Extracción de Relaciones basadas en Entidades Nombradas en Español utilizando Técnicas de Procesamiento de Lenguaje Natural

Relationship Extraction based on Named Entities in Spanish using Natural Language Processing Techniques

TRABAJO FIN DE GRADO
GRADO EN INGENIERÍA INFORMÁTICA
CURSO 2023–2024

Mauro Díaz Lupone
Yasser Takfa Ghazal

Directores
Luis Javier García Villalba
Luis Alberto Martínez Hernández

Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Madrid, Mayo de 2024

Agradecimientos

Deseamos manifestar nuestro más profundo agradecimiento a todas las personas que colaboraron en la elaboración del Trabajo de Fin de Grado. En primer lugar, queremos dar las gracias a nuestros tutores de TFG, Luis Javier García Villalba y Luis Alberto Martínez Hernández, por su guía y respaldo incondicional durante todo el proceso de investigación. Queremos tambíen agradecer por el apoyo y contribuciones al proyecto a Elena Almaraz Luengo, que estuvo presente en las últimas fases brindando toda la ayuda posible.

También queremos agradecer a todos nuestros profesores, amigos y familiares por habernos dado su respaldo y motivación mientras avanzábamos con el proyecto. Sus opiniones, recomendaciones y críticas constructivas fueron fundamentales para mejorar nuestro trabajo y para alcanzar las metas que nos habíamos fijado.

Sin el apoyo de todas estas personas, este Trabajo de Fin de Grado no habría sido posible.

Índice General

Índice de Figuras

Índice de Tablas

Lista de Acrónimos

ALBETO	*Lite version of BETO*
BERT	*Bidirectional Encoder Representations from Transformers*
BETO	*Spanish BERT*
CNN	*Convolutional Neural Networks*
DARE	*Data Augmented Relation Extraction*
DistilBETO	*Distil version of BETO*
ERN	*Modelos neuronales de extracción de relaciones*
FNN	*Feedforward Neural Network*
GPT	*Generative Pre-trained Transformer*
GPT-2	*Generative Pre-trained Transformer 2*
GRU	*Gated Recurrent Unit*
HERBERTa	*Hybrid Entity and Relation extraction BERT*

IE	*Information Extraction*
KB	*Knowledge base*
LSTM	*Long Short Term Memory*
M-BERT	*Multilingual Bidirectional Encoder Representations from Transformers*
ML	*Machine Learning*
MLM	*Modelado del Lenguaje Enmascarado*
NER	*Named Entity Recognition*
NLP	*Natural Language Processing*
NSP	*Predicción de la Siguiente Frase*
PE	*Positional Embedding*
RE	*Relation Extraction*
ReLU	*Rectified Linear Unit*
RNN	*Recurrent Neural Networks*
RoBERTa	*Robustly Optimized BERT Approach*
TL	*Transfer Learning*

Abstract

In this Final Degree Project, we conduct a comprehensive and in-depth study of several methods, architectures, and models for Relation Extraction in recognized named entities in Spanish texts. The main objective of this project is to develop a model for Relation Extraction through the use of the most innovative technologies available today, such as the *Transformer* architecture and the models derived from BERT, along with their integration with a Named Entity Recognition model to extract information from Spanish texts efficiently and effectively.

The study delves into the field of Natural Language Processing, exploring its several and complex algorithms and architectures derived from machine learning and its applications in the information extraction research domain. A thorough investigation is also presented on the best pre-trained language models in Spanish, such as RoBERTa and DistilBETO, and some of the largest datasets for Relation Extraction in Spanish.

The results obtained reflect the development of a model based on the *Transformer* architecture, specifically the use of RoBERTa, which is highly effective in linking named entities found in Spanish texts. This effectiveness is reflected in the metrics presented in the paper.

In conclusion, this work presents a thorough study of the methods for Relation Extraction and the effectiveness of models derived from BERT. The results achieved demonstrate the potential and efficiency of the proposed model, confirming its value for use in Spanish texts.

Keywords: Natural Language Processing, Named Entity Recognition, Relation Extraction, Information Extraction, Artificial Intelligence, Neural Networks.

Resumen

En el presente Trabajo de Fin de Grado (TFG) se lleva a cabo un estudio completo y profundo sobre distintos métodos, arquitecturas y modelos para la Extracción de Relaciones en entidades nombradas en textos en español. El objetivo principal de este proyecto es el desarrollo de un modelo para la ext a través del uso de las tecnologías más innovadoras en la actualidad, como es la arquitectura *Transformer* y los modelos derivados de BERT, y su respectiva unificación con un modelo de Reconocimiento de Entidades Nombradas para poder extraer información de textos en español de forma eficiente y eficaz.

En el estudio, se profundiza en el campo del Procesamiento del Lenguaje Natural, en sus distintos y complejos algoritmos y arquitecuras derivantes del aprendizaje automático y en sus aplicaciones al área de investigación de la Extracción de Información. Se presenta además una gran investigación en los mejores modelos de lenguaje preentrenados en español como son RoBERTa o DistilBETO y algunos de los mayores dataset para la Extracción de Relaciones en español.

Los resultados obtenidos reflejan la obtención de un modelo basado en la arquitectura *Transformer*, en particular el uso de RoBERTa, muy efectivo y que logra relacionar bien las entidades encontradas en textos en español. Esto se puede ver reflejado en las mñetricas presentadas en el trabajo.

En conclusión, este trabajo presenta un estudio exhaustivo de los métodos para la Extracción de Relaciones y de la efectividad de los modelos derivados de BERT. Los resultados alcanzados demuestran el potencial y la eficiencia del modelo presentado, confirmando el valor de su uso para textos en español.

Palabras clave: Procesamiento del Lenguaje Natural, Reconocimiento de Entidades Nombradas, Extracción de Relaciones, Extracción de Información, Inteligencia Artificial, Redes nueronales.

Capítulo 1

Introducción

1.1. Motivación

En los últimos años, se ha visto un interés creciente por la Inteligencia Artificial y sus aplicaciones en el ámbito de la interpretación y reproducción del lenguaje, debido a la gran cantidad de datos accesibles por cualquier persona mediante Internet. Las tecnologías de Procesamiento del Lenguaje Natural (del inglés, *Natural Language Processing* (NLP)) han experimentado un gran desarrollo en estas últimas décadas: desde la primera arquitectura moderna Word2Vec [MCCD13], se produjeron varios avances como las *Recurrent Neural Networks* (RNN) y las *Convolutional Neural Networks* (CNN) [RNN21] hasta llegar a la tecnología más utilizada hoy en día, llamada *Transformer*, que fue presentada en 2017 por Vaswani et al [VSP+23]. Esta nueva tecnología revolucionó el mundo del NLP y actualmente es la vanguardia y el paradigma dominante.

Una tarea importante del NLP es la *Information Extraction* (IE), en la que se identifica y clasifica información de interés de un texto y se puede dividir en dos subtareas que son la NER y RE. Actualmente la IE es un área de investigación muy importante por sus muchas aplicaciones [WWRM+18]. A partir de [VSP+23] se han desarrollado modelos como BERT [DCLT19] o *Generative Pre-trained Transformer* (GPT) [RNSS18] y sus respectivas variantes y mejoras, que han demostrado experimentalmente de ofrecer los mejores resultados para tareas de NLP.

A diferencia que para la subtarea NER, se tiene la problemática que para la RE no ha sido todavía desarrollado un modelo con estas nuevas tecnologías para textos en español. A partir de los buenos resultados de los modelos basados en BERT y sus variantes para el inglés, surge el interés del desarrollo de uno basado en la misma tecnología pero para el español, de la misma manera que se hizo en [Rod22a] para el portugués o en [PF23] para textos clínicos españoles.

1.2. Contexto

El presente Trabajo Fin de Grado se enmarca dentro de un proyecto de investigación titulado Novel Strategies to Fight Child Sexual Exploitation and Human Trafficking Crimes and Protect their Victims – HEROES, aprobado por la Comisión Europea dentro del Programa Marco Horizonte 2020 (convocatoria H2020-SU-SEC-2020) en virtud del

acuerdo de subvención número 101021801 y en el que participa como coordinador del proyecto el Grupo GASS de la Universidad Complutense de Madrid (Grupo de Análisis, Seguridad y Sistemas, https://gass.ucm.es, grupo 910623 del catálogo de grupos de investigación reconocidos por la UCM).

Además de la Universidad Complutense de Madrid participan en HEROES 21 entidades ubicadas en 17 países: 11 de países de la UE (Austria, Bélgica, Bulgaria, Francia, Grecia, Irlanda, Letonia, Lituania, Portugal, España, Reino Unido), 1 país asociado (Suiza) y 5 terceros países (Bangladesh, Brasil, Colombia, Perú, Uruguay). Dichas entidades son: University of Kent (Reino Unido), The Free University of Brussels (Bélgica), The French National Research Institute for Digital Science and Technology – INRIA (Francia), Center for Security Studies – KEMEA (Grecia), International Centre for Migration Policy Development – ICMPD (Austria), International Center for Missing and Exploited Children – ICMEC (Suiza), IDENER Research & Development Agrupación de Interés Económico (España), Athena Research Center – ARC (Grecia), Trilateral Research and Consulting (Reino Unido), Centre for Women and Children Studies – CWCS (Bangladesh), Center Against Human Trafficking and Exploitation – KOPZI (Lituania), Portuguese Association for Victim Support – APAV (Portugal), Fundación Renacer (Colombia), The Greek Council for Refugees – GCR (Grecia), Brazilian Association for the Defense of Children of Children and Youth – ASBRAD (Brasil), Hellenic Police (Grecia), Latvia National Police (Letonia), General Directorate for the Fight against Organized Crime (Bulgaria), Dirección General de la Policía – DGP (España), Federal Police (Brasil), Federal Highway Police (Brasil), Secretaría de Inteligencia Estratégica de Estado – Presidencia de la República Oriental del Uruguay (Uruguay).

Tienen más información en:

https://cordis.europa.eu/project/id/101021801

https://heroes-fct.eu

1.3. Objeto de la Investigación

El objetivo del proyecto es la generación de un modelo de NLP que sea capaz de extraer relaciones entre entidades nombradas en español en un texto para un enfoque general. A partir de este modelo se pretende automatizar procesos tales como búsqueda de información específica, identificación de conexiones entre individuos, empresas u organizaciones, entre otros.

Además, se consigue comprender la arquitectura interna de las herramientas más revolucionarias de los últimos años, las inteligencias artificiales basadas en aprendizaje automático, y más en concreto, las que tienen como finalidad el NLP. Se adentrará en sus detalles, tratando de descubrir la raíz de su comportamiento y efectividad.

1.4. Plan de Trabajo

El desarrollo de este trabajo se ha realizado en tres fases:

1. **Investigación**: En esta fase se llevó a cabo un proceso de familiarización con los términos que más adelante se utilizarían con frecuencia, así como conocer la

situación de las tecnologías a utilizar y la búsqueda de los recursos que podrían ser utilizados en fases futuras. Para ello, se concertaron reuniones semanales entre los alumnos y el tutor en las cuales se presentaban nuevos conceptos que serían de vital importancia para la culminación del proyecto, además de tener acceso a una serie de cursos que abordaban temas de introducción, como por ejemplo las bases del aprendizaje automático, aprendizaje profundo y teoría sobre redes neuronales. Posteriormente, después de una obtención de una base teórica en el área de investigación, se comenzó una búsqueda a través de la herramienta *Google Scholar*, de otros trabajos, herramientas y modelos usados para la IE. Finalmente, se comenzó la búsqueda de un conjuntos de datos que se utilizaría en las siguientes fases para el entrenamiento del modelo. Estos datos debían encontrarse ya etiquetados, listos para su preprocesamiento y utilización, además de representar texto en el idioma español, que resultó ser la parte más difícil. Para ello, se comenzó una subfase dentro de la propia fase de investigación que consistía en descubrir todas las alternativas que existen en el español, hallando la primera dificultad, ya que todos los conjuntos de datos conocidos y probados se encontraban en inglés. A pesar de ello, se consiguieron resultados y más adelante se pasó a la fase de desarrollo.

2. **Desarrollo**: En esta fase, y tras haber adquirido los conocimientos necesarios, se empezaron a plasmar las ideas en código. Para ello, se empezaron a probar las distintas librerías y herramientas estudiadas en la fase anterior, se inició la adecuación al caso real de los conjuntos de datos de entrenamiento hallados y se empezaron a resolver incompatibilidades y errores resultantes debidos a la adecuación de las herramientas. En esta fase se mantuvieron las reuniones semanales para monitorizar el avance y tratar de evitar la caída en puntos muertos y situaciones de estancamiento.

3. **Experimentación**: Esta fase fue la última y se centró en un estudio mayoritario de como mejorar el modelo que se estaba desarrollando. Para ello se estudiaron las métricas que ofrecía la herramienta utilizada en el entrenamiento del modelo y se optimizaron a través del cambio de hiperparámetros, datos y arquitecturas *Transformer*. Es por ello que en esta fase se siguió tanto investigando y desarrollando: búsqueda exhaustiva del mejor modelo en español aplicable a nuestro proyecto, estudio de trabajos referentes en nuestra área de investigación, análisis y modificación de los datos para obtener mejores resultados y entrenamiento del modelo con un cambio de hiperparámetros. Para optimizar esta fase se estuvo además probando el entrenamiento del modelo en ordenadores más potentes que ofrecían mayor rapidez de procesamiento.

1.5. Estructura del Trabajo

El resto del trabajo está organizado en 7 capítulos con la estructura que se comenta a continuación:

El Capítulo 2 introduce algunos conceptos importantes para llegar a la comprensión sobre el modelo que se construye en el proyecto. Se presentan los términos y áreas fundamentales para el entendimiento del NLP, en particular la RE y la NER. Además se estudian, tanto en funcionamiento como en aplicaciones, las tecnologías más avanzadas en la actualidad para el NLP.

El Capítulo 3 presenta un estudio sobre las distintas técnicas usadas para la RE: tanto métodos más tradicionales como métodos que utilizan ya varios conceptos de aprendizaje profundo. Se exponen los trabajos para la RE que a través del uso de modelos de lenguaje preentrenados llegan al estado del arte en la tarea. Se incluye además un estudio de los mejores modelos de lenguaje preentrenados para el idioma español, buscando ventajas y desventajas de cada uno de ellos

El Capítulo 4 presenta las aportaciones de este trabajo. Se muestra el estudio hecho sobre los distintos dataset encontrados para la RE en español y la respectiva uniformización y preprocesamiento que se hizo. Se muestra también un análisis sobre las herramientas utilizadas para la generación del modelo y el *pipeline* que se ha seguido para su construcción.

El Capítulo 5 describe los experimentos que se han realizado tanto para los datos presentados en el Capítulo 4 como para los modelos estudiados en el Capítulo 3. Se presentan los resultados obtenidos.

El Capítulo 6 muestra las conclusiones obtenidas posteriores a la experimentación y el estudio de los resultados del Capítulo 5. Se presentan además posibles líneas futuras de investigación

Los Capítulos 7 y 8 son las traducciones al inglés de la Introducción y de las Conclusiones.

Capítulo 2

Contexto de la Investigación

En este Capítulo se explica todo el contexto necesario para el entendimiento de este trabajo. En la Sección 2.1 se introduce el NLP y mientras que la Sección 2.2 se centra en la tarea particular de la que trata el proyecto: IE. Se procede después en la Sección 2.3 a explicar alguna terminología técnica habitualmente usada en el trabajo. En la Sección 2.4 se introduce la arquitectura *Transformer*, detallando cada una de las capas utilizadas, como la *Feed Forward* o la *Add and Norm*. Finalmente en la Sección 2.5 se explican los modelos de lenguaje preentrenados derivantes de la arquitectura *Transformer* y en la Sección 2.6 los distintos tipos de dataset existentes para entrenar modelos de NLP.

2.1. Procesamiento del Lenguaje Natural

El **Procesamiento de Lenguaje Natural** (NLP) es un área de estudio interdisciplinaria que combina técnicas y conocimientos de campos como la inteligencia artificial, la lingüística, la estadística y la computación. Su propósito principal es desarrollar algoritmos y sistemas capaces de comprender, analizar, generar e interpretar el lenguaje humano de manera inteligente, abarcando diversos formatos como texto, voz o lenguaje de signos [PMS23].

El campo del NLP ha experimentado una gran evolución a lo largo del tiempo, adaptándose a las demandas y aprovechando los avances tecnológicos. Inicialmente, las técnicas de NLP se basaban en reglas y modelos probabilísticos para llevar a cabo tareas como el etiquetado de partes del discurso y el análisis sintáctico [cou21]. Sin embargo, con la llegada de los *word embeddings*, como *Word2Vec* [MCCD13] y *GloVe* [PSM14], surgió una nueva forma de representar palabras como vectores en un espacio vectorial, lo que permitió capturar de manera más efectiva el significado semántico y mejorar el rendimiento en distintas tareas de NLP.

Posteriormente, las Redes Neuronales Recurrentes (RNN) revolucionaron el campo al posibilitar el modelado de secuencias de texto. A pesar de su potencia, las RNN presentaban limitaciones como el problema de la desaparición del gradiente y de que al ser un modelo secuencial, no permitía paralelismo [cou21]. La introducción de modelos de atención, como el *Transformer* [VSP+23], marcó un hito en la evolución del NLP [PMS23]. Estos modelos presentaron una arquitectura innovadora que, junto a permitir el paralelimso, se lograba capturar las relaciones entre las palabras en una oración sin depender exclusivamente de la recursividad de las RNN, lo que condujo a mejoras

significativas en tareas como la traducción automática y el análisis de sentimientos [PMS23].

En cuanto a las aplicaciones del NLP, estas abarcan una amplia gama de campos, como algunas de las descritas en [PMS23]:

1. **Reconocimiento de voz:** convertir el habla en texto, utilizado en sistemas de asistentes virtuales y aplicaciones de transcripción automática.
2. **Traducción automática:** traducir texto de un idioma a otro, siendo Google Translate un ejemplo destacado.
3. **Análisis de sentimientos:** determinar la actitud emocional expresada en un texto, utilizado en redes sociales, reseñas de productos y análisis de opinión pública.
4. **Resumen automático:** generar resúmenes concisos de documentos largos o extensos, útil en la extracción de información relevante de grandes volúmenes de texto.
5. NER: identificar y clasificar entidades importantes en un texto, como nombres de personas, lugares, organizaciones, fechas, etc. [OC23]
6. RE: identificar y clasificar las relaciones entre las entidades nombradas en un texto, proporcionando información sobre cómo están conectadas las diferentes partes de la información.

2.2. Extracción de información

La Extracción de Información (IE en inglés) es una tarea del NLP, dedicada a estructurar y extraer información relevante y significativa de fuentes de datos no estructuradas o semiestructuradas [LFF+19]. En este proceso, se busca convertir datos sin procesar, como texto, en una forma más organizada y fácilmente interpretable para su posterior análisis. Dos técnicas clave utilizadas en la Extracción de Información son el Reconocimiento de Entidades Nombradas (NER) y la Extracción de Relaciones (RE).

- **Reconocimiento de Entidades Nombradas [OC23]:** Esta técnica se enfoca en identificar y clasificar entidades dentro del texto en categorías predefinidas, como nombres de personas, organizaciones, lugares, fechas, cantidades, etc. El NER permite etiquetar automáticamente las menciones de estas entidades en el texto, lo que facilita su análisis posterior y su integración en sistemas de procesamiento de lenguaje natural y otras aplicaciones de extracción de información. Un ejemplo de ello se presenta en la Figura 2.1.

- **Extracción de relaciones:** el objetivo de la extracción de relaciones es predecir las relaciones entre dos o más entidades a partir de texto sin formato. Igual que explican en [ROD22b], cojamos la frase "Enel Green es una empresa de energía eléctrica y limpia con sede en Roma" (Figura 2.2). Nos fijamos que contiene dos entidades con nombre y una relación que puede representarse como una tupla: `(Enel Green,`

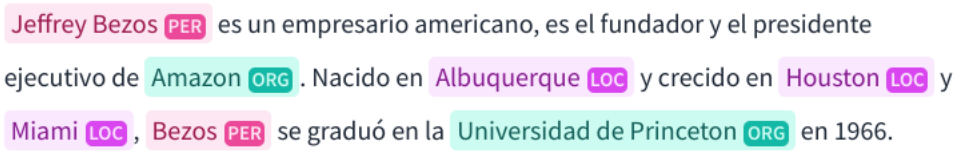

Figura 2.1: NER para una frase de ejemplo en el modelo de lenguaje *BERT* en español

`ubicación_de_sede, Roma)`. RE puede extraer información estructurada de varias fuentes diferentes que luego puede aplicarse a tareas posteriores. En los ultimos años se han propuesto varios *Modelos neuronales de extracción de relaciones* (ERN). Estos distintos métodos para abordar el problema de la detección y extracción de relaciones de documentos de texto se profundiza en el Capítulo 3.

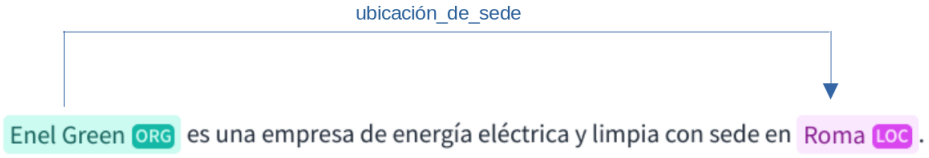

Figura 2.2: RE para una frase de ejemplo

2.3. Conceptos específicos

En esta Sección se abordan conceptos especializados, ofreciendo una explicación de su significado y relevancia dentro del contexto específico en el que se aplican. Aquí, se definen, como en [cou21], términos clave con el fin de proporcionar una comprensión profunda y precisa de los fundamentos que sustentan este campo de estudio particular.

- **Corpus**: conjunto de textos utilizado en el NLP para un estudio y análisis del lenguaje. Los elementos del conjunto representan textos reales que permiten a los desarrolladores utilizarlos para entrenar y evaluar algoritmos y modelos para tareas del NLP.
- **Token**: un token se refiere a una unidad básica de un texto, que puede ser una palabra, un número, un signo de puntuación u otro elemento significativo del texto. La *tokenización* es el proceso de dividir un texto en tokens individuales para su posterior análisis.
- **Época**: una *época* se refiere a una iteración completa a través de todo el conjunto de datos de entrenamiento. Durante cada época, el modelo pasa por todas las muestras de entrenamiento una vez, ajustando gradualmente sus parámetros para minimizar la función de pérdida o error.
- **Pipeline**: una *pipeline* es una secuencia de pasos que se aplican para un objetivo. En el ámbito del NLP la *pipeline* de un modelo indica los pasos para obtener ciertos resultados dado un input. Mientras que la *pipeline* de un proyecto son las fases por las que ha pasado.

2.4. Arquitectura *Transformer*

2.4.1. Embedding y Word2Vec

El *Word Embedding* es una técnica utilizada en NLP que asigna una representación vectorial a palabras en un espacio multidimensional. Esta representación captura las relaciones semánticas y sintácticas entre las palabras, lo que permite a los modelos de NLP comprender y trabajar con el significado de las palabras de manera más efectiva. En esencia, el *Word Embedding* transforma palabras en números, lo que facilita su procesamiento por parte de algoritmos de aprendizaje automático.

El algoritmo **Word2Vec** [MCCD13] es una técnica específica que utiliza redes neuronales para aprender representaciones vectoriales de palabras a partir de grandes cantidades de texto no etiquetado. Word2Vec se basa en dos arquitecturas principales: Continuous Bag of Words (CBOW) y Skip-gram. CBOW se entrena para predecir una palabra objetivo a partir de un contexto circundante, mientras que Skip-gram se entrena para predecir palabras del contexto dado una palabra objetivo. Ambas arquitecturas ajustan los vectores de palabras de manera que palabras con contextos similares tengan vectores similares en el espacio vectorial. Esto permite que Word2Vec capture relaciones semánticas y sintácticas entre palabras.

La implementación e idea detrás de cualquiera de las dos arquitecturas de *Word2Vec* es de entrenar una red neuronal de clasificación múltiple que devuelve el *embedding* óptimo para un *token* dada como función de pérdida la verosimilitud construida a través del cálculo de la probabilidad de una palabra dada la anterior en el texto. Un ejemplo de clasificación en 2 dimensiones de varios tokens de tres grupos semánticos distintos sería la de la Figura 2.3

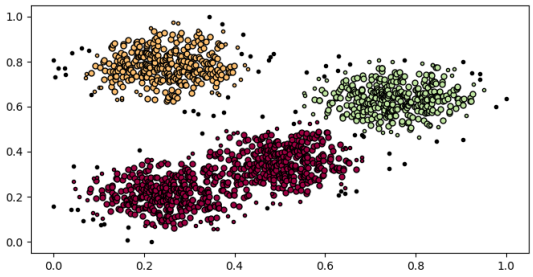

Figura 2.3: Representación normalizada del embedding en 2 dimensiones de varias palabras y coloreadas por cercanía

2.4.2. Atención

Inspirados en la idea fundamental de capturar y comprender las relaciones contextuales entre palabras, como *Word2Vec* donde se estudian las palabras en función de su contexto cercano para generar representaciones vectoriales significativas, surgieron las RNN [RNN21], que adoptaron un enfoque similar al considerar la información pasada para comprender mejor el contexto en el que aparece una palabra. Las RNN, inicialmente utilizadas en tareas de modelado de secuencias, fueron refinadas con la arquitectura

encoder-decoder, que permitía no solo comprender sino también generar secuencias de texto. En esta arquitectura de codificador-decodificador, se supone que el decodificador empieza a hacer predicciones fijándose sólo en la salida final del paso del codificador, que tiene información acumulada.

Sin embargo, las RNN presentaban limitaciones en términos de memoria a largo plazo [RNN23]. En respuesta, surgieron los modelos de **Atención** [VSP+23], que mejoraron la capacidad de las redes neuronales para capturar relaciones a largo plazo y dirigir su enfoque a partes específicas de la secuencia de entrada. En la arquitectura basada en la atención se observa cada nodo del codificador en cada paso temporal y luego realiza predicciones tras obtener unos pesos para cada otro nodo dependiendo de cuánta información puede ofrecer. Ejemplo de ambas estructuras lo encontramos en la Figura 2.4.

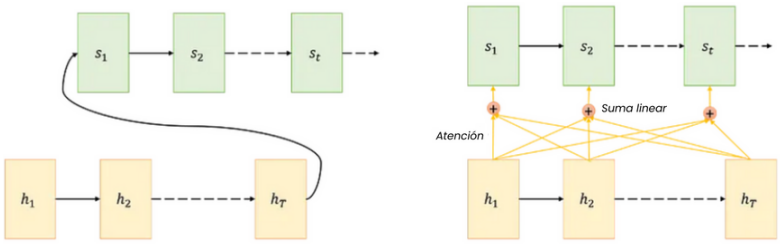

Figura 2.4: A la izquierda una representación de la estructura codificador-decodificador. A la derecha la misma estructura pero con el mecanismo de atención. Ambos presentados en [RNN21].

2.4.3. Multi-head attention

Para Vaswani *et al* en [VSP+23] la *Atención* descrita anteriormente se maneja como una consulta y un conjunto de pares clave-valor, todos asignados a una salida, donde todos los elementos descritos se representan como vectores. A los distintos valores resultantes de los pares se les asignan diferentes pesos mediante una función que mide la compatibilidad entre la consulta y la clave correspondiente, y tras obtener estos pesos se realiza una suma ponderada para obtener el resultado. Esto nos ofrece una codificación de una atención dada a cada palabra en referencia al contexto de la frase, se puede ver más intuitivamente con que para cada palabra se le asignará una consulta q, una clave k y una valor v y q representará una pregunta sobre el contexto de esa palabra en la frase que se le hará a las otras palabras, que responderán con sus respectivos pares de clave-valor. Esta atención particular se llama en [VSP+23] ***Scaled Dot-Product Attention*** y el input está formado por consultas y claves de dimensión d_k, y valores de dimensión d_v. Calculamos el producto escalar de las consultas con todas las claves, dividimos luego cada uno por $\sqrt{d_k}$, y aplicamos una capa *Softmax* y obtenemos los "pesos" para cada valor, como vemos en la Figura 2.5. En la práctica y de forma más óptima, calculamos la función de atención sobre un conjunto de consultas simultáneamente, juntadas en una matriz Q. Las claves y los valores también se juntan en las matrices K y V respectivamente. Calculamos la matriz de resultados como:

$$\text{Atención}(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1}$$

En lugar de implementar una sola función de atención con claves, valores y consultas d_{model}-dimensionales en [VSP+23] sugirieron proyectar de manera lineal las consultas, claves y valores h veces con diferentes, y aprendidas, proyecciones lineales de d_q, d_k y d_v dimensiones respectivamente. Para cada una de estas proyecciones se aplica la función de atención en paralelo, obteniendo valores de output de dimensión d_v. Estos irán después concatenados y otra vez proyectados, obteniendo así los resultados finales como se muestra en la Figura 2.6 para uno de los h procesos en paralelo.

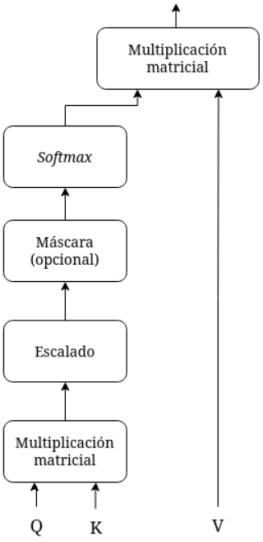

Figura 2.5: *Scaled Dot-Product Attention* 2.1

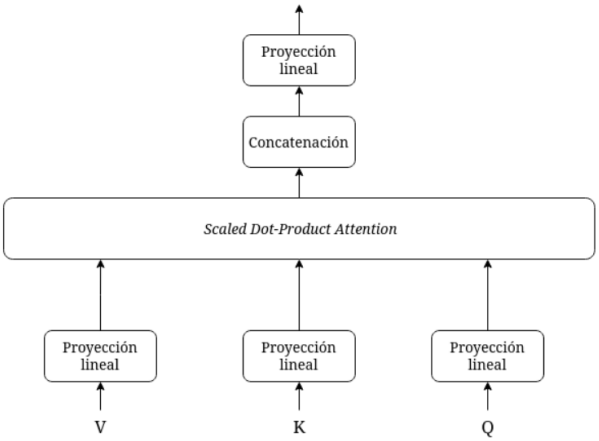

Figura 2.6: Cálculo de *Multi-head Attention* para una proyección lineal i-ésima de las h existentes y dada por la fórmula 2.2

La ***Multi-head Attention*** permite al modelo atender conjuntamente la información de distintos subespacios de representación en distintas posiciones. Con una sola capa de atención, se inhibe este proceso. Matemáticamente lo vemos como:

$$\begin{aligned} \text{MultiHead}(Q,K,V) &= \text{Concat}(head_1,\dots,head_h)W^O \\ \text{donde} \quad head_i &= \text{Atención}(QW_i^Q, KW_i^K, VW_i^K) \end{aligned} \tag{2.2}$$

Donde las proyecciones son matrices paramétricas $W_i^Q \in \mathbb{R}^{d_{model}\times d_q}$, $W_i^K \in \mathbb{R}^{d_{model}\times d_k}$, $W_i^V \in \mathbb{R}^{d_{model}\times d_v}$ y $W^0 \in \mathbb{R}^{h\times d_{model}\times d_k}$.

2.4.4. Positional-Encoding

Dado que el modelo *Transformer* presentado en [VSP+23] no tiene ni recurrencias ni convoluciones, para que el modelo tenga en cuenta el orden de la secuencia de los *tokens*, se decide introducir información sobre la posición relativa o absoluta de los *tokens* en la secuencia. Para ello, se añade *Positional Embedding* (PE) a los *embeddings* de las palabras de entrada del texto. Estas codificaciones tienen la misma dimensión d_{model} que las embeddings, de modo que ambas puedan sumarse. Hay muchas opciones de codificaciones posicionales que se describen en [SQ19], para esta arquitectura se elige:

$$\mathrm{PE}_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right)$$
$$\mathrm{PE}_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right)$$

donde *pos* es la posición numérica de la palabra y donde i representa la posición dentro del vector *embedding*.

2.4.5. Feed-Forward

Además de las subcapas de atención, cada una de las capas de nuestro *Transformer* contiene una red *Feedforward Neural Network* (FNN) totalmente conectada que se aplica a cada posición por separado y de forma idéntica. Esta consiste en dos transformaciones lineales con una activación *Rectified Linear Unit* (ReLU) intermedia.

$$\mathrm{FNN}(x) = W_2 \max(0, W_1 x_i + b_1) + b_2$$

Donde $x, b_1, b_2 \in \mathbb{R}^{d_{model}}$ y $W_1, W_2 \in \mathbb{R}^{d_{model} \times d_{model}}$

2.4.6. Estructura

La estructura completa del *Transformer* es la de la Figura 2.7:

Aquí, el codificador mapea una secuencia de entrada de tokens $(x_1, \ldots, x_n)$ a una secuencia $z = (z_1, \ldots, z_n)$. A partir de z, el descodificador genera una secuencia de salida $(y_1, \ldots, y_n)$ de tokens, un elemento a la vez. En cada paso, el modelo es autorregresivo, es decir, va consumiendo los tokens generados previamente como entrada adicional al generar el siguiente. El *Transformer* se rige por esta arquitectura general utilizando capas apiladas de Multi-head attention, interconectadas entre sí, tanto para el codificador como para el decodificador, que se muestran en las mitades izquierda y derecha de la Figura 2.7, respectivamente. En [VSP+23] también se muestran algunas ventajas de la elección de capas de autoatención frente a capas recurrentes o convolucionales. Una es la complejidad computacional total por capa que se junta con la cantidad de cálculo que se puede paralelizar, medida por el número mínimo de operaciones secuenciales necesarias. Otra es la longitud del camino entre las dependencias de largo alcance en la red. Un factor determinante para la capacidad de aprender dichas dependencias es la longitud de los caminos que las señales deben recorrer en los dos sentidos: cuanto más cortos sean los caminos entre todas las combinaciones de posiciones en las secuencias de entrada y salida, la facilidad de aprendizaje de las dependencias de largo alcance será mayor.

- **Add & Norm**: es la unión de dos tareas, la primera, es conexión residual que une la entrada de cada capa con la salida de la misma, esto nos permite mantener los pesos a lo largo del tiempo de ejecución. Dicha metodología fue ampliamente explotada en las redes ResNet [HZRS15]. Y la segunda es una tarea de normalización de capa

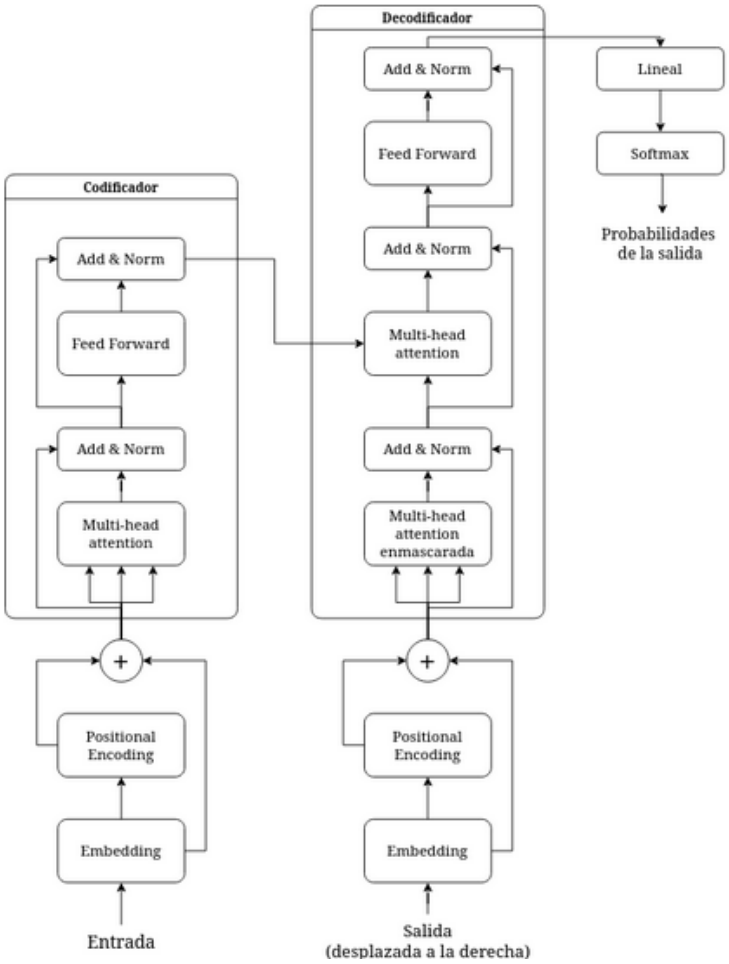

Figura 2.7: Estructura del transformer

[BKH16], que se asegura de que no se produzcan cambios muy bruscos sobre los datos al pasar de una capa a otra y así acelerar el entrenamiento del modelo con un mayor nivel de generalización.

- **Codificador**: el codificador está compuesto por una pila de $N = 6$ capas idénticas, que,a su vez, contienen dos subcapas cada una. La primera de estas subcapas es un mecanismo de *Multi-Head Attention*, y la segunda es una red de *Feed-Forward* simple, totalmente conectada en función de la posición. Alrededor de las dos subcapas se emplea una conexión residual, seguida de una normalización de capas *Add and Norm*. El resultado $z = (z_1, \ldots, z_n)$ pasa directamente al *Decodificador* que lo recibirá en una de sus subcapas como los valores K y V de la *Atención* descrita anteriormente.

- **Multi-Head Attention enmascarada**: en las tareas de generación de texto, la red neuronal a veces no tiene acceso a los *tokens* que aún no ha generado; en su contexto sólo tiene a su alcance los *tokens* pasados. Para obligar a la red a predecir con exactitud el *token* $N + 1$ utilizando únicamente los *tokens* anteriores 1 a N sin trampas ni miradas furtivas a *tokens* futuros (aún no generados) se emplea la técnica de enmascaramiento de los pesos de atención en la diagonal superior derecha, como se muestra en la Figura 2.8:

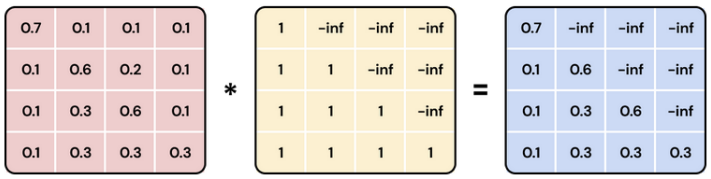

Figura 2.8: *Multi-Head Attention enmascarada* en [Ngo21]. Primera matriz son los *embedding* normalizados, la segunda matriz es la de enmascaramiento con donde $-\text{inf} := -\infty$ y la tercera son los *embedding* enmascarados.

- **Decodificador**: el decodificador, así como el codificador [VSP+23] está compuesto por una pila de $N = 6$ capas idénticas. Además de las dos subcapas del codificador, el decodificador añade una tercera subcapa, que realiza la atención sobre la salida de la pila del codificador. Al igual que en el codificador, se emplean conexiones residuales alrededor de cada una de las subcapas, seguidas de una normalización *(Add and Norm)*. También se realiza una pequeña modificación en la subcapa de autoatención de la pila para evitar que se atienda a posiciones posteriores *(Masked Multi-Head Attention)*.

  Este enmascaramiento, combinado con el hecho de que los *embedding* de salida están desplazados de una posición, garantiza que las predicciones de la posición i sólo pueden depender de las salidas conocidas en posiciones inferiores a i. La salida del codificador esta conectada al decodificador mediante la capa segunda capa de atención del mismo, donde se vuelven a realizar las funciones sobres las matrices Q, K, V, solo que en este caso V, K son las salidas del codificador y V es lo que se recibe de la subcapa enmascarada.

2.5. Modelos de lenguaje preentrenados

El Aprendizaje por Transferencia (en inglés, *Transfer Learning* (TL)) es una técnica en el ámbito del aprendizaje automático donde un modelo entrenado en una tarea específica se utiliza como punto de partida para entrenar un modelo en una tarea relacionada pero diferente. En lugar de comenzar desde cero y entrenar un modelo completamente nuevo para cada tarea, el TL aprovecha el conocimiento aprendido por el modelo pre-entrenado y lo aplica en una nueva tarea. Esto puede resultar beneficioso cuando se dispone de conjuntos de datos pequeños o cuando el entrenamiento desde cero sería computacionalmente costoso. El proceso de transferencia puede involucrar el **ajuste fino** del modelo pre-entrenado en la nueva tarea o el *extracción de características* para un nuevo modelo.

- **Ajuste fino**: se refiere a la técnica de tomar un modelo pre-entrenado y continuar el entrenamiento con datos específicos de la nueva tarea. En lugar de reentrenar completamente todas las capas del modelo, se realiza un ajuste sobre los parámetros del modelo en una etapa adicional de entrenamiento utilizando el nuevo conjunto de datos. Esto permite al modelo adaptarse mejor a los nuevos datos mientras conserva el conocimiento previamente aprendido.

- **Extracción de características**: implica tomar las características aprendidas por un modelo pre-entrenado en una tarea y utilizar esas características como entrada para un nuevo modelo en una tarea diferente. En este enfoque, el modelo pre-entrenado actúa como un extractor de características, y estas características extraídas se utilizan como entrada para un nuevo modelo, que puede ser más simple o tener una arquitectura diferente. Esto es especialmente útil cuando se dispone de pocos datos para la nueva tarea o cuando el entrenamiento desde cero no es factible debido a limitaciones computacionales. Un ejemplo sería usar el modelo para crear *embeddings* efectivos de las palabras para otro modelo.

2.5.1. Modelo BERT

Como se demostró en [DCLT19], BERT es un modelo preentrenado de representación del lenguaje que ha obtenido resultados de vanguardia en una amplia gama de tareas de NLP. BERT está diseñado para representar eficientemente el lenguaje a partir de texto sin etiquetar a través de la obtención del cntexto izquierdo y derecho de cada palabra gracias a su bidireccionalidad. Por ello, basta utilizar BERT y conectar su salida a una capa *Softmax* para poder aplicar su representación del lenguaje a alguna tarea de NLP. El modelo BERT es construido a base de codificadores de la arquitectura *Transformer* y la especialización del modelo en una tarea es gracias a dos pasos: el pre-entrenamiento y el ajuste fino.

En [DCLT19] explican la diferencia entre el preentrenamiento de BERT y el de otros modelos que utilizan modelos lingüísticos tradicionales de izquierda a derecha o de derecha a izquierda.. BERT representa las entradas utilizando tres niveles de información Tabla 2.1, como el *embedding* de tokens, el *embedding* de segmentos y el PE y aprende dos tareas no supervisadas: *Modelado del Lenguaje Enmascarado* (MLM) y *Predicción de la Siguiente Frase* (NSP):

- MLM: en la etapa de preprocesamiento, se selecciona aleatoriamente el 15 % de los tokens en una frase y se aplican distintas estrategias de enmascaramiento. Dentro de este 15 %, aproximadamente el 80 % de los tokens se enmascaran, lo que significa que se ocultan sus valores originales para que el modelo los prediga. Sin embargo, el enmascaramiento no siempre ocurre: aproximadamente el 10 % de los tokens se reemplazan por el valor de otro token seleccionado al azar, mientras que el último 10 % se deja sin modificar, conservando su valor original. Este proceso de enmascaramiento y sustitución se realiza para entrenar al modelo en la tarea de predicción de palabras enmascaradas, lo que le permite comprender y generar texto de manera más precisa.
- NSP: se entrena el modelo para que, dado un par de frases A y B, sea capaz de determinar si la frase B sigue a la frase A. Para lograr esto, el 50 % de las veces se etiqueta la frase B como *isNext* y el otro 50 % como *NotNext*. Esta técnica resulta extremadamente útil para tareas como la respuesta a preguntas.

2.5.2. Aplicaciones del modelo BERT

En la actualidad, los modelos de lenguaje que llegan a un estado del arte en la tarea específica para la cual han sido entrenados, se basan mayoritariamente en la

Entrada	*CLS*	Mi	gato	es	mono	*SEP*	Le	gusta	jugar	*SEP*
Embedding del token	E_{CLS}	E_{mi}	E_{gato}	E_{es}	E_{mon}	E_{SEP}	E_{le}	E_{gust}	E_{juga}	E_{SEP}
Sumar	+	+	+	+	+	+	+	+	+	+
Positional encoding	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9
Sumar		+	+	+	+	+	+	+	+	+
Embedding del segmento	E_A	E_A	E_A	E_A	E_A	E_A	E_B	E_B	E_B	E_B

Tabla 2.1: Niveles de información (mediante *embeddings*) de BERT

arquitectura *Transformer*. Particularmente los modelos derivados de BERT irrumpieron en la investigación debido a cómo resolvían ciertas limitaciones y problemas que tenían otros modelos de aprendizaje supervisado. En particular, en todos los problemas de clasificación (asignar un conjunto predefinido de categorías a una secuencia de texto determinada) el modelo BERT es muy usado, ya que permite añadiendo una sola capa neuronal (capa *Softmax*) obtener un método innovador, más eficiente y óptimo que los tradicionales.

En [Kor21] se estudia bien las distintas formas que hay para conseguir mejores resultados usando BERT, pero básicamente se generaliza en que para adaptar el modelo a una tarea particular se necesita:

- **Pre-entrenamiento adicional**: BERT es entrenado inicialmente en textos de propósito general, que pueden tener una distribución de datos distinta a los textos de la tarea objetivo. Por tanto es buena técnica re-entrenarlo en un corpus de texto específico de la tarea que es de interés y con un conjunto de datos de la misma área temática.
- **Estrategias de re-entreno**: hay muchas formas de usar BERT para un objetivo específico. Por ejemplo, se puede utilizar la representación proporcionada por el modelo como una característica añadida en el modelo de clasificación, o se pueden utilizar sus capas internas para obtener información sobre el texto.

Por ejemplo, y muy útil para la investigación, ha sido la manera efectiva en la cual BERT crea *embedding* de los tokens. Esto se debe a que este modelo es un primer y buen intento de uno de los mayores objetivos de los investigadores en NLP que es crear un modelo de texto general (universal) preentrenado con un gran corpus de texto de uso general. Esto implica que BERT es muy bueno en entender la semantica y gramática de las palabras y es bueno en relacionarlas en ámbito general, con que luego si se re-entrena en un ámbito particular es más eficiente que un método tradicional, construido específicamente y sólo para esa tarea.

Es lógico que por todo esto han aparecido algunos modelos especializados basados en la arquitectura y metodología de enseñanza de BERT pero previamente entrenados en corpus de texto específico. Por ejemplo en [BLC19] se presentó SciBERT un modelo entrenado con un corpus de textos científicos. En la Tabla 2.2 se muestran los resultados dados en unos análisis de [BLC19] en particular para las tareas de RE y NER en unos dataset en inglés:

Campo	Tarea	Dataset	Estado del arte	BERT-base	SciBERT
Bio	NER	BC5CDR	88.85	86.62	90.01
Bio	RE	ChemProt	76.68	79.14	83.64
Info	NER	SciERC	64.20	65.24	67.57
Info	RE	SciERC	nada	78.71	79.97

Tabla 2.2: *F1-Score* de BERT y SciBERT comparados con lo que era el estado del arte anterior. Datos de [BLC19].

2.5.3. Modelo RoBERTa

Robustly Optimized BERT Approach (RoBERTa), presentado en [LOG+19], se fundamenta en la táctica de enmascaramiento lingüístico empleada en BERT, en la que el modelo es también entrenado para predecir partes del texto no etiquetado que se han ocultado intencionalmente. RoBERTa, implementado en PyTorch [PGM+19], ajusta hiperparámetros clave de BERT, como la eliminación del objetivo de pre-entrenamiento de BERT conocido como NSP, y emplea entrenamiento con minibatches y tasas de aprendizaje mucho más altas. Esto permite a RoBERTa tener un rendimiento mucho mejor a la hora de la realización del modelado del lenguaje enmascarado en comparación con su predecesor,BERT, lo que repercute positivamente en la tarea posterior. Otro cambio concreto importante es que se produce una sustitución del MLM usado en BERT, por un enmascarado dinámico, que consiste en duplicar la frase 10 veces y entregarla en cada época cambiando el token enmascarado. Con esto se consigue que el modelo vea distintas versiones de la misma frase, cosa que mejora la precisión.

2.5.4. Modelo GPT

GPT, presentado en [RNSS18], está construido exclusivamente con decodificadores, una característica que lo distingue de otros modelos como BERT. Los decodificadores en GPT son capaces de generar secuencias de texto, lo que permite al modelo generar continuaciones coherentes y contextualmente relevantes para una entrada dada. El entrenamiento de GPT se basa en dos etapas: la primera etapa consiste en aprender un modelo lingüístico de gran capacidad sobre un gran corpus de texto. A continuación, se lleva a cabo una fase de ajuste en la que se adapta el modelo a una tarea específica con datos etiquetados. Una de las principales diferencias entre GPT y BERT radica en su arquitectura y enfoque de entrenamiento. Mientras que GPT utiliza decodificadores para generar texto secuencialmente, BERT emplea un modelo codificador bidireccional que aprende representaciones contextualizadas de palabras mediante la predicción de palabras enmascaradas en el contexto de una oración. Esto significa que GPT está diseñado para la generación de texto, mientras que BERT se centra en la comprensión del lenguaje natural y tareas relacionadas, como el análisis de sentimientos o la clasificación de texto.

En Febrero de 2019, OpenAI publica una versión mejorada de GPT llamada *Generative Pre-trained Transformer 2* (GPT-2) [RNSS18] en la que básicamente se entrena la versión anterior con un corpus más grande y se expande y se hace más complejo el modelo. Mientras que GPT original fue entrenado con alrededor de 117 millones de parámetros,

GPT-2 cuenta con 1.5 mil millones de parámetros. En cuanto a los datos utilizados para el entrenamiento, GPT-2 se entrenó con un corpus de texto aún más amplio y diverso que GPT. Además este nuevo modelo presenta unos mayores niveles de control sobre la generación de texto, lo que implica que puede ser ajustado para producir diferentes estilos de texto, de distintos temas cambiando un poco sus parámetros.

2.6. Dataset

Interesa entonces re-entrenar, hacer un *ajuste fino* de modelos pre-entrenados para que sean mejores en una tarea específica. En particular para entrenar modelos específicos en el ámbito de la RE para entidades nombradas, es crucial contar con datos adecuados que proporcionen ejemplos anotados de texto con relaciones entre entidades. Estos conjuntos de datos, conocidos como **datasets**, juegan un papel fundamental en el desarrollo y la evaluación de algoritmos de extracción de relaciones. Existen varios tipos de datasets que son de interés para la extracción de relaciones, entre ellos:

- **Datasets de Conocimiento Estructurado:** estos conjuntos de datos contienen información estructurada en forma de grafos, bases de datos o tablas, donde las relaciones entre entidades están explícitamente definidas.
- **Datasets Semánticos:** estos conjuntos de datos contienen ejemplos de texto donde se describen relaciones semánticas entre entidades, como *nació en*, *trabaja en*, *es padre de*, etc. Permiten entrenar modelos para capturar el significado y la semántica de las relaciones entre entidades.
- **Datasets Gramaticales:** en estos datasets se encuentran ejemplos que destacan las relaciones sintácticas entre entidades en el texto. Contienen estructuras gramaticales que indican relaciones, como las dependencias entre palabras en una oración.

Otra división importante en el ámbito de los dataset para tareas de NLP es su idioma: al ser los datos texto, este puede cambiar significado dependiendo del idioma, dialecto o variedad lingüistica en el que esté definido, por ello es importante que dicho dataset contenga esa información. Es por ello que nos podemos encontrar:

- **Datasets Multilingües:** los datasets multilingües contienen datos en varios idiomas y son valiosos para el desarrollo de modelos de NLP que funcionan en entornos multilingües. Estos datasets permiten a los investigadores entrenar modelos que pueden comprender y generar texto en múltiples idiomas. Un ejemplo de dataset multilingüe en el contexto de la extracción de relaciones semánticas es el TAC KBP (Knowledge Base Population). Este dataset se utiliza en la conferencia TAC (Text Analysis Conference), que tiene como objetivo evaluar sistemas de extracción de información y población de bases de conocimiento en varios idiomas. Por ejemplo los participantes pueden recibir documentos en inglés y español que contienen información sobre eventos y personas, junto con las relaciones semánticas que existen entre ellas, como la afiliación institucional de una persona o la ubicación de un evento.
- **Datasets Monolingües:** los datasets monolingües se centran en un solo idioma y son esenciales para tareas específicas de NLP dentro de un idioma particular. Estos datasets pueden incluir corpus de texto anotado con etiquetas gramaticales,

entidades nombradas, relaciones semánticas y más. Un ejemplo es el conjunto de datos CoNLL, que incluye varios conjuntos de datos anotados en inglés y otros idiomas para tareas como reconocimiento de entidades nombradas, análisis de dependencias y extracción de relaciones.

Capítulo 3

Estado del Arte

En este Capítulo se realiza una búsqueda y revisión de los trabajos actuales relacionados con este proyecto. En la Sección 3.1 se procede a dar una visión general de los métodos existentes que han sido mayormente usado para la RE: se describen técnicas tradicionales, con el uso de *Machine Learning* (ML), con RNN y con CNN. En la Sección 3.2 se hace una revisión de los modelos de lenguaje preentrenados basados en BERT aptos para ser usados para textos en español: se hace un estudio de los modelos multilingües y de los entrenados directamente y exclusivamente en español. Por último, en la Sección 3.3 se revisan los trabajos y modelos actuales para la extracción de relaciones de entidades nombradas a través del uso de *Transformer* multilingües o en español.

3.1. Extracción de relaciones

La extracción de relaciones (RE) es una de las tareas clave de la extracción de información. Se refiere a la clasificación de las relaciones semánticas que pueden existir entre entidades. Este tipo de información es esencial para construir Base de Conocimiento Semántico (en inglés, *Knowledge base* (KB)), que pueden emplearse posteriormente para inferir las relaciones que existen entre varias entidades. Además, RE es útil para desarrollar sistemas de respuesta a preguntas, resumir textos y crear taxonomías de conceptos. Muchas aproximaciones a la RE hacen uso de entidades nombradas ya extraídas y posteriormente establecen vínculos entre ellas mediante algoritmos heurísticos o basados en el aprendizaje automático [WQZ+21]. Los algoritmos más recientes tienden a resolver el problema de la NER y la RE de forma conjunta. En esta Sección nos centraremos en estudiar algunos de los enfoques más importantes en extracción de relaciones, se pueden observar en la Tabla 3.1

3.1.1. Métodos tradicionales

- **Métodos de construcción manual de patrones**: estos métodos requieren la cooperación entre expertos en la materia y lingüistas para construir un conjunto de conocimientos de patrones basados en palabras, partes de discurso o la semántica. Con estos conocimientos lingüísticos y conocimiento profesional del dominio, la RE puede realizarse en el fragmento de idioma preprocesado con los patrones. Si coinciden, se puede decir que el enunciado tiene la relación del patrón correspondiente

Propuesta	Técnica	Enfoque	Alcance
Marti et al. [Hea92]	Construcción manual de patrones	Tradicional	Frase
Feldman et al. [Int04]	Sistema *KnowItAll*	Semi-supervisado	Frase
Mintz et al. [MBSJ09]	Dataset de relaciones semánticas	Supervisado a distancia	Frase
Nguyen et al. [NG15]	Redes neuronales convolucionales	Supervisado	Frase
Zhang et al. [ZZHY15]	Redes neuronales recurrentes LSTM	Supervisado	Frase
Seganti et al. [SFS+21]	BERT	Supervisado	Frase
Landeghem et al. [SVL23]	BERT y derivados	Supervisado	Frase
Liu et al. [LSN+23]	BERT y Redes neuronales de grafos	Supervisado	Documento
Papanikolau et al. [PP20]	GPT2	Supervisado	Dataset

Tabla 3.1: Tabla resumen de los métodos para la RE investigados.

[Hea92]. Un ejemplo lo podemos sacar del estudio del estado del arte hecho en [WQZ+21] donde nos encontramos el patrón de *hiponimia* (relación semántica entre dos términos donde uno de ellos tiene un significado más específico que el otro). Por ejemplo dada la base de conocimiento "Para Y como X" que nos representa un patrón de relación, si tenemos el corpus "... trabaja para autores como Herrick y Shakespeare" entonces nos encontrará las relaciones *Hiponimia*(autor, Shakespeare) y *Hiponimia*(autor, Herrick).

- **Métodos semi-supervisados**: un enfoque semisupervisado para la extracción de relaciones es un método que utiliza tanto datos etiquetados como no etiquetados para entrenar un modelo de extracción de relaciones. En este enfoque, se aprovechan tanto los datos anotados manualmente (etiquetados) como los datos sin etiquetar para mejorar el rendimiento del modelo. Un sistema pionero que realiza RE de forma autónoma es el sistema *KnowItAll* presentado en [Int04]. Este sistema consta de cuatro módulos principales: un extractor de datos, una interfaz para el motor de búsqueda, un módulo de evaluación denominado *Assesor*, y una base de datos. Utiliza *ontologías* (especificación formal y explícita de los conceptos, relaciones y propiedades relevantes dentro de un dominio de conocimiento específico) y *bootstrapping* (método iterativo que utiliza una versión inicial simple de un modelo o algoritmo para generar versiones más complejas y precisas a partir de los datos de entrada) para extraer relaciones. *Assesor* es el módulo que se encarga de asignar probabilidades a las relaciones extraídas.

- **Métodos supervisados**: se ve RE como un problema de clasificación. Estos enfoques se dividen en dos tipos: basados en *características* y basados en *kernel* [PPB17]. En los basados en características, cada instancia de relación en los datos etiquetados se utiliza para entrenar un clasificador. En comparación, los métodos

basados en kernel rara vez necesitan pasos explícitos de preprocesamiento. Sin embargo depende más del rendimiento de la función kernel diseñada. A partir de estos métodos llegamos a los **Métodos supervisados a distancia** que son un tipo de método basado en el conocimiento presentados en [MBSJ09]. Todo este trabajo se basa en esta hipótesis: si dos entidades participan en una relación, todas las frases que mencionan estas dos entidades pueden expresar esa relación. De este modo, la supervisión a distancia intenta extraer las relaciones entre entidades del texto utilizando una KB. Cuando una frase y una KB se refieren al mismo par de entidades, este método marca la frase de forma heurística con la relación correspondiente en la KB.

3.1.2. Métodos basados en CNN y RNN

El primer modelo basado en CNN para RE fue presentado en [LSCC13]. Este modelo transforma la frase de un texto en una serie de vectores de palabras, que alimentan a una CNN y una capa de salida *softmax* para obtener una probabilidad de clasificación. En un esfuerzo por mejorar la selección de características y clasificar, en [XFHZ15] se introdujo múltiples filtros y módulos de max-pooling para CNN. Basado en este trabajo, otros estudios describieron una CNN dinámica que utilizaba un operador de max-pooling dinámico para seleccionar algunas características del resultado de la CNN. Ambos modelos lograron un alto rendimiento en diferentes tareas. Para reducir la dependencia de herramientas NLP, en [ZLL+14] se propuso el concepto de PE entre entidades y combinaron la información léxica de las entidades con las características a nivel de oración extraídas por CNN. Posteriormente, muchos modelos de CNN adoptaron el método PE para extraer relaciones con menos herramientas de NLP, o incluso sin utilizar ninguna información más que el *embedding* de palabras. Sin embargo, estos modelos se centraron principalmente en características locales y pasaron por alto las características globales y las dependencias a largo plazo. Para integrar más información estructurada, en [NG15] combinaron el concepto de filtros con múltiples tamaños de ventana para la extracción de relaciones. Este enfoque demostró que los filtros con múltiples tamaños de ventana proporcionan más información estructurada al modelo, lo que resulta en una mejora en la arquitectura de la CNN. Muchos modelos posteriores han adoptado esta técnica.

En el ámbito de la RE, las CNN enfrentan ciertas limitaciones. Uno de los problemas prominentes es la falta de consideración de características globales, especialmente cuando se trata de dependencias a larga distancia entre pares de entidades. Para abordar estas debilidades, los avances recientes se han centrado en el empleo de RNN, *Long Short Term Memory* (LSTM) y *Gated Recurrent Unit* (GRU). Estos métodos son eficientes en modelar secuencias de datos, ofreciendo soluciones potenciales a los desafíos que enfrentan las CNN en la captura de relaciones complejas dentro de un texto. Para aprender relaciones dentro de un texto extenso considerando la información temporal, se empezó a utilizar una arquitectura de RNN bidireccionales. Las RNN combinan la salida de cada estado oculto y se logra representar la característica contextual a nivel de oración. Al final del modelo, se realiza una operación de max-pooling para seleccionar algunas características de palabras desencadenantes para la predicción. Tener información completa y secuencial sobre todas las palabras en la oración es beneficioso para la RE, en [ZZHY15] se aplicaron redes LSTM bidireccionales para obtener la representación a nivel de oración y también utilizaron varias características léxicas. Los resultados experimentales mostraron que el uso de la *embeddings* de palabras como características de entrada era suficiente para lograr

resultados de vanguardia.

3.2. Modelos de lenguaje en español

El mayor problema actual con los modelos de lenguaje preentrenados para las tareas de NLP es que están mayoritariamente para el idioma inglés. Esto se debe a que es el idioma más usado y la mayoría de las investigaciones en mejoras de las métricas para tareas se hacen con dataset en inglés. Es por ello que empezaron a aparecer desarrollos de BERT o RoBERTa entre otros en los cuales se re-entrenaban los modelos en otros idiomas. Primero aparecieron modelos *multilingual*, los cuales eran entrenados en grandes corpus de texto en distintos idiomas y ofrecían resultados buenos en tareas como la traducción de texto. Más adelante, una vez también hubo desarrollo de grandes corpus de texto en otros idiomas, aparecieron modelos entrenados explícitamente en una sola lengua. En esta Sección se presentarán algunos de los modelos derivados de BERT que actualmente presentan el estado del arte de muchas de las tareas de NLP en el idioma español.

3.2.1. Modelos *multilingual*

En el mismo año que se presentaba BERT en [DCLT19], se publicó también la versión multilingüe *Multilingual Bidirectional Encoder Representations from Transformers* (M-BERT) que ha servido como base para posteriores modelos multilingües que son actualmente muy utilizados. En [PSG19] se hace un análisis exhaustivo de M-BERT y de como logra llegar al estado del arte (en 2019) en tareas como puede ser NER. Es por ello que cobra muchísima importancia y todos los modelos posteriores adquieren sus mismas ideas y métodos para mejores versiones. M-BERT está construido con los mismos 12 codificadores que tiene BERT pero se diferencia en el entrenamiento: se entrena con las páginas de Wikipedia de 104 idiomas usadas como vocabulario de palabras. No utiliza ningún marcador que denote el idioma de entrada del texto, y no tiene ningún mecanismo explícito para fomentar que los pares equivalentes en idioma tengan representaciones similares. Sorprendentemente, se demostró experimentalmente que este modelo logra ofrecer muy buenos resultados manejando la transferencia entre diferentes sistemas de escritura y de idioma bastante bien. En cuanto a por qué M-BERT generaliza entre los idiomas, se hipotiza que el hecho de tener tokens de palabras utilizadas en todos los idiomas (números, URLs...) que deben ser mapeadas a un area cercana del espacio vectorial de los *embeddings* fuerza a que palabras similares estén en áreas similares, extendiendo así el efecto a otras palabras, hasta que los diferentes idiomas estén cerca en el espacio vectorial. Actualmente de los modelos más importantes y que ha llegado a mejores resultados para la tarea de RE es el presentado por [SFS+21] llamado *Hybrid Entity and Relation extraction BERT* (HERBERTa). Primero de todo se hace un trabajo de investigación y de desarrollo de un dataset multilingüe de 14 idiomas específico para la tarea de RE y de NER en los que se juntan varios otros dataset y un trabajo propio de preprocesamiento de datos. El enfoque de [SFS+21] es utilizar dos modelos pre-entrenados de BERT y una *pipeline* no convencional. El primer modelo es BERT, ajustado para la tarea de clasificación de secuencias de texto. Su entrada es una secuencia de tokens, y su salida consta de una una salida que representa el contexto general de la oración (a partir del primer token de la secuencia: `[CLS]`). Esta última se pasa a un softmax para clasificar y encontrar la relación. El segundo modelo es una implementación de BERT para la NER: su entrada es la misma secuencia para el primer modelo y la relación del primer modelo y su salida es la relacion

con sus respectivas entidades. Los modelos BERT varían dependiendo del idioma.

3.2.2. Modelos BETO

Spanish BERT (BETO) es el primer modelo derivado de BERT entrenado enteramente en un corpus en español y fue publicado en [CCF+23]. Esto era necesario porque aunque los modelos multilingües llegaban a buenos resultados, no eran lo suficientemente buenos para algunas tareas, en particular en NER fallaban bastante en las predicciones. El modelo BETO tiene la misma estructura que el de BERT, siempre con 12 codificadores y tiene más de 110 millones de parámetros entrenados con un dataset que junta texto de distintas fuentes como pueden ser Wikipedia, Ted Talks y periódicos. El corpus total llega a tener una dimensión de más de 3 billones de palabras, mientras que el vocabulario tiene 31K palabras. Para medir cuanto de buenos son los resultados, se utilizan distintas métricas y dataset para cada tarea. En [CCF+23] se mide por ejemplo el NER con el dataset ConLL, mientras que para RE no se mide. El entrenar BERT únicamente en el idioma español aporta muchas ventajas, como es la especificidad y exactitud en el idioma, pero no puede tomar las ventajas que tiene un modelo multilingüe, que al ser entrenado en un corpus más grande aunque de más idiomas, consigue encontrar otros patrones o en algunos casos crear un *embedding* más efectivo de las palabras. Es por ello que en algunas tareas, a través de los resultados experimentales de [CCF+23], se llega al estado del arte en español mientras que en otros como NER, aunque cerca, siguen siendo mejor modelos multilingües basados en RoBERTa. Se induce entonces a mejorar los modelos entrenados en español con más datos y preentrenamiento distinto.
Es entonces que se publican otros modelos como pueden ser *Lite version of BETO* (ALBETO) y *Distil version of BETO* (DistilBETO) que resuelven ciertos problemas de los modelos anteriores como la dificultad de usarlos en aplicaciones que funcionan en tiempo real. Al ser modelos muy grandes tardan bastante en predecir y calcular luego el tener otros modelos más pequeños pero también efectivos puede ser mucha utilidad. Estos dos son presentados en [CDBM+23] y cogen la misma estructura que los respectivos modelos en inglés de BERT en los cuales hay menos parámetros y menos capas y se entrena con un gran corpus en español. Los resultados son buenos aunque siempre en tareas como NER son peores que el respectivo modelo BETO.

3.2.3. Modelos RoBERTa en español

Al haberse demostrado en [LOG+19] los buenos resultados que el modelo RoBERTa ofrece en comparación al modelo BERT se empezaron a desarrollar versiones tanto multilingües como específicas de un idioma para estos modelos. Simultáneamente, debido gran desarrollo de los modelos GPT, también había interés de especializar estos últimos a un idioma particular. Es por ello que en 2022 se publica en [GAP+21] una familia de modelos de lenguaje preentrenados en un corpus de más de 570 GB en español para RoBERTa, GPT y GPT-2. Se demuestra experimentalmente que estos modelos superan con creces las métricas publicadas para otros modelos en español como pueden ser BETO para las tareas que ellos especifican.
Los datos de entrenamiento utilizados para el reentrenamiento provienen todos de la Biblioteca Nacional Española (BNE) y están divididos en tres grandes categorías: basados en temas, eventos relevantes y dominios con riesgo de desaparición. Esto es además muy innovativo porque hasta ahora la mayoría de corpus estaban compuestos por información

sacada de Wikipedia. Además de ello, ofrecen modelos en los cuales después de todo el gran entrenamiento con el corpus anterior, se les hace un *ajuste fino* con otros dataset (publicados por otros grupos) para ofrecer ya un modelo perfectamente usable para esas tareas. En particular, nos encontramos ya un modelo entrenado para NER a través del dataset CoNLL-NERC. Los modelos utilizados son justamente los descritos en el Capítulo 2 como son GPT-2 y RoBERTa, en el que para este último se utiliza una versión *large* en la cual tenemos el doble de capas de codificadores y de parámetros. Los resultados son muy buenos, para el dataset de "*CoNLL-NERC*", el modelo *RoBERTa-base* llega a un *F1-Score* de 0.8851 mientras que entre los otros modelos en español existentes sólamente BETO se acerca con un 0.8759. Con otros dataset ocurre algo parecido: para "*CoNLL-NERC*" tenemos que *RoBERTa-large* llega a un 0.9051 mientras que M-BERT 0.8810 y para "*PAWS-X*" *RoBERTa-large* llega a un 0.9150 mientras que M-BERT 0.9.Concluimos que en general el modelo sobrepasa de largo los demás para tareas en los que los dataset para los cuales se evalua no están basados en Wikipedia.

3.3. Extracción de relaciones con modelos BERT y GPT

En esta Sección se analiza como lograr hacer RE de entidades en un texto con modelos de lenguajes preentrenados implementados con *Transformers*. Esto ha sido muy útil y actualmente los mejores resultados para el idioma inglés en esta tarea son con esta tipología de modelos. Aunque parte de la investigación se ha centrado en cómo se aplican los modelos para el inglés, al ser lo más avanzado, aquí relataremos mayormente los resultados conseguidos en español. Actualmente para este idioma solo existen modelos específicos para el área de la medicina y de la biología, de los cuales se puede coger inspiración pero no se asemeja a este trabajo.

3.3.1. Extraccion de relaciones con BERT en otros idiomas

La mayoría de los avances recientes en la extracción de relaciones, evaluados en diversos y reconocidos conjuntos de datos, se han basado en modelos derivados de BERT. Estos modelos han establecido nuevos estándares en el campo, demostrando su efectividad en una variedad de temas y bases de conocimiento.
En [Rod22a] se hace un estudio muy efectivo de estos avances y se implementa un modelo de extracción de relaciones multilingüe para los idiomas portugués e inglés. En particular lo hacen para extraer relaciones en textos ciber-forenses. Primero de todo se hace un estudio extenso de los distintos tipos de transformers actuales y se llega a la conclusión de que los mejores resultados se pueden obtener con modelos derivados de BERT, ya que GPT están enfocados en otros tipos de tareas como generación de texto. Para los datos que utilizarán, se basan mayoritariamente en la famosa base de conocimiento llamada *Dbpedia*, en la que se construyen grafos de relaciones semánticas basados en la Wikipedia. El problema de estos datos es que están en el formato llamado RDF, en el que se organizan en tripletas, que consisten en un sujeto, un predicado y un objeto y cada tripleta representa una declaración sobre un recurso. Para idiomas como el inglés se presentan ya también preprocesados en formato JSON, mucho más útil para entrenar una red neuronal, mientras que por ejemplo en español no existen. En [Rod22a] la herramienta que utilizarán para crear todo el modelo es OpenNRE, de la que se habla más en profundidad en el Capítulo 4. El *pipeline* que crean para la extracción de relaciones es simple y muy bien estructurado, en el que primero de todo preprocesan los datos para dejarlos en un formato en el cual la

herramienta OpenNRE los acepte y hacen un *ajuste fino* específico para los transformer que utilizarán para cada idioma para la tarea NER y otro para la tarea RE. Despúes se hace un *script* de python en el cual los datos preprocesados se pasan al modelo en inglés o en portugués dependiendo del idioma y posteriormente se extraen las entidades de los datos. Entonces se tiene, después de esta fase, los mismos datos (tokens del texto en el que estamos interesados) pero con las entidades anotadas. Se les pasará entonces por el modelo especializado en extracción de relaciones y se sacará la probabilidad de cada relación para cada par de entidades encontradas. Por último, con un *script* final se eligen las relaciones con una probabilidad aceptable (elegida por ellos) y se muestran en pantalla como un grafo de relaciones semánticas. Los resultados son muy prometedores, ya que por ejemplo para el portugués consiguen con el equivalente BETO para ese idioma el estado del arte para todos los dataset con los cuales ellos prueban.

3.3.2. Extraccion de relaciones con BERT en español

Para el idioma en español, como explicado anteriormente, actualmente solo se encuentran modelos especializados en temática médica y/o de biomedicina. En [SPMGB+23] hacen un estudio de los mejores modelos actuales de deep learning para la extracción de información en la temática médica del cancer. En la publicación se demuestra que los modelos basados en BERT son mejores que los tradicionales y hacen una comparativa de las métricas obtenidas para sus dataset usando distintos Transformer. Se concluye que el reentrenamiento de los datos en la temática que concierne el objetivo del estudio es un paso clave para obtener buenos resultados: se compara el rendimiento de BETO, de M-BERT y de RoBERTa en dos versiones, una especializada en biomedicina y otra la versión propuesta en [GAP+21] entrenada con la Biblioteca Nacional Española y sale que las mejores métricas las obtiene el modelo especializado en biomedicina. Se observa entonces, que aunque haya modelos entrenados con mayor cantidad de datos, la temática y proveniencia de los datos asume una igual o más importancia. En [CLP+22] se aplica justo esto último para el campo de la biomedicina. Se hace una gran tarea de producción de un dataset propio y se reentrena un modelo base de RoBERTa con sus datos. Los resultados que se consiguen son mejores que si se usase BETO o RoBERTa BNE.
Por último es interesante el *pipeline* construido en [PF23], ya que es la misma idea que para el que se hizo en [Rod22a] pero utilizando otra herramienta: SpaCy3. Esto es interesante porque justamente esta librería publicó hace poco [SVL23] un modelo en software libre para hacer ajuste fino de un transformer para RE, que se describe en mayor profundidad en el Capítulo 4. Todos los ejemplos e investigaciones actuales están en muchas ocasiones prefiriendo utilizar esta herramienta en vez de OpenNRE y en [PF23] es de los pocos casos en los que se aplica en otro idioma que no sea el inglés. Es de mucha utilidad porque muestran cómo hacer el preprocesamiento de los datos para SpaCy 3 y un pipeline ejemplar para la extracción de información. En este caso particular, en el que sus datos están especializados en historias clínicas electrónicas, consiguen buenos resultados reentrenando el modelo RoBERTa en español con el dataset de la biblioteca nacional española.

3.3.3. Extraccion de relaciones y GPT2

Es ampliamente reconocido que las arquitecturas de modelos de lenguaje como GPT son altamente efectivas en la generación de texto coherente y relevante. Debido a su capacidad para predecir palabras y secuencias de manera contextualizada que se ve en el Capítulo 2,

GPT se destaca en tareas como puede ser la generación de contenido creativo. En [PP20] aprovechan la capacidad de este modelo para proponer una forma de reentrenar BERT para la tarea de RE cuando no se tiene disponible un dataset adecuado. El método se denomina *Data Augmented Relation Extraction* (DARE) y básicamente utilizan GPT-2 para generar más datos parecidos a los del dataset inicial que se tiene para entrenar el modelo. Se empieza anotando para cada oración en el dataset inicial sus entidades y las respectivas relaciones existentes entre ellas. Es entonces que se generan n subconjuntos del dataset en el que en cada uno de estos se guardan todas las oraciones que tienen la misma relación entre entidades. Se procede después a un *ajuste fino* del modelo GPT-2 con cada uno de los subconjuntos para especializarlo en cada específica relación entre entidades. Posteriormente se le pedirá a cada modelo reentrenado para una relación que genere más frases distintas las cuales añadir al nuevo dataset. Se reentrena con este nuevo dataset el modelo BERT o derivado de nuestra preferencia. En [PP20] se demuestra empíricamente, con experimentación y análisis de varias modificaciones del método general aquí descrito, es especialmente adecuado para lidiar con un desequilibrio de clases o con datos limitados.

Capítulo 4

Metodología del proyecto

En este Capítulo se realiza una descripción detallada de cual ha sido el flujo de trabajo para llegar a generar el modelo de extracción de relaciones. En la Sección 4.1 se realiza un estudio de las dos herramientas principales para la generación de un modelo de RE mediante el reentreno de un *Transformer*, son comparados y se elige SpaCy3 para el proyecto. En la Sección 4.2 se presenta la *pipeline* que ha seguido este trabajo. Las siguientes secciones proceden a describir con más profundidad el *pipeline presentado*: en la 4.3, la 4.4 y la 4.5 se presentan los datos elegidos para el entrenamiento del modelo y el respectivo análisis y preprocesado hecho; en la Sección 4.6 y 4.7 se presenta el modelo *Transformer* y el modelo NER elegido.

4.1. Herramientas

Como se ha visto anteriormente, hacer buenos *embeddings* de las palabras del texto es suficiente para lograr resultados de vanguardia. Para ello en los últimos años lo que más ha servido es la arquitectura *Transformer* descrita en el Capítulo 2. Es por eso que se han creado varias herramientas de software libre dedicadas a la NER y la RE utilizando *Transformers*. Aquí presentamos las dos más usadas en los últimos años.

4.1.1. Spacy

SpaCy [SVL23] es una herramienta avanzada de NLP, reconocida por su eficiencia, velocidad y precisión. Con su última versión, SpaCy3, esta biblioteca ha avanzado aún más, ofreciendo una gama ampliada de capacidades y funcionalidades avanzadas para el análisis y procesamiento de texto. Desde tareas como análisis sintáctico y tokenización hasta reconocimiento de entidades y extracción de relaciones, se proporciona un conjunto completo de herramientas para satisfacer diversas necesidades en el ámbito del NLP. Tiene una arquitectura modular que permite a los desarrolladores combinar y personalizar componentes según los requisitos específicos de cada proyecto, facilitando así la creación de *pipelines* de procesamiento de texto adaptadas a distintos contextos y aplicaciones. Además, SpaCy3 ha sido optimizado para ofrecer un rendimiento y escalabilidad mejorados, lo que lo convierte en una opción destacada tanto para proyectos de investigación y desarrollo como para implementaciones a gran escala en entornos de producción.

El artículo [SVL23] ofrece una implementación de Spacy3 de un nuevo componente personalizado desde cero. Construyen un modelo de aprendizaje automático en *Thinc* (biblioteca de Python diseñada para construir y entrenar modelos de aprendizaje automático, que ofrece una API de programación funcional), implementando un nuevo componente de SpaCy3, entrenado con un nuevo sistema de configuración y demostrando cómo usar un modelo preentrenado de transformers de la biblioteca *Hugging Face Transformers* para mejorar el rendimiento. El desafío específico que se plantean en [SVL23] es implementar un componente personalizado para predecir relaciones entre entidades nombradas.

El modelo básicamente sigue los siguientes pasos, dados en las imágenes 4.1 y 4.2. Se explicará mediante un ejemplo de un texto de medicina:

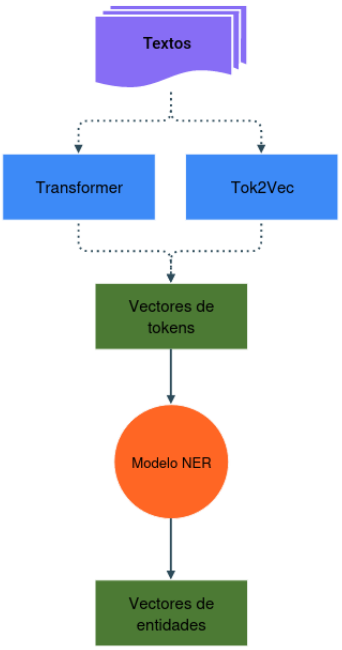

Figura 4.1: Primera parte de Spacy en el proceso de RE

Vamos a basar el ejemplo en la frase: "`Paciente presentaba biopsia negativa`". Es una frase con 4 tokens y con dos entidades: *biopsia* y *negativa* con la correspondiente etiquetación de *EVENTO* (médico) y *MEDIDA*. En el primer paso se tokeniza el texto y se pasa por una capa que calcula una representación vectorial para cada uno de ellos. Esta capa puede ser elegida entre 2 posibles:

- **Tok2Vec:** componente propia de las *pipelines* de SpaCy3 que recibe un *token* y devuelve un respectivo *embedding* obtenido a través de un entrenamiento previo. Es una capa que funciona de forma parecida a *Word2Vec*, es entrenada para ofrecer los mejores embeddings posibles para la *pipeline* completa de Spacy3 que se está entrenando. Por los resultados obtenidos en [SVL23], la capa *Tok2Vec* no llega a ofrecer tan buenos resultados como BERT en la creación de embeddings pero es mucho más rápido y necesita menos recursos para ser entrenado.

- **Transformer:** se puede utilizar cualquier transformer disponible en la biblioteca *Hugging Face Transformers* como componente del *pipeline* en la obtención de una buena representación vectorial de un *token*.

Se quedaría entonces la tokenización [`Paciente`, `presentaba`, `biopsia`, `negativa`] y después del Transformer/Tok2Vec los vectores $[x_1, x_2, x_3, x_4]$ donde $x_i \in \mathbb{R}^n$ y n es la dimensión del output de la capa anterior. Posteriormente se pasa la matriz obtenida por el modelo de NER utilizado y se obtiene los vectores de entidades. Si una entidad está compuesta por más de un token (luego vector) se sacará la media entre ellos. En este caso se obtienen $[x'_3, x'_4]$ las entidades etiquetadas adecuadamente. Se sigue el proceso acorde a la Figura 4.2.

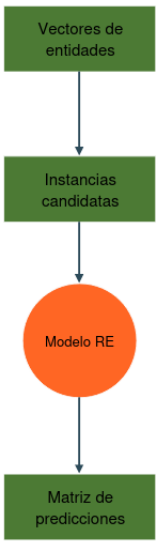

Figura 4.2: Segunda parte de Spacy en el proceso de RE

Se crean instancias candidatas para calcular la probabilidad de una relación. Simplemente crearán todos los pares de entidades posibles entre las encontradas y se concatenan cada una de estas instancias obtenidas creando así una matriz de instancias. En este caso se obtendrían las instancias $[x'_3, x'_4]$ y $[x'_4, x'_3]$. Una vez obtenidas, se pasan por el modelo de RE que se entrena y se obtienen las probabilidades sobre cada relación impuesta en el modelo. Por ejemplo, si se tiene solamente la relación *Pertenece a*, puede salir una probabilidad de 0.9 en la primera instancia y de 0.1 en la segunda. Se impondrá un **umbral** de probabilidad $\theta \in [0, 1]$ que permite seleccionar solo las relaciones precedidas que superen ese umbral. Por ejemplo, si $\theta = 0.5$ entonces el modelo devolverá la instancia $[x'_3, x'_4]$ con las respectivas entidades y relación asociada pero no $[x'_4, x'_3]$.

4.1.2. OpenNRE

OpenNRE [HGY+] es un proyecto contenido en el marco de proyectos de OpenSKL que proporciona una herramienta de código abierto disponible en GitHub que contiene la implementación necesaria tanto para reentrenar ciertos modelos RE como para crear un

nuevo modelo desde cero. Para ello, esta herramienta ha sido diseñada de tal manera que se prioriza la eficiencia computacional mediante la implementación basada en bibliotecas como PyTorch o TensorFlow, para un entrenamiento rápido, además de asegurarse de que la herramienta fuese fácil de utilizar para que aquellas personas que no tuviesen mucha idea de los detalles técnicos pudiesen crear sus propios modelos.

Para ello, OpenNRE contiene algunos modelos reentrenables en su implementación, todos ellos entrenados previamente con un corpus de datos extraído de New York Times y FreeBase. Además, el proyecto [HGY+] contiene la implementación de todas las fases de un modelo, con gran facilidad para extenderlas según las necesidades del programador. Ahora, explicaremos las fases de la implementación de esta herramienta:

- **Tokenización:** OpenNRE implementa la tokenización a nivel de palabra y a nivel de subpalabra, para tratar de ahorrar tiempo a los programadores y cubrir un gran espectro de necesidades. Para ello, existe la clase BasicTokenizer que puede ser extendida para cubrir necesidades específicas en este proceso.

- **Módulo:** esta componente contiene varios módulos neuronales, entre ellos funciones de activación para las distintas capas de red o funciones de agrupamiento. Estos pueden ser utilizados y configurados libremente para un entrenamiento totalmente personalizado.

- **Codificador:** el codificador, el encargado de convertir la entrada en formato de texto en formato legible por las redes neuronales sin olvidar plasmar los puntos importantes de la entrada, también está implementado en OpenNRE de tal manera que se puede extender su funcionalidad a partir de la clase BaseEncoder, en la cual están implementados codificadores como el pre-entrenado de BERT y codificadores convolucionales.

- **Modelo:** en este caso, difiere la implementación dependiendo del uso que se le quiera dar a la herramienta: por un lado están los desarrolladores que no necesitan implementar sus propios modelos, sino que solo quieren reentrenar modelos ya existentes y proporcionados por OpenNRE; y por otro lado incluye la casuística de querer crear un propio modelo de RE. Además, este módulo contiene funciones para mejorar los modelos proporcionados, lo que provoca que la propia herramienta contenga un tutorial de uso muy extenso, ya que se podría replicar un caso de entrenamiento del modelo sin necesidad de ningún recurso más de los proporcionados.

A partir de la inclusión de las modificaciones contenidas en OpenNRE, se consiguieron en la mayoría de los casos resultados de desempeño muy parecidos a los de los papers originales de los distintos modelos y Transformers, y en algunas situaciones, OpenNRE ha sido capaz de mejorar los resultados.

En comparación a Spacy3, OpenNRE tiene tanto ventajas como desventajas:

- **Ventajas:** OpenNRE es una herramienta muy útil en el contexto de investigación, debido a la facilidad en su modificación y a la compatibilidad de esta herramienta con casi cualquier formato de datos estructurados conocido. Trabajos como el de [ROD22b] respaldan los buenos resultados ofrecidos por la herramienta entrenando modelos también en otros idiomas.

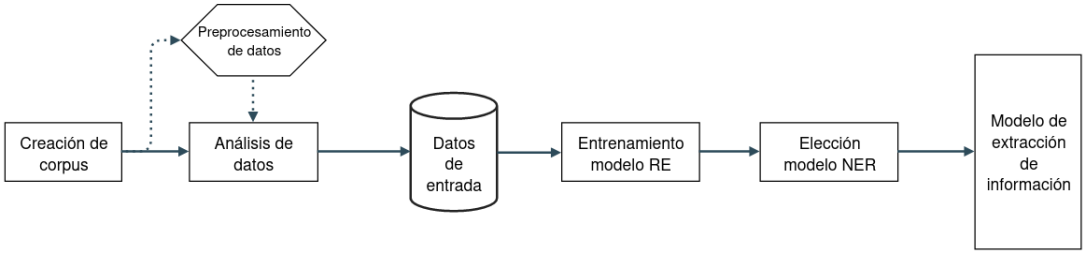

Figura 4.3: Pipeline

- **Desventajas:** OpenNRE ha recibido actualizaciones hasta 2021, por ello se pudieron contemplar ciertos errores de compatibilidad, como la necesidad de descarga de dependencias que ya se encontraban instaladas en los ordenadores de trabajo. Además de ello, la herramienta solo permite reentrenar modelos BERT, mientras que los derivados como RoBERTa, que generalmente ofrecen mejores resultados, no son compatibles.

Es por todo lo anterior que aunque por un tiempo se estuvo trabajando en OpenNRE, finalmente el modelo será generado a través de la herramienta SpaCy3.

4.2. Pipeline

Elegida SpaCy3 como la herramienta principal para la creación y entrenamiento del modelo de RE, se realiza una propuesta de pipeline para la IE en un texto en español, y se presenta en la Figura 4.3. En el primer paso se realiza la creación de un *Corpus* con el cual se entrena el modelo; en el paso 2 se procede a hacer un análisis de datos del corpus, en el cual se limpian los datos y se uniformizan. Posteriormente, en el paso 3 se convierten a formato que acepte Spacy para el entrenamiento, que es ejecutado en el paso 4. Finalmente, en el paso 5 se elige un modelo de NER para unirlo al ya entrenado de RE y obtener en el paso 6 el modelo completo de IE.

4.3. Creación de corpus

Como señalado en el Capítulo 1, antes de comenzar con el desarrollo se debían encontrar los conjuntos de datos con los que se entrenaría al modelo. Para ello, habría que realizar una labor de investigación con el objetivo de hallar una fuente de datos en idioma español lo suficientemente extensa para realizar un entrenamiento con buenos resultados. Se procede en esta Sección a explicar cuales han sido los dataset encontrados y la elección para la creación del corpus.

4.3.1. Dis-Rex

Como comentan sus propios creadores en el artículo [BBM22] este conjunto de datos ha sido creado a partir de la supervisión a distancia, tratando de cumplir dos objetivos que

se consideran críticos: la adición de frases que no contienen ninguna relacion y la adición de frases que contienen más de una relación. Este conjunto de datos contiene 1.5 millones de frases en 4 idiomas distintos, entre ellos español, y está formado por 36 clases de relaciones distintas, además de la clase *Ninguna relación*. Los desarrolladores trataron de crear un conjunto mucho más balanceado de lo que se suele encontrar para este tipo de tareas, y para ello se basaron en Wikipedia, de la que extraen todas las frases necesarias, además de basarse en los números de identificación asignados en Wikidata para cada entidad, lo que permite relacionar palabras que se refieren a la misma entidad (como USA y EEUU). Finalmente, tras este proceso seleccionan las relaciones con mayor representación en el dataset y limitan aquellas relaciones cuya representación en el conjunto supera las 10000 muestras. Con este último ajuste pretenden conseguir un equilibrio entre todas las entidades, evitando una sobreestimación de los modelos entrenados por este conjunto.

4.3.2. MultiCrossRE

Este conjunto de datos, presentado en el artículo [BGP+23] parte del conjunto de datos CrossRE, presentado un año antes en el artículo [BP22], que consta de anotaciones manuales e incluye 17 tipos de relación, además de estar dividido en 6 distintos campos, tales como la música o las noticias. Por tanto, el artículo se centra en el proceso de traducción del conjunto de datos CrossRE. Para ello, se emplea el software comercial DeepL, un modelo que traduce el texto introducido de manera muy eficiente. Gracias a esta herramienta, se puede hallar este conjunto de datos en multitud de idiomas.

4.3.3. RebelDataset

Este conjunto de datos, cuya explicación queda plasmada en el artículo [HCTNN23], tiene como motivación expandir el foco de los conjuntos de datos utilizados en el ámbito de extracción de relaciones a otros idiomas, valiéndose de la anotación automática para conseguir producir conjuntos extensos, aunque introduciendo algunas mejoras para que no hagan mella los efectos de la anotación automática. El proceso de extracción de datos es algo parecido al mencionado en Dis-Rex, las frases son extraídas tanto de Wikidata como de Wikipedia en 18 idiomas distintos, produciendo conjuntos de datos que superan las 400 relaciones. Tras este proceso, y para asegurar una mayor eficacia en la anotación automática de las entidades y su relación, se procedió a anotar manualmente una pequeña porción de los datos como ejemplo. Este último ajuste se realizó en todos los idiomas en los que se encuentra el conjunto. Finalmente, se obtuvo un conjunto de datos que contiene más de 45 millones de frases, distribuidas en los distintos idiomas disponibles.

4.4. Preprocesamiento y análisis de datos

Debido a la distinta procedencia de los conjuntos de datos, se tuvo que realizar un trabajo de uniformización y adecuación a la herramienta elegida para realizar el entrenamiento, para así no provocar incompatibilidades. Este proceso fue diferente dependiendo del conjunto de datos a procesar.

4.4.1. Dis-Rex

En este caso, el conjunto de datos consistía en una gran cantidad de frases extraídas de Wikipedia, que, tras un trabajo de procesado se convertían en líneas JSON que contenían tanto la frase, como información acerca de las entidades, como la posición que ocupaban en la frase y el identificador de la entidad según wikidata. Finalmente, se indicaba el tipo de relación entre estas dos entidades. Estas líneas JSON, en muchos casos, contenían erratas, por tanto, se diseñó un programa que trataba de cargar las líneas JSON, y si estas no se cargaban, se trataba de corregirlas mediante ciertos procedimientos. Cuando se obtuvo el conjunto de datos con los conflictos de carga de JSON ya resueltos, se procedió a solucionar el mayor problema que albergaba este conjunto de datos: las entidades indicadas no contenían el tipo de entidad al que pertenecían, lo que provocaría la obtención de un modelo muy poco eficiente. Para solucionar este contratiempo, se realizó una investigación acerca de todos los tipos de relaciones que se encontraban en al menos alguna oración en todo el conjunto de datos, y, aplicando simplemente la lógica, se trató de inferir los tipos de entidades que podrían formar tales relaciones. Finalmente, y tras descartar aquellas relaciones que se consideraban poco útiles para el contexto de la investigación y aquellas que no daban suficiente información para poder inferir los tipos de entidades que las formaban, se obtuvo el conjunto de datos definitivo con los tipos de entidad indicados.

4.4.2. MultiCross

Para este conjunto de datos, esta etapa fue muy sencilla, ya que en su formato original tenía toda la información necesaria para cada frase. Por tanto, solo se tuvo que evaluar si las líneas JSON contenían errores mediante el script mencionado anteriormente, y así, se obtendría el archivo final listo para su utilización.

4.4.3. RebelDataset

Este caso tuvo algo más de complicación, ya que tras eliminar las líneas JSON que provocaban algún tipo de conflicto, se trató de comprobar la compatibilidad con la herramienta a utilizar para el entrenamiento del modelo, sin éxito. Debido a la poca información que contenían las ventanas de error generadas al tratar de realizar este proceso, se comenzó a investigar acerca del error, hasta que finalmente, observando el propio dataset, se pudo comprobar que el error lo generaban los índices de posición de las entidades en el archivo, ya que el índice que indicaba el final de la entidad en el texto no se correspondía con el índice real. Se procedió a implementar un script que modificaba los valores de los índices de todas las entidades contenidas, y se volvió a comprobar la compatibilidad, ahora sí, con éxito. Más adelante, y ya entrados en la fase de entrenamiento, a partir de resultados muy poco satisfactorios, se incluyó una modificación que consistía en evaluar todas las relaciones que aparecían en el conjunto de datos, y eliminar gran parte de estas, ya que tenían muy poca representación, y provocaban un entrenamiento muy poco eficiente. Esto redujo el tamaño del conjunto de datos, pero se mantuvo un tamaño muy considerable para el contexto de nuestro trabajo.

4.5. Datos de entrada

Tras la realización de todas estas modificaciones sobre los distintos conjuntos de datos, se procedió a implementar un script que originaba el archivo binario que utilizaría SpaCy3 para las siguientes fases. Para ello, anteriormente se decidió utilizar para la división de datos los siguientes valores: 80 % del total en datos de entrenamiento, 10 % en datos de validación y 10 % en datos de test. La división se realizó un paso antes ya que el entrenamiento con SpaCy3 obliga a tener en distintos archivos los datos dependiendo de su papel en la fase de entrenamiento. Una vez, divididos, se procedió a crear los archivos con extensión "`.spacy`", a partir de la librería DocBin del propio SpaCy3, que primeramente convertía los datos en formato binario interpretable por SpaCy3, para luego crear un archivo y volcarlos en éste.

4.6. Entrenamiento del modelo

Una vez los datos están en formato para SpaCy3, se procede a la creación y entrenamiento del modelo de RE. Este modelo es el implementado en [SVL23] que está en dos versiones: una sin el uso de *Transformer* y otra con su uso. Debido a la magnitud del corpus creado, se usó la primera versión (con Tok2Vec) para que fuese entrenada con el corpus completo descrito anteriormente. Se entrena también la versión con *Transformer*, pero con 1/10 del corpus, ya que los recursos de memoria solicitados por el entrenamiento eran demasiado grandes. El entrenamiento y los hiperparámetros elegidos se presentan en el Capítulo 5.

El transformer elegido para el modelo RE es RoBERTa de [GAP+21] entrenado con un corpus de 570GB de la Biblioteca Nacional Española. Esta elección se debe principalmente a que está demostrado experimentalmente, como descrito en el Capítulo 3, que modelos más grandes ofrecen mejores resultados en tareas como NER y RE. Por ejemplo, en la presentación de modelos como BETO o ALBETO se observa que logran con menos hiperparámetros llegar a casi los mismos resultados en cuanto a métrica en casi todas las tareas de NLP que BERT, menos en casos como NER en el que la diferencia es mayor. Además el modelo de este proyecto no está pensado para ser usado en aplicaciones *a tiempo real* (que funcionan en un período de tiempo que el usuario percibe como inmediato), por ello no es un óbstaculo que la latencia en extraer información de un texto sea más o menos larga. Entre los modelos grandes, los principales son RoBERTa y BERT: se elige RoBERTa al optimizar BERT en todas las tareas y al tener una versión en español como es la presentada en [GAP+21].

4.7. Elección del modelo NER

El modelo elegido para la extracción de entidades es el construido en [OC23]. Los autores dedicaron su investigación en crear un modelo basado en RoBERTa para NER especializado en textos legales en español. Para ello utilizan el mismo Transformer de [GAP+21] que será utilizado para el modelo de RE, en el cual han aplicado un *fine tuning* para textos legales. La herramienta que utilizaron para el re-entrenamiento sigue siendo SpaCy3, lo que ha facilitado enormemente la tarea de acoplamiento de su modelo con el de extracción de relaciones. En [OC23] presentan unos buenos resultados en detectar clases como puede ser PER (Persona) y DATE (Fecha) con unos respectivos *F1-Score* del 84 %

y 74 %. Un ejemplo de extracción de entidades nombradas en textos legales españoles es la presentada en la Figura 4.4.

Figura 4.4: Ejemplo de entidades encontradas por el modelo de [OC23].

Capítulo 5

Experimentos y Resultados

En este Capítulo se describe la experimentación hecha para la generación del modelo. Se incluye la descripción del corpus y su preprocesamiento en la Sección 5.1, su entrenamiento y evaluación en la Sección 5.2 y 5.3 y por último un ejemplo de la salida del modelo en la Sección 5.4.

5.1. Preprocesamiento de los datos

En este caso, como se explica en el Capítulo 4, la fase de preprocesamiento varía dependiendo del formato utilizado en cada conjunto de datos. Tras la adecuación de todos los conjuntos de datos, y debido a cuestiones de tamaño y compatibilidad de entidades, se decidió utilizar una mezcla de los conjuntos DisRex y RebelDataset. Tras un extenso estudio acerca de los tipos de relaciones contenidos en cada conjunto, se decide filtrar los datos de estos conjuntos para conservar solo las 16 relaciones con mayor peso. Tras este filtrado, se realiza un estudio acerca del número de apariciones de estas relaciones, y tras observar los datos obtenidos, se decide sustraer ejemplos de algunas relaciones para evitar el sobreentrenamiento para ellas, y obtener como resultado un conjunto más equilibrado. Finalmente, se obtienen las relaciones indicadas en la Tabla 5.1.

Tras obtener el conjunto de datos definitivo, con los pesos de las relaciones ya conocidos, se crea el script que convertiría el archivo de texto contenedor en un archivo listo para ser utilizado por los comandos de entrenamiento de la herramienta SpaCy3, presentada en el Capítulo 4. Este script recibe un vector de líneas JSON, cuyo contenido son las distintas frases y anotaciones de estas frases, y va convirtiendo una a una todas las líneas en objetos de tipo *Doc*, un objeto de la librería SpaCy3 que contiene toda la información necesaria de la frase, con anotaciones acerca de las posiciones de los tokens que la componen, las entidades que participan en la relación contenida en la frase, así como el tipo de entidad y finalmente el tipo de relación que forman. Un punto clave para crear estos objetos, es informar de la adición de una nueva anotación a estos objetos de tipo *Doc*, que sería el tipo de relación que contiene, ya que en su formato original no existe esta anotación. Para ello, antes de ejecutar el bucle de cálculo, se debe ejecutar la siguiente instrucción:

```
1 Doc.set_extension("rel", default={})
```

Tras ejecutar esta instrucción, los objetos Doc aceptan la adición de una relación entre dos entidades a las anotaciones. Por tanto, se puede pasar al relleno de información en estos objetos, que en este caso se compondría por la creación de las dos entidades, con

Nombre de la relación	Número de apariciones
Lugar de nacimiento	77256
Fecha de nacimiento	115547
País	100000
Lugar de fallecimiento	25425
Fecha de fallecimiento	40201
Miembro de equipo	20708
Liga	15656
Sigue a	11312
Hijo/a	8468
Descubridor o inventor	8381
Idioma oficial	7090
Esposo/a	6831
Hermano/a	5760
Capital	4669
Fundado por	2148
Obra notable	2001

Tabla 5.1: Tabla de relaciones y número de apariciones.

sus posiciones de inicio y final relativas a la frase en la que se encuentran, y la etiqueta del tipo de entidad a la que pertenecen. Para el primer punto, debido a la naturaleza del conjunto de datos escogido, se debería traducir las posiciones de las entidades que se encuentran en el archivo fuente; es decir, se deben convertir las posiciones en número de caracteres anotados en el archivo en posiciones con respecto a la cantidad de tokens que se encuentran en la frase. Para ello, SpaCy3 cuenta con una función que realiza esta traducción de manera automática, lo que facilita este proceso. La instrucción a ejecutar sería:

```
    span = doc.char_span(inicio, fin, label=etiqueta)
```

Y el proceso que seguirían todos los *tokens* sería el de la Figura 5.1.

Finalmente, se añadiría la relación contenida en la frase al objeto Doc referenciando las posiciones de inicio anteriormente calculadas que corresponden a las entidades que forman esta relación.

El siguiente paso sería traducir este nuevo conjunto de objetos de tipo Doc al lenguaje binario, el legible por las herramientas de SpaCy. Para ello, la librería cuenta con este par de funciones que realizan la labor y vuelcan los datos a un archivo nuevo cuya extensión sería ".spacy":

```
    train_docbin = DocBin(docs=train_docs, store_user_data=True)
    train_docbin.to_disk("train_Rebel.spacy")
```

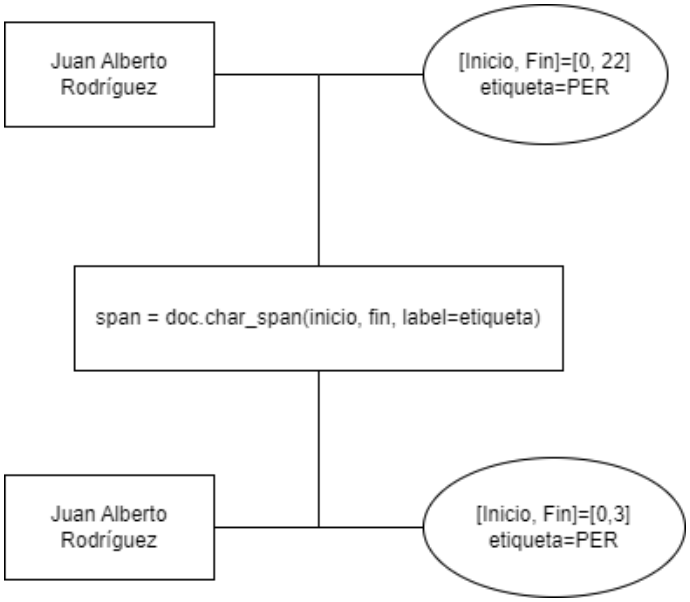

Figura 5.1: *F1-Score* en relación del número de pasos para *Conversión de las posiciones de las entidades*.

Tras cargarse satisfactoriamente el documento, este estaría listo para ser utilizado en las herramientas de entrenamiento de SpaCy sin provocar ninguna incompatibilidad. Este punto se logró tras realizar muchas pruebas y localizar muchos errores de formato una vez comenzada la fase de entrenamiento, ya que muchos de estos errores no eran percibibles hasta que se trataba de realizar el entrenamiento del modelo, además de localizar ciertas desventajas del conjunto de datos una vez finalizados los entrenamientos del modelo con particiones del conjunto. Por tanto, el preprocesamiento de datos fue constante en la siguiente fase.

5.2. Entrenamiento

Se procedió a realizar dos entrenamientos: uno para un modelo sin el uso de los *Transformer*, creando los *embeddings* con la herramienta Tok2Vec de SpaCy3 y otro con RoBERTa. Para los expermientos sobre el entrenamiento, tanto de hiperparámetros como de dataset, se utilizó un ordenador con Windows 10, memoria RAM 8 GB y procesador Intel Core i5. Una vez solucionado y comprobado que Spacy lograba entrenar el modelo con pequeñas partes del dataset, se procedió a pasar todos los *scripts* de Python y los datos al ordenador más potente y se entrenó completamente el modelo.

Para la primera versión del modelo (con *Tok2Vec*), se utilizó todo el corpus explicado en la Sección anterior y bastó entrenarlo con el procesador propio del ordenador descrito anteriormente. Para el segundo entrenamiento, aplicando ya la arquitectura *Transformer*, se hicieron bastantes pruebas cambiando los modelos que se utilizaban y la cantidad del corpus con la cual se entrenaban. Estas ejecuciones se hicieron en un ordenador con Windows 10, con una memoria RAM de 32 GB, y un procesador de Intel(R) Core(M) i7-8700B. Para una ejecución más rápida, se utilizó una GPU NVIDIA GeForce RDX 3070 con 8 GB de VRAM.

El entrenamiento del modelo, una vez descargadas las dependencias y configurado todo

Hiperparámetro	Valor
Pasos totales	700
Tamaño *batch*	256
Umbral	0.5
Early stop patience	1600000
Tamaño *batch Transformer*	32
Optimizador	AdamW con *corrección de sesgo*
Scheduler	Linear con *Warmup*
Valor de regularización $L2$	0.01
Término de suavizado	$1 * 10^{-8}$
Valor inicial tasa de aprendizaje	$5 * 10^{-5}$
Pasos *warmup*	250
max_length	100

Tabla 5.2: Tabla de hiperparámetros durante el entrenamiento para la versión con *Transformer*.

de [SVL23] correctamente, se ejecutó con la instrucción

```
!spacy project run train_gpu
```

En particular, lo más importante en esta instrucción era el llamado archivo de configuración en el cual se indica todo lo necesario para el entrenamiento con los datos. SpaCy3 se encarga de cargar los *paths* de cada uno de los *splits* del dataset en entrenamiento, validación y test y del archivo llamado `rel_trf.cfg`, en el cual se indica el transformer utilizado y los hiperparámetros impuestos que son los presentados en la Tabla 5.2.Para el modelo de *Transformer* elegido, se hicieron pruebas tanto con BETO como con DistilBETO, pero ofrecieron resultados con baja precisión respecto a los modelos que hacen uso de Tok2Vec y RoBERTa (en particular, `PlanTL-GOB-ES/roberta-base-bne` de [GAP+21]).

Durante el entrenamiento, Spacy va evaluando el modelo que está siendo entrenado cada *eval_frecuency* pasos, en el que se impone, teniendo como pasos totales 10000, *eval_frecuency* = 500 para Tok2Vec y pasos totales 700, *eval_frecuency* = 100 para Transformer. En el entrenamiento cada *eval_frecuency* se mostrarán lás métricas en el datataset de evaluación para el modelo. Los resultados del *F1-Score* son mostrados en la Figura 5.2 para Tok2Vec y 5.3 para Transformer. El máximo alcanzado para Tok2Vec es 0.95 y en Transformer es 0.88.

5.3. Evaluación

Se presentan los resultados obtenidos en el modelo explicado anteriormente para el conjunto de datos de *test*. Para ello, el propio Spacy tiene su proprio *script* ejecutado con el comando a continuación. En este mismo, se hace un estudio de las métricas *Precision*,

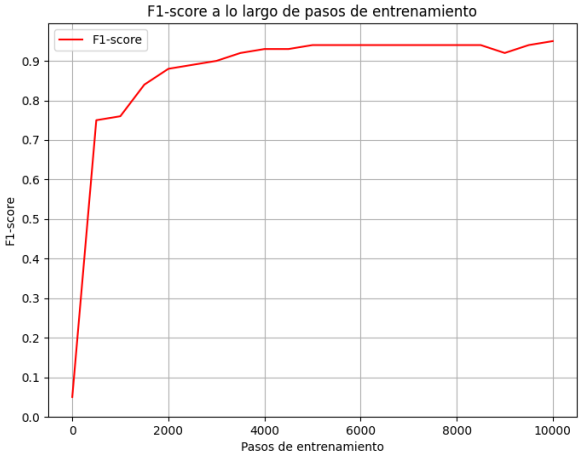

Figura 5.2: *F1-Score* en relación del número de pasos para *Tok2Vec*.

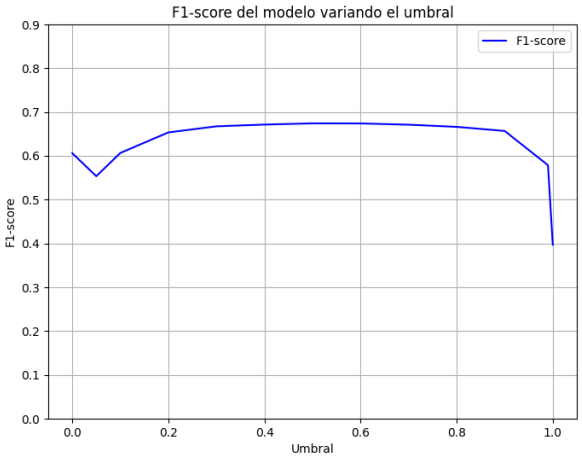

Figura 5.3: *F1-Score* en relación del número de pasos para *Transformer*.

Recall y *F1-Score* variando el umbral de probabilidad entre 0 y 1 para decidir el óptimo que dejar por defecto en el modelo.

```
!spacy project run evaluate
```

5.3.1. *Precision* y *Recall*

La métrica *Precision* se calcula una vez medidos el número de los verdaderos positivos y falsos positivos en la clasificación (donde se ve como clasificación el asignar una relación u otra a un par de entidades). La *Precision* se define, como en [Pah17], mediante la fórmula:

$$\text{Precision} = \frac{\text{Total de verdaderos positivos}}{\text{Total de verdaderos positivos} + \text{Total de falsos positivos}} \in [0,1] \qquad (5.1)$$

Esta métrica indica una proporción de cuantas veces que el algoritmo indica una relación, esa relación es cierta realmente. En las Figuras 5.4 y 5.5 tenemos la *Precision* del modelo Tok2Vec y Transformer respectivamente para distintos valores del umbral de probabilidad θ. Se observa en que el modelo con Tok2Vec tiene un sobreentrenamiento porque llega a valores de precisión muy elevados para la mayoría de umbrales, mientras que con Transformer la prograsión es más normal, al tener valores muy altos de precisión solo cuando el umbral es elevado (entonces solo clasificará relaciones de las que el nivel de seguridad del modelo sea muy alto).

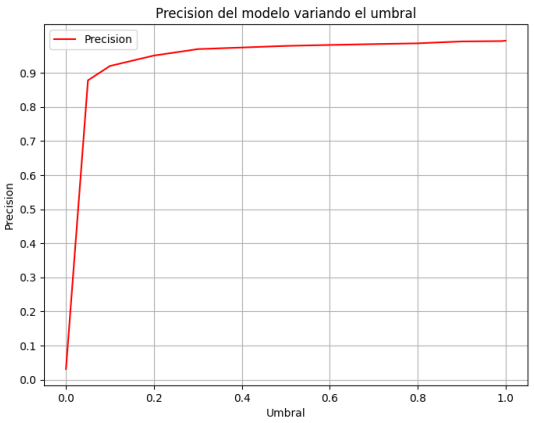

Figura 5.4: *Precision* en relación del umbral para *Tok2Vec* ya entrenado.

Por otro lado, tenemos *Recall* que mide cuanto el modelo se equivoca y no predice (dentro del umbral impuesto) una cierta relación entre dos entidades. Después de la obtención del número de falsos negativos, la *Recall* se define, como en [Pah17], mediante la fórmula:

$$\text{Recall} = \frac{\text{Total de verdaderos positivos}}{\text{Total de verdaderos positivos} + \text{Total de falsos negativos}} \in [0,1] \qquad (5.2)$$

En las Figuras 5.6 y 5.7 tenemos la *Recall* del modelo Tok2Vec y Transformer respectivamente para distintos valores del umbral de probabilidad θ. Se observa otra vez el sobreentrenamiento del modelo con Tok2Vec, donde solo con umbral muy alto la métrica baja, mientras que con Transformer con un umbral intermedio la métrica se queda en un nivel intermedio constante.

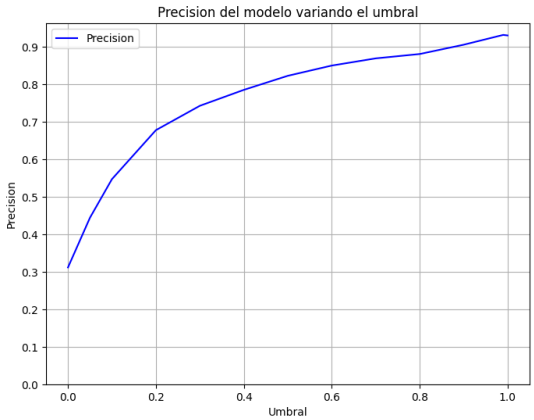

Figura 5.5: *Precision* en relación del umbral para *Transformer* ya entrenado.

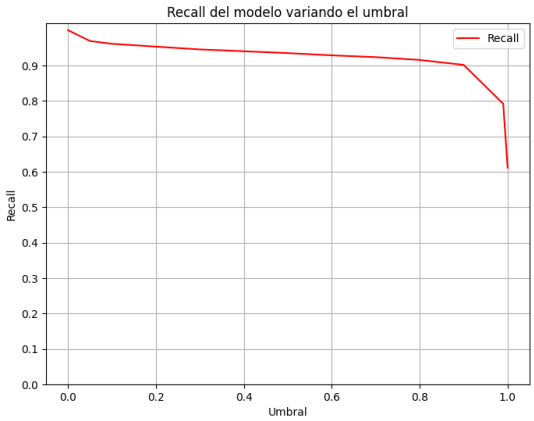

Figura 5.6: *Recall* en relación del umbral para *Tok2Vec* ya entrenado.

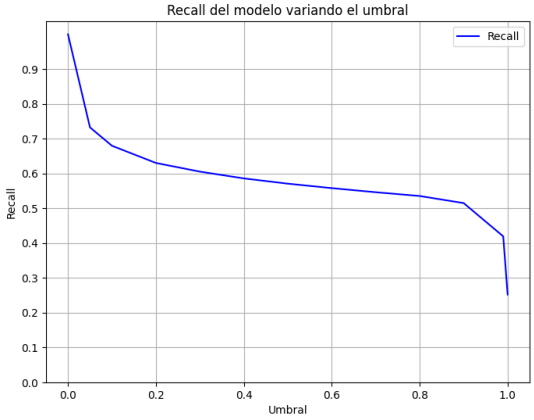

Figura 5.7: *Recall* en relación del umbral para *Transformer* ya entrenado.

5.3.2. *F1-Score*

Para tomar las conclusiones más acertadas para elegir el modelo más óptimo en relación a su umbral, y ver su desempeño en el *test set*, se toma la métrica que balancea las dos anteriores: *F1-Score*. Esta es particularmente útil y común cuando los datos no están del todo balanceados, que es este caso, al tener muchas más relaciones que de otras. La métrica se calcula, como en [Pah17], con la fórmula:

$$\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \in [0, 1] \tag{5.3}$$

Con la fórmula 5.3, se obtiene la siguiente gráfica dependiente del umbral en la Figura 5.8 para el modelo Tok2Vec y en la Figura 5.9 en el modelo de Transformer.

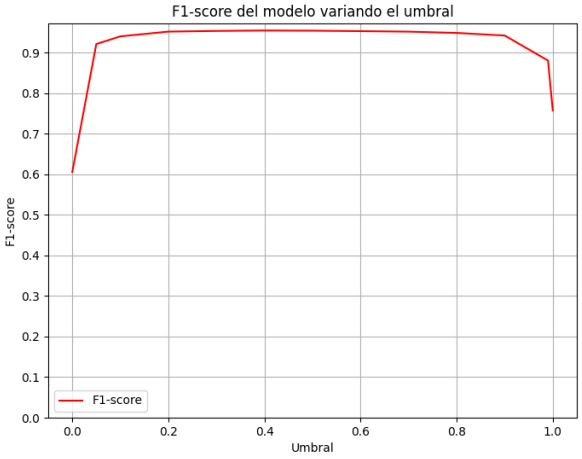

Figura 5.8: *F1-Score* en relación del umbral para *Tok2Vec* ya entrenado.

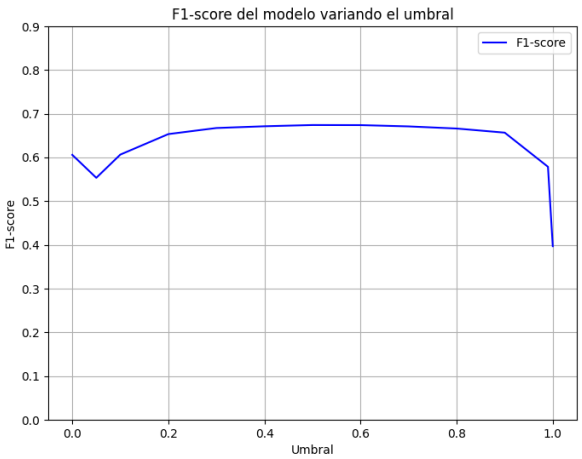

Figura 5.9: *F1-Score* en relación del umbral para *Transformer* ya entrenado.

En cuanto a los resultados visuales, podemos concluir que, en el caso del modelo entrenado con Tok2Vec, debido a la poca demanda de recursos de la fase de entrenamiento, lo que facilitó la posibilidad del entrenamiento con la totalidad de los datos disponibles, las

relaciones entre entidades que representan lugares son muy fácilmente interpretadas por el modelo, teniendo un porcentaje de éxito muy alto. Sin embargo, cuando los tipos de entidades varían, pasando a representar personas, organizaciones o fechas, el modelo no es capaz de extraer la relación entre éstas con una confianza mayor al 50 %, cometiendo errores de interpretación en las relaciones entre dos personas o entre una persona y una fecha.

Este no es el caso con el modelo entrenado con el *Transformer*, ya que aún teniendo en cuenta la limitación en cuanto a cantidad de datos a entrenar,en los casos en los que el *Tok2Vec* no es capaz de extraer las relaciones correctas, este modelo se comporta de manera muy satisfactoria, siendo capaz de extraer relaciones mucho más ambiguas y con menos contexto, además de comprender muy bien el contexto que rodea a estas entidades, sacando conclusiones a partir de los distintos componentes del texto.

5.3.3. Matriz de confusión

La matriz de confusión es una herramienta muy útil en el contexto del aprendizaje automático para visualizar la efectividad del modelo obtenido. En nuestro caso, es una matriz cuadrada de orden 16 en la cual cada fila y cada columna representan una de las relaciones de nuestro modelo. Las columnas contienen el número de predicciones de cada clase, y la diagonal principal es el lugar donde coinciden las predicciones con la relación real. Como podemos ver en la Figura 5.10, las relaciones entre personas son aquellas donde más falsos positivos se dan, mientras que las relaciones entre localizaciones y otro tipo de entidades son predichas de manera satisfactoria. También se puede puntualizar que solo 4 de las 16 relaciones son predichas de manera correcta en menos del 50 % de las ocasiones; estas son las relaciones de *hijo*, *trabajo reseñable*, *idioma oficial* y *capital*, lo que puede ser totalmente lógico debido a la ambigüedad del contexto en el que nos podemos encontrar con estas relaciones y el parecido con otras también evaluadas.

A partir de esta matriz de confusión se puede generar una tabla en la que se muestren las distintas relaciones y la frecuencia con la que se han predicho de manera correcta, y de manera incorrecta, para así poder comparar el rendimiento del modelo con las distintas relaciones. De tal modo, obtenemos la tabla 5.3, en la cual podemos destacar:

5.4. Ejemplo de salida del modelo

El modelo completo de IE se construye uniendo el modelo NER de [OC23] y el generado anteriormente con RoBERTa. Por el análisis de las métricas de la Sección anterior entre el modelo *Tok2Vec* y el modelo *Transformer*, se concluye que este último es mejor elección. Analizado en la evaluación el *F1-Score* variando el umbral de probabilidad en un intervalo de $[0, 1]$ (Figura 5.9), se obtiene que el valor óptimo del parámetro es $\theta = 0.5$.

A continuación se muestran ejemplos de la extracción de información que obtiene el modelo dado un texto en español.

"Nikola Tesla, nacido el 10 de julio de 1856 en Smiljan, en la región de la actual Croacia, fue un visionario inventor e ingeniero serbio-estadounidense que dejó una huella indeleble en la historia de la tecnología y la ciencia. Sus padres, Milutin Tesla y Georgina Đuka Tesla, procedían de familias serbias y se dedicaban a la agricultura. Desde pequeño, Tesla mostró una curiosidad innata por el mundo que le rodeaba, así como unas habilidades

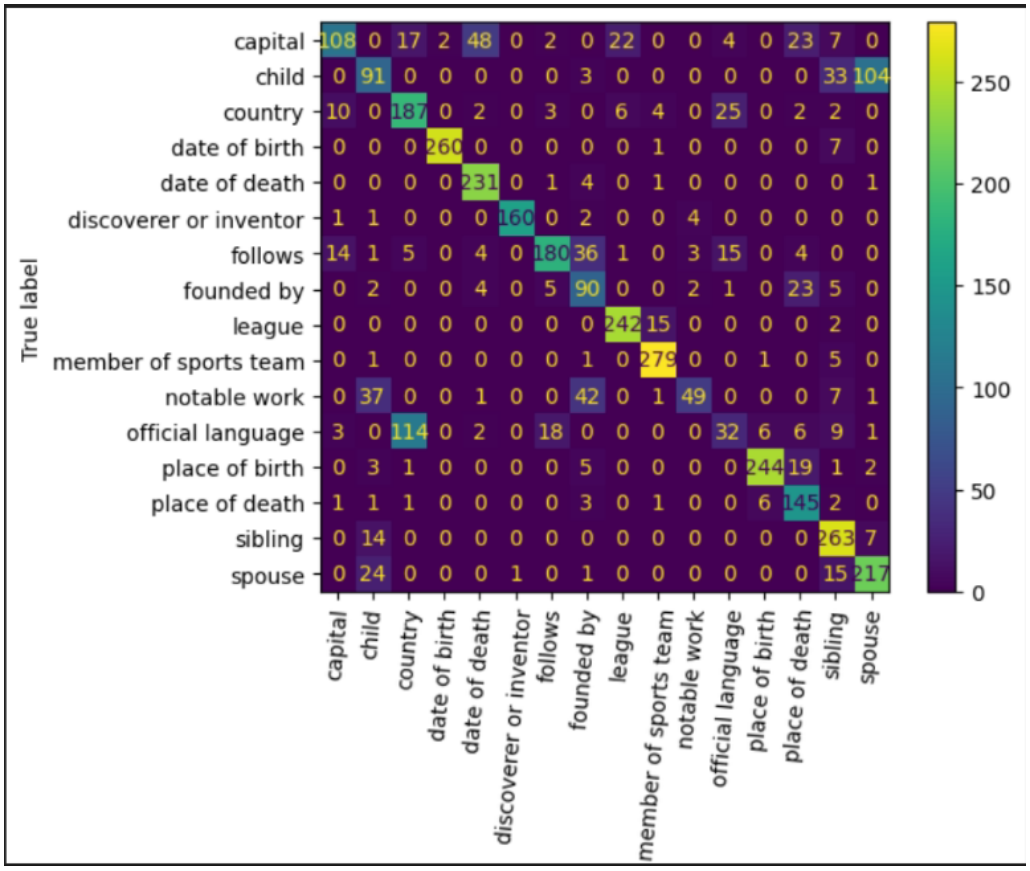

Figura 5.10: Matriz de confusión del modelo entrenado al evaluarlo con los datos de test.

Nombre de la relación	**Nº aciertos**	**Nº fallos**	**% aciertos**
Lugar de nacimiento	244	41	86 %
Fecha de nacimiento	260	8	97 %
País	187	54	78 %
Lugar de fallecimiento	145	15	91 %
Fecha de fallecimiento	231	7	97 %
Miembro de equipo	279	8	97 %
Liga	242	17	93 %
Sigue a	180	83	69 %
Hijo/a	91	140	39 %
Descubridor o inventor	160	8	95 %
Idioma oficial	32	159	17 %
Esposo/a	217	41	84 %
Hermano/a	263	21	93 %
Capital	108	125	46 %
Fundado por	90	42	68 %
Obra notable	49	89	36 %

Tabla 5.3: Tabla de relaciones y efectividad de predicción.

excepcionales en matemáticas y mecánica. Tras completar su educación secundaria en Croacia, Tesla continuó sus estudios en la Universidad Técnica de Graz, en Austria, antes de trasladarse a la Universidad Caroline de Praga y, finalmente a la Universidad Técnica de Graz. Durante este tiempo, Tesla comenzó a concebir ideas revolucionarias, como el concepto de motor de corriente alterna, que más tarde se convertiría en uno de sus mayores logros. En 1884, Tesla emigró a Estados Unidos en busca de oportunidades para desarrollar y compartir sus inventos. Rápidamente se unió a la compañía de Thomas Edison, pero sus ideas sobre la corriente eléctrica provocaron una división. Tesla se asoció con George Westinghouse y juntos cambiaron el panorama energético al desarrollar y promover el sistema de corriente alterna, que resultó ser más eficiente que el sistema de corriente continua de Edison. A lo largo de su vida, Tesla patentó más de 300 inventos y descubrimientos, entre ellos la bobina de Tesla, el transformador de Tesla y el motor de inducción. Su obsesión por la transmisión inalámbrica de energía y la comunicación a larga distancia también dio lugar a proyectos como la Torre de Wardenclyffe Tower, aunque muchos de ellos no llegaron a completarse por dificultades financieras. La vida de Tesla estuvo marcada por éxitos innovadores, pero también por luchas personales y desafíos económicos. A pesar de su genialidad, Tesla murió el 7 de enero de 1943 en Nueva York, en una relativa oscuridad. Hoy, su legado perdura y sus contribuciones siguen influyendo en la tecnología moderna, dejando una huella indeleble en la historia de la ciencia y la electricidad."

El modelo, en menos de 20 segundos, localiza en primer lugar las entidades que se muestran en la Figura 5.11. Como se puede observar se logran obtener muchas de las entidades claras de PERSONA o de LOCALIZACIÓN como pueden ser `Thomas Edison` o `Universidad Caroline de Praga`. Si se buscan otro tipo de entidades, bastaría cambiar el modelo de NER a uno entrenado con más precisión para una temática concreta de textos. De hecho [OC23] es un modelo especializado en textos legales, del cual tenemos un ejemplo posterior a este.

```
Entidades en el texto:
{'Tipo': 'LOC', 'título': 'Universidad Técnica de Graz'}
{'Tipo': 'LOC', 'título': 'Croacia'}
{'Tipo': 'LOC', 'título': 'Austria'}
{'Tipo': 'PER', 'título': 'Thomas Edison'}
{'Tipo': 'MISC', 'título': 'inventor'}
{'Tipo': 'PER', 'título': 'Milutin'}
{'Tipo': 'PER', 'título': 'Đuka Tesla'}
{'Tipo': 'PER', 'título': 'Georgina'}
{'Tipo': 'LOC', 'título': 'Nueva York'}
{'Tipo': 'LOC', 'título': 'Universidad Caroline de Praga'}
{'Tipo': 'PER', 'título': 'Tesla'}
{'Tipo': 'ORG', 'título': 'Universidad de Graz, Austria'}
{'Tipo': 'PER', 'título': 'George'}
{'Tipo': 'LOC', 'título': 'Estados Unidos'}
{'Tipo': 'PER', 'título': 'Nikola'}
{'Tipo': 'LOC', 'título': 'Smiljan'}
```

Figura 5.11: Subconjunto de las entidades encontradas en el primer texto por el modelo de [OC23].

Posteriormente se muestran las posibles relaciones entre las entidades encontradas, que son las expuestas en la Figura 5.12. Se observa la buena confianza con la cual se identifican las relaciones de posición y fecha. Otras relaciones aparecen con menos confianza, como puede ser `founded_by`. Relaciones entre personas no se muestran directamente al tener un nivel de confianza demasiado bajo.

Se procede a dar otro ejemplo de un texto de área temática general, en este caso en vez de ser una biografía es una descripción de una empresa.

"La empresa SpaceX, fundada por Elon Musk en 2002, es conocida por su papel innovador en la industria aeroespacial. Con sede en Hawthorne, California, SpaceX ha lanzado varios cohetes y ha llevado a cabo misiones para la NASA. El objetivo principal de SpaceX es hacer que los viajes espaciales sean más accesibles y eventualmente llevar humanos a Marte. Elon Musk, quien también es CEO de Tesla, tiene una larga historia de involucrarse en proyectos tecnológicos revolucionarios. Tesla, la empresa de automóviles eléctricos, ha liderado la industria de vehículos eléctricos, produciendo modelos como el Model S, Model 3, y Model X. Musk ha expresado su visión para una transición global hacia la energía sostenible. SpaceX tuvo su primer gran éxito en 2010 cuando el cohete Falcon 9 completó su primera misión con éxito. Desde entonces, SpaceX ha desarrollado y lanzado el cohete Falcon Heavy, uno de los cohetes más potentes del mundo. En 2020, SpaceX hizo historia al

```
Relaciones entre entidades encontradas en el texto:
{'origen': 'Tesla', 'destino': 'Nueva York', 'tipo': 'place of death', 'confianza': 0.96289915}
{'origen': 'Tesla', 'destino': '7 de enero de 1943', 'tipo': 'date of death', 'confianza': 0.93527657}
{'origen': 'Smiljan', 'destino': '10 de julio de 1856', 'tipo': 'date of birth', 'confianza': 0.9537328}
{'origen': 'Universidad Técnica de Graz', 'destino': 'Croacia', 'tipo': 'country', 'confianza': 0.90856236}
{'origen': 'Smiljan', 'destino': 'Croacia', 'tipo': 'country', 'confianza': 0.92596585}
{'origen': 'Nikola', 'destino': '10 de julio de 1856', 'tipo': 'date of birth', 'confianza': 0.9677365}
{'origen': 'Tesla', 'destino': 'George', 'tipo': 'founded by', 'confianza': 0.6421633}
{'origen': 'inventor', 'destino': 'Croacia', 'tipo': 'country', 'confianza': 0.5304043}
{'origen': 'inventor', 'destino': '10 de julio de 1856', 'tipo': 'date of birth', 'confianza': 0.95928425}
{'origen': 'Nikola', 'destino': 'Smiljan', 'tipo': 'place of birth', 'confianza': 0.9027927}
{'origen': 'Tesla', 'destino': '10 de julio de 1856', 'tipo': 'date of birth', 'confianza': 0.9671552}
{'origen': '10 de julio de 1856', 'destino': 'Croacia', 'tipo': 'country', 'confianza': 0.778833}
{'origen': 'Universidad Técnica de Graz', 'destino': 'Tesla', 'tipo': 'founded by', 'confianza': 0.5031833}
{'origen': 'Tesla', 'destino': 'Smiljan', 'tipo': 'place of birth', 'confianza': 0.9031767}
{'origen': 'Universidad Técnica de Graz', 'destino': 'Austria', 'tipo': 'country', 'confianza': 0.9089221}
{'origen': 'Croacia', 'destino': '10 de julio de 1856', 'tipo': 'date of birth', 'confianza': 0.91926306}
{'origen': '10 de julio de 1856', 'destino': 'Smiljan', 'tipo': 'place of birth', 'confianza': 0.70484555}
{'origen': '7 de enero de 1943', 'destino': 'Nueva York', 'tipo': 'place of death', 'confianza': 0.75557387}
{'origen': 'Nueva York', 'destino': '7 de enero de 1943', 'tipo': 'date of death', 'confianza': 0.8723529}
```

Figura 5.12: Relaciones encontradas en el primer texto por el modelo obtenido en el presente trabajo.

ser la primera empresa privada en enviar astronautas a la Estación Espacial Internacional como parte del programa NASA Commercial Crew. La empresa también ha desarrollado el sistema de lanzamiento Starship, con el objetivo de llevar misiones tripuladas a Marte y otros destinos en el espacio profundo. SpaceX ha establecido instalaciones de lanzamiento en lugares como Cabo Cañaveral, Florida, y Boca Chica, Texas. Además, la empresa tiene acuerdos con la NASA para futuras misiones espaciales y planea lanzar una serie de satélites para proporcionar servicios de internet de alta velocidad a nivel mundial. Elon Musk ha sido una figura controvertida por sus declaraciones y acciones públicas. En 2018, enfrentó críticas por sus comentarios en Twitter y tuvo problemas legales con la SEC (Comisión de Bolsa y Valores de Estados Unidos). Sin embargo, sigue siendo un líder influyente en la industria tecnológica y aeroespacial."

Se estudian primero las entidades encontradas por el modelo (Figura 5.13) donde la ejecución tiene una duración de 15 segundos. Se puede observar que en este caso son también muy acertadas: todas las encontradas tienen sentido con su etiquetación de entidad y son las más importantes en el texto. Las entidades de tipo ORG (organización), que predominan en este texto, las ubica también de manera correcta el modelo.

Posteriormente se muestran las relaciones encontradas entre las entidades anteriores con un nivel de confianza mayor de 0.5. Se observa que aparecen relaciones no presentes en el texto anterior. Por ejemplo la relación `follows`, con unos niveles de confianza en algunos casos muy altos, como por ejemplo con las entidades `Model X` y `Model S`. Otras relaciones como `founded_by`, con alta confianza, acierta muy bien. En general entonces el resultado

```
{'Tipo': 'ORG', 'título': 'SEC'}
{'Tipo': 'LOC', 'título': 'Hawthorne, California'}
{'Tipo': 'LOC', 'título': 'Estación Espacial Internacional'}
{'Tipo': 'LOC', 'título': 'cohete Falcon Heavy'}
{'Tipo': 'LOC', 'título': 'cohete Falcon 9'}
{'Tipo': 'ORG', 'título': 'SpaceX'}
{'Tipo': 'ORG', 'título': 'NASA Commercial'}
{'Tipo': 'ORG', 'título': 'Comisión de Bolsa y Valores'}
{'Tipo': 'LOC', 'título': 'Boca Chica'}
{'Tipo': 'LOC', 'título': 'Cabo Cañaveral'}
{'Tipo': 'ORG', 'título': 'Tesla'}
{'Tipo': 'PER', 'título': 'Elon Musk'}
```

Figura 5.13: Subconjunto de las entidades encontradas en el segundo texto por el modelo de [OC23].

es completo y acertado.

```
Relaciones entre entidades encontradas en el texto:
{'origen': 'NASA Commercial', 'destino': 'SpaceX', 'tipo': 'founded by', 'confianza': 0.6063625}
{'origen': 'SpaceX', 'destino': 'cohete Falcon 9', 'tipo': 'follows', 'confianza': 0.82867277}
{'origen': 'Tesla', 'destino': 'Model S', 'tipo': 'follows', 'confianza': 0.74395806}
{'origen': 'Model 3', 'destino': 'Model S', 'tipo': 'follows', 'confianza': 0.93571216}
{'origen': 'SEC', 'destino': '2018', 'tipo': 'follows', 'confianza': 0.844762}
{'origen': 'Comisión de Bolsa y Valores', 'destino': '2018', 'tipo': 'follows', 'confianza': 0.8311179}
{'origen': 'SpaceX', 'destino': '2020', 'tipo': 'follows', 'confianza': 0.7499766}
{'origen': 'Model X.', 'destino': 'Model S', 'tipo': 'follows', 'confianza': 0.9885029}
{'origen': 'Estación Espacial Internacional', 'destino': '2020', 'tipo': 'follows', 'confianza': 0.7830144}
{'origen': 'NASA Commercial', 'destino': '2020', 'tipo': 'follows', 'confianza': 0.82530427}
{'origen': 'Model X.', 'destino': 'Model 3', 'tipo': 'follows', 'confianza': 0.96325797}
{'origen': 'SpaceX', 'destino': 'Hawthorne, California', 'tipo': 'founded by', 'confianza': 0.5444168}
{'origen': 'SpaceX', 'destino': 'Elon Musk', 'tipo': 'founded by', 'confianza': 0.9965586}
{'origen': 'Model X.', 'destino': 'Tesla', 'tipo': 'follows', 'confianza': 0.572393}
{'origen': 'SpaceX', 'destino': 'cohete Falcon Heavy', 'tipo': 'notable work', 'confianza': 0.5252624}
```

Figura 5.14: Relaciones encontradas en el segundo texto por el modelo obtenido en el presente trabajo.

Capítulo 6

Contribuciones

6.1. Mauro Díaz Lupone

Durante las primeras fases del proyecto, en particular el primer mes y medio desde la asignación de esta línea temática, se estuvo mayoritariamente estudiando y aprendiendo todo lo necesario para poder hacer posteriormente la investigación presentada en el trabajo. Para ello primeramente se cursaron clases online de [cou21], llamados programas especializados, dado que los conocimientos relacionados con la Inteligencia Artificial eran nulos: se cursó un programa de 100 horas sobre ML donde se estudiaba todo lo relacionado con aprendizaje supervisado y no supervisado, posteriormente un curso de 130 horas sobre *aprendizaje profundo* en los cuales ya se analizaban con profundidad las RNN y CNN junto a otras arquitecturas como los *Transformers* y por último un programa de NLP de 100 horas en el que en particular se profundizaba en los modelos de atención. Además, dado que en este campo el lenguaje de programación más recurrente es Python, se estudió y practicó en este lenguaje mediante cursos ofrecidos por los tutores.

Una vez obtenidos todos los conocimientos necesarios, se empezó con el trabajo de investigación. Para ello primeramente se estuvieron buscando trabajos relacionados con la RE y se analizaron. Se hizo una comparativa y resumen sobre cuales eran los métodos actuales con mejores resultados para esta área temática y se empezó a profundizar sobre ello. Decidido que se optaba por un modelo con BERT, se procedió a una búsqueda exhaustiva sobre los varios modelos derivados de BERT, en qué mejoraban y en qué empeoraban y sobre todo si había disponibilidad de encontrar una variante en español. En este proceso de investigación, también se analizaron otros modelos como GPT y sus respectivas aplicaciones a la RE presentadas en el Capítulo 3.

Terminada esta primera fase de investigación, se inició una nueva más enfocada al desarrollo en la cual se investigaron las herramientas actuales para generación de modelos de RE con BERT. Se decidió utilizar en un primer momento OpenNRE y se estudió exhaustivamente su funcionamiento. Posteriormente, se empezó la búsqueda de datasets en español para la extracción de relaciones. Fue particularmente difícil encontrar conjuntos de datos *open source* para esta tarea, así que fué una fase duradera.

Una vez teniendo ya los datasets, se procedió a un preprocesamiento de los datos en el cual se intentaban poner un formato en el cual OpenNRE los aceptaba, que era en líneas JSON particulares. Fue un trabajo largo en el que se tuvo que entender muy bien el formato presentado de los datos para una conversión particular a JSON. Una vez uniformizados,

se procedió a una limpieza de las líneas de texto que contenían errores, que en ciertos casos representaron hasta un 20 % del dataset. Esto se debe a que a la hora de intentar entrenar modelos con OpenNRE con los datos uniformizados salían siempre errores, y hasta que no se limpiaron bien las líneas JSON no se logró entrenar un modelo. Teniendo ya los datos, se crearon *scripts* de Python para el entrenamiento en OpenNRE, herramienta que dió muchísimos problemas de compatibilidades junto a otros defectos que hizo que después de más un mes de trabajo desarrollando en esta herramienta se tomase la decisión de cambiar a SpaCy3. Se hizo entonces un trabajo de investigación y desarrollo en esta herramienta hasta que se logró dejar un primer *script* funcional para el entrenamiento en cpu de un modelo para que pudiese continuar el compañero de proyecto con ello. Se resolvieron los problemas de compatibilidades que podía traer SpaCy3 a través de la creación de un entorno virtual y de una extensa preparación de dependencias adecuadas para así poder, dado unos datos correctos (que fue trabajo del compañero), que el código entrenase un modelo con la cpu.

Mientras que el compañero de proyecto Yasser seguía trabajando en el desarrollo, se procedió a crear la memoria del trabajo. Se hizo todo el Capítulo 2 y todo el Capítulo 3 y se crearon todas las imágenes presentadas en todo Capítulo. Todo lo escrito está sacado de la investigación descrita anteriormente. Una vez obtenidos ya unos primeros resultados de entrenamiento, se siguió con la memoria de los siguientes Capítulos. Para el Capítulo 4 se contribuyó con toda la descripción de la herramienta SpaCy3 y del pipeline (con todas sus secciones derivadas) del proyecto. Se generó además el código de Python que junta un modelo NER con el modelo RE generado con el entrenamiento para la extracción de información. Una vez con el código y el modelo funcional se procedió a la escritura del Capítulo 5, en particular se contribuyó con la descripción del entrenamiento y de la evaluación del modelo, se crearon los gráficos de las métricas obtenidas por el compañero de proyecto y la tabla de los hiperparámetros usados. En las fases finales se ayudó en la corrección de la memoria, en particular de todas las imágenes, limpieza de Capítulos y mejora de la Introducción.

6.2. Yasser Takfa Ghazal

Durante la primera fase del proyecto, debido a la incorporación de manera tardía a la dinámica de este, se trató de familiarizarse con los principios básicos que se iban a tratar más adelante. Para ello, se hizo uso de algunos cursos en línea disponibles en [cou21], los cuales sirvieron como una perfecta introducción a las herramientas que se utilizarían más adelante. Debido al poco conocimiento sobre el lenguaje de programación de Python, también se procedió a realizar cursos de iniciación en este lenguaje, enfocando esta fase a las herramientas que ofrecía el propio Python para el contexto del trabajo.

En segunda instancia, se pasó a investigar acerca de los conjuntos de datos hallados por el compañero Mauro Díaz, para facilitar más adelante la tarea de preprocesado de estos. En esta fase se indagó acerca de los objetivos de los autores de estos conjuntos de datos, la metodología utilizada para su creación y los resultados que estos obtenían con herramientas y Transformers investigados por el compañero Mauro y propuestos para su uso en las siguientes fases. Una vez extraída toda la información necesaria acerca de estos conjuntos de datos, se procedió a la realización de *scripts* que sirvieran para, principalmente, uniformizar los datos debido a su distinta procedencia, y seguidamente, conseguir la compatibilidad de estos con la herramienta OpenNRE, que fue la primera

herramienta considerada para realizar el entrenamiento del modelo. Este fue un proceso tedioso, y en la fase de pruebas, se concluyó que la herramienta OpenNRE no podría ser utilizada debido a ciertas incompatibilidades con los ordenadores de trabajo debido a dependencias utilizadas por la herramienta que no se podían descargar de manera correcta. A pesar de este contratiempo, se pudo extraer de esta fase la uniformización de los conjuntos de datos y la corrección de errores en el formato de estos, lo que facilitó mucho la compatibilización del conjunto de datos final con la herramienta que se decidió utilizar, SpaCy.

La fase de compatibilización con la herramienta SpaCy fue la más larga y tediosa debido a la poca documentación acerca del formato aceptado por SpaCy, sin embargo, tras hallar ciertos ejemplos de código, se consiguió crear un *script* inicial que realizaba la conversión de una parte de los datos a formato spacy de manera satisfactoria. Gracias a este avance, se pudo proceder a realizar los primeros entrenamientos con conjuntos de datos muy pequeños, lo que supuso un gran avance, ya que se confirmó la compatibilidad total de estos pequeños conjuntos con la herramienta SpaCy, y se pudo guiar el esfuerzo hacia la manera de proceder correcta para la obtención de la totalidad del conjunto de datos en formato *.spacy*.

Tras cierto tiempo intentando mejorar el *script* para la obtención de todo el conjunto de datos en el formato correcto, y el entrenamiento del modelo con los conjuntos de datos pequeños para realizar pruebas y obtener los primeros resultados, se consiguió extraer todo el conjunto de datos; sin embargo, al tratar de realizar el entrenamiento final con todo el conjunto de datos se localizó otro problema: la falta de potencia de los ordenadores de trabajo para asumir todo el esfuerzo de cómputo necesario para realizar el entrenamiento. Se procedió a comunicar al tutor acerca de esta dificultad, y mientras éste hallaba la forma de conseguir acceso a un ordenador con mayor potencia, se centraron los esfuerzos en la finalización de la memoria. Debido a que Mauro ya había comenzado con las partes más teóricas de la redacción de la memoria, se brindó apoyo completando el capítulo 1, capítulo 4, y el comienzo de la redacción del capítulo 5, poniendo especial hincapié en la explicación de la parte más práctica del proyecto.

Más adelante, cuando el tutor brindó acceso a un ordenador de trabajo capaz de soportar el entrenamiento del modelo final, se comenzó a estudiar las herramientas que permitían el uso de una tarjeta gráfica para la ejecución de código. Tras enfrentar ciertos problemas e incompatibilidades con la herramienta elegida(Cuda) se consiguió realizar el entrenamiento con la tarjeta gráfica integrada en el ordenador de trabajo, lo que resolvió los problemas de escasa potencia hallados en anteriores fases. Finalmente, se consiguieron los primeros resultados con el conjunto de datos definitivo, y debido a la disponibilidad del ordenador para su uso, se procedió a realizar pruebas modificando ciertos hiperparámetros de la fase de entrenamiento, así como probando con distintos Transformers para conocer si existían diferencias significativas en el rendimiento de estos. Tras elegir un modelo final, se procedió a realizar pruebas con distintos textos complejos para evaluar su rendimiento sobre escenarios de uso reales, además de completar la memoria, integrando los resultados finales en el capítulo 5 y completando el capítulo 6.

Finalmente, se comenzó la fase de corrección de la memoria a partir de los comentarios del tutor para así obtener la memoria definitiva lo más completa posible.

Capítulo 7

Conclusiones y Trabajo Futuro

7.1. Conclusiones

La extracción de relaciones es una de las tareas más importantes en el complejo entramado del NLP, por tanto, ha sufrido grandes cambios en los últimos años, sobre todo a partir del desarrollo de la tecnología *Transformer*. Este desarrollo de la metodología para esta tarea del NLP, junto al profundo estudio de esta última tecnología es el foco principal de este trabajo, en el cual se abordan distintas variantes del *Transformer* para tratar de sacar conclusiones sobre el contexto de uso de cada una y sus fortalezas y debilidades. Algunas de las mencionadas son BERT, BETO, M-BERT o RoBERTa. Además de ello se hace un extenso estudio de los modelos y datos actualmente disponibles para el idioma español en esta área de investigación, concluyendo que hay una gran ausencia y mayor necesidad de desarrollo en el ámbito.

En el presente trabajo se utiliza la variante RoBERTa para desarrollar un modelo de extracción de relaciones entre pares de entidades en un ámbito general. Para ello, se destaca la importancia de la búsqueda de un conjunto de datos de calidad, con anotaciones precisas, y lo suficientemente amplio para obtener buenos resultados, además de una investigación exhaustiva de las diferentes herramientas de entrenamiento de modelos de NLP para dar con la más adecuada para el contexto.

Como conclusiones finales del trabajo expuesto, se puede extraer que las herramientas para el entrenamiento de modelos de NLP han avanzado mucho en los últimos años, ofreciendo en sus últimas versiones un amplio repertorio de funcionalidades para personalizar el entrenamiento del modelo y extraer de todo el proceso un resultado basado en las necesidades del individuo; además, se puede concluir que la tarea más tediosa de todo el proceso es la búsqueda de un conjunto de datos en idioma español etiquetado y preparado para su entrenamiento sin necesidad de una fase de preprocesado muy compleja, a parte de la escasa presencia de conjuntos sobre ámbitos de investigación específicos, lo que complicaría la tarea de generar un modelo que funcione sobre textos de un ámbito concreto.

7.2. Trabajo Futuro

Como posibles trabajos futuros pueden señalarse los siguientes:

- La creación de un conjunto de datos en idioma español más extenso y con anotaciones

más precisas. Se podría explorar la posibilidad tanto de crear un conjunto de datos de manera automatizada como la generación de un conjunto de datos mediante supervisión humana, ya que se ha podido comprobar la inexactitud de las anotaciones en los casos en los que el conjunto se ha generado de manera automatizada.

- Relacionado con el anterior punto, se podría explorar la implementación de una herramienta de creación de conjuntos de datos en ámbitos específicos, o, debido a la complejidad que podría conllevar, un conjunto de datos que contenga elementos de distintos ámbitos especializados, para así acceder a estos de manera sencilla y entrenar el modelo dependiendo de la aplicación que se le daría.

- Profundizar en el análisis del proceso de entrenamiento para comprender las facilidades o dificultades a las que se enfrenta el modelo. De esta manera se podría comprender el por qué de las deficiencias del modelo ante la predicción de una relación entre entidades concretas, como la observada en el presente trabajo entre entidades que representan personas, o la importancia del contexto que rodea a estas entidades, para así poder desarrollar modelos más completos y más eficientes en contextos difíciles de interpretar.

Capítulo 8

Introduction

8.1. Motivation

In recent years, there has been a growing interest in Artificial Intelligence and its applications in the realm of language interpretation and reproduction, driven by the vast amount of data accessible to anyone via the Internet. Natural Language Processing (NLP) technologies have seen significant development in the last few decades. Since the first modern architecture, Word2Vec [MCCD13], there have been several advances such as Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN) [RNN21], leading up to today's most widely used technology, known as the *Transformer*, introduced in 2017 by Vaswani et al. [VSP+23]. This new technology revolutionized the world of NLP and is currently at the forefront and the dominant paradigm.

An important task in NLP is Information Extraction (IE), which involves identifying and classifying information of interest from a text. This can be divided into two subtasks: Named Entity Recognition (NER) and Relation Extraction (RE). Currently, IE is a very important research area due to its numerous applications [WWRM+18]. Since [VSP+23], models such as BERT [DCLT19] and GPT [RNSS18], along with their respective variants and improvements, have been developed, demonstrating experimentally that they offer the best results for NLP tasks.

Unlike with the NER subtask, there is a problem in that, for RE, a model using these new technologies has not yet been developed for Spanish texts. Given the successful results of BERT-based models and their variants for English, there is interest in developing a model based on the same technology for Spanish, similar to what was done in [Rod22a] for Portuguese or in [PF23] for Spanish clinical texts.

8.2. Context

This Final Degree Project is part of a research project called Novel Strategies to Fight Child Sexual Exploitation and Human Trafficking Crimes and Protect their Victims - HEROES, approved by the European Commission within the Horizon 2020 Framework Programme (call H2020-SU-SEC-2020) under grant agreement number 101021801 and in which the GASS Group of the Universidad Complutense de Madrid (Grupo de Análisis, Seguridad y Sistemas, `https://gass.ucm.es`, group 910623 of the catalogue of research

groups recognised by the UCM).

In addition to the Universidad complutense de Madrid, 21 organisations from 17 countries are participating in HEROES: 11 from EU countries (Austria, Belgium, Bulgaria, France, Greece, Ireland, Latvia, Lithuania, Portugal, Spain, United Kingdom), 1 associated country (Switzerland) and 5 third countries (Bangladesh, Brazil, Colombia, Peru, Uruguay). These entities are: University of Kent (UK), The Free University of Brussels (Belgium), The French National Research Institute for Digital Science and Technology - INRIA (France), Center for Security Studies - KEMEA (Greece), International Centre for Migration Policy Development - ICMPD (Austria), International Center for Missing and Exploited Children - ICMEC (Switzerland), IDENER Research & Development Agrupación de Interés Económico (Spain), Athena Research Center - ARC (Greece), Trilateral Research and Consulting (United Kingdom), Centre for Women and Children Studies - CWCS (Bangladesh), Center Against Human Trafficking and Exploitation - KOPZI (Lithuania), Portuguese Association for Victim Support - APAV (Portugal), Fundación Renacer (Colombia), The Greek Council for Refugees - GCR (Greece), Brazilian Association for the Defense of Children of Children and Youth - ASBRAD (Brazil), Hellenic Police (Greece), Latvia National Police (Latvia), General Directorate for the Fight against Organized Crime (Bulgaria), Dirección General de la Policía - DGP (Spain), Federal Police (Brazil), Federal Highway Police (Brazil), Secretaria de Inteligencia Estratégica de Estado - Presidencia de la Republica Oriental del Uruguay (Uruguay).

More information is available at:

https://cordis.europa.eu/project/id/101021801
https://heroes-fct.eu

8.3. Object of the Investigation

The goal of the project is to generate an NLP model capable of extracting relationships between named entities in Spanish text for a general-purpose approach. This model is intended to automate processes such as the search for specific information, identification of connections between individuals, companies, or organizations, among others.

Moreover, this project aims to understand the internal architecture of the most revolutionary tools of recent years, artificial intelligences based on machine learning (ML), and more specifically, those focused on NLP. It will delve into their details, seeking to uncover the root of their behavior and effectiveness.

8.4. Workplan

The development of this work has been carried out in three phases:

1. **Research** In this phase, a process of familiarization was conducted with the terms that would later be used frequently, as well as understanding the state of the technologies to be used and searching for resources that could be utilized in future phases. To achieve this, weekly meetings were held between the students and the tutor where new concepts, crucial to the project's completion, were presented. Additionally, access to a series of introductory courses was provided, covering topics

like the basics of machine learning, deep learning, and neural network theory. After building a theoretical foundation in the research area, a search through *Google Scholar* was initiated to find other works, tools, and models used for IE. Finally, a search for datasets to be used in the following phases for model training began. These datasets needed to be pre-labeled, ready for preprocessing and use, and represented text in Spanish, which proved to be the most challenging part. Thus, a subphase within the research phase itself was initiated to discover all the existing alternatives in Spanish. The first challenge emerged when it was found that all the known and tested datasets were in English. Despite this, some results were obtained, allowing the project to move on to the development phase.

2. **Development**: In this phase, after acquiring the necessary knowledge, ideas began to be transformed into code. This involved testing various libraries and tools studied in the previous phase, adapting the training datasets to the real case, and resolving resulting incompatibilities and errors due to the adjustment of the tools. Weekly meetings were held to monitor progress and avoid falling into dead ends and stagnation.

3. **Results**: This final phase focused mainly on improving the model being developed. To achieve this, the metrics offered by the tool used to train the model were analyzed and optimized through changes in hyperparameters, data, and *Transformer* architectures. Thus, this phase continued both research and development: an exhaustive search for the best Spanish model applicable to our project, study of benchmark works in our research area, data analysis and modification for better results, and training the model with hyperparameter adjustments. To optimize this phase, the training was also tested on more powerful computers that offered faster processing speeds.

8.5. Struture of the Work

The rest of the work is organized in 9 chapters with the structure explained below:

Chapter 1 is an introduction of this work, which is the translation in spanish of this chapter.

Chapter 2 introduces some important concepts to facilitate the understanding of the model built in this project. It presents the fundamental terms and areas for understanding NLP, particularly RE and NER. Additionally, the chapter explores the most advanced technologies in NLP today, covering both their functioning and applications.

Chapter 3 presents a study on the different techniques used for RE, including both traditional methods and those that incorporate various deep learning concepts. It showcases works on RE that achieve state-of-the-art results through the use of pre-trained language models. The chapter also includes an examination of the best pre-trained language models for the Spanish language, highlighting their advantages and disadvantages.

Chapter 4 contains the contributions of this work. It features the study conducted on the various datasets found for RE in Spanish, as well as the corresponding standardization and preprocessing that was done. Additionally, it provides an analysis of the tools used to generate the model and the pipeline that was followed for its construction.

Chapter 5 describes the experiments conducted on both the data presented in Chapter 4 and the models studied in Chapter 3. The obtained results are presented.

Chapter 6 outlines the conclusions drawn from the experimentation and the study of the results in Chapter 5. It also introduces potential future lines of research.

Chapters 7 and 8 contain the English translations of the Introduction and the Conclusions, respectively.

Capítulo 9

Conclusions and Future Work

9.1. Conclusions

Relation extraction is one of the most important tasks in the complex framework of NLP, and as such, it has undergone significant changes in recent years, especially following the development of *Transformer* technology. This development of the methodology for this NLP task, along with a thorough study of this latest technology, is the main focus of this work, in which various *Transformer* variants are addressed to draw conclusions about the context in which each is used, as well as their strengths and weaknesses. Some of the mentioned ones are BERT, BETO, M-BERT, or RoBERTa. All of these variants provide a base model, with the possibility of retraining it in Spanish using the most appropriate dataset for the project's context. Additionally, an extensive study of the models and data currently available in Spanish for this research area is conducted, concluding that there is a significant gap and a greater need for development in the field.

In this work, the RoBERTa variant is used to develop a model for relation extraction between pairs of entities in a general context. To achieve this, the importance of finding a quality dataset, with accurate annotations, and sufficiently large to obtain good results, is emphasized, along with a thorough investigation of the different training tools for NLP models to find the most suitable one for the context.

As final conclusions of the presented work , it can be inferred that the tools which are available for the training of models of NLP have advanced significantly in recent years, offering in their las versions a very complete set of functionalities to customize the training of the model, and be able to have a result that matches the needs of the user. Additionally, it can be concluded that the most demanding task of the project has been the search for a labeled Spanish language dataset prepared for training without having to get into a very complex phase of preprocessing, apart from the absence of specific fields datasets, which would surely complicate the task of generating a model that works on texts from a particular domain.

9.2. Future Work

The following can be highlighted as potential future work:

- Creating a more extensive Spanish-language dataset with more precise annotations.

One could explore the possibility of creating a dataset both in an automated manner and through human supervision, as the inaccuracy of annotations has been observed in cases where the set was generated automatically.

- Related to the previous point, one could explore the implementation of a dataset creation tool in specific domains, or, due to the complexity that might entail, a dataset containing elements from various specialized domains, allowing easy access and the ability to train the model depending on the intended application.
- Deepen the analysis of the training process to understand the challenges or advantages that the model faces. This could help explain the model's deficiencies when predicting a relationship between specific entities, like the one observed in this work involving entities that represent people, or the importance of the context surrounding these entities, to be able to develop more comprehensive and efficient models for difficult-to-interpret contexts.

Bibliografía

[BBM22] Abhyuday Bhartiya, Kartikeya Badola, and Mausam. DiS-ReX: A multilingual dataset for distantly supervised relation extraction. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 849–863, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[BGP+23] Elisa Bassignana, Filip Ginter, Sampo Pyysalo, Rob van der Goot, and Barbara Plank. Multi-CrossRE a multi-lingual multi-domain dataset for relation extraction. In Tanel Alumäe and Mark Fishel, editors, *Proceedings of the 24th Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 80–85, Tórshavn, Faroe Islands, May 2023. University of Tartu Library.

[BKH16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[BLC19] Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text, 2019.

[BP22] Elisa Bassignana and Barbara Plank. CrossRE: A cross-domain dataset for relation extraction. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3592–3604, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[CCF+23] José Canyete, Gabriel Chaperon, Rodrigo Fuentes, Jou-Hui Ho, Hojin Kang, and Jorge Pérez. Spanish pre-trained bert model and evaluation data, 2023.

[CDBM+23] José Canyete, Sebastián Donoso, Felipe Bravo-Marquez, Andrés Carvallo, and Vladimir Araujo. Albeto and distilbeto: Lightweight spanish language models, 2023.

[CLP+22] Casimiro Pio Carrino, Joan Llop, Marc Pàmies, Asier Gutiérrez-Fandiño, Jordi Armengol-Estapé, Joaquín Silveira-Ocampo, Alfonso Valencia, Aitor Gonzalez-Agirre, and Marta Villegas. Pretrained biomedical language models for clinical NLP in Spanish. In *Proceedings of the 21st Workshop on Biomedical Language Processing*. Association for Computational Linguistics, 2022.

[cou21] Natural Language Processing Specialization. https://www.coursera.org/specializations/natural-language-processing, 2021.

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[GAP+21] Asier Gutiérrez-Fandiño, Jordi Armengol-Estapé, Marc Pàmies, Joan Llop-Palao, Joaquín Silveira-Ocampo, Casimiro Pio Carrino, Aitor Gonzalez-Agirre, Carme Armentano-Oller, Carlos Rodríguez Penagos, and Marta Villegas. Spanish language models. *CoRR*, 2021.

[HCTNN23] Pere-Lluís Huguet Cabot, Simone Tedeschi, Axel-Cyrille Ngomo, and Roberto Navigli. Redfm: a filtered and multilingual relation extraction dataset. pages 4326–4343, 01 2023.

[Hea92] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *COLING 1992 Volume 2: The 14th International Conference on Computational Linguistics*, 1992.

[HGY+] Xu Han, Tianyu Gao, Yuan Yao, Deming Ye, Zhiyuan Liu, and Maosong Sun. OpenNRE: An open and extensible toolkit for neural relation extraction. In Sebastian Padó and Ruihong Huang, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[Int04] *WWW '04: Proceedings of the 13th international conference on World Wide Web*. Association for Computing Machinery, 2004.

[Kor21] M. V. Koroteev. Bert: A review of applications in natural language processing and understanding, 2021.

[LFF+19] J. Licheng, Z. Fan, L. Fang, Y. Shuyuan, L. Lingling, F. Zhixi, and Q. Rong. Neural information extraction pipeline for cyber forensics with pre-trained language models. *IEEE Access*, 7:1–22, October 2019.

[LOG+19] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[LSCC13] ChunYang Liu, WenBo Sun, WenHan Chao, and WanXiang Che. Convolution neural network for relation extraction. 2013.

[LSN+23] Yiming Liu, Hongtao Shan, Feng Nie, Gaoyu Zhang, and George Xianzhi Yuan. Document-level relation extraction with local relation and global inference. *Information*, 14, 2023.

[MBSJ09] Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. Distant supervision for relation extraction without labeled data. 2009.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[NG15] Thien Huu Nguyen and Ralph Grishman. Relation extraction: Perspective from convolutional neural networks. 2015.

[Ngo21] Chuong Ngo. Transformers: The Nuts and Bolts. https://www.revistek.com/posts/transformer-architecture, April 2021.

[OC23] Alvaro López Olmos and ZhenBo Chen. Reconocimiento de entidades nombradas en textos en español utilizando roberta, 2023.

[Pah17] Ramit Pahwa. Micro-Macro Precision,Recall and F-Score. https://medium.com/@ramit.singh.pahwa/micro-macro-precision-recall-and-f-score-44439de1a044, September 2017.

[PF23] Joel Pardo Ferrera. Entity and relation extraction from electronic health records using spacy. 2023.

[PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[PMS23] Narendra Patwardhan, Stefano Marrone, and Carlo Sansone. Transformers in the real world: A survey on nlp applications. *Information*, 14, 2023.

[PP20] Yannis Papanikolaou and Andrea Pierleoni. Dare: Data augmented relation extraction with gpt-2, 2020.

[PPB17] Sachin Pawar, Girish K. Palshikar, and Pushpak Bhattacharyya. Relation extraction : A survey, 2017.

[PSG19] Telmo Pires, Eva Schlinger, and Dan Garrette. How multilingual is multilingual bert?, 2019.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. Doha, Qatar, 2014. Association for Computational Linguistics.

[RNN21] A simple overview of RNN, LSTM and Attention Mechanism. https://medium.com/swlh/a-simple-overview-of-rnn-lstm-and-attention-mechanism, January 2021.

[RNN23] RNN vs. LSTM vs. Transformers: Unraveling the Secrets of Sequential Data Processing. https://medium.com/@mroko001/rnn-vs-lstm-vs-transformers-unraveling-the-secrets-of-sequential-data, September 2023.

[RNSS18] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf*, 2018.

[Rod22a] Fillipe Barros Rodrigues. Neural information extraction pipeline for cyber forensics with pre-trained language models, 2022.

[ROD22b] Fillipe Barros RODRIGUES. Neural information extraction pipeline for cyber forensics with pre-trained language models. MSc Thesis, Universidade de Brasília, Brasília, August 2022.

[SFS+21] Alessandro Seganti, Klaudia Firlag, Helena Skowronska, Michal Satawa, and Piotr Andruszkiewicz. Multilingual entity and relation extraction dataset and model. 2021.

[SPMGB+23] Oswaldo Solarte-Pabón, Orlando Montenegro, Alvaro García-Barragán, Maria Torrente, Mariano Provencio, Ernestina Menasalvas, and Víctor Robles. Transformers for extracting breast cancer information from spanish clinical narratives. *Artificial Intelligence in Medicine*, 2023.

[SQ19] Vighnesh Shiv and Chris Quirk. Novel positional encodings to enable tree-based transformers. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[SVL23] Victoria Slocum Sofie Van Landeghem. Implementing a custom trainable component for relation extraction. https://explosion.ai/blog/relation-extraction, April 2023.

[VSP+23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[WQZ+21] Hailin Wang, Ke Qin, Rufai Yusuf Zakari, Guoming Lu, and Jin Yin. Deep neural network based relation extraction: An overview, 2021.

[WWRM+18] Yanshan Wang, Liwei Wang, Majid Rastegar-Mojarad, Sungrim Moon, Feichen Shen, Naveed Afzal, Sijia Liu, Yuqun Zeng, Saeed Mehrabi, Sunghwan Sohn, and Hongfang Liu. Clinical information extraction applications: A literature review. *Journal of Biomedical Informatics*, pages 34–49, 2018.

[XFHZ15] Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. Semantic relation classification via convolutional neural networks with simple negative sampling. 2015.

[ZLL+14] Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. Relation classification via convolutional deep neural network. 2014.

[ZZHY15] Shu Zhang, Dequan Zheng, Xinchen Hu, and Ming Yang. Bidirectional long short-term memory networks for relation classification. 2015.