].

[65] Nikita Kitaev, Łukasz Kaiser y Anselm Levskaya. *Reformer: The Efficient Transformer*. 2020. arXiv: 2001.04451 [cs.LG].

[66] Wayne Xin Zhao et al. *A Survey of Large Language Models*. 2023. arXiv: 2303.18223 [cs.CL].

[67] OpenAI. *ChatGPT*. URL: https://openai.com/chatgpt. Último acceso: 19/10/2023.

[68] Hugo Touvron et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. 2023. arXiv: 2307.09288 [cs.CL].

[69] Rohan Anil et al. *PaLM 2 Technical Report*. 2023. arXiv: 2305.10403 [cs.CL].

[70] Google. *Google Drive*. URL: https://www.google.com/intl/es/drive/. Último acceso: 16/10/2023.

[71] *Repositorio del proyecto*. Disponible en https://github.com/nicolasferraro/proyecto_de_grado_new(solicitar acceso).

Anexo A

Reemplazo de nombres de participantes

Para mantener el anonimato de los participantes mencionados en las sentencias judiciales, se reemplazan sus nombres con la lista definida a continuación, dependiendo de la etiqueta que tengan asignada. Los nombres fueron elegidos de forma aleatoria y no contienen ningún significado especial.

Etiqueta	**Nombre**	**Apellido**
AA	Julio	Villar
BB	Gisela	Segui
CC	Elias	Fariña
DD	Nicole	Alcantara
EE	Alonso	Arevalo
FF	Nahia	Grau
GG	Adan	Castro
HH	Ana Cristina	Coronado
II	Roberto Carlos	Macias
JJ	Yasmina	Borrego
KK	Alberto	Merchan
LL	Tania	Quiroga
MM	Jesus Antonio	Medrano
NN	Agustina	Baez
OO	Antonio Javier	Exposito
PP	María	Fuente
QQ	Luca	Miro
RR	Paola	Pérez
SS	Placido	Mariño
TT	Delia	Montes
UU	Brian	Palau
VV	Rosario	Rubio
WW	Bartolome	Liu
XX	Balbina	Pacheco
YY	Julio	Muriel
ZZ	Maria Juana	Bellido

Cuadro A.1: Nombres y apellido asignados a cada etiqueta

Anexo B

Gestión de código

Para alojar y versionar el código se utilizó un repositorio de Git, y tal como se mencionó en el capítulo 4, gran parte de la ejecución de pruebas se realizó en Google Colab[50], el cuál utiliza Google Drive[70] para su almacenamiento y manejo de algunos archivos.

B.1. Repositorio

El corpus de sentencias judiciales de la BJN del que parte este proyecto es almacenado en un repositorio privado en la instancia de Gitlab que provee la Facultad de Ingeniería.

El código implementado para la realización de pruebas de este proyecto se encuentra versionado en un repositorio privado[71] de GitHub[44].

El código para la solución de los problemas abordados por este informe se organiza en los archivos del cuadro B.1. Además cuenta con una carpeta "plots" que contiene los archivos de código necesarios para generar las gráficas utilizadas en el informe.

El repositorio es de acceso restringido porque como se menciona en el cuadro B.1, contiene un *dump* de la base de datos con fragmentos de las sentencias judiciales no anonimizadas, documentos que no deben ser de acceso libre.

La mayoría de las pruebas realizadas requirieron de cambios en el código (principalmente las del capítulo 4), no solo modificación de parámetros o variables de ambiente. Para cada uno de esos casos se creó una nueva rama. La figura B.1 contiene un diagrama de la organización de las ramas utilizadas. Los colores agrupan ramas por objetivo, los cuales coinciden con diferentes secciones de este documento.

Archivo	Finalidad
requirements.txt	Listado de dependencias utilizadas, incluyendo sus versiones.
config.py	Incluye constantes y variables de ambiente utilizados por el resto de los archivos.
db.py	Crear la base de datos, cargarla con los *chunks* de las sentencias judiciales y realizar consultas.
dump_pg_anonimo.sql	*Dump* de la base de datos. Es utilizado por el código de Google Colab para cargar la base de datos al crearla, previo a la ejecución de cada prueba.
anonym.py	Modelo para NER.
split_files.py	Dividir todos los chunks en conjuntos de entrenamiento, validación y evaluación.
fine_tuning.py	Dar el formato de entrada requerido por el modelo de NER al dataset de *chunks*.
check_performance.py	Generar matriz de confusión y evaluar el desempeño del modelo NER.
image_generator.py	Generar imagenes de fragmentos de sentencias con sus etiquetas, utilizadas en el informe.
generate_name_dataset.py	Generar el conjunto de nombres utilizado para entrenar la red siamesa.
siamese.py	Definición y ajuste de la red siamesa.
clustering.py	Agrupamiento jerárquico sobre el conjunto de nombres.

Cuadro B.1: Archivos del repositorio

B.2. Google Drive

Los archivos de Google Colab se encuentran en una carpeta privada de Google Drive. Cada prueba crea un archivo de extensión .txt que contiene los logs de la ejecución, incluyendo los resultados obtenidos. Esos archivos se almacenan en Google Drive, al igual que el modelo entrenado.

Las pruebas de Google Colab clonan el repositorio para obtener el código fuente, por lo que tienen acceso a él. El repositorio clonado y la base de datos creada es eliminada luego de cada ejecución, y los archivos de salida no contienen fragmentos de las sentencias judiciales, por lo que en Google Drive no se almacena información sensible. Igualmente la carpeta debe ser de acceso restringido por contener archivos de Google Colab con acceso al repositorio, el cuál como se mencionó antes, sí contiene sentencias judiciales.

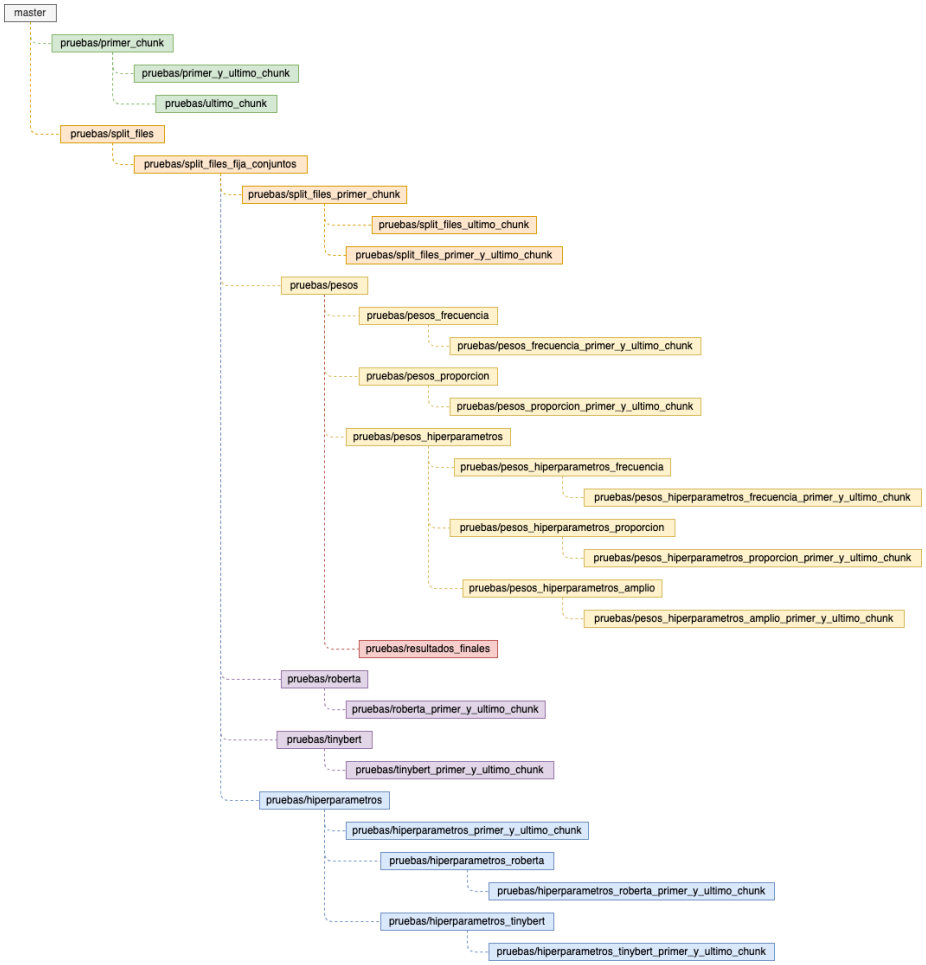
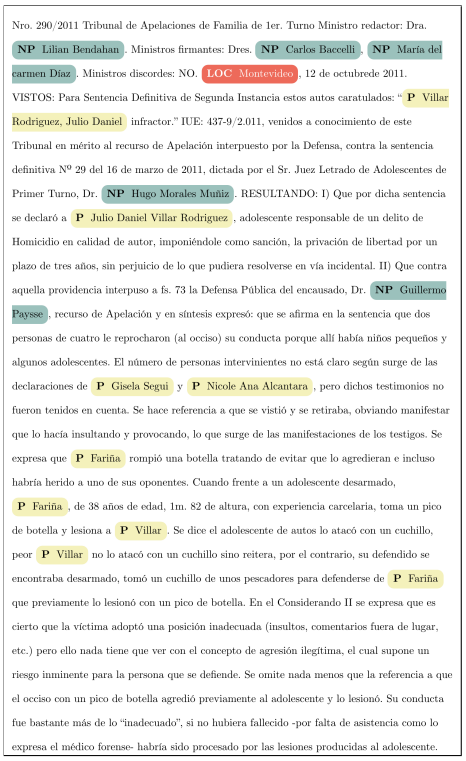


Figura B.1: Árbol de ramas en el repositorio de Github.

Anexo C

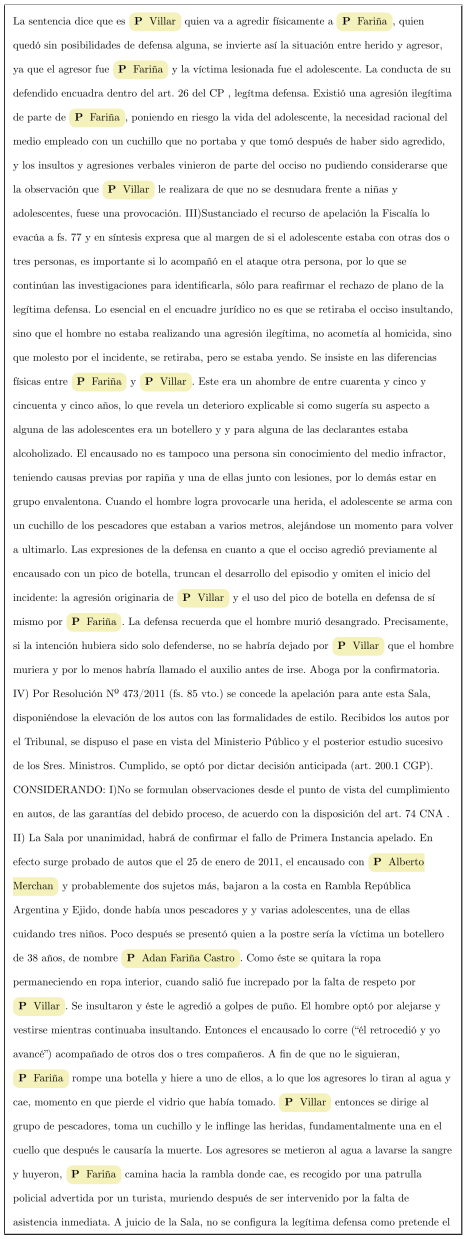
Sentencia completa anotada por el modelo

Las figuras C.1, C.2 y C.3 corresponden a la sentencia 290/2011 con sede en el Tribunal de Apelaciones de Familia de 1er. Turno anotada con las etiquetas predichas por la solución presentada en el capítulo 4.



Nro. 290/2011 Tribunal de Apelaciones de Familia de 1er. Turno Ministro redactor: Dra. NP Lilian Bendahan. Ministros firmantes: Dres. NP Carlos Baccelli, NP María del carmen Díaz. Ministros discordes: NO. LOC Montevideo, 12 de octubrede 2011. VISTOS: Para Sentencia Definitiva de Segunda Instancia estos autos caratulados: " P Villar Rodriguez, Julio Daniel infractor." IUE: 437-9/2.011, venidos a conocimiento de este Tribunal en mérito al recurso de Apelación interpuesto por la Defensa, contra la sentencia definitiva Nº 29 del 16 de marzo de 2011, dictada por el Sr. Juez Letrado de Adolescentes de Primer Turno, Dr. NP Hugo Morales Muñiz. RESULTANDO: I) Que por dicha sentencia se declaró a P Julio Daniel Villar Rodriguez, adolescente responsable de un delito de Homicidio en calidad de autor, imponiéndole como sanción, la privación de libertad por un plazo de tres años, sin perjuicio de lo que pudiera resolverse en vía incidental. II) Que contra aquella providencia interpuso a fs. 73 la Defensa Pública del encausado, Dr. NP Güillermo Paysse, recurso de Apelación y en síntesis expresó: que se afirma en la sentencia que dos personas de cuatro le reprocharon (al occiso) su conducta porque allí había niños pequeños y algunos adolescentes. El número de personas intervinientes no está claro según surge de las declaraciones de P Gisela Segui y P Nicole Ana Alcantara, pero dichos testimonios no fueron tenidos en cuenta. Se hace referencia a que se vistió y se retiraba, obviando manifestar que lo hacía insultando y provocando, lo que surge de las manifestaciones de los testigos. Se expresa que P Fariña rompió una botella tratando de evitar que lo agredieran e incluso habría herido a uno de sus oponentes. Cuando frente a un adolescente desarmado, P Fariña, de 38 años de edad, 1m. 82 de altura, con experiencia carcelaria, toma un pico de botella y lesiona a P Villar. Se dice el adolescente de autos lo atacó con un cuchillo, peor P Villar no lo atacó con un cuchillo sino reitera, por el contrario, su defendido se encontraba desarmado, tomó un cuchillo de unos pescadores para defenderse de P Fariña que previamente lo lesionó con un pico de botella. En el Considerando II se expresa que es cierto que la víctima adoptó una posición inadecuada (insultos, comentarios fuera de lugar, etc.) pero ello nada tiene que ver con el concepto de agresión ilegítima, el cual supone un riesgo inminente para la persona que se defiende. Se omite nada menos que la referencia a que el occiso con un pico de botella agredió previamente al adolescente y lo lesionó. Su conducta fue bastante más de lo "inadecuado", si no hubiera fallecido -por falta de asistencia como lo expresa el médico forense- habría sido procesado por las lesiones producidas al adolescente.

Figura C.1: Sentencia anotada por completo (primera parte). Fragmento de la sentencia número 290/2011[3].



La sentencia dice que es P Villar quien va a agredir físicamente a P Fariña , quien quedó sin posibilidades de defensa alguna, se invierte así la situación entre herido y agresor, ya que el agresor fue P Fariña y la víctima lesionada fue el adolescente. La conducta de su defendido encuadra dentro del art. 26 del CP , legítma defensa. Existió una agresión ilegítima de parte de P Fariña , poniendo en riesgo la vida del adolescente, la necesidad racional del medio empleado con un cuchillo que no portaba y que tomó después de haber sido agredido, y los insultos y agresiones verbales vinieron de parte del occiso no pudiendo considerarse que la observación que P Villar le realizara de que no se desnudara frente a niñas y adolescentes, fuese una provocación. III)Sustanciado el recurso de apelación la Fiscalía lo evacúa a fs. 77 y en síntesis expresa que al margen de si el adolescente estaba con otras dos o tres personas, es importante si lo acompañó en el ataque otra persona, por lo que se continúan las investigaciones para identificarla, sólo para reafirmar el rechazo de plano de la legítima defensa. Lo esencial en el encuadre jurídico no es que se retiraba el occiso insultando, sino que el hombre no estaba realizando una agresión ilegítima, no acometía al homicida, sino que molesto por el incidente, se retiraba, pero se estaba yendo. Se insiste en las diferencias físicas entre P Fariña y P Villar . Este era un ahombre de entre cuarenta y cinco y cincuenta y cinco años, lo que revela un deterioro explicable si como sugería su aspecto a alguna de las adolescentes era un botellero y y para alguna de las declarantes estaba alcoholizado. El encausado no es tampoco una persona sin conocimiento del medio infractor, teniendo causas previas por rapiña y una de ellas junto con lesiones, por lo demás estar en grupo envalentona. Cuando el hombre logra provocarle una herida, el adolescente se arma con un cuchillo de los pescadores que estaban a varios metros, alejándose un momento para volver a ultimarlo. Las expresiones de la defensa en cuanto a que el occiso agredió previamente al encausado con un pico de botella, truncan el desarrollo del episodio y omiten el inicio del incidente: la agresión originaria de P Villar y el uso del pico de botella en defensa de sí mismo por P Fariña . La defensa recuerda que el hombre murió desangrado. Precisamente, si la intención hubiera sido solo defenderse, no se habría dejado por P Villar que el hombre muriera y por lo menos habría llamado el auxilio antes de irse. Aboga por la confirmatoria. IV) Por Resolución Nº 473/2011 (fs. 85 vto.) se concede la apelación para ante esta Sala, disponiéndose la elevación de los autos con las formalidades de estilo. Recibidos los autos por el Tribunal, se dispuso el pase en vista del Ministerio Público y el posterior estudio sucesivo de los Sres. Ministros. Cumplido, se optó por dictar decisión anticipada (art. 200.1 CGP). CONSIDERANDO: I)No se formulan observaciones desde el punto de vista del cumplimiento en autos, de las garantías del debido proceso, de acuerdo con la disposición del art. 74 CNA . II) La Sala por unanimidad, habrá de confirmar el fallo de Primera Instancia apelado. En efecto surge probado de autos que el 25 de enero de 2011, el encausado con P Alberto Merchan y probablemente dos sujetos más, bajaron a la costa en Rambla República Argentina y Ejido, donde había unos pescadores y y varias adolescentes, una de ellas cuidando tres niños. Poco después se presentó quien a la postre sería la víctima un botellero de 38 años, de nombre P Adan Fariña Castro . Como éste se quitara la ropa permaneciendo en ropa interior, cuando salió fue increpado por la falta de respeto por P Villar . Se insultaron y éste le agredió a golpes de puño. El hombre optó por alejarse y vestirse mientras continuaba insultando. Entonces el encausado lo corre ("él retrocedió y yo avancé") acompañado de otros dos o tres compañeros. A fin de que no le siguieran, P Fariña rompe una botella y hiere a uno de ellos, a lo que los agresores lo tiran al agua y cae, momento en que pierde el vidrio que había tomado. P Villar entonces se dirige al grupo de pescadores, toma un cuchillo y le inflinge las heridas, fundamentalmente una en el cuello que después le causaría la muerte. Los agresores se metieron al agua a lavarse la sangre y huyeron, P Fariña camina hacia la rambla donde cae, es recogido por una patrulla policial advertida por un turista, muriendo después de ser intervenido por la falta de asistencia inmediata. A juicio de la Sala, no se configura la legítima defensa como pretende el

Figura C.2: Sentencia anotada por completo (segunda parte). Fragmento de la sentencia número 290/2011[3].

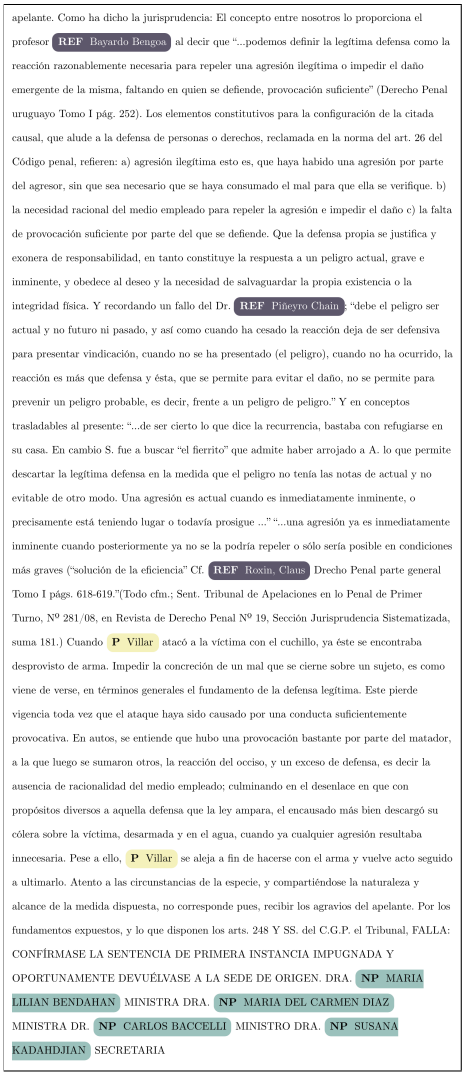


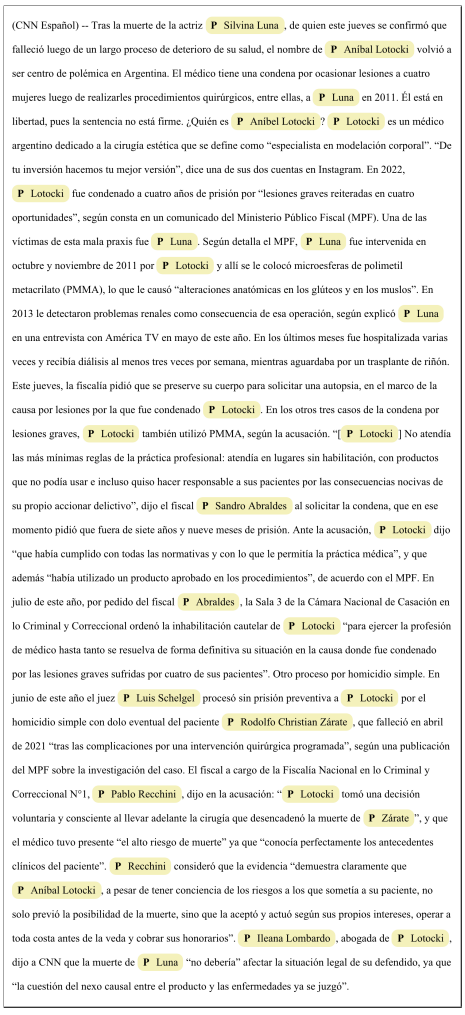
Figura C.3: Sentencia anotada por completo (tercera parte). Fragmento de la sentencia número 290/2011[3].

Anexo D

Agrupamiento de entidades sobre artículo

A continuación se incluye el texto completo del artículo utilizado como ejemplo en la sección *Resultados* del capítulo *Vinculación de entidades* 5.5. El artículo proviene de CNN[1] y trata sobre la muerte de Silvina Luna tras las operaciones realizadas por el cirujano Aníbal Lotocki. La figura D.1 contiene las entidades etiquetadas manualmente. Mientras que la figura D.2 contiene el resultado de agrupar dichas entidades utilizando $\alpha = 0{,}55$, y la figura D.3 $\alpha = 0{,}30$, que obtiene los mejores resultados.

[1] `https://cnnespanol.cnn.com/2023/09/05/quien-es-anibal-lotocki-medico-que-opero-silvina-luna-orix-arg/`



(CNN Español) -- Tras la muerte de la actriz P Silvina Luna , de quien este jueves se confirmó que falleció luego de un largo proceso de deterioro de su salud, el nombre de P Aníbal Lotocki volvió a ser centro de polémica en Argentina. El médico tiene una condena por ocasionar lesiones a cuatro mujeres luego de realizarles procedimientos quirúrgicos, entre ellas, a P Luna en 2011. Él está en libertad, pues la sentencia no está firme. ¿Quién es P Aníbel Lotocki ? P Lotocki es un médico argentino dedicado a la cirugía estética que se define como "especialista en modelación corporal". "De tu inversión hacemos tu mejor versión", dice una de sus dos cuentas en Instagram. En 2022, P Lotocki fue condenado a cuatro años de prisión por "lesiones graves reiteradas en cuatro oportunidades", según consta en un comunicado del Ministerio Público Fiscal (MPF). Una de las víctimas de esta mala praxis fue P Luna . Según detalla el MPF, P Luna fue intervenida en octubre y noviembre de 2011 por P Lotocki y allí se le colocó microesferas de polimetil metacrilato (PMMA), lo que le causó "alteraciones anatómicas en los glúteos y en los muslos". En 2013 le detectaron problemas renales como consecuencia de esa operación, según explicó P Luna en una entrevista con América TV en mayo de este año. En los últimos meses fue hospitalizada varias veces y recibía diálisis al menos tres veces por semana, mientras aguardaba por un trasplante de riñón. Este jueves, la fiscalía pidió que se preserve su cuerpo para solicitar una autopsia, en el marco de la causa por lesiones por la que fue condenado P Lotocki . En los otros tres casos de la condena por lesiones graves, P Lotocki también utilizó PMMA, según la acusación. "[P Lotocki] No atendía las más mínimas reglas de la práctica profesional: atendía en lugares sin habilitación, con productos que no podía usar e incluso quiso hacer responsable a sus pacientes por las consecuencias nocivas de su propio accionar delictivo", dijo el fiscal P Sandro Abraldes al solicitar la condena, que en ese momento pidió que fuera de siete años y nueve meses de prisión. Ante la acusación, P Lotocki dijo "que había cumplido con todas las normativas y con lo que le permitía la práctica médica", y que además "había utilizado un producto aprobado en los procedimientos", de acuerdo con el MPF. En julio de este año, por pedido del fiscal P Abraldes , la Sala 3 de la Cámara Nacional de Casación en lo Criminal y Correccional ordenó la inhabilitación cautelar de P Lotocki "para ejercer la profesión de médico hasta tanto se resuelva de forma definitiva su situación en la causa donde fue condenado por las lesiones graves sufridas por cuatro de sus pacientes". Otro proceso por homicidio simple. En junio de este año el juez P Luis Schelgel procesó sin prisión preventiva a P Lotocki por el homicidio simple con dolo eventual del paciente P Rodolfo Christian Zárate , que falleció en abril de 2021 "tras las complicaciones por una intervención quirúrgica programada", según una publicación del MPF sobre la investigación del caso. El fiscal a cargo de la Fiscalía Nacional en lo Criminal y Correccional N°1, P Pablo Recchini , dijo en la acusación: " P Lotocki tomó una decisión voluntaria y consciente al llevar adelante la cirugía que desencadenó la muerte de P Zárate ", y que el médico tuvo presente "el alto riesgo de muerte" ya que "conocía perfectamente los antecedentes clínicos del paciente". P Recchini consideró que la evidencia "demuestra claramente que P Aníbal Lotocki , a pesar de tener conciencia de los riesgos a los que sometía a su paciente, no solo previó la posibilidad de la muerte, sino que la aceptó y actuó según sus propios intereses, operar a toda costa antes de la veda y cobrar sus honorarios". P Ileana Lombardo , abogada de P Lotocki , dijo a CNN que la muerte de P Luna "no debería" afectar la situación legal de su defendido, ya que "la cuestión del nexo causal entre el producto y las enfermedades ya se juzgó".

Figura D.1: Artículo con las entidades etiquetadas manualmente

Figura D.2: Resultado de agrupar las entidades con $\alpha = 0{,}55$

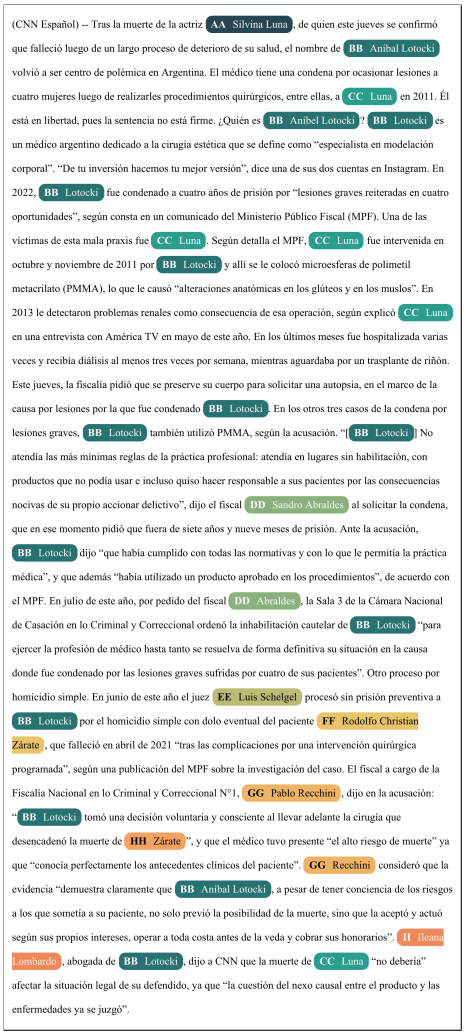


Figura D.3: Resultado de agrupar las entidades con $\alpha = 0{,}30$