Trabajo Fin de Máster: Reconocimiento de entidades y extracción de relaciones en texto biomédico

Trabajo Fin de Máster

Aitana Villaplana Moreno

Trabajo de investigación para el

Máster en Tecnologías del Lenguaje

Universidad Nacional de Educación a Distancia

Dirigido por

Prof. Dra. Raquel Martínez Unanue

Prof. Dra. Soto Montalvo Herranz

Febrero 2023

Agradecimientos

A mis padres que siempre me han apoyado e inspirado en todo. A Tomás por su apoyo incondicional y su ayuda. Y a mis tutoras por su paciencia y ayuda para llevar a cabo este trabajo.

Resumen

El análisis de documentos médicos supone un gran reto a día de hoy. Existe mucha información desestructurada, que difícilmente puede ser analizada. Es por esto, que una de las tareas de Procesamiento del Lenguaje Natural es poder ser capaces de extraer las Entidades Nombradas en los textos, así como también ciertas relaciones que existen entre éstas, lo que facilita en gran medida el posterior análisis de los datos biomédicos. Para realizar estas tareas existen modelos avanzados, basados en modelos de lenguaje entrenados con grandes cantidades de datos. Lo que se propone en este trabajo, es utilizar modelos *transformers* junto con modelos de aprendizaje automático como SVM o Redes Neuronales para realizar la tarea de extracción de entidades, ya sea utilizando modelos previamente entrenados en grandes cantidades de texto biomédico en español, o bien realizando el entrenamiento del modelo a partir de estos modelos previamente entrenados. Esta última aproximación ha conseguido superar los resultados de otros sistemas del estado del arte para la tarea del reconocimiento de entidades. Respecto a la tarea de extracción de relaciones, se han utilizado también estos modelos *transformers* entrenados con texto biomédico en español junto con modelos de Redes Neuronales, además de utilizar técnicas de aumento de datos como SMOTE-NC y reducción de dimensionalidad como LDA, lo que ha dado como resultado sistemas comparables a los del estado del arte.

Abstract

The analysis of medical documents is a considerable challenge nowadays. There is a lot of unstructured information. For this reason one of the tasks of Natural Language Processing is to be able to extract the Named Entities in the texts, as well as certain relationships that exist between them, which greatly facilitates the subsequent analysis of biomedical data. To perform these tasks, advanced models are available, based on language models trained on large amounts of data. In this work, we propose to use *transformers* models together with machine learning models such as SVM or Neural Networks to perform the task of entity extraction, either by using models previously trained on large amounts of biomedical text in Spanish, or by training the model from these previously trained models, which has managed to overcome the results of other state-of-the-art systems for the task of entity recognition. Regarding the relation extraction task, we have also used these *F1Score* models trained with biomedical text in Spanish together with Neural Network models, in addition to using data augmentation techniques such as SMOTE-NC and dimensionality reduction such as LDA, which has resulted in comparable state-of-the-art systems.

Índice general

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

1.1. Motivación

Actualmente nos encontramos en una época en la cual la medicina está continuamente desarrollando nuevas técnicas de diagnóstico y tratamiento gracias a la tecnología, lo que permite mejorar la salud general de la población. Estas técnicas incluyen el uso de tecnología como escáneres o herramientas quirúrgicas, pero también software para ayudar y facilitar el trabajo a los profesionales de la salud. Debido a la gran cantidad de datos que se han digitalizado, como por ejemplo, informes, diagnósticos, historiales, etc., también se han utilizado múltiples técnicas de inteligencia artificial para gestionar y estructurar esta información, así como también aprovechar ese conocimiento para la automatización de tareas.

Centrándonos en concreto en el área del Procesamiento del Lenguaje Natural (PLN), existen muchos retos que llevar a cabo mediante el análisis de documentos médicos, tales como la clasificación de documentos, reconocimiento de entidades médicas nombradas en el texto, o la extracción de relaciones entre esas Entidades Nombradas (ENs). Un ejemplo de textos etiquetados utilizados para el reconocimiento de entidades es el corpus NCBI Disease Corpus Doğan et al. (2014), que contiene las siguientes entidades: Mención compuesta, como por ejemplo: cáncer de mama o de ovario (*breast or ovarian cancer*); Modificador, como por ejemplo: tumor (*tumour*); Enfermedad específica, como por ejemplo: hemocromatosis (*hemochromatosis*), y por último, Tipo de enfermedad, como por ejemplo tumores (*tumours*).

Otros corpus muy utilizados son BioCreative II Gene Mention para nombres de genes Smith y Tanabe (2008) y el corpus BioCreative V Chemicals

Disease Relationship para nombres de enfermedades y sustancias químicas Li et al. (2016). Cabe destacar que estos corpus son en inglés, de ahí la importancia de fomentar el uso de corpus en castellano para este tipo de tareas.

La recopilación relaciones entre estas sustancias, enfermedades o entidades de forma general también tiene una importancia significativa para la investigación biomédica, pero se ve afectada por su elevado coste en tiempo y el rápido crecimiento de la literatura biomédica. En los últimos años, ha crecido el interés por desarrollar enfoques computacionales para la extracción automática de relaciones químico-enfermedad, como por ejemplo, la tarea BioCreative V Chemical Disease Relation (CDR)[1], cuyo corpus está formado por 1500 artículos de PubMed Wei et al. (2016). Dicho corpus, relaciona enfermedades con químicos, por ejemplo, en el texto: "*Lithium carbonate may be a factor [...] It also causes neurologic depression, cyanosis [...]*", la enfermedad *cyanosis* mantiene relación con el químico *Lithium carbonate*.

Todos estos textos son en lengua inglesa, pero existen también algunos corpus en castellano, como los utilizados en este trabajo. Por ejemplo, para la frase "Esta afecta principalmente a las personas mayores de 60 años", encontramos la relación entre la acción "afecta" y el predicado "mayores", entre otras.

Reconocer estas entidades y sus relaciones puede facilitar en gran medida el análisis de los informes médicos, notas, y demás material con el que trabajan los profesionales de la salud, pudiendo incluso descubrir factores o relaciones desconocidas hasta la fecha. Así pues, la motivación de este trabajo es resolver un problema de reconocimiento de entidades médicas y sus relaciones entre sí, para textos médicos en castellano, ayudando así a la investigación y mejora de los sistemas de procesamiento de texto biomédico en español.

1.2. Objetivos

En este trabajo, se propone utilizar varios modelos de *Machine Learning* y *transformers* para realizar las tareas de reconocimiento de entidades y extracción de relaciones en textos biomédicos en español.

A lo largo de la memoria se explicarán cada uno de los modelos propues-

[1] https://biocreative.bioinformatics.udel.edu/tasks/biocreative-v/track-3-cdr/

tos y sus resultados, así como también, las técnicas utilizadas. Por lo tanto, los objetivos que se busca conseguir con este trabajo son los siguientes:

- Aplicación de algoritmos de *Machine Learning* a partir de las características extraídas de un modelo *transformer* a las tareas de reconocimiento de entidades y extracción de relaciones.
- Realizar un estudio comparativo entre distintos modelos de *transformers* entrenados previamente en textos biomédicos en español, y su eficacia para ambas tareas.
- Ajustar un modelo completo de *transformers* entrenado previamente con grandes cantidades de texto, para realizar la tarea de reconocimiento de entidades.
- Resolver los problemas encontrados durante el análisis del estado del arte, tales como el desbalanceo entre las distintas entidades y relaciones a clasificar. Así como también se propone el uso de modelos entrenados con texto biomédico en español.
- Evaluar los modelos realizados y su desempeño, y compararlos con los modelos utilizados por otros sistemas para estas mismas tareas.
- Realizar un buen análisis de errores, y encontrar puntos de mejora de cara a futuros trabajos.

1.3. Estructura del documento

Este trabajo se estructura en varios capítulos dedicados a tratar distintos aspectos del trabajo, dicha estructura se explica a continuación.

En el capítulo 2, se hace una revisión de los trabajos realizados para las tareas de reconocimiento de entidades y extracción de relaciones, así como también se especifican y explican los modelos y técnicas utilizadas en el desarrollo de este trabajo, y su posición en el estado del arte actual. También se realiza un análisis crítico, detectando posibles problemas a mejorar.

En el capítulo 3, se explica detalladamente la problemática a resolver, y se analiza el conjunto de datos con los que se va a trabajar.

En el capítulo 4, se explica la propuesta que se plantea en este trabajo y se profundiza en los modelos y técnicas utilizadas en la propuesta.

En el capítulo 5, se indica la metodología y las métricas de evaluación del trabajo, y se presentan los resultados obtenidos, frente a los resultados de otros investigadores que resolvieron la misma tarea.

En el capítulo 6, se discute y se analiza sobre los resultados obtenidos en el capítulo 5, se realiza un análisis de la problemática encontrada, y se analizan los modelos que mejores resultados han proporcionado.

Por último, en el capítulo 7, se concluyen los resultados del trabajo, además de plantear líneas de trabajo futuro.

Capítulo 2

Estado del arte

Este trabajo involucra varios conceptos sobre inteligencia artificial y diversas técnicas que se explicarán en este capítulo, introduciendo el Procesamiento del Lenguaje Natural, así como también, se describirán las diversas técnicas que se utilizan en cada etapa del desarrollo de un modelo, fundamentalmente las utilizadas en este trabajo.

También se dará una visión sobre las técnicas y la evolución de estos modelos aplicados en específico al procesamiento de textos biomédicos, tanto en inglés como en castellano.

2.1. Aprendizaje automático

El aprendizaje automático (*Machine Learning* en inglés) es un subcampo de las ciencias de la computación y una rama de la inteligencia artificial, cuyo objetivo es desarrollar técnicas que permitan que las computadoras aprendan[1].

La estadística y el *Machine Learning* implican el desarrollo o uso de algoritmos que permitan a un programa inferir patrones sobre datos. Para el caso del aprendizaje supervisado, los algoritmos realizan estas inferencias generalmente sobre datos de entrenamiento, que a su vez le permiten "generalizar" y hacer predicciones sobre nuevos datos. Dichos algoritmos se conocen como "modelos" en *Machine Learning*. Durante la fase de aprendizaje, los parámetros numéricos que caracterizan el modelo de un determinado algoritmo se calculan optimizando una medida numérica, normalmente

[1] https://es.wikipedia.org/wiki/Aprendizaje_autom%C3%A1tico

a través de un proceso iterativo Nadkarni et al. (2011).

Por normal general, el aprendizaje se puede dividir en cuatro ramas, aprendizaje supervisado, no supervisado, semi-supervisado y por refuerzo. Siendo las dos principales ramas más utilizadas la supervisada y la no supervisada. En este trabajo se utilizan modelos de aprendizaje supervisado, por lo que se profundizará más en este tipo de aprendizaje.

2.1.1. Aprendizaje supervisado

En el aprendizaje supervisado, cada elemento de los datos de entrenamiento está etiquetado con la respuesta correcta, por lo que el modelo tiene que basarse en la información obtenida en el entrenamiento para ser capaz de predecir la etiqueta de salida correctamente. En este tipo de modelos se corre el riesgo de realizar un sobreajuste en el entrenamiento (*over-fitting*). El modelo puede ajustarse a los datos de ejemplo casi a la perfección, pero haciendo malas predicciones para los nuevos casos que no hayan sido vistos anteriormente. Esto se debe a que puede aprender el ruido aleatorio de los datos de entrenamiento en lugar de sólo sus características esenciales y deseadas Nadkarni et al. (2011).

Para minimizar ese riesgo de sobreajuste se utilizan técnicas como la validación cruzada (*cross-validation*), que divide los datos de ejemplo de forma aleatoria en conjuntos de entrenamiento y de prueba para validar internamente las predicciones del modelo. Este proceso de partición de datos, entrenamiento y validación se repite a lo largo de varias rondas, y los resultados de la validación se promedian entre todas estas Refaeilzadeh et al. (2009).

En esta sección, se introducirán los modelos utilizados durante el desarrollo del trabajo, mientras que en las secciones 4.2, 4.3 y 4.4 se hablará más en detalle de su uso en el mismo.

2.1.1.1. Máquinas de Vectores de Soporte

Uno de los modelos de *Machine Learning* utilizados en este trabajo, son las Máquinas de Vectores de soporte o *Support Vector Machine* en inglés (SVM) Hearst et al. (1998). Este algoritmo se utiliza en clasificación binaria, y se basa en realizar una separación lineal de los datos en el espacio, donde cada clase deberá ser representada a un lado de la frontera de decisión,

si estos son linealmente separables. En caso de que los datos no sean linealmente separables, se deberá aplicar una función *kernel* Patle y Chouhan (2013) que proyecte los datos a un espacio en el cual sean linealmente separables, generalmente a un espacio de mayores dimensiones. En la Figura 2.1 se puede observar como se realiza una proyección de un espacio no linealmente separable de dos dimensiones, a uno de tres dimensiones en el cual los datos se puedan separar[2].

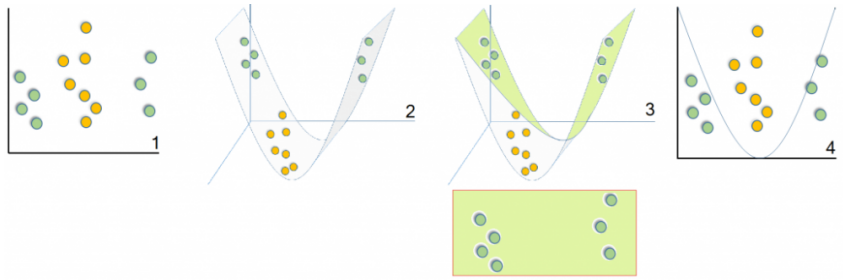


Figura 2.1: Proyección de un espacio con una función *kernel*.

El algoritmo se compone de dos vectores de soporte, paralelos a dicha frontera de decisión, y el objetivo del algoritmo es maximizar la distancia entre ambos vectores de soporte, lo que se conoce como margen. Es decir, se trata de maximizar la distancia entre clases. En la Figura 2.2, se pueden observar los vectores de soporte que separan ambas clases[3].

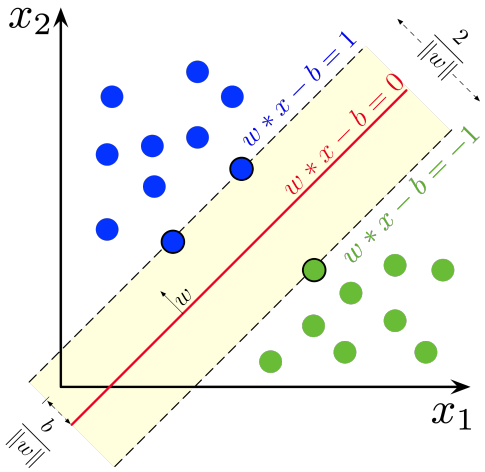


Figura 2.2: Clasificación binaria con SVM.

[2] https://numerentur.org/nucleo-kernel-de-las-svm/
[3] https://en.wikipedia.org/wiki/Support-vector_machine

Al tratarse de un algoritmo de clasificación binaria, si se quisiera aplicar a un problema con múltiples clases, es necesario construir K clasificadores para separar una clase de todas las demás siguiendo una estrategia *one-to-rest* Zhou et al. (2005). En la Figura 2.3 se muestra un ejemplo de aplicar dicha aproximación a un conjunto de datos de tres clases[4].

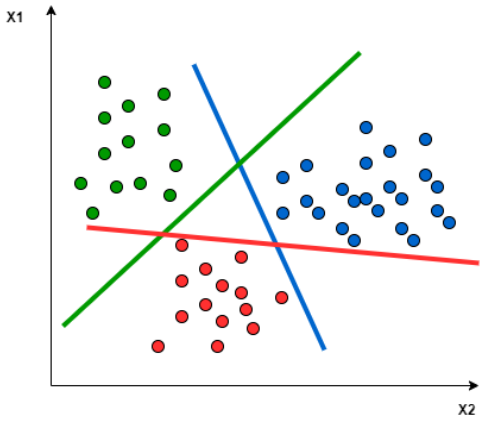


Figura 2.3: Clasificación multiclase *one-to-rest* con SVM.

Otro enfoque que se puede utilizar es el conocido como *one-to-one*. En este enfoque, se genera un hiperplano para separar entre cada dos clases, ignorando los puntos del resto de clases. Esto significa que la separación sólo tiene en cuenta los puntos de las dos clases que se estén dividiendo. En la Figura 2.4[4] se muestra la aplicación de esta técnica con el mismo conjunto de datos de tres clases que se ha mostrado en la Figura 2.3.

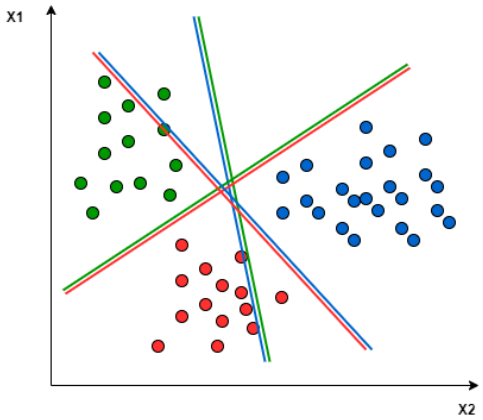


Figura 2.4: Clasificación multiclase *one-to-one* con SVM.

[4] https://www.baeldung.com/cs/svm-multiclass-classification

2.1.1.2. *Random Forest*

Dentro del *Machine Learning*, existen una serie de técnicas para tratar de mejorar el rendimiento de los modelos, una de esas técnicas se conoce como entrenamiento ensamblado. Dicha técnica consiste en entrenar varios modelos, y combinar las predicciones de éstos a la hora de dar la predicción final. Esta técnica aporta una serie de ventajas respecto al entrenamiento de un único modelo, como por ejemplo, reducir el sobre-ajuste, ya que al combinar los resultados de varios modelos, se reduce la probabilidad de error Sagi y Rokach (2018). En este trabajo se ha utilizado el modelo en ensamblado *Random Forest*.

Antes de comenzar a explicar este modelo, es necesario introducir el modelo de Árbol de Decisión. Los árboles de decisión son modelos con estructura de árbol, formados por un nodo raíz, nodos internos y nodos hoja. Este tipo de estructura permite discriminar los datos entre varios nodos basándose en las características de los datos de entrada Song y Ying (2015). El modelo selecciona las características que utilizar en cada nodo mediante el algoritmo CART, el cual se basa en la idea de ir dividiendo de manera consecutiva los datos de entrenamiento en dos grupos, hasta alcanzar una profundidad máxima, o lograr la máxima pureza de los nodos hoja Loh (2011).

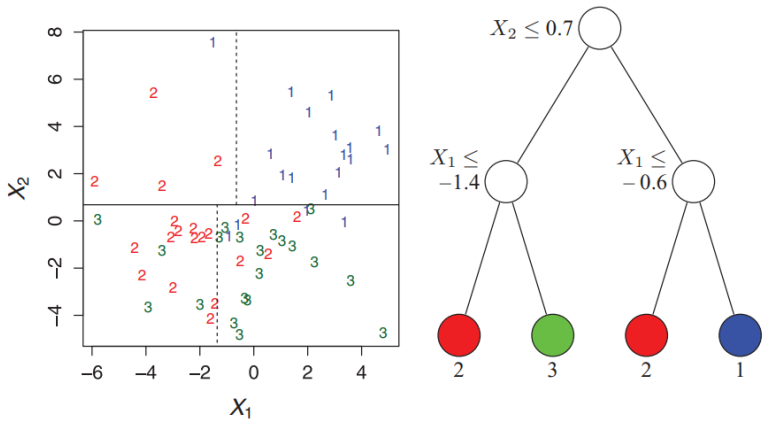
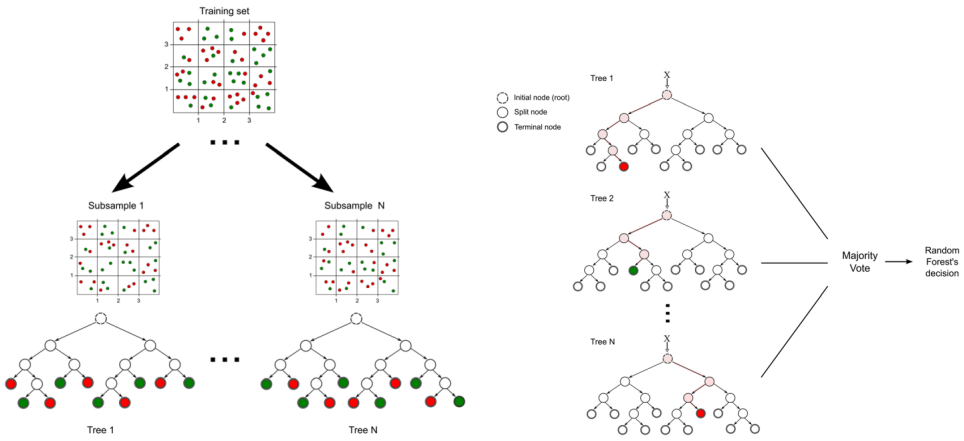


Figura 2.5: Árbol de decisión y sus predicciones.

En la Figura 2.5 se puede observar en la parte derecha un ejemplo de árbol de decisión, y en la parte izquierda las predicciones del mismo para un determinado conjunto de datos Loh (2011).

El modelo *Random Forest* Breiman (2001) está formado por una combinación de árboles de decisión, de forma que cada árbol depende de los valores de un vector aleatorio muestreado independientemente del resto de árboles, y con la misma distribución para todos los árboles del bosque. En la Figura 2.6a se puede ver como se distribuyen los datos de manera aleatoria entre los distintos árboles que componen el bosque Machado et al. (2015).



(a) Distribución de datos en *Random Forest*. (b) Arquitectura de *Random Forest*.

Figura 2.6: Distribución de datos y arquitectura de *Random Forest*.

Por otro lado, en la Figura 2.6b, se puede observar como el modelo se compone de varios árboles, entrenados con subconjuntos del conjunto de test, y a su vez, los resultados proporcionados por estos modelos se combinarán en un único resultado Machado et al. (2015).

2.1.1.3. Perceptrón

El modelo Perceptrón Rosenblatt (1958) fue creado basándose en el funcionamiento de una neurona del cerebro. El modelo más básico de Perceptrón sería equivalente a una única neurona. El potencial de este modelo es la capacidad de asociarse con otras neuronas, para formar lo que llamaríamos una Red Neuronal. Las Redes Neuronales (NN) son modelos complejos que contienen varias capas de neuronas unidas entre sí, que se comunican mediante las conexiones. Esto se explicará en detalle más adelante, en la sección 2.1.1.4.

En la Figura 2.7 se puede observar la arquitectura del modelo Perceptrón, en este caso para cinco entradas, siendo x_i los datos de entrada, y w_1 los

pesos asociados a cada entrada[5].

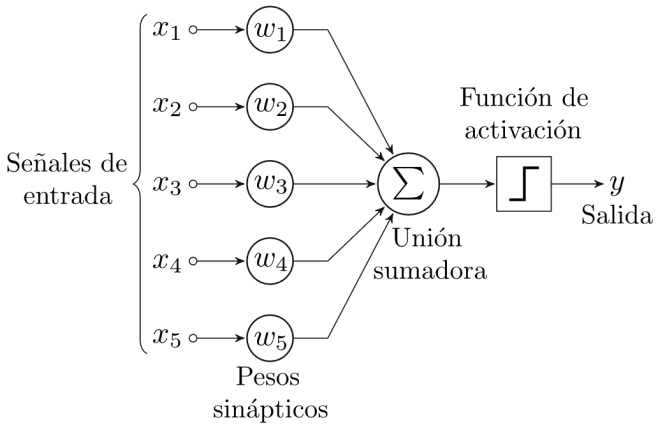


Figura 2.7: Arquitectura de Perceptrón con cinco señales de entrada.

La forma de aprendizaje de este modelo es mediante la actualización de los pesos de cada dato de entrada, buscando que éstos sean los que consigan que la salida del algoritmo sea la deseada. La fórmula para calcular si un dato pertenece o no a una clase, se muestra en la ecuación 2.1, siendo x el vector de entrada, w el vector de pesos para cada entrada, y u el umbral, como un margen para fijar un nivel mínimo de actividad de la neurona.

$$f(x) = \begin{cases} 1, & \text{Si } w \cdot x - u > 0 \\ 0, & \text{Otro caso.} \end{cases} \tag{2.1}$$

Existe también un tipo más avanzado de Perceptrón, llamado Perceptrón multicapa, que combina varias neuronas para obtener un modelo más complejo. Dicho modelo es la forma más básica de una Red Neuronal, que se explicará en detalle en la sección 2.1.1.4.

2.1.1.4. Aprendizaje profundo

El aprendizaje profundo o *Deep Learning* (DL) es un área del *Machine Learning* cuya premisa se basa en el uso de un tipo de modelo, las Redes Neuronales, cuyo comportamiento se asemeja al del cerebro humano. La arquitectura del modelo se basa en una serie de unidades, llamadas neuronas, conectadas entre sí a lo largo de varias capas, una de entrada de datos,

[5] https://es.wikipedia.org/wiki/Perceptr%C3%B3n

otra de salida, y una o varias intermedias, llamadas capas ocultas. Dichas neuronas cuentan con la arquitectura del modelo Perceptrón. En la Figura 2.8 se muestra un esquema de una Red Neuronal[6].

Cada neurona, se conecta una a otra y tiene un peso y un sesgo asociado. Además, se utiliza una función de activación que crea un umbral determinado para cada neurona; si la salida de dicha neurona está por encima del valor umbral, ésta se activa, enviando datos a la siguiente capa de la red, de lo contrario, no se envían los datos. El aprendizaje consiste en encontrar pesos y sesgos que hagan que la Red Neuronal muestre el comportamiento deseado Schmidhuber (2015).

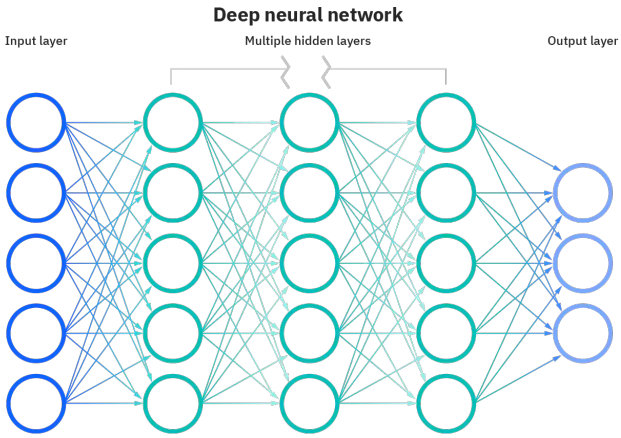


Figura 2.8: Arquitectura de una Red Neuronal.

Una de las principales diferencias entre el *Machine Learning* (ML) y el *Deep Learning* (DL), es que los algoritmos de ML necesitan recibir como entrada vectores de características numéricas extraídas previamente mediante otras técnicas de extracción de características, como se hablará en la sección 2.2.2. Sin embargo, los modelos de DL son capaces de calcular en el propio modelo las características de los datos que sean óptimas para el modelo. Además de esto, se pueden optimizar esas características mediante el propio proceso de aprendizaje de la red. En la Figura 2.9 se puede observar dicha diferencia Khan et al. (2021).

[6] https://www.ibm.com/cl-es/cloud/learn/neural-networks

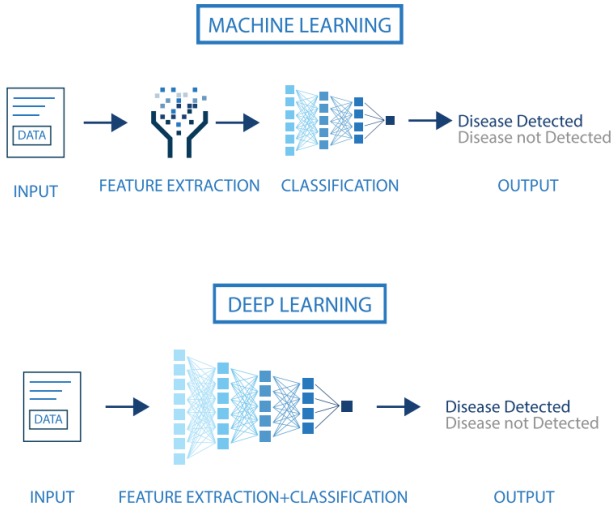


Figura 2.9: Diferencias entre el *Machine Learning* y el *Deep Learning*.

La forma más común de aprendizaje de las Redes Neuronales, es el algoritmo de *Backpropagation*, el cual se utiliza para regular el peso y el sesgo de cada neurona de la capa. Dicho algoritmo, se basa en una función de coste que recoge el valor de salida de la neurona y calcula cuan distinto es al valor de salida esperado (lo que se denomina la pérdida o *loss*); y en una función de optimización, que se basa en la función de coste, para actualizar el valor del peso y el sesgo. En la ecuación 2.2 se puede observar la fórmula que calcula la salida de la j-ésima neurona de la i-ésima capa de la Red Neuronal, siendo w y b, el peso (*weight*) y el sesgo (*bias*), y z la salida asociada a esa neurona[7].

$$z_j^{[i]} = w_j^{[i]} x + b_j^{[i]} \tag{2.2}$$

El proceso de *BackPropagation* tiene dos fases: la primera, propagación hacia delante (*forward propagation*), donde la señal de entrada es propagada desde la capa de entrada, hasta la capa de salida, donde el valor de los pesos y el sesgo se mantiene constante. La segunda, propagación hacia atrás (*backward propagation*), utiliza la función de coste para calcular las pérdidas, y dicha pérdida se propaga desde la capa de salida hasta la capa de entrada, donde los pesos y sesgos se regulan mediante el cálculo de la función de optimización, para ello, se calcula la derivada parcial de la función de coste

[7] https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-deep-learning

para el parámetro θ (peso y sesgo), tal y como se muestra en la ecuación 2.3, siendo a el resultado de aplicar a z la función de activación de la neurona[8]. Este proceso se repite hasta que el modelo es capaz de devolver la salida deseada Li et al. (2012).

$$\frac{\partial L(z,y)}{\partial \theta} = \frac{\partial L(z,y)}{\partial a} \times \frac{\partial a}{\partial \theta} \times \frac{\partial z}{\partial \theta} \tag{2.3}$$

Las funciones de coste pueden variar según el modelo, se calculan en base a una función de pérdida. En la Figura 2.10 se pueden observar los ejemplos más comunes de funciones de pérdidas[9]. En la ecuación 2.4 se muestra la función de coste $J(\theta)$ en función de la ecuación de pérdida $L(\hat{y}^{(i)}, y^{(i)})$, siendo θ los parámetros del modelo (peso y sesgo), $h_\theta(x^{(i)})$ y $\hat{y}^{(i)}$ la predicción del modelo para el dato de entrada $x^{(i)}$, y $y^{(i)}$ la salida real para dicho dato de entrada[9].

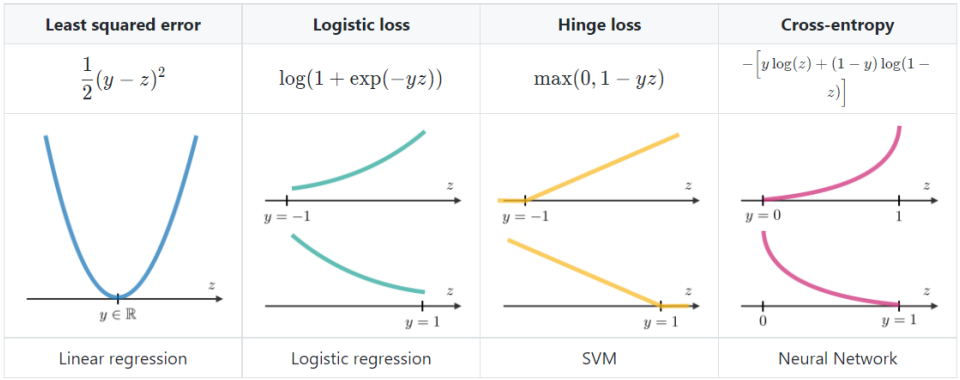


Figura 2.10: Funciones de pérdida.

$$J(\theta) = \sum_{i=1}^{m} L(h_\theta(x^{(i)}), y^{(i)}) \tag{2.4}$$

Por otra parte, como función de optimización la más utilizada es el Gradiente Descendente, el cual actualiza los pesos y el umbral en función del valor obtenido en la función de coste y un factor de aprendizaje, dicha fórmula se muestra en la ecuación 2.5, siendo α el factor de aprendizaje, que sirve para determinar cuánto se actualiza el parámetro[8].

[8] https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-deep-learning
[9] https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-supervised-learning

$$\theta = \theta - \alpha \frac{\partial L(z, y)}{\partial \theta} \tag{2.5}$$

En este trabajo se han aplicado modelos basados en Redes Neuronales, debido a que funcionan especialmente bien para las tareas de PLN, dichos modelos son versiones más complejas de esta misma arquitectura, que se explican a continuación.

- **Redes Neuronales Convolucionales (CNN)** LeCun et al. (1998): son redes enfocadas a la extracción de características invariantes dentro del texto. Este tipo de redes son más utilizadas en modelos de imágenes, ya que fueron diseñadas para lidiar con datos de 2 dimensiones, pero también se han desarrollado bastante en el área de PLN. Presentan una organización jerárquica, localidad e invariación espacial (los objetos se detectan aunque estén rotados, escalados o transformados). La arquitectura de las CNN se basa en 3 tipos de capas distintas: la capa de entrada, que recibe un mapa de características, generalmente de 2 dimensiones; la capa de convolución, que principalmente se compone de un *kernel* (núcleo) que se desplaza sobre los datos de entrada, calculando el producto escalar y extrayendo de esta manera un mapa de características, los coeficientes de cada *kernel* se aprenden durante el entrenamiento gracias al descenso del gradiente y al algoritmo de *backpropagation*. Mientras que una capa densa captura patrones y características globales, una convolucional captura patrones y características locales. Por último, la capa de *pooling*, cuyo objetivo fundamental es reducir el número de características utilizadas, para ello, realiza operaciones que utilizan ventanas para agrupar valores de la entrada, ya sea tomando el valor máximo (*Max Pooling*, utilizada generalmente) o el valor medio (*Average Pooling*). En la Figura 2.11a, se puede observar su arquitectura Yin et al. (2017).

- **Redes Neuronales Recurrentes (RNN)** Elman (1990): éstas redes están enfocadas en extraer información de datos a lo largo del tiempo. Su arquitectura permite a las capas procesar tanto la información actual, como la información anterior, de esa forma se tiene una concepción del contexto y el histórico de los datos. Esta aproximación es ampliamente utilizada para PLN por su naturaleza secuencial. Los primeros modelos de RNN tenían un inconveniente conocido, aplicados a

largas cadenas de texto el significado del contexto se perdía debido a que el gradiente se va desvaneciendo a valores muy pequeños, impidiendo eficazmente el peso de cambiar su valor, lo que puede impedir que la Red Neuronal continúe su entrenamiento. Este problema es conocido como el problema de desvanecimiento de gradiente (*Vanish Gradient*). Por este motivo, existen modelos de RNN que resuelven este problema: *Long Short-Term Memory* (LSTM) Hochreiter y Schmidhuber (1997) y *Gated Recurrent Unit* (GRU) Cho et al. (2014), los cuales tienen neuronas más complejas, con mayor número de puertas, encargadas de seleccionar la información que se mantiene y se traslada a la siguiente capa. Dichas arquitecturas se pueden ver en la Figura 2.11b y c, siendo r y f las puertas que indican el reseteo de la secuencia, y z la capa que indica cuanta información o qué información se mantiene.

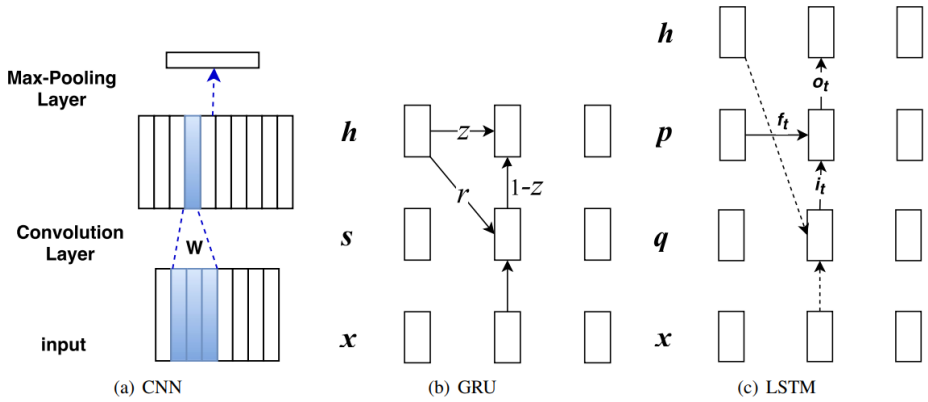


Figura 2.11: Arquitectura de las 3 Redes Neuronales más utilizadas.

2.1.1.5. *Transformers*

Los modelos de *transformers* Vaswani et al. (2017) son modelos de aprendizaje profundo que incorporan un mecanismo de atención denominado *self-attention*, que permite atender a distintos elementos de una secuencia sin que tengan que atravesar procesamientos intermedios, tal y como se hace en las RNN. La Figura 2.12 muestra la arquitectura del Transformer, siendo la parte izquierda del modelo lo que se denomina *encoder*, y la parte derecha el *decoder* Vaswani et al. (2017).

Dichas partes se pueden utilizar de distintas maneras, usando de forma aislada cualquiera de las partes, o ambas en conjunto, debido a ello, exis-

ten tres tipos de modelos que utilizan la arquitectura del *transfomer*. Los modelos que utilizan ambas partes son conocidos como modelos *sequence-to-sequence*, el más utilizado es el T5 Raffel et al. (2020) sobre todo para tareas de traducción. Los modelos auto-regresivos o generadores, solamente hacen uso de la parte del *decoder*, siendo GPT el más conocido Radford et al. (2018). Por último, los modelos auto-regresivos o discriminativos hacen uso exclusivo del *encoder*, siendo *Bidireccional Representations from Transformers* (BERT) Devlin et al. (2018) el modelo más común. En secciones posteriores (4.3.4 y 4.3.5) se explicará más en detalle la variedad de modelos de *encoder* utilizados tanto de forma general, como en este trabajo.

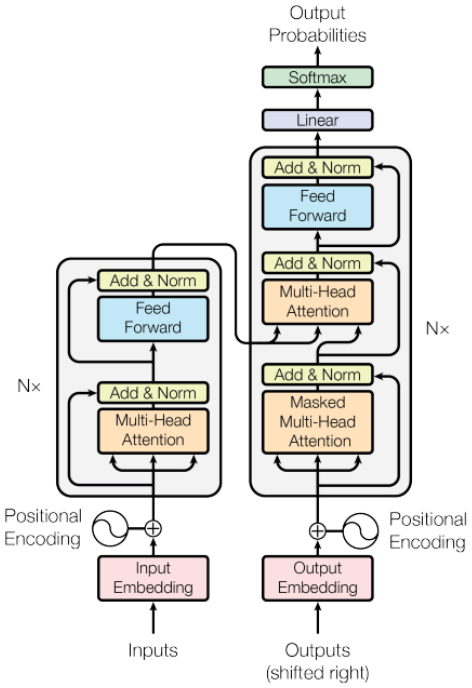


Figura 2.12: Arquitectura del Transformer.

El *encoder* recibe la secuencia de entrada y genera una representación de alto nivel de dicha entrada que puede ser usada tanto por el *decoder* como por capas adicionales para realizar distintos tipos de tareas (clasificación, predicción, entre otras). Por otra parte, el *decoder* utiliza la salida del *encoder* y otras entradas para generar una salida. Con el mecanismo de *self-attention* que usan los *transformers*, la representación de cada elemento a la entrada depende de los demás elementos, dando lugar a una representación

contextualizada. En la ecuación 2.6 se muestra la función de atención.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_K}})V \tag{2.6}$$

La función de atención está formada por una consulta Q y un conjunto de pares clave-valor como salida. La consulta Q representa el vector de una palabra, las claves K son el resto de palabras de la secuencia, de dimensión d_K y V el valor vectorial de la palabra que se procesa en ese instante. En otras palabras, en este mecanismo de atención el vector de resultante se calcula como una suma ponderada de los valores, donde el peso asignado a cada valor se calcula mediante una especie de "función de compatibilidad" de la consulta con la clave correspondiente.

Respecto a la función de auto-atención o *self-attention*, esta varía ciertamente de la atención convencional. La diferencia es que los vectores Q, K y V no son los vectores de entrada originales, si no una transformación de éstos, W_Q, W_K, W_V. La transformación es simplemente una multiplicación matricial entre los vectores de entrada y las matrices de transformación correspondientes. La razón de hacer esto, es que si no transformamos los vectores de entrada, el producto escalar para calcular el peso del valor de cada entrada siempre dará una puntuación de peso máxima para el propio token de entrada. En otras palabras, cuando calculamos los n pesos de atención (j para $j = 1, 2, ..., n$) para el token de entrada en la posición i, el peso en i ($j == i$) es siempre mayor que los otros pesos en $j = 1, 2, ..., n (j <> i)$. Este puede no ser conveniente. Por ejemplo, para el token del pronombre, necesitamos que atienda a su referente, no al propio token del pronombre.

Además de este motivo, también es posible que la transformación produzca mejores representaciones para la los vectores Q, K y V. Los pesos de las matrices de transformación se entrenan a medida que se realiza el entrenamiento del modelo.

La mayoría de modelos actuales utiliza lo que se conoce como *multi-head attention*, lo cual se basa en añadir varias unidades de *self-attention*, que en lugar de ejecutar una función de atención sola cada vez, proyecta Q, K y V a varios espacios lineales para realizar la función de atención en paralelo en cada uno de ellos, produciendo así vectores de salida de dimensión d_V que posteriormente se concatenan.

Al tener varias capas de atención tendremos varias matrices de pesos W_Q, W_K y W_V. Cada conjunto de matrices se inicializa de forma aleato-

ria y después del entrenamiento, cada conjunto de matrices se utiliza para proyectar los vectores de entrada en un subespacio diferente. Esto permite al modelo que preste atención conjuntamente a información de representaciones sobre diferentes subespacios y posiciones, lo cual permite generalizar mejor García García (2021).

2.1.2. Aprendizaje no supervisado

En el aprendizaje no supervisado, no se tiene información sobre la salida correcta correspondiente con los datos que se tiene, por lo que el proceso de aprendizaje trata de reconocer patrones automáticamente, aplicando varios tipos de inferencia.

Las tareas que llevan a cabo los modelos de aprendizaje no supervisado suelen basarse en agrupación o asociación. La agrupación (*clustering*), es un proceso por el cual se agrupan los datos basándose en la similitud que tengan entre sí en base a diferentes criterios de similitud, tratando que los datos de cada grupo (*cluster*) sean lo más parecidos entre sí, y lo más distintos a los miembros del resto de grupos Sharma et al. (2020).

2.1.3. Aprendizaje semi-supervisado

En el aprendizaje semi-supervisado se trata de aprender en presencia de datos etiquetados y no etiquetados. Tradicionalmente, el aprendizaje se ha estudiado en el paradigma no supervisado, en el que todos los datos están sin etiquetar, o en el paradigma supervisado, en el que todos los datos están etiquetados. El objetivo del aprendizaje semi-supervisado es entender cómo la combinación de datos etiquetados y no etiquetados puede cambiar el comportamiento del aprendizaje, y diseñar algoritmos que aprovechen dicha combinación.

A su vez, el aprendizaje semi-supervisado se puede utilizar principalmente en tres tareas, clasificación, regresión y agrupamiento. Los modelos de clasificación y regresión son entrenados tanto con datos etiquetados como no etiquetados, esperando así mejorar los resultados de los modelos supervisados. Por otro lado, los modelos de agrupamiento o *clustering* se entrenan con los datos no etiquetados y aportando información al modelo con los datos etiquetados, mejorando así las divisiones entre los distintos *clusters* Yang et al. (2022).

El aprendizaje semi-supervisado es de gran interés en el aprendizaje automático y la minería de datos porque puede utilizar datos no etiquetados fácilmente disponibles para mejorar las tareas de aprendizaje supervisado cuando los datos etiquetados son escasos o caros. Algunos ejemplos de modelos semi-supervisados son los modelos de autoentrenamiento, los modelos de mezcla y el coentrenamiento Zhu y Goldberg (2009).

2.1.4. Aprendizaje por refuerzo

El aprendizaje por refuerzo se inspira en la psicología conductivista, que permite determinar qué acciones debe escoger el modelo en un entorno dado, con el objetivo de conseguir una "recompensa" o premio de algún tipo.

Este tipo de modelos se utiliza principalmente en teoría de juegos u optimización, para tratar de encontrar la decisión o decisiones óptimas a un problema en base a las recompensas obtenidas para cada decisión tomada, buscando maximizar dichas recompensas Sutton y Barto (2018).

Los problemas más destacados del Procesamiento del Lenguaje Natural para este tipo de algoritmos son el análisis sintáctico, la comprensión del lenguaje, los sistemas de generación de textos, la traducción automática y los sistemas conversacionales Uc-Cetina et al. (2022).

2.2. Procesamiento del Lenguaje Natural

La información no estructurada digitalizada ha sufrido un gran aumento en los últimos años en volumen, accesibilidad y relevancia debido a la revolución digital de este siglo. Actualmente se han digitalizado millones de documentos de todos los tipos, generando un volumen de datos inmenso.

En esencia, el Procesamiento del Lenguaje Natural es una disciplina en la que converge la Inteligencia Artificial, el aprendizaje automático y la lingüística computacional. Esta disciplina, trata de hacer entender a las computadoras el lenguaje humano, basándose en establecer una comunicación humano-máquina Khurana et al. (2017). La minería de textos (*text mining*) o el PLN, presentan una serie de aproximaciones capaces de explorar grandes cantidades de información textual de una manera eficiente, permitiendo que los investigadores obtengan interpretaciones realistas de los datos Antons et al. (2020). El proceso de análisis de textos se puede dividir en un conjunto de fases secuenciales, las cuales se muestran en la Figura

2.13 Antons et al. (2020).

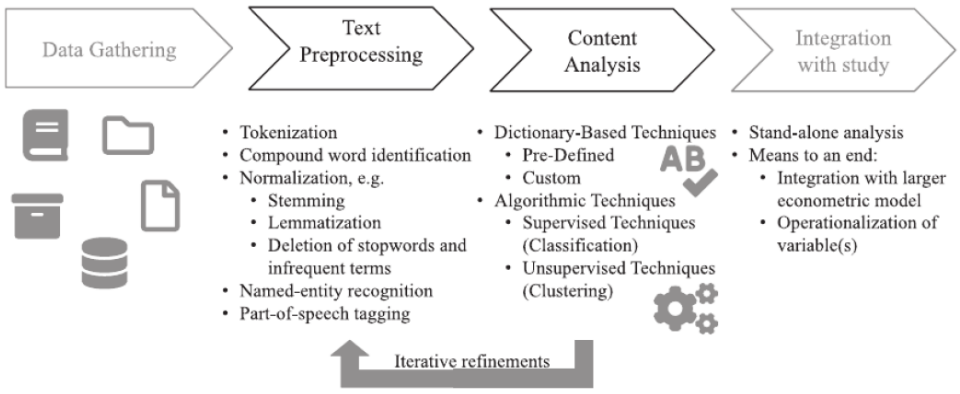


Figura 2.13: Procesos y técnicas de minería de textos.

La fase inicial, es la recolección de los datos que se van a utilizar y habitualmente dichos datos son extraídos de páginas web, redes sociales, revistas, artículos y libros. Para el caso de los textos biomédicos, la mayor parte de los datos son extraídos de artículos o *abstracts* científicos, o informes médicos. La segunda fase, es el preprocesamiento del texto obtenido mediante técnicas de procesamiento del lenguaje, que se explicarán en la sección 2.2.1; dicho proceso está dividido en dos partes principales, el preproceso de los datos y la extracción de características Naseem et al. (2021). La tercera fase, es el análisis del contenido, en el cual se introducen dichas características al modelo apropiado de *Machine Learning*, tal y como se introdujo en la sección 2.1. Por último, se integran los resultados del estudio en el área interesada.

2.2.1. Preprocesado de los datos textuales

Los modelos utilizados para el PLN, requieren que los datos de entrada introducidos hayan sido tratados y representados previamente de forma numérica. Por esta razón, es vital realizar un preprocesado del conjunto de datos textuales a utilizar y, posteriormente, realizar una representación de dichos datos o una extracción de características, generalmente a un espacio vectorial de diversas dimensiones.

Los datos textuales utilizados para los modelos generalmente contienen lenguaje informal, como abreviaciones, signos de puntuación o errores ortográficos. Este tipo de imperfecciones del lenguaje pueden causar ruido, por lo que es importante aplicar ciertas técnicas de preprocesado de datos y

hacer lo posible por mitigar o reducir dicho ruido. El texto a su vez puede verse como una colección de documentos, frases, palabras o incluso caracteres, según lo que sea más interesante para el modelo en cuestión, según el tipo de información que se quiera extraer.

El primer paso para entrenar un modelo de PLN es convertir los datos a un lenguaje que entienda la máquina. Los *datasets* de texto usualmente contienen palabras sin significado, signos de puntuación, errores ortográficos, etc., lo que puede producir un empeoramiento de la efectividad del modelo. Por ello, existen una serie de técnicas que se suelen aplicar para el preprocesado del texto:

- **Tokenización**: se trata de un proceso por el cual se transforma el texto en *tokens*. Los *tokens* pueden ser frases, palabras o incluso símbolos, son pequeñas unidades de significado textual. Generalmente es el primer paso del preprocesamiento Giachanou et al. (2017) y se suele aplicar para dividir el texto en palabras.

- **Reemplazo de caracteres**: en los datos textuales habitualmente se utilizan abreviaturas o acrónimos, que pueden ser confusos para el modelo, por lo que si es posible se reemplazan por la forma larga.

- **Corrección de ortografía**: existe la posibilidad de que el texto utilizado contenga errores de ortografía, por lo que existen herramientas para corregir estos errores y que el modelo no interprete estas palabras como distintas. Una de estas herramientas es la librería Textblob[10], en *Python*.

- **Eliminación de *stop-words***: en el lenguaje existen palabras que no aportan significado a la frase, sólo sirven como nexo entre los sustantivos o verbos, dichas palabras son las preposiciones, conjunciones, artículos, etc. Son muy frecuentes en el texto y no sirven de ayuda a los modelos de clasificación, por lo que usualmente se eliminan. Existen diversas herramientas para ello, como por ejemplo Natural Language Toolkit[11] o scikit-learn[12], ambas librerías en *Python*.

- ***Stemming***: una palabra puede tener diversas formas (prefijos, sufijos, tiempos verbales, etc), sin embargo su significado raíz sigue siendo el

[10] https://textblob.readthedocs.io/en/dev/
[11] https://www.nltk.org/
[12] https://scikit-learn.org/stable/

mismo, por lo que usualmente se sustituyen estas palabras por su lexema o raíz, con el fin de que las variaciones de las palabras no interfieran en el significado Mejova y Srinivasan (2011). Existen distintos algoritmos para realizar el *stemming*, como el algoritmo de Porter Willett (2006), de Lancaster o *Snowball*.

- **Lematización**: un proceso parecido al *Stemming*, pero en este caso las palabras no se "cortan" por la raíz, si no que se traducen a su forma base, lo que supone un aumento de la complejidad de la técnica. Esta lematización se puede aplicar utilizando bases de datos léxicas, como por ejemplo, WordNet[13], que contiene palabras y sus relacionadas léxicas y semánticas. Dicha técnica se puede aplicar también con las herramientas vistas anteriormente (NLTK y Textblob, entre otras).
- ***Part of Speech Tagging (POS)* o Etiquetado de parte del discurso**: se trata de una técnica para etiquetar a las palabras con sus categorías gramaticales, lo que en ocasiones puede facilitar el entrenamiento del modelo.

Las técnicas de preprocesamiento que se han utilizado en este trabajo se detallarán más adelante, en la sección 4.3.1.

2.2.2. Extracción de características

La extracción de características es el proceso mediante el cual se transforma el texto previamente preprocesado en información que el modelo de *Machine Learning* pueda entender, para ello existen diversas aproximaciones. Algunas de ellas utilizadas durante el desarrollo de este trabajo y se muestran a continuación:

- **Representación categórica**. En las formas más básicas de representación, se representan las palabras mediante un vector con valores 1s ó 0s, como por ejemplo en los modelos *one-hot enconding* y *Bag-of-words* (BoW).
  - *One-hot enconding*: en este modelo cada palabra del vocabulario está representada mediante un vector del tamaño del vocabulario, donde todos los elementos tienen un valor de 0, a excepción del elemento que representa dicha palabra, cuyo valor es 1.

[13] https://wordnet.princeton.edu/

- *Bag-of-words*: este modelo es una extensión de *One-hot enconding*, en el cual se pueden representar grupos de palabras, o frases realizando una combinación de los *one-hot vectors* de las palabras que lo forman. En la Figura 2.14 Naseem et al. (2021) se muestra un ejemplo de dichas representaciones. Cabe destacar que ambos modelos ignoran las relaciones sintácticas y semánticas entre palabras.

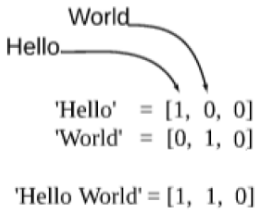


Figura 2.14: Ejemplo de *one-hot enconding* y *Bag-of-words*.

- **Representación frecuencial**. La siguiente forma a nivel de complejidad son las representaciones frecuenciales. La más común es *Term Frequency-Inverse Document Frequency* (TF-IDF) Willett (1988), la cual se utiliza para representar las palabras basándose en su frecuencia de aparación en los documentos, pero reduciendo el efecto de las palabras muy comunes, como conjunciones, preposiciones, etc, y dando valor a las palabras poco frecuentes. La función de cálculo de dicha representación se muestra en la ecuación 2.7, siendo t los términos, d cada documento, D la colección de documentos, y df_t la suma de los documentos que contienen el término i Naseem et al. (2021).

$$TF - IDF(t, d, D) = TF(t, d) \times \log(\frac{D}{df_t}) \tag{2.7}$$

La función TF se muestra en la ecuación 2.8, siendo fd_t y fd_i la frecuencia del término t e i en el documento d, respectivamente, y T_d el número de términos del documento d. De esta forma se calcula el peso del término t en el documento d, mediante la división de las veces que éste aparece en el documento entre las palabras totales del mismo[14].

[14] https://es.wikipedia.org/wiki/Tf-idf

$$TF(t,d) = \frac{fd_t}{\sum_{i=0}^{T_d} fd_i} \tag{2.8}$$

Sin embargo, todas estas formas de representación no son capaces de representar y capturar la sintaxis y semántica de las palabras, y comienzan a tener problemas cuando se trata de grandes cantidades de datos, debido a que las representaciones son proporcionales al tamaño del vocabulario, llegando a tener vectores de miles de dimensiones. Por ello, es necesario desarrollar modelos que sean capaces de representar dicho significado semántico, además de hacerlo para espacios de representación más reducidos.

- **Representación continua de palabras**. En el área del procesamiento del lenguaje, se crearon modelos de aprendizaje no supervisado para representar el significado textual en base a la semántica y el contexto de las palabras, como los modelos de *Word Embeddings*, un método de aprendizaje que extrae las características del texto, mapeando las palabras a un vector de N dimensiones. Para ello, existen diversos algoritmos, principalmente tres: *Word2Vec*, GloVe y *FastText*.

  - *Word2Vec* Mikolov et al. (2013b): es un modelo de Redes Neuronales que genera un vector para cada palabra, para ello, se sirve principalmente de dos algoritmos distintos para tratar de captar el significado semántico de cada palabra. El primer algoritmo, *Continuous Bag of Words* (CBOW), trata de predecir una palabra en función de su contexto, representando a cada palabra como una combinación de las palabras de su alrededor, así, palabras utilizadas en contextos similares tienen representaciones similares y cercanas en el espacio. El segundo algoritmo es *Skip-Gram*, funciona de la forma opuesta, el cual trata de predecir el contexto a raíz de una palabra central.

    El modelo CBOW es más rápido y suele ser mejor para grandes conjuntos de datos, mientras que *Skip-Gram* suele ser mejor para conjuntos más pequeños, y en la representación de palabras poco frecuentes. En la Figura 2.15 se pueden observar ambas arquitecturas Mikolov et al. (2013a).

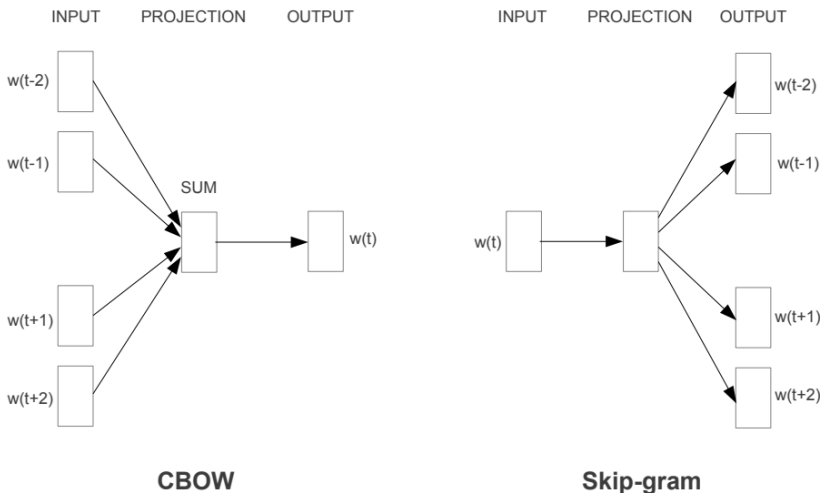


Figura 2.15: Arquitectura CBOW y *Skip-Gram*.

Este modelo de representación puede entrenarse directamente con un conjunto de datos, o bien puede utilizarse un modelo previamente entrenado con un conjunto de datos similar, mucho más grande que el que se va a utilizar. Esto permite obtener una representación mejor, y más precisa de las palabras.

- *Global Vectors* (GloVe) Pennington et al. (2014): este modelo se basa en las ocurrencias de una palabra en el corpus, en primer lugar creando la matriz de co-ocurrencias del corpus, para posteriormente realizar una factorización para obtener los vectores N-dimensionales que representarán a cada palabra. Sin embargo, tanto GloVe como *Word2Vec* no son capaces de representar a palabras fuera del vocabulario (OOV en inglés) con el que fueron entrenados los modelos.

- *FastText* Bojanowski et al. (2017): es un modelo basado en CBOW, que también se entrena mediante una Red Neuronal. A diferencia de los dos modelos anteriores, este modelo preprocesa las palabras, dividiéndolas en n-gramas en lugar de utilizar la palabra completa. Este modelo proporciona muy buenos resultados para el caso de palabras raras, y reduce el tiempo de entrenamiento del modelo *Word2Vec*, además de poder representar palabras OOV. En este trabajo se utiliza un modelo preentrenado de *FastText*, que se explicará en la sección correspondiente 4.3.3.2.

- **Representación contextual**. En la representación continua, se generan los vectores de palabras de forma estática, basándose en el contexto global de la palabra, generando una sola representación de esa palabra. Esto puede provocar un problema a la hora de representar palabras polisémicas, ya que todos los significados de esa palabra tendrían la misma representación, y además ésta sería una mezcla de todos los distintos contextos en los que aparece, lo cual no parece ser muy buena forma de representar su significado.

  Dadas estas limitaciones, se ha tratado de crear *embeddings* dependientes del contexto de la palabra para así tener mejores representaciones de éstas. Los modelos más comunes son ELMo Peters et al. (2018), BERT Devlin et al. (2018) y GPT-2 Radford et al. (2019). Estos modelos de lenguaje son capaces de generar representaciones para cada palabra dado un contexto, pero esa misma palabra dentro de otro contexto tendría otra representación distinta, ajustada a dicho contexto. El éxito de este enfoque sugiere que estas representaciones capturan propiedades altamente transferibles y agnósticas del lenguaje Ethayarajh (2019). En la Figura **??** se puede observar como se representarían distintos vectores para la palabra ratón (*mouse*) en los distintos contextos donde esta adopta varios significados[15].

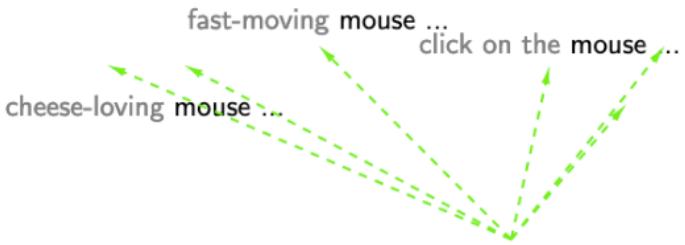


Figura 2.16: Distintas representaciones para palabra polisémica *mouse*.

2.2.3. Procesamiento del lenguaje en textos biomédicos

Como la mayoría de los descubrimientos biomédicos se comunican en publicaciones académicas, encontrar y leer información relevante en la literatura es esencial para cualquier investigador en ciencias de la salud. Además de la literatura científica, también es importante estudiar y extraer información

[15] https://ai.stanford.edu/blog/contextual/

sobre la historia clínica electrónica, la cual es generada en las instituciones sanitarias con respecto a la atención a los pacientes.

Por este motivo, la investigación en PLN biomédica ha aumentado enormemente en los últimos 30 años. Se ha convertido en una actividad destacada debido a la necesidad crítica de aprovechar la cantidad de información en textos relacionados con la investigación biomédica y la atención clínica, así como también la información relacionada con la salud en Internet, y la necesidad de utilizar la información para aplicaciones relacionadas con muchas facetas de la salud Friedman et al. (2013).

Dado que las publicaciones académicas, los historiales, informes y notas clínicas son contenido textual, el procesamiento del lenguaje natural adquiere cada vez más importancia en la investigación biomédica, ya que puede facilitar en gran medida la productividad de la investigación al extraer información clave del texto libre y convertirla en conocimiento estructurado para la comprensión humana. Desde finales de la década de 1990, la colaboración interdisciplinaria entre las comunidades de PLN y biomédica se ha hecho más común, formando una nueva área de investigación conocida como procesamiento de lenguaje natural biomédico (BioPLN), con el objetivo de desarrollar métodos de PLN para varios tipos de aplicaciones biomédicas Huang y Lu (2016).

Para ello, los desarrolladores de BioPLN incorporan tecnologías de extracción de información (IE), como por ejemplo, extracción de eventos o extracción de entidad-relación, para identificar el segmento de texto que puede representar un foco de información objetivo. El foco puede ser una interacción entidad-entidad, por ejemplo, la interacción fármaco-fármaco o DDI Segura-Bedmar et al. (2011), la interacción proteína-proteína o PPI Krallinger et al. (2008), o la relación entidad-entidad, como la relación entre genes Nédellec (2005), entre otras.

Por ello, nace la necesidad de desarrollar nuevos paradigmas de PLN que involucren e integren estadística, conocimiento lingüístico, conocimiento específico e inteligencia artificial del ámbito biomédico y clínico, tal y como se muestra en la Figura 2.17 Friedman et al. (2013).

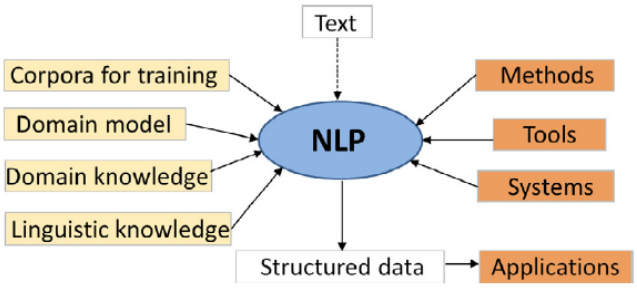


Figura 2.17: Aspectos de un sistema PLN.

A pesar de contar con una gran cantidad de texto con el que se puede trabajar, la dificultad de la aplicación de los algoritmos de aprendizaje supervisado reside en la inaccesibilidad a gran cantidad de corpus clínicos anónimos, lo cual es necesario para el desarrollo y la evaluación de sistemas de PLN Friedman et al. (2013). Este problema, es el principal cuello de botella en cuanto al desarrollo del BioPLN se refiere. El acceso a notas e historiales clínicos es limitado, debido a las políticas de protección de datos y confidencialidad con los pacientes. Por lo general, sólo los investigadores afiliados con un centro médico pueden acceder a la información clínica con aprobación institucional, pero dichos documentos no pueden ser compartidos a la comunidad de PLN a menos que éstos estén anonimizados y se hayan establecido acuerdos de uso de datos. Para la anonimización existen modelos como *MITRE Identification Scrubber Toolkit* (MIST) Aberdeen et al. (2010), que se encargan de detectar información sensible en los informes médicos.

Respecto a textos en español existe una campaña de evaluación de sistemas de anonimización, MEDDOCAN: Medical Document Anonymization Track [16], que cuenta con un corpus de 1000 casos clínicos seleccionados manualmente. Dicha tarea se estructuró en base a dos subtareas, la primera fue el reconocimiento de entidades, y la segunda la detección de partes sensibles en el texto.

Otro problema que existe es la dificultad de obtener grandes cantidades de datos médicos etiquetados, debido al alto coste que esto supone. Por este motivo, es importante explorar otras técnicas como el *Transfer Learning*, que permite utilizar el conocimiento aprendido por un modelo en otro modelo, para así mejorar los resultados sin la necesidad de entrenar un nuevo modelo con grandes cantidades de datos. Por ejemplo, algunos investigadores reali-

[16] https://temu.bsc.es/meddocan/

zaron un entrenamiento de un modelo lingüístico y transfirieron sus pesos para preentrenar un modelo de NER biomédico con la misma arquitectura, lo que obtuvo buenos resultados Sachan et al. (2018).

Con el fin de facilitar la tarea de obtención de documentos, existen bases de datos médicas dedicadas a publicar información clínica, como citas o *abstracts* de artículos de la literatura biomédica, así como también ciertas partes de artículos médicos, con el fin de ayudar al entrenamiento y evaluación de los modelos biomédicos. Por ejemplo, PubMed[17] es una base de datos bibliográfica que comprende más de 21 millones de citas de la literatura de investigación biomédica; las entradas están anotadas con términos del Medical Subject Heading (MeSH)[18].

Cabe destacar también que en España existe una iniciativa por parte del gobierno de España llamada Plan de Impulso de las Tecnologías del Lenguaje (Plan TL)[19], que tiene como objetivo fomentar el desarrollo del PLN, la traducción automática y los sistemas conversacionales en lengua española y lenguas cooficiales. Es por esto que han proporcionado una serie de modelos de *transformer* entrenados con conjuntos de textos en español, y algunos de ellos, específicamente en el dominio biomédico en español[20]. Algunos de estos modelos se utilizan en este trabajo, y se hablará específicamente de ellos en la sección dedicada al desarrollo del trabajo 4.2.

2.2.3.1. Retos de BioPLN

Para promover esta colaboración interdisciplinar, evaluar y hacer avanzar las técnicas de PLN en biomedicina, se han organizado a lo largo de los años un sinfín de tareas y/o retos compartidos de BioPLN. Las evaluaciones de los retos BioPLN se agrupan principalmente por el género del texto: las tareas biológicas se centran en las publicaciones académicas, mientras que las tareas clínicas se centran en los registros clínicos. Las tareas de desafío se introdujeron por primera vez en la comunidad de BioPLN en la ACM KDD Cup[21] en 2002, seguida por TREC Genomics[22] en 2003, ambas utilizando corpus en inglés, y muchas otras en los siguientes años.

[17] http://www.ncbi.nlm.nih.gov/pubmed
[18] http://www.nlm.nih.gov/meshd
[19] https://plantl.mineco.gob.es/Paginas/index.aspx
[20] https://huggingface.co/PlanTL-GOB-ES
[21] https://kdd.org/kdd-cup/view/kdd-cup-2002/Intro
[22] https://trec.nist.gov/data/t12_genomics.html

Los primeros retos, como la KDD Cup y la TREC Genomics, se centraban en las tareas de recuperación de documentos o de clasificación de los mismos. En 2004, BioCreative[23] y JNLPBA empezaron a abordar la necesidad de detectar automáticamente entidades biomédicas en textos libres. En concreto, la tarea de mención de genes (GM) de BioCreative I tenía como objetivo la detección de nombres de genes, mientras que la tarea de JNLPBA Kim et al. (2004) incluía múltiples tipos de entidades, como el ADN, el ARN y el tipo de célula. El Reconocimiento de Entidades Nombradas (NER) en el ámbito biológico es esencial, ya que constituye la base de muchas tareas de PLN de nivel superior, como la interacción proteína-proteína o la extracción de la regulación génica (GR).

En 2013 se puso a prueba un nuevo evento de evaluación denominado ShARe/CLEF eHealth[24]. En él se abordaron tres tareas distintas: NER tradicional sobre nombres de enfermedades en notas clínicas y normalización, mapeo de acrónimos y abreviaturas en documentos clínicos a UMLS CUIs y recuperación de documentos relevantes para responder a las preguntas que los pacientes puedan tener al leer resúmenes de alta, cabe mencionar que los textos utilizados en está tarea están escritos en español.

También en 2014 el track TAC BiomedSumm[25] desafió a los participantes a aprovechar el conjunto de oraciones de citación que hacen referencia a un artículo específico ('citaciones') para su resumen, un problema importante en la investigación BioPLN. Más adelante, se pusieron en marcha nuevos retos, como IberEval 2017[26], que propuso una tarea de reconocimiento y resolución de abreviaturas utilizando *abstracts* de artículos biomédicos en español. En el año 2018, se puso en marcha la segunda edición del reto en IberEval 2018[27], que extendía el material introduciendo textos clínicos y casos de estudio.

En el año 2018, también se comenzó un nuevo reto llamado eHealth-KD Challenge 2018[28], que planteó una serie de tareas con textos médicos en español. La primera tarea de este reto era la identificación de palabras clave, la segunda la clasificación de fichas palabras, y la última la extracción de relaciones entre las mismas. Al año siguiente, se presentó el reto eHealth-

[23] https://biocreative.bioinformatics.udel.edu/
[24] https://sites.google.com/site/shareclefehealth/
[25] https://tac.nist.gov/2014/BiomedSumm/
[26] https://temu.bsc.es/BARR/
[27] https://temu.bsc.es/BARR2/
[28] http://www.sepln.org/workshops/tass/2018/task-3/

KD 2019[29], que contaba también con dos tareas de clasificación de palabras clave y sus relaciones. Por último, los retos eHealth-KD 2020[30] y eHealth-KD 2021[31] se basaron en reconocimiento de entidades y extracción de relaciones. Se describirá ampliamente el reto eHealth-KD 2021 en la sección 2.3, ya que se trata del reto que se aborda en este trabajo.

Además de los retos ya descritos, en el año 2020, la tarea IberLEF 2020[32] planteó un reto de minería de textos relacionados con el cáncer (CANTEMIST: CANcer TExt Mining Shared Task), el cual se componía de varias fases. En primer lugar se realizaba una tarea de NER, para reconocer menciones sobre tumores, después se realizaba una normalización, y finalmente se codificada para generar un índice de menciones para cada documento. El año siguiente, el reto IberLEF 2021[33] MEDDOPROF: MEDical DOcuments PROFessions recognition shared task, se basaba en la detección de profesiones en textos médicos, con la intención de detectar posibles riesgos y accidentes laborales entre los profesionales de la salud. Ambas tareas fueron con textos en español.

Por último, en el año 2022 se propuso el reto DISTEMIST: DISease TExt Mining Shared Task, el cual se estructura en dos tareas independientes. La primera tarea consiste en detectar menciones de enfermedades en casos clínicos públicos, mientras que la segunda tarea asignaba un código SNOMED CT a dichas menciones Miranda-Escalada et al. (2022). SNOMED CT es una terminología clínica integral, multilingüe y codificada. Puede utilizarse para codificar, recuperar, comunicar y analizar datos clínicos permitiendo a los profesionales de la salud representar la información de forma adecuada, precisa e inequívoca[34].

2.2.4. Reconocimiento de Entidades Nombradas

El Reconocimiento de Entidades Nombradas (NER) tiene como objetivo reconocer las menciones que se realizan a entidades de todo tipo en los textos. Dichas entidades pertenecen a tipos semánticos predefinidos, como persona, lugar, organización, etc. Nadeau y Sekine (2007). NER no sólo actúa como

[29] https://knowledge-learning.github.io/ehealthkd-2019/
[30] https://knowledge-learning.github.io/ehealthkd-2020
[31] https://knowledge-learning.github.io/ehealthkd-2021
[32] https://temu.bsc.es/cantemist/
[33] https://temu.bsc.es/meddoprof/
[34] https://www.sanidad.gob.es/profesionales/hcdsns/areaRecursosSem/snomed-ct/quees.htm

una herramienta independiente para la extracción de información (IE), sino que también desempeña un papel esencial en diversas aplicaciones de PLN como la comprensión de textos, la recuperación de información, el resumen automático de textos, la respuesta a preguntas y la traducción automática.

Normalmente se divide la tarea de NER en dos categorías: NER genéricas (persona, lugar, etcétera) o NER de dominio específico, como por ejemplo el dominio biomédico. En cuanto a las técnicas aplicadas en NER, hay cuatro corrientes principales: enfoques basados en reglas, que no necesitan datos anotados ya que se basan en reglas elaboradas a mano; enfoques de aprendizaje no supervisado, que se basan en algoritmos no supervisados sin ejemplos de entrenamiento etiquetados a mano; enfoques de aprendizaje supervisado basados en características y, por último, enfoques basados en el aprendizaje profundo, que descubren automáticamente las representaciones necesarias para la clasificación y/o la detección a partir de la entrada en crudo Li et al. (2020).

Antes de comenzar a explicar las distintas técnicas y aproximaciones, es importante definir la tarea de manera formal. Una entidad nombrada es una palabra o una frase que identifica claramente un elemento de un conjunto de otros elementos que tienen atributos similares Sharnagat (2014). Ejemplos de entidades nombradas son los nombres de organizaciones, personas y nombres de lugares en el ámbito general; nombres de genes, proteínas, fármacos y enfermedades en el ámbito biomédico.

NER es el proceso de localizar y clasificar las entidades nombradas en el texto en categorías de entidades predefinidas. Formalmente, dadas una secuencia de tokens $s =< w_1, w_2, ..., w_N >$, NER obtiene como resultado una lista de tuplas, de la forma $< I_s, I_e, t >$, donde cada elemento es una entidad mencionada en la secuencia s, siendo $I_s \in [1, N]$ e $I_e \in [1, N]$, el inicio y el final de los índices de la entidad respectivamente, y siendo t el tipo de entidad de un conjunto de tipos de entidades predefinidas. En la Figura 2.18 se observa un ejemplo de un sistema NER reconociendo las entidades de una frase Li et al. (2020).

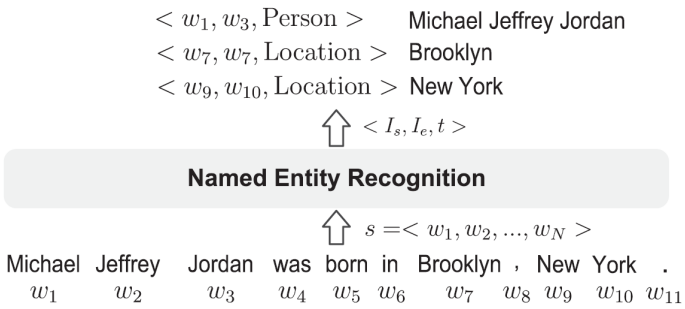


Figura 2.18: Sistema NER reconociendo una frase.

A la hora de gestionar entidades que sean formadas por más de una palabra se suele utilizar la notación IOB (*Inside–Outside–Beginning*) para representar las entidades. IOB proporciona etiquetas para indicar los límites de la entidad. Estas etiquetas son las siguientes: *B-entity*, para indicar la primera palabra de la entidad; *I-entity*, para las palabras siguientes; y *O* para indicar palabras sin entidad Ramshaw y Marcus (1999), siendo *entity* el tipo de entidad en cuestión. Este tipo de etiquetado es ampliamente utilizado en NER.

De forma similar a como se ha comentado en la sección 2.2.3 sobre los recursos que existen para tareas de PLN, existen también una serie de recursos disponibles para sistemas NER, como *datasets* anotados para el caso específico de NER biomédico, como por ejemplo, NCBI Disease Corpus[35], una colección de 793 resúmenes de PubMed, anotando menciones a enfermedades.

Respecto a las métricas de evaluación utilizadas, éstas suelen calcularse comparando sus resultados con anotaciones humanas. La comparación puede cuantificarse por coincidencia exacta o relajada. Si se utiliza la comparación exacta, para que una entidad sea reconocida como correcta, tienen que ser correctos sus dos índices, tanto el inicial como el final. Por otro lado, la coincidencia relajada Grishman y Sundheim (1996), permite que un caso sea válido siempre y cuando se le asigne a la palabra el tipo correcto de entidad, independientemente de sus límites, siempre y cuando haya un solapamiento entre los límites predichos y los reales.

[35] https://www.ncbi.nlm.nih.gov/research/bionlp/Data/disease/

2.2.4.1. Modelos y aproximaciones NER

Dentro de la tarea de reconocimiento de entidades, existen cuatro aproximaciones principales para realizar la misma:

- **Modelos basados en reglas**: Los sistemas NER basados en reglas se basan en reglas elaboradas a mano, o también de forma automática a partir de ciertas directrices. Las reglas pueden diseñarse a partir de una nomenclatura específica de un dominio y de patrones sintáctico-léxicos, como por ejemplo, FACILE Black et al. (1998) o SRA Aone et al. (1998). Estos sistemas se basan principalmente en reglas semánticas y sintácticas elaboradas a mano para reconocer entidades, por lo que funcionan muy bien cuando el léxico es exhaustivo. El problema de este tipo de sistemas es que debido a las reglas específicas del dominio, y si no se cuenta con diccionarios completos, no suelen obtener unos buenos resultados, además de que no pueden transferirse a otros dominios.

- **Modelos de aprendizaje no supervisado**: Un enfoque posible del aprendizaje no supervisado es el *clustering* Nadeau y Sekine (2007). Los sistemas NER basados en *clustering* extraen las entidades nombradas de los elementos agrupados basándose en la similitud del contexto. La idea clave es que los recursos léxicos, los patrones léxicos y las estadísticas calculadas en un gran corpus pueden utilizarse para inferir las entidades nombradas. Aplicado al dominio biomédico, existe un enfoque en el cual el modelo recurre a las terminologías, a las estadísticas del corpus (por ejemplo, la frecuencia inversa de los documentos y los vectores de contexto) y al conocimiento sintáctico (por ejemplo, la fragmentación de las frases) para extraer las entidades nombradas Zhang y Elhadad (2013).

- **Modelos de aprendizaje supervisado**: Aplicando el aprendizaje supervisado, NER se convierte en una tarea de clasificación multi-clase o de etiquetado de secuencias. A partir de muestras de datos anotadas, las características se diseñan para representar cada ejemplo de entrenamiento. A continuación, se utilizan algoritmos de aprendizaje automático para aprender un modelo que permita reconocer patrones similares a partir de datos no vistos. Los principales modelos supervisados utilizados son: Modelos Ocultos de Markov (HMM) Morwal et al.

(2012), Árboles de Decisión, modelos de Máxima Entropía Chieu y Ng (2003) y SVM.

- **Modelos de aprendizaje profundo**: En este tipo de modelos las características se aprenden entrenando el propio modelo, a diferencia de los modelos supervisados tradicionales, lo que favorece una mejor representación de los datos, ya que existe la posibilidad de que el modelo encuentre características ocultas. Existen muchos trabajos que utilizan el aprendizaje profundo para tareas de NER en el ámbito biomédico, por ejemplo utilizando arquitecturas CNN Yao et al. (2015), o redes LSTM bidireccionales (BiLSTM) Cho y Lee (2019), proporcionando resultados bastante buenos dentro de este ámbito.

- **Modelos de transformers**: Los modelos de transformers, han proporcionado muy buenos resultados en muchos ámbitos de PLN, incluida la tarea de NER. Para esta tarea, una posibilidad es utilizar la arquitectura del modelo BERT Devlin et al. (2018). Éste calculará las etiquetas IOB de salida para cada palabra de una secuencia de palabras dada. En la Figura 2.19 se muestra un ejemplo de la arquitectura de este modelo, compuesto por una capa de *embedding*, seguida de una capa de *encoding* posicional, y finalmente una capa densa para calcular las probabilidades de salida Arkhipov et al. (2019).

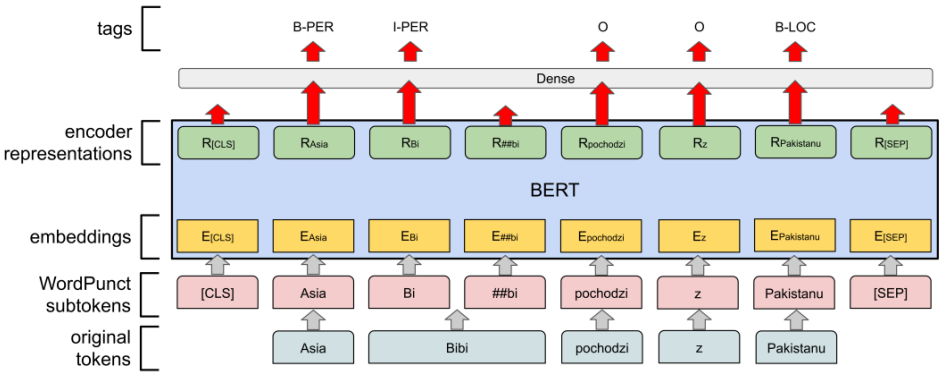


Figura 2.19: Arquitectura del modelo BERT para la tarea NER.

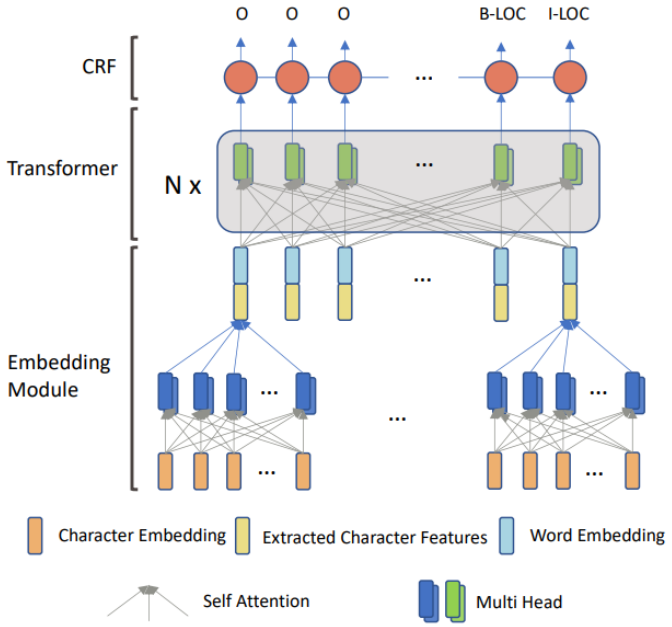


Figura 2.20: Arquitectura del modelo TENER para la tarea NER.

Otro ejemplo de la utilización de estos modelos es el compuesto por una capa de *embedding*, seguida de una capa de *encoding*, y finalmente una capa de *Conditiona Random Field* (CRF), en lugar de una capa densa, la cual calcula la probabilidad final de que una palabra pertenezca a una entidad. La arquitectura de este modelo se muestra en la Figura 2.20 Yan et al. (2019).

2.2.5. Relaciones entre entidades nombradas

Las entidades, como se ha visto en la sección 2.2.4, constituyen una unidad básica de información. Las ocurrencias de las entidades en una frase suelen estar vinculadas a través de relaciones bien definidas; por ejemplo, las ocurrencias de persona y organización en una frase pueden estar vinculadas a través de relaciones como *empleado en*.

La tarea de la extracción de relaciones (RE, *Relation Extraction* en inglés) consiste en identificar tales relaciones automáticamente. La extracción de relaciones es un paso muy útil en la IE, después del NER. El éxito de la RE requiere detectar tanto las entidades nombradas, como los tipos de entidad, para posteriormente determinar el tipo de relación que existe entre ellas Pawar et al. (2017). En la Figura 2.21 se puede observar el resultado

de ambos modelos, y la relación que existe entre ellos Zhong y Chen (2020).

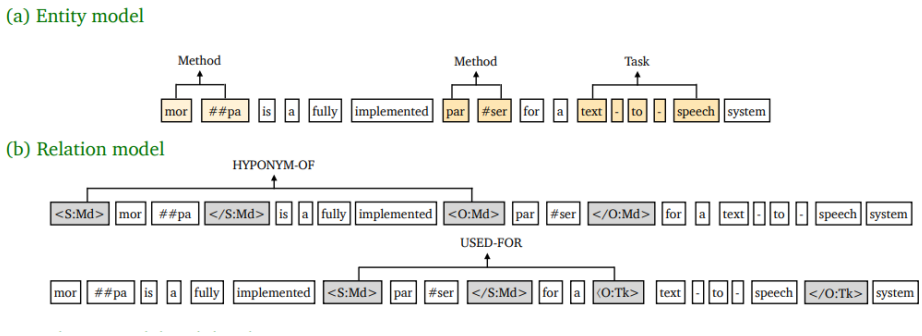


Figura 2.21: Sistema NER y sistema RE.

Formalmente, una relación se define como una tupla, de la forma $r = (e_1, e_2, ..., e_N)$, siendo e_i las entidades de una relación predefinida r. La mayoría de los sistemas de extracción de relaciones se centran en extraer relaciones binarias, ya que la extracción de entidades no binarias resulta una tarea compleja Bach y Badaskar (2007).

Uno de los principales problemas que surgen en los métodos supervisados de RE es el del desequilibrio de clases. Esto ocurre porque el número de instancias negativas (es decir, los pares de entidades que no tienen ninguna relación significativa) supera en gran medida el número de instancias positivas (es decir, los pares de entidades que tienen alguno de los tipos de relación predeterminados). Este desequilibrio de clases da lugar a que el *recall* de los sistemas sea bajo, ya que los clasificadores tienden a clasificar con la clase sin relación. Existen una serie de soluciones a este problema, una de ellas se basa en la votación entre un conjunto de clasificadores que mejoran significativamente los resultados del modelo en estas situaciones Kambhatla (2006). Otra posible solución es tratar de balancear los datos añadiendo instancias de las clases menos representadas, lo que se conoce como *Data Augmentation*, está práctica está detallada en la sección 2.2.6.

De la misma forma que NER tiene *datasets* públicos para realizar la tarea, en el caso de RE el conjunto de datos más utilizado en la literatura es ACE 2004[36], junto con dos conjuntos publicados posteriormente, ACE 2005[37] y ACE 2007[38], todos ellos plurilingües. Los corpus del 2004 y 2005

[36] https://catalog.ldc.upenn.edu/LDC2005T09
[37] https://catalog.ldc.upenn.edu/LDC2006T06
[38] https://catalog.ldc.upenn.edu/LDC2014T18

contenían textos de distintos géneros, entre ellos noticias de varias fuentes (radio o periódicos), y estaban escritos en inglés, árabe y chino. Respecto al corpus lanzado en 2007, éste contenía exclusivamente datos de prensa en español y blogs en árabe. Cabe mencionar que estos *datasets* son de temática general, y no pertenecen al ámbito biomédico.

Respecto a las métricas de evaluación utilizadas, suelen ser las tradicionales para un sistema de clasificación multi-clase, precisión, *recall*, y *F1Score*, cuyo cálculo se explicará la sección 5.3.

2.2.5.1. Modelos y aproximaciones RE

Las modelos de RE tienen tres enfoques principales, aprendizaje supervisado, semi-supervisado y no supervisado. Dentro de éstos, existen varios tipos de modelos, como se muestran a continuación:

- **Modelos de aprendizaje supervisado**: con los modelos de aprendizaje supervisado la tarea se orienta como una tarea de clasificación multi-clase, donde dada una frase $s = w_1, w_2, ..., e_1..., w_j, ...e_2, ..., w_n$ siendo e_i entidades, se puede construir una función G tal y como se muestra en la ecuación 2.9.

$$G_R(f(s)) = \begin{cases} +1, & \text{Si } e_1 \text{ y } e_2 \text{ están relacionadas mediante } R. \\ 0, & \text{Otro caso.} \end{cases} \tag{2.9}$$

  Siendo $f(s)$ las características de la frase s Bach y Badaskar (2007). Un ejemplo de modelo de clasificación utilizado para está tarea es SVM. También son utilizados otros modelos ya mencionados, como Redes Neuronales Lin et al. (2016) o *transformers*, estos modelos se utilizan en este trabajo, en la sección 4.4.3 se ahondará más en los parámetros y modelos utilizados en detalle.

- **Modelos de aprendizaje semi-supervisado**: Debido a la escasez de datos etiquetados para realizar modelos supervisados, el uso de modelos no supervisados o semi-supervisados aumenta considerablemente. Existen dos técnicas principales para el enfoque no-supervisado, una de ellas es el *Bootstrapping*, que se basa en inducir un clasificador partiendo de un pequeño conjunto de datos etiquetados y un gran

conjunto de datos no etiquetados Abney (2002), uno de los sistemas de RE que utiliza esta técnica es DIPRE Brin (1998).

La otra técnica utilizada es conocida como *Active Learning*. La idea clave es que el algoritmo de aprendizaje puede pedir las etiquetas verdaderas de algunas instancias seleccionadas no etiquetadas. La principal ventaja, es que se consigue un rendimiento comparable al de los métodos supervisados con muy pocas instancias etiquetadas. Un ejemplo es el modelo LGCo-Testing Sun y Grishman (2012).

- **Modelos de aprendizaje no supervisado**: Uno de los primeros enfoques para una RE completamente no supervisado es un algoritmo de *clustering*. El algoritmo parte de la base de que los pares de entidades que se dan en un contexto similar pueden agruparse y que cada par de un *cluster* es una instancia de la misma relación. Las relaciones entre entidades se descubren mediante este proceso de agrupación de *clusters* Hasegawa et al. (2004).

Todas las técnicas de RE explicadas en la sección anterior suponen que se conocen de antemano los límites y los tipos de menciones de entidades. Si no se dispone de estos conocimientos, para utilizar estas técnicas de extracción de relaciones es necesario aplicar primero alguna técnica de NER. Una vez identificadas las entidades y sus tipos, se pueden aplicar las técnicas de RE. Este método "en cadena" (*pipeline*) es propenso a la propagación de errores de la primera fase (NER) a la segunda fase (RE), por lo que es interesante aplicar sistemas que sean capaces de realizar esta tarea de manera conjunta.

2.2.6. Estrategias de balanceo de datos

Como se ha comentado en anteriores secciones, uno de los problemas de las tareas de NER y RE (especialmente RE) es que siempre va a existir un desequilibrio entre las palabras o pares sin entidad o relación, que con ellas. Esto genera que una de las clases tenga mucha representación con respecto a las demás, dando lugar a modelos ampliamente sesgados y tendentes a clasificar los datos como no relación / entidad.

2.2.6.1. Aumento de datos textuales

Una práctica común para mitigar este problema es la ya mencionada *Data Augmentation*. Está práctica tiene múltiples técnicas posibles a la hora

de ser aplicada, ya sea a nivel de palabra, de frase o incluso a nivel de caracteres. Un ejemplo de estas técnicas es *Back Translation*, que se realiza a nivel de palabra, y consiste en traducir la palabra en cuestión a otro idioma con algún modelo de traducción, y volver a traducirla a la lengua original, con alta probabilidad de que la palabra no sea exactamente igual, pero sí con el mismo significado Edunov et al. (2018).

Otra técnica utilizada a nivel de palabra es la utilización de sinónimos seleccionados a partir de una red de palabras como WordNet. Lo que se traduce en utilizar palabras sinónimas para aumentar los datos de cierta clase, sin utilizar la misma palabra.

Ambas técnicas son utilizadas en este trabajo, para la tarea de NER, cuyos detalles se explicarán en la sección 4.3.2. A parte de éstas, existen otras técnicas que no son abordadas en este trabajo, como por ejemplo la generación de frases similares mediante algún modelo de lenguaje generador Kafle et al. (2017), como puede ser algún modelo basado en *transformers*, como GPT.

2.2.6.2. Aumento de datos sobre el espacio de características

Las técnicas explicadas anteriormente sirven para realizar el aumento de datos a nivel de datos textuales (palabras, en este caso), pero existen otras técnicas utilizadas para aumentar el número de datos una vez que ya se han calculado las características de los mismos. Es decir, aplicar *Data Augmentation* a una matriz de características para generar muestras sintéticas similares a las que ya pertenecen a la clase que se pretende aumentar. Un ejemplo de esta técnica es *Synthetic Minority Oversampling Technique* (SMOTE) Chawla et al. (2002).

El algoritmo SMOTE se basa en generar ejemplos sintéticos para la o las clases minoritarias. Esto se hace operando directamente sobre el espacio de características, a diferencia de las técnicas explicadas anteriormente, utilizando para ello el algoritmo de los K vecinos más próximos (KNN) Cover y Hart (1967). Las muestras sintéticas se generan de la siguiente forma: se calcula la diferencia entre el vector de características sobre el que se pretende trabajar y su vecino más cercano. Después se multiplica esta diferencia con un número aleatorio entre 0 y 1, y se suma al vector de características original. Este nuevo vector sería una muestra sintética.

Esta generación de nuevas muestras aumenta el peso de la clase en el

modelo, además de ayudar a la generalización del mismo, ya que se dispone de más datos distintos para una misma clase. Esto se puede combinar también con la reducción de muestras de la clase mayoritaria, conocido como *under-sampling*.

Existen variantes del algoritmo original de SMOTE, como por ejemplo SMOTE-NC, que está diseñado para trabajar con características categóricas y continuas (a diferencia de SMOTE, que no puede manejar características categóricas). SMOTE-NC cambia ligeramente la forma en que se genera una nueva muestra. Para la generación de muestras sintéticas categóricas se calcula la distancia euclídea entre el vector de características y los demás vectores de características (muestras de la clase minoritaria) utilizando el espacio de características continuas. Las características continuas se calculan de la misma forma que en SMOTE Chawla et al. (2002).

En este trabajo se han utilizado la técnicas SMOTE-NC para abordar la tarea de RE, cuyos detalles se explicarán más adelante, en la sección 4.4.2.

2.2.7. Estrategias de reducción de dimensionalidad

Además de las técnicas explicadas en la sección 2.2.6, otra forma para mejorar los resultados en corpus desbalanceados es aumentar la separabilidad de las clases entre sí, para que al modelo le resulte más fácil discriminar entre estas. Una de las técnicas para ello es la reducción de dimensionalidad a un espacio de características donde los datos sean más separables. Existen dos técnicas principales de reducción de dimensionaldiad, *Principal Component Analysis* (PCA) y *Linear Discriminant Analysis* (LDA), que es la utilizada en este trabajo.

La técnica LDA se basa en maximizar la varianza entre clases y minimizar la varianza dentro de la clase, garantizando así la máxima separabilidad. En el caso del PCA, la forma y la ubicación de los conjuntos de datos originales cambian cuando se transforman a un espacio diferente, mientras que el LDA no cambia la ubicación, sino que sólo intenta proporcionar una mayor separabilidad de las clases y dibujar una región de decisión entre las clases dadas Balakrishnama y Ganapathiraju (1998).

La función objetivo del LDA consiste en buscar una matriz de proyección W tal que los puntos de datos en el nuevo espacio que pertenecen a la misma clase estén muy cerca, mientras que los puntos de datos de clases diferentes estén lejos unos de otros Gado et al. (2017). En la ecuación 2.10 se muestra

el ratio a maximizar por la matriz W.

$$J(W) = \arg\max_{x} \frac{det(W^T S_b W)}{det(W^T S_w W)} \tag{2.10}$$

Siendo S_b la matriz de dispersión entre clases y S_w la matriz de dispersión intraclases, las cuales se definen en la ecuación 2.11, donde m es la media global de todas las muestras, m_i, N_i y x_j, son la media, el número de elementos y el elemento jotaésimo de la clase i, respectivamente.

$$\begin{aligned} S_b &= \sum_{i=1}^{K} N_i (m_i - m)^T (m_i - m) \\ S_w &= \sum_{i=1}^{K} \sum_{x_j \in C_i} (x_j - m_i)^T (x_j - m_i) \end{aligned} \tag{2.11}$$

Una vez llegado a este punto, W se obtiene extrayendo los vectores propios de la matriz $S_W^{-1} S_b$. Por definición, un vector propio de una matriz representa un subespacio invariante de una dimensión del espacio vectorial en el que se encuentra la matriz. Los vectores cuyos valores propios son distintos de cero son linealmente independientes. Así pues, cualquier espacio vectorial puede representarse en términos de combinaciones lineales de los vectores propios. Gracias a esto se puede obtener un conjunto no redundante de características, considerando a los vectores propios que tengan un valor propio mayor que cero Balakrishnama y Ganapathiraju (1998).

La técnica de LDA se ha utilizado en este trabajo para reducir la complejidad computacional, además de transformar los datos a unos con mayor separabilidad lineal. En la sección 4.4.2 se entrará más en detalle en cómo se ha utilizado esta técnica.

2.3. Reto eHealth Knowledge Discovery

Actualmente existen retos relacionados con las tareas de PLN en biomedicina, tal y como se mencionó en la sección 2.2.3.1. En esta sección se va a profundizar en el reto eHealth Knowledge Discovery[39], en concreto el del año 2021[40], el último que se celebró, ya que es el que se aborda en este trabajo.

[39] https://ehealthkd.github.io/
[40] https://ehealthkd.github.io/2021

Este reto propone las tareas de NER y RE en textos clínicos en español.

El reto se compone de tres tareas: reconocimiento de entidades, extracción de relaciones y una tarea que combina ambas, donde se parte de un texto sin entidades, en el cual se deben extraer tanto las entidades como las relaciones entre éstas de forma conjunta. En este trabajo sólo se abordará la tarea de NER y de RE por separado.

En esta sección se hablará de los modelos y enfoques utilizados por los participantes que obtuvieron los mejores resultados. La descripción detallada de las tareas, el conjunto de datos y la evaluación, se tratarán más adelante en los capítulos 3 y 5, específicamente en las secciones 3.1, 3.2 y 5.2 respectivamente.

2.3.1. Modelos utilizados en la tarea de NER

El equipo que obtuvo los mejores resultados en la tarea de NER fue PUCRJ-PUCPR-UFMG Pavanelli et al. (2021). El modelo que presentaron es un modelo basado en *transformers*, la versión plurilingüe de BERT (mBERT) Devlin et al. (2018). Presentaron una arquitectura *end-to-end*, que realiza ambas tareas de forma conjunta, tal y como se muestra en la Figura 2.22.

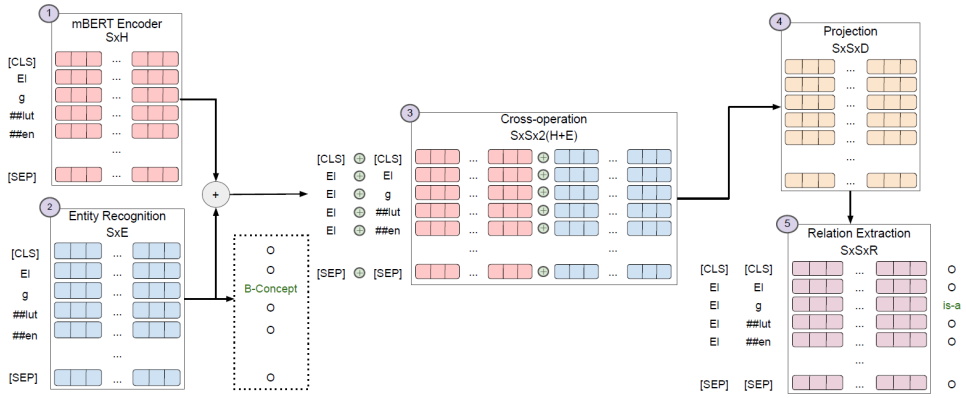


Figura 2.22: Sistema presentado por el equipo PUCRJ-PUCPR-UFMG en eHealth-KD 2021.

En primer lugar, se realiza una codificación del texto de entrada mediante el *encoder* mBERT (punto 1 de la Figura 2.22). Una vez hecho esto, los vectores se introducen en un clasificador *softmax* para realizar el reconocimiento de entidades (punto 2).

El equipo que obtuvo el segundo puesto en esta tarea fue Vicomtech Cuadros (2021), el cual presentó tres sistemas, que se evaluaron para cada una de las tareas. Para el caso de la tarea de NER, el que obtuvo mejores resultados fue el modelo con un clasificador conjunto, formado por una Red Neuronal que recibía los tokens de entrada y emitía conjuntamente predicciones para las dos tareas, en la Figura 2.23 se puede observar su arquitectura.

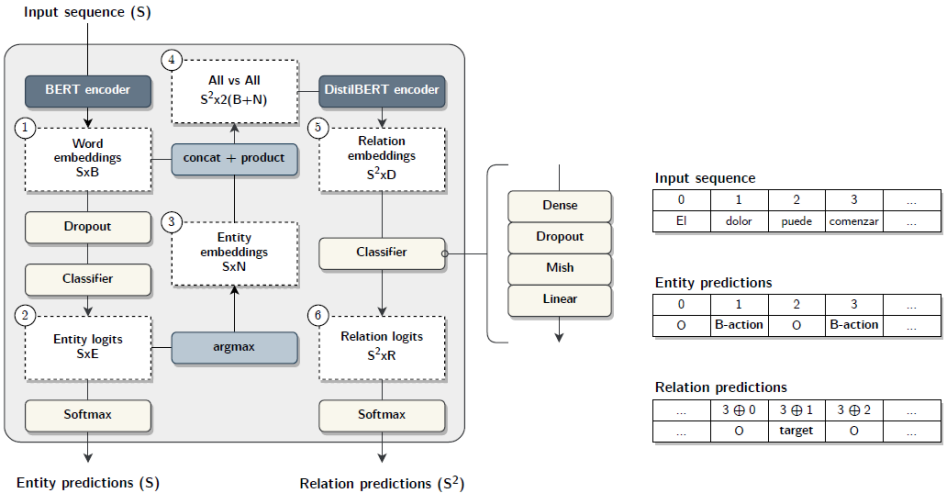


Figura 2.23: Sistema presentado por el equipo Vicomtech en eHealth-KD 2021.

En primer lugar, el texto se introducía en el *encoder* de un modelo de *transformer* IXAmBERT Otegi et al. (2020), el cual se trata de un modelo plurilingüe para el inglés, el español y el euskera. Una vez obtenidos los vectores codificados, se introducen en un clasificador. Cabe destacar que este equipo también realizó una prueba aplicando el modelo de *transformer* BETO Canete et al. (2020), un modelo entrenado con grandes conjuntos de textos en español, pero los resultados con IXAmBERT fueron ligeramente mejores.

El tercer equipo en conseguir los mejores resultados fue IXA Andrés (2021). El sistema presentado se diseñó como un *pipeline* de clasificadores, cada uno de ellos ajustados independientemente para cada subtarea. Para la tarea de NER, se codificaron los datos utilizando un modelo de *transformer* XML-RoBERTa Conneau et al. (2019). Como clasificador se utilizó una Red Neuronal *Feed-Forward-Network* (FNN), que clasificaba cada token con su entidad correspondiente.

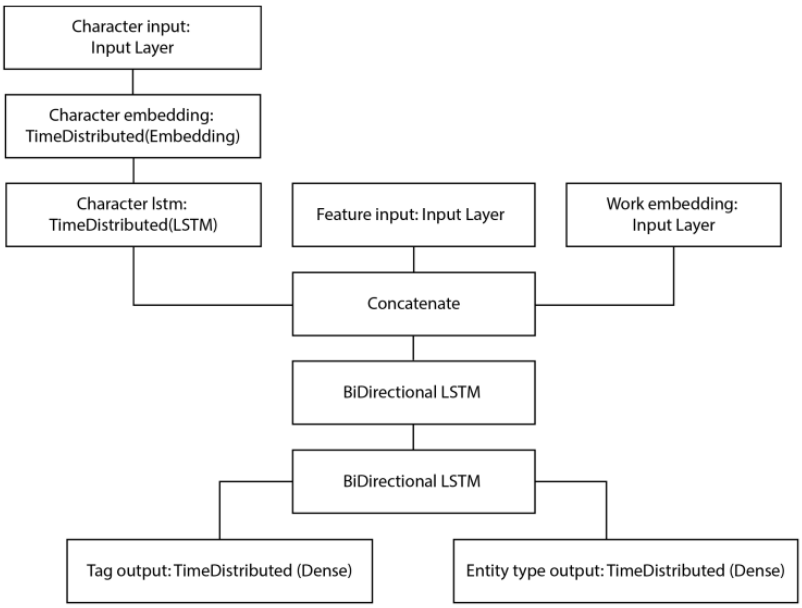


Figura 2.24: Sistema presentado por el equipo Vicomtech en eHealth-KD 2021.

Los tres equipos abordaron la tarea de NER y RE de forma conjunta, pero el equipo UM-MMM Monteagudo-Garcıa et al. (2021) alcanzó el cuarto puesto en la tarea de NER abordándola de forma independiente. Este equipo presentó un modelo de Redes Neuronales construido con varias capas apiladas de LSTM y LSTM bidireccionales. El modelo recibió tres entradas, la primera fueron las entidades a nivel de carácter, que posteriormente pasarían a una capa de *embedding*, la segunda fue una concatenación de las *POS tags* de las palabras y los lemas de éstas (extraídos mediante lematización y etiquetado de parte del discurso), y la tercera entrada serían los vectores extraídos del modelo BERT. Dicha arquitectura se muestra en la Figura 2.24 Monteagudo-Garcıa et al. (2021).

2.3.2. Modelos utilizados en la tarea de RE

Los equipos que obtuvieron los tres mejores resultados fueron IXA, Vicomtech y uhKD4 Alfaro-González et al. (2021). El equipo IXA, como se indicó en la sección anterior 2.3.1, realizó dos clasificadores distintos para cada tarea. Para el caso de la tarea de RE, se realizó la misma arquitectura que para la tarea NER.

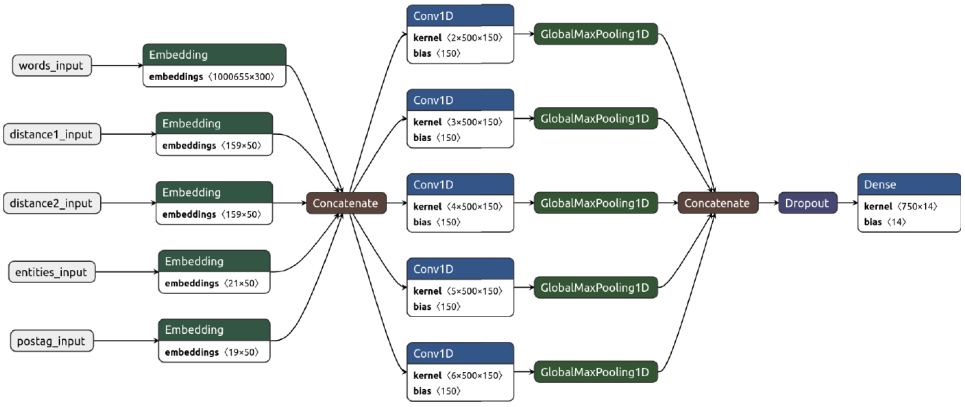


Figura 2.25: Sistema presentado por el equipo uhKD4 en eHealth-KD 2021 para la tarea de RE.

El segundo equipo en obtener los mejores resultados, Vicomtech, utilizó un modelo conjunto para ambas tareas, cuya arquitectura se mostró en la Figura 2.23. Para la tarea de RE, se recibe como entrada los tokens codificados por el modelo de *transformer*, así como las entidades *gold standard* que se proporcionan en el reto, para que el modelo se enfoque exclusivamente en la tarea de RE, en lugar de la tarea conjunta. Dada la entrada, se obtienen todas las combinaciones de pares de tokens posibles, para posteriormente introducirlos en un modelo DistilBERT Sanh et al. (2019). Una vez codificados estos pares, se pasan finalmente a un clasificador.

Por último, el tercer equipo en conseguir los mejores resultados fue uhKD4. A diferencia de los dos sistemas anteriores que abordaban las tareas de forma conjunta, el sistema presentado por uhKD4 se basó en dos arquitecturas independientes basadas en DL, una para cada tarea. La tarea de RE se aborda como un problema de clasificación multiclase utilizando una CNN. La arquitectura de la red se muestra en la Figura 2.25.

2.4. Valoración del estado del arte e identificación de la línea de trabajo

Tras realizar un estudio del estado del arte se ha comprobado que el reconocimiento de entidades y la extracción de relaciones es un tema que ha sido de interés desde los años 90 hasta la actualidad. Es un interés debido a que proporciona una gran ayuda a otro tipo de sistemas, como recuperación o

extracción de información, el resumen automático de textos y la traducción automática, entre otros.

Por este motivo, durante años se han desarrollado todo tipo de sistemas para realizar estas tareas, de forma general y específica para el ámbito biomédico. Sin embargo, para el dominio médico puede no ser suficiente, ya que es un dominio complejo, con múltiples entidades y palabras técnicas específicas del dominio, por lo que es complicado generalizar un modelo. Debido a que se trata de un dominio complejo, es importante centrar esfuerzos en el mismo, tratando de encontrar soluciones a cuestiones aún sin resolver dentro de NER y RE en textos biomédicos.

Durante el análisis del estado del arte, se han identificado principalmente tres problemas abiertos. Uno de ellos es la centralización de la mayoría de corpus de datos en inglés. La mayoría de modelos de *transformers* que se entrenan lo hacen con textos en inglés. Si bien es cierto que existen modelos entrenados con múltiples lenguas (modelos plurilingües, como mBERT), éstos son minoría en comparación con modelos entrenados en inglés. También existen modelos de español, como BETO, pero a pesar de ello, están entrenados con textos muy generales, por lo que no están del todo ajustados al dominio médico que estamos tratando.

Otro de los problemas encontrado, es que la mayoría de trabajos realizados de NER y RE biomédico, no utilizan técnicas de *Transfer Learning* (introducido en la sección 2.2) con otros modelos del ámbito biomédico, si no que se limitan a entrenar modelos generalistas, con el conjunto de datos que se utilice en cuestión, lo que muy posiblemente esté limitando la representación de términos del ámbito médico o biológico.

Por último, existe un problema de desbalance entre las clases sin entidad para el caso de NER y sin relación para el caso de RE, ya que lo más común es que las palabras no sean una entidad, o no formen parte de una relación. En el caso de RE es más grave, ya que en las relaciones se dan entre los pares de entidades de una oración, y la mayoría de estos no forman ninguna relación, lo que deja en una proporción muy baja a los pares con relación, especialmente si se trata de una frase larga. Este problema y algunas de sus posibles soluciones ya han sido nombrados anteriormente en las secciones 2.2.6 y 2.2.7, mientras que en la sección 4.4.2 se profundizará en cómo se han utilizado en este trabajo.

Capítulo 3

Caso de estudio

En este capítulo se va a describir en profundidad el caso de estudio con la problemática que se ha de resolver.

3.1. Descripción del caso de estudio

El caso a resolver se trata de la tarea IberLEF eHealth Knowledge Discovery Challenge 2021[1]. El eHealth-KD 2021 propone modelar el lenguaje humano en un escenario en el que los documentos digitales médicos en español puedan ser legibles para una máquina desde un punto de vista semántico. Con esta tarea, se espera fomentar el desarrollo de tecnologías de software para extraer automáticamente una gran variedad de conocimiento de los documentos médicos escritos en lengua española.

La estructura semántica propuesta modela cuatro tipos de entidades. Cada una de ellas representa una interpretación semántica específica, y hacen uso de trece relaciones semánticas entre ellas, que se explicará en las secciones posteriores 3.1.1 y 3.1.2 Piad-Morfis et al. (2021).

El reto se compone de tres subtareas[2]: la primera subtarea trata sobre NER; la segunda subtarea trata sobre RE basándose en las entidades ya existentes en el propio conjunto de datos. La última tarea es una combinación de ambas, debiendo extraer las entidades y las relaciones de forma conjunta. En este trabajo sólo se abordan las tareas de NER y RE por separado.

[1] https://ehealthkd.github.io/2021
[2] https://ehealthkd.github.io/2021/tasks

3.1.1. Subtarea A: Reconocimiento de entidades

Dada una lista de documentos con frases extraídas de documentos clínicos escritos en español, cuya explicación detallada está en la sección 3.2, el objetivo de esta subtarea es identificar todas las entidades por documento y sus tipos. Estas entidades son todos los términos relevantes (de una o varias palabras) que representan elementos semánticamente importantes en una frase. La Figura 3.1 muestra las entidades relevantes que aparecen en un conjunto de frases de ejemplo, con sus respectivos tipos.

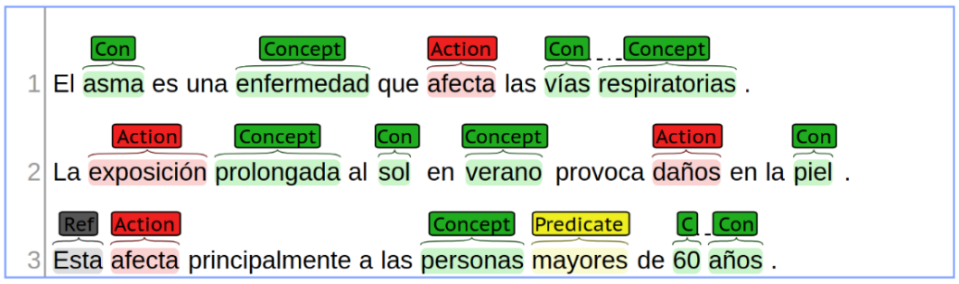


Figura 3.1: Entidades detectadas en una frase de ejemplo.

Es importante tener en cuenta que algunas entidades, como "vías respiratorias" y "60 años", abarcan más de una palabra. Las entidades siempre estarán formadas por una o más palabras completas, es decir, no un prefijo o un sufijo de una palabra, y nunca incluirán símbolos de puntuación circundantes, paréntesis, etc. Dicho esto, existen cuatro tipos de entidades:

- **Concept**: identifica un término, concepto o idea relevante en el dominio de la frase.
- **Action**: identifica un proceso o modificación de otras entidades. Puede indicarse mediante un verbo o una construcción verbal, como por ejemplo "afecta", pero también mediante sustantivos, como "exposición", donde denota el acto de exponerse al Sol, y "daños", donde denota el acto de dañar la piel, tal y como se observa en la Figura 3.1. También puede utilizarse para indicar relaciones funcionales no verbales, como "padre", entre otras.
- **Predicate**: identifica una función o filtro de otro conjunto de elementos, que tiene una etiqueta semántica en el texto, como "mayores",

y se aplica a una entidad, como “personas” con algunos argumentos adicionales como “60 años”.

- **Reference**: identifica un elemento textual que se refiere a una entidad, de la misma frase o de otra, que puede indicarse mediante pistas textuales como “esta” o “aquella”.

3.1.2. Subtarea B: Extracción de relaciones

La subtarea B continúa a partir del resultado de la subtarea A, relacionando las entidades detectadas y etiquetadas en el documento de entrada. El objetivo de esta subtarea es reconocer todas las relaciones semánticas relevantes entre las entidades reconocidas. En la Figura 3.2 se puede ver un ejemplo de ocho de las trece relaciones.

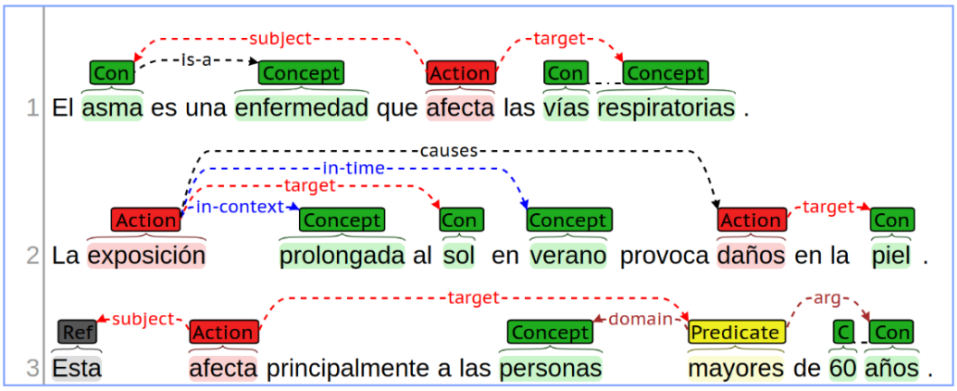


Figura 3.2: Entidades y sus relaciones detectadas en una frase de ejemplo.

Las relaciones semánticas se dividen en diferentes categorías, las cuales indican siempre una relación binaria entre dos entidades:

- **Relaciones generales**: relaciones de propósito general entre dos conceptos. Las cuales son:
  - **is-a**: indica que una entidad es un subtipo, instancia o miembro de la clase identificada por la otra.
  - **same-as**: indica que dos entidades son semánticamente iguales.
  - **has-property**: indica que una entidad tiene una determinada propiedad o característica.
  - **part-of**: indica que una entidad es parte constitutiva de otra.

- **causes**: indica que una entidad provoca la existencia o aparición de otra.
- **entails**: indica que la existencia de una entidad implica la existencia o aparición de otra.

- **Relaciones contextuales**: permiten refinar una entidad añadiendo modificadores a ésta. Las cuales son:
  - **in-time**: indica que algo existe, ocurre o se circunscribe a un marco temporal, como por ejemplo: "exposición" *in-time* "verano".
  - **in-place**: indica que algo existe, ocurre o se circunscribe a un lugar o localización.
  - **in-context**: indica un contexto general en el que ocurre algo, como un modo, una manera o un estado, como en "exposición" *in-context* "prolongada".
- **Roles de acción**: indican qué rol desempeñan las entidades relacionadas con una *Action*. Las cuales son:
  - **subject**: indica quién realiza la acción, por ejemplo: el "asma" *afecta* [...]
  - **target**: indica quién recibe el efecto de la acción, por ejemplo: [...] *afecta* las "vias respiratorias". Las acciones pueden tener varios sujetos y objetivos, en cuyo caso la interpretación semántica es que la unión de los sujetos realiza la acción sobre cada uno de los objetivos.
- **Roles del predicado**: indica qué papel desempeñan las entidades relacionadas con un *Predicate*. Las cuales son:
  - **domain**: indica la entidad principal sobre la que se aplica el predicado.
  - **arg**: indica una entidad adicional que especifica un valor para que el predicado tenga sentido. La semántica exacta de este argumento depende de la semántica de la etiqueta del predicado, por ejemplo en "mayores" de "60 años", donde la etiqueta "mayores" indica que "60 años" es una cantidad, que restringe la edad mínima para que el predicado sea verdadero.

3.2. Análisis del conjunto de datos

El conjunto de datos con los que se va a entrenar el modelo es proporcionado por eHealth en su repositorio[3]. En primer lugar, se deberá entrenar el modelo con el conjunto de entrenamiento, el cual está formado por 1200 frases extraídas de MedlinePlus[4], y 300 frases extraídas de Wikinews[5].

En segundo lugar, se proporciona un conjunto de desarrollo, que se permite utilizar también para entrenar los modelos, y así poder realizar el *fine-tuning* o ajustado. Este conjunto de desarrollo, está formado por 25 frases de MedlinePlus, 25 frases de Wikinews, y añade 50 frases extraídas del corpus CORD-19 Wang et al. (2020), el cual cuenta con textos y artículos relacionados con COVID-19, SARS-CoV-2 y coronavirus relacionados. Este corpus que no se encuentra en el conjunto de entrenamiento.

Además, la mayoría de las frases están en español, pero también se incluye un pequeño conjunto de frases en inglés, las cuales corresponden a las extraídas del corpus del dominio biomédico CORD-19, para evaluar la generalización entre dominios, aunque dicha evaluación no se aborda en este trabajo.

3.2.1. Análisis del conjunto de datos de entrenamiento

A continuación, se va a realizar un análisis de los datos de entrenamiento. En la Figura 3.3 se muestra una gráfica con la frecuencia de cada entidad. Como se puede observar, la entidad más frecuente es *Concept*, mientras que la menos frecuente es *Reference*, por lo que es muy probable que algunos de los modelos sean incapaces de detectarla.

Por otra parte, es importante también analizar la frecuencia de las relaciones que existen en el *dataset* de entrenamiento. En la Figura 3.4 se puede ver que la clase mayoritaria es *target*, y que las relaciones menos representadas son *has-property*, *same-as*, *entails* y *part-of*. Es importante tener la frecuencia en cuenta, ya que sirve para determinar que clases minoritarias deben ser reforzadas, por ejemplo, agregando más peso a dichas clases dentro del modelo.

[3] https://github.com/ehealthkd/corpora
[4] https://medlineplus.gov/
[5] https://www.wikinews.org/

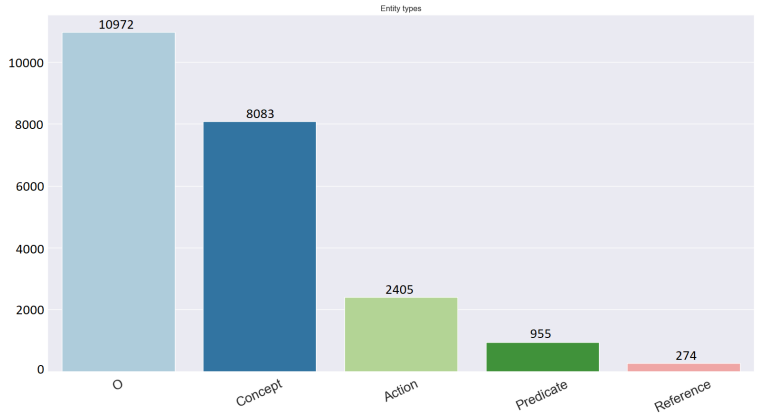


Figura 3.3: Frecuencia de aparición de las entidades en el conjunto de entrenamiento.

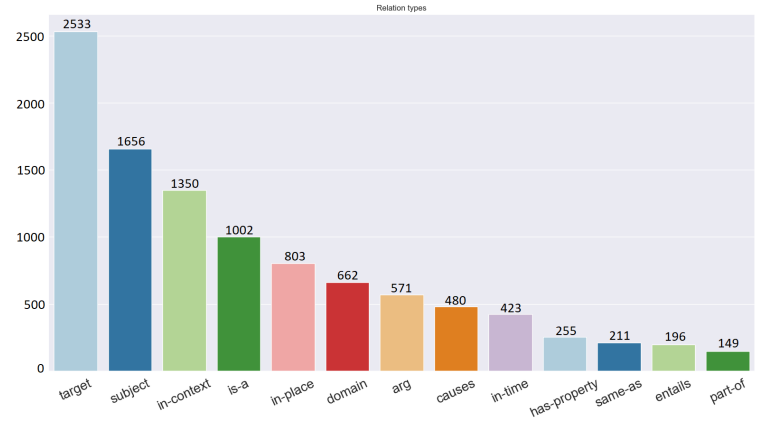


Figura 3.4: Frecuencia de aparición de las relaciones en el conjunto de entrenamiento.

También hay que tener en cuenta que la frecuencia de las relaciones que se muestra en la Figura 3.4 es ínfima al lado de la frecuencia que tienen los pares de entidades sin relación, ya que esta frecuencia asciende a casi de 550 mil pares sin relación, es decir, un 98.18 % de los pares de palabras posibles no tienen relación. Lo que de nuevo supone un desafío para los modelos, por el gran desequilibrio que supone, esto se tratará de mitigar durante el entrenamiento de los modelos.

Otro análisis interesante que se puede realizar es estudiar qué palabras son las más frecuentes en el corpus. En la Figura 3.5 se muestran las 10 palabras más frecuentes en el conjunto de entrenamiento. Se puede observar

que son palabras generales, sin terminología médica, pero algunas de ellas son palabras relacionadas con el dominio biomédico, como “enfermedad”, “sangre”, “cuerpo” o “tratamiento”. Esto puede jugar un papel fundamental a la hora de utilizar *Transfer Learning* con modelos entrenados en este dominio.

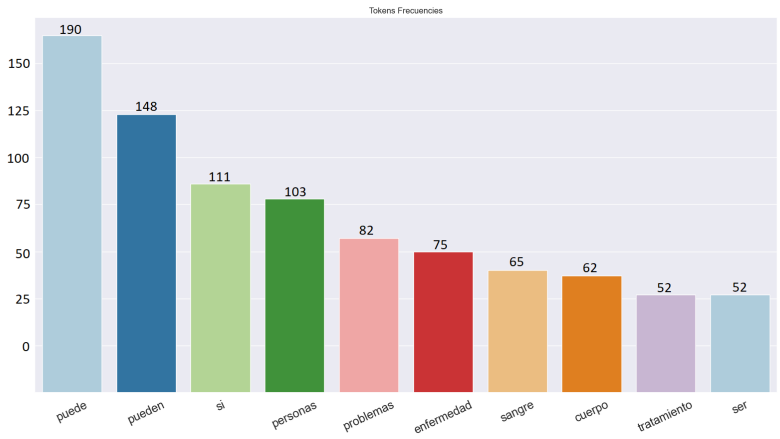


Figura 3.5: Las diez palabras más frecuentes en el conjunto de entrenamiento, eliminando *stop-words*.

3.2.2. Análisis del conjunto de datos de desarrollo

Una vez realizado el análisis del conjunto de entrenamiento, es importante también hacerlo para el conjunto de desarrollo, ya que los modelos se ajustarán con este conjunto. De la misma forma que en la sección anterior, en la Figura 3.6 se muestran las frecuencias de los tipos de entidad para éste.

Se puede observar que el conjunto de desarrollo tiene la misma estructura de entidades en cuanto a frecuencia se refiere, lo que significa que no se puede contar con que el conjunto de desarrollo ayude a la hora de balancear las clases. Respecto a las relaciones, en la Figura 3.7 se muestran las relaciones de este conjunto. En este conjunto aparece un tipo de relación que no está en el conjunto de entrenamiento, *has-part*. Esta relación viene introducida por el corpus CORD-19, como se comentó al principio de esta sección 3.1. Esta relación también está incluida en la colección de test, por lo que hay que reentrenar los modelos con este conjunto de desarrollo.

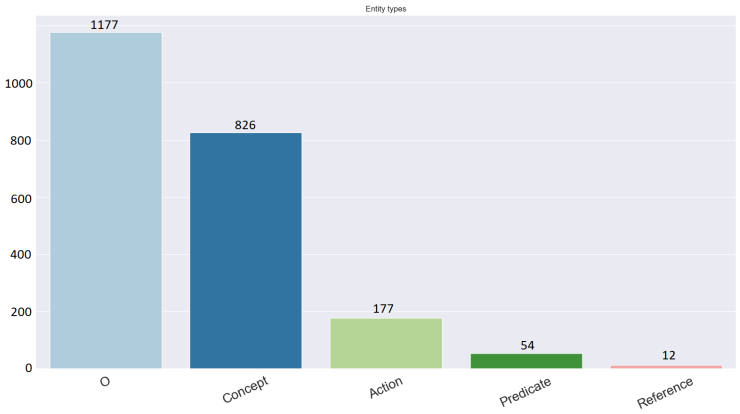


Figura 3.6: Frecuencia de aparición de las entidades en el conjunto de desarrollo.

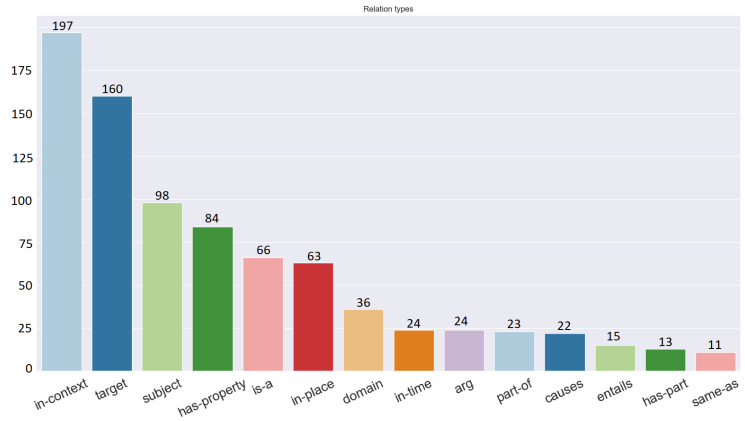


Figura 3.7: Frecuencia de aparición de las relaciones en el conjunto de entrenamiento.

También destaca el hecho de que la frecuencia de las relaciones es distinta con respecto al conjunto de entrenamiento, además de la adición de esta nueva relación. Esto no es necesariamente malo, ya que puede ayudar a que el conjunto de datos con el que está entrenado el modelo final tenga ligeramente menor desbalance de relaciones.

Por último, como en la sección anterior, es interesante observar qué palabras son las más recurrentes en este conjunto de desarrollo. En la Figura 3.8 se muestran las 10 palabras más frecuentes del mismo. Lo principal que se puede notar es que hay muchas referencias a la enfermedad COVID-19, y el virus que la produce, SARS-CoV-2. Esto no es de extrañar, ya que el corpus

más representativo de este conjunto es el CORD-19. Además se observan palabras en inglés, como se ha comentado anteriormente en la sección 3.1.

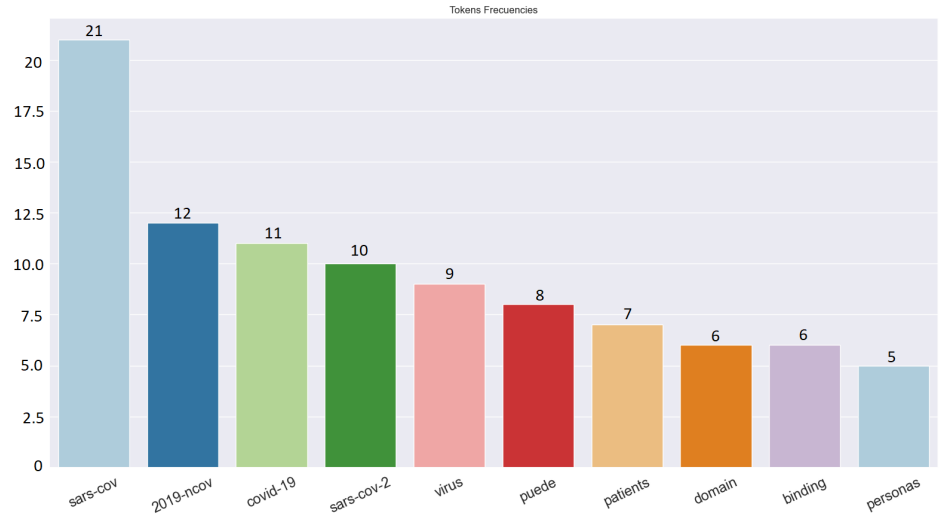


Figura 3.8: Las diez palabras más frecuentes en el conjunto de desarrollo, eliminando *stop-words*.

Tras este análisis, se puede ver que existen varias diferencias sobre el tipo de datos entre los conjuntos de entrenamiento y desarrollo, lo que aporta generalidad al modelo, y facilita el desempeño con el conjunto de test.

3.2.3. Análisis de la influencia de las entidades en las relaciones

La tarea de RE también puede verse influenciada en gran medida por el tipo de entidades que forman parte de la relación. Pueden existir ciertas entidades que son más propensas a formar ciertas relaciones. Por este motivo, se ha decidido analizar qué parejas de entidades son más comunes en según que relaciones.

En primer lugar, se extrajeron la cantidad de relaciones de cada tipo que existen entre según qué pares de entidades, en la Figura 3.9 se muestran los pares de entidades, y las líneas representan la cantidad de muestras de un tipo específico de relación, para ese par. La etiqueta definida para pares sin relación, es la letra O. Las entidades están representadas por su inicial, siendo A-A la pareja de entidades *Action* y *Action*, C-P *Concept* y *Predicate*, y así sucesivamente. Por cuestiones de visualización, en esta Figura 3.9 sólo

se muestran pares de entidades, ya que en caso de que una de las palabras del par no sea una entidad, siempre se trata de un par sin relación.

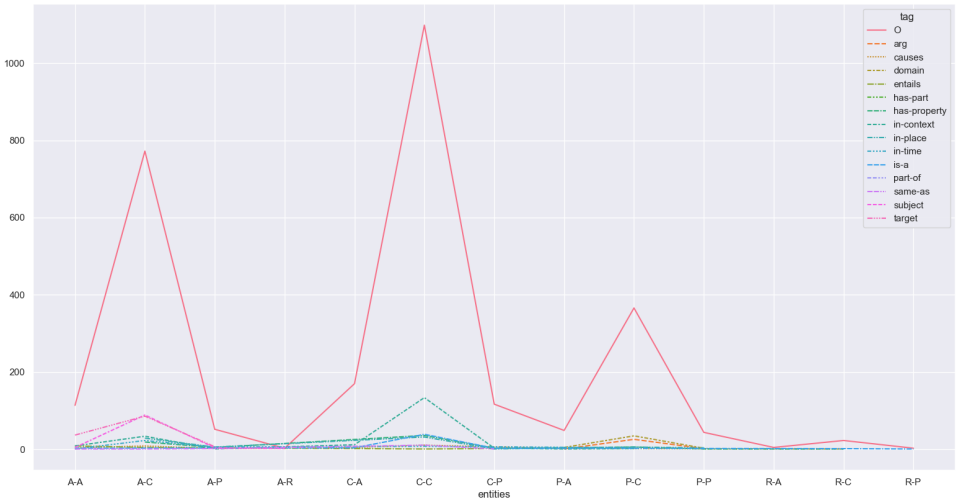


Figura 3.9: Cantidad de relaciones por cada par de entidades.

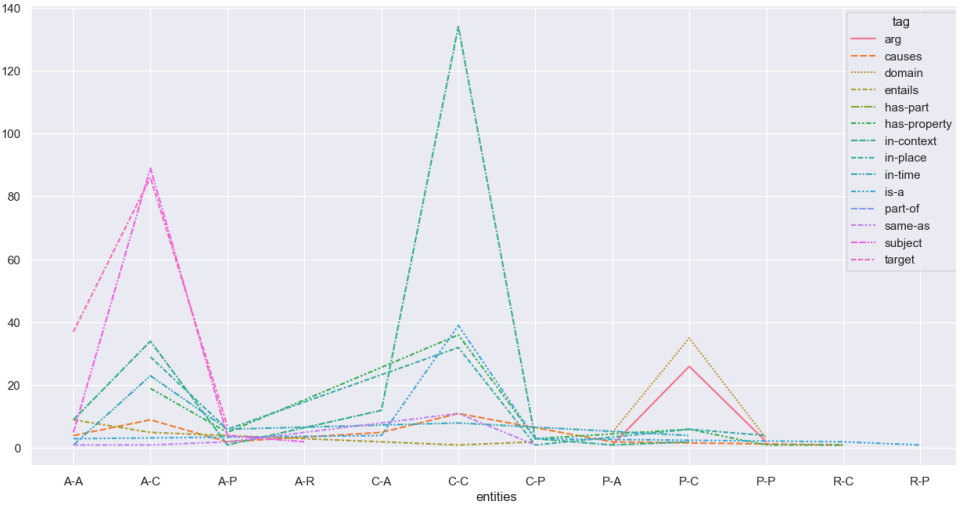


Figura 3.10: Cantidad de relaciones por cada par de entidades sin incluir la clase sin relación.

Como se puede observar, la cantidad de pares sin relación es muy alta en comparación con el resto. Otra cosa destacable es que hay ciertos pares de entidades que son más frecuentes que otros de forma general, y estos son: el par *Action-Concept*, *Concept-Concept* y *Predicate-Concept*. El problema de esto es que también se ve que la mayoría de relaciones se juntan en

esos tres pares, lo que puede dificultar la diferenciación entre las distintas relaciones. En la Figura 3.10 se muestra la cantidad de relaciones para cada par, teniendo sólo en cuenta las relaciones, sin mostrar los casos sin relación.

En esta Figura 3.10 se sigue observando que hay un predominio de pares de entidades que acogen a la mayoría de las relaciones, aunque en este caso se aprecian más diferencias. Por ejemplo, en la relación *subject* y *target* se aprecia una predominancia del par *Action-Concept*. También el par *Concept-Concept* está muy relacionado con la entidad *in-context*, y por último, *Predicate-Concept* con las relaciones *domain* y *arg*.

Hay que tener en cuenta que esta Figura 3.10 esta sesgada por la cantidad de datos que hay con cada relación, por lo que las relaciones con más ocurrencias coinciden con los tres picos principales de la figura. Para tratar de tener una visión más relativa, en la Figura 3.11 se muestra el porcentaje de cada una de las relaciones para cada uno de los pares.

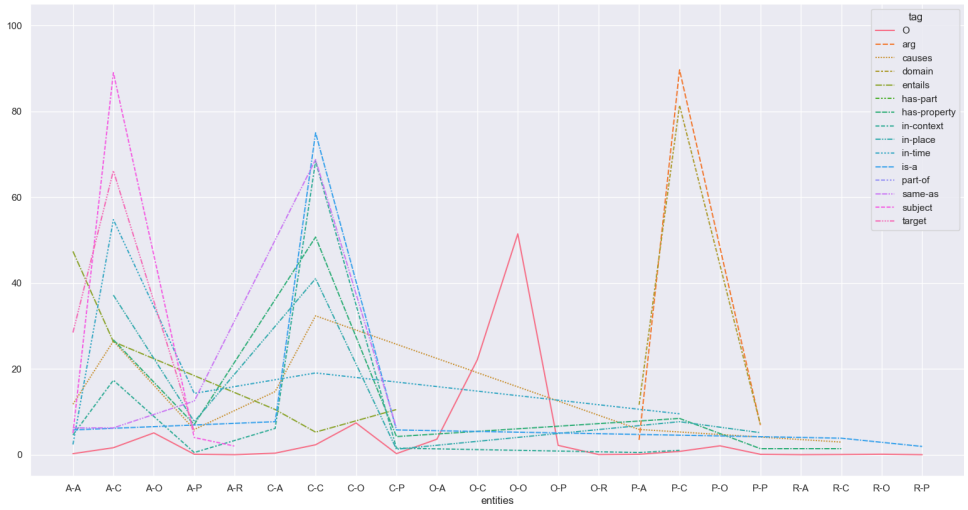


Figura 3.11: Frecuencia de las relaciones por cada par de entidades.

En esta Figura 3.11, se aprecia más que existe una variedad de relaciones entre las distintas entidades. Aún así, se aprecian ciertos patrones, como por ejemplo que las relaciones *domain* y *arg* sólo se dan pares que contengan una entidad de tipo *Predicate*. También se confirma que los pares sin relación se dan principalmente cuando una de las palabras o ambas no tiene entidad, aunque si que se aprecia que también se da entre pares con entidades, como es el caso de *Concept-Concept* o *Action-Concept*. También en esta Figura 3.11 se aprecian los picos que se observaban en las figuras anteriores entre

los pares *Action-Concept* y *Concept-Concept*.

Tras este análisis, es posible sacar ciertos patrones que pueden servir de utilidad a la hora de desarrollar una propuesta para esta tarea, como el hecho de poner más peso en ciertas relaciones, según que tipo de entidades estén involucradas, esto se explicará más en detalle en la sección de la implementación de la tarea de RE, la sección 4.4.3.

Capítulo 4

Propuesta

A la vista de las necesidades encontradas durante la valoración del estado del arte, lo que se propone es la resolución del reto de eHealth Knowledge Discovery, concretamente las tareas de NER y RE, intentando mejorar los resultados de los anteriores participantes. En las sucesivas secciones se detallará la propuesta realizada en este trabajo.

4.1. Propuesta detallada

Lo que se propone para la resolución de la tarea de NER es construir dos arquitecturas de modelos distintas. La primera de ellas consiste en aplicar un modelo de *transformer* preentrenado para codificar la entrada de datos y posteriormente introducir esos en un clasificador, para así realizar el *Transfer Learning* de modelos ya entrenados previamente. La selección de los modelos utilizados se detalla en la sección 4.2. La diferencia entre esta propuesta y la realizada por otros sistemas del estado del arte, es que los modelos *transformer* utilizados estarán previamente entrenados con texto del dominio biomédico en español, por lo que se espera que esto proporcione buenos resultados.

La segunda consiste en reentrenar uno de estos modelos preentrenados (tanto con texto general como con textos del ámbito médico) con los datos de entrenamiento. Con esto se pretende evaluar el uso de modelos previamente entrenados en texto biomédico, y si esto supone una gran mejoría en los resultados o por el contrario es más conveniente entrenar el modelo desde cero con los datos de entrenamiento.

Además de esto, y a diferencia de las propuesta realizadas anteriormente

por otros sistemas, se propone utilizar las técnicas de *Data Augmentation* para tratar de mitigar el desbalanceo entre las distintas clases de entidades y la clase sin entidad, aplicando las técnicas de aumento de datos textuales (sección 2.2.6.1) a los modelos de *transformer* que se reentrenen.

Para la tarea de RE, el primer enfoque que se propone es realizar dos clasificadores, SVM y Redes Neuronales, para realizar la clasificación de los pares de entidades. Como entrada el clasificador recibirá los vectores de características resultantes de aplicar un modelo de *transformer* preentrenado a los datos de entrada, tal y como se realizan para la tarea de NER. Los modelos utilizados para ello son los mismo utilizados en la tarea de NER, descritos en la sección 4.2. De esta forma se aporta también en esta tarea la novedad de utilizar estos modelos ya entrenados en texto biomédico en español.

Como se trata de un problema con muchas clases, además de que existe una clase sobre-representada, se propone un segundo enfoque que consiste en utilizar un segundo clasificador de Redes Neuronales (NN) entrenado sólo con los datos que tengan relación, para así repuntuar con este segundo clasificador las decisiones tomadas por el primer clasificador, pretendiendo así discriminar mejor entre las entidades detectadas.

Como en el caso de NER, una de las aportaciones de esta propuesta es la utilización de técnicas de aumento de datos. En esta ocasión se aumentarán las muestras en el espacio de características antes de introducirlas en el clasificador, esperando mejorar así la representatividad de las entidades con relación.

Además de esto, al tratarse de un modelo que recibe como entrada dos vectores de características (uno por palabra), también se plantea utilizar un modelo de reducción de dimensionalidad para que además de reducir las dimensiones y facilitar el entrenamiento, aumente la separabilidad de los datos, tal y como se explica en la sección 2.2.7.

4.2. Selección de modelos de preentrenados

La selección de modelos determina en gran medida los resultados obtenidos. Lo primero que hay que tener en cuenta es la lengua con la que están entrenados los modelos, en este caso en español. Lo segundo más importante es con que tipo de datos ha sido entrenado, ya que en este trabajo se quiere

explorar el uso de un modelo entrenado con textos del dominio biomédico. Por ello, se proponen una serie de modelos, algunos de ellos más generales y otros más específicos del dominio.

De los cinco modelos que se presentan a continuación, los dos primeros (secciones 4.2.1 y 4.2.1) son generales para el español. Los últimos tres modelos (secciones 4.2.3, 4.2.4 y 4.2.5) son modelos preentrenados con textos de naturaleza biomédica y clínica en español, proporcionados por el Plan de Tecnologías del Lenguaje del Gobierno de España[1], como se comentó anteriormente. Con estos modelos se pretender realizar *Transfer Learning* y aprovechar su preentrenamiento para aplicarlo a las tareas que se están abordando, ya que no se dispone de material suficiente para entrenar un modelo de tales características.

4.2.1. BERT multilingual base model

El primer modelo con el que se va a entrenar es el modelo plurilingüe de BERT (mBERT) Devlin et al. (2018). Este modelo ha sido entrenado en 104 idiomas con datos de Wikipedia, y fue entrenado para realizar modelado lingüístico enmascarado (*Masked Language Modeling*) (MLM). También es sensible a las mayúsculas y minúsculas y distingue entre inglés y castellano. El modelo puede encontrarse en el repositorio oficial Google Research[2], y en la librería HuggingFace[3].

mBERT es un modelo de *transformers* preentrenado en un gran corpus de datos plurilingües de forma autosupervisada. Esto significa que fue preentrenado sólo con los textos en bruto, sin que éstos fueran etiquetados de ninguna manera.

De este modo el modelo aprende una representación interna de las lenguas en el conjunto de entrenamiento que puede utilizarse para extraer características útiles para tareas posteriores, por lo que es factible utilizar este modelo para entrenar un clasificador utilizando las características producidas por el modelo mBERT como entradas.

Respecto a las características de este modelo, son las siguientes: 12 capas de atención, 768 estados ocultos (lo que significa que los vectores de características de la entrada tendrán 768 dimensiones), 12 cabezas de atención y

[1] https://plantl.mineco.gob.es/tecnologias-lenguaje/PTL/Paginas/plan-impulso-tecnologias-lenguaje.aspx

[2] https://github.com/google-research/bert

[3] https://huggingface.co/bert-base-multilingual-cased

110 millones de parámetros.

4.2.2. BETO: Spanish BERT

El segundo modelo a utilizar es BETO[4] Canete et al. (2020), un modelo BERT entrenado con un gran corpus español. El modelo fue entrenado con textos de Wikipedia, Wikinews, Wikiquotes en español, además de fuentes del proyecto OPUS con textos en español, textos de las Naciones Unidas, charlas TED, publicaciones del gobierno, entre otras fuentes[5]. Modelos como BERT-base y mBERT cuentan con 12 capas de atención, 768 estados ocultos, 12 cabezas de atención y 110 millones de parámetros. Este modelo tiene el mismo número de parámetros y capas de atención, pero a diferencia de estos, cuenta con 1024 estados ocultos y 16 cabezas de atención.

Tanto este modelo como mBERT son modelos entrenados con datos generales sin ser aplicados al dominio biomédico. Se ha decidido seleccionarlos para poder evaluar las diferencias entre utilizar un modelo preentrenado con textos generales y uno preentrenado con textos del dominio biomédico.

4.2.3. Biomedical language model for Spanish

El modelo RoBERTa$_{Bio}$ se trata de un modelo lingüístico biomédico en el español[6] Carrino et al. (2021b). Es un modelo basado en RoBERTa Liu et al. (2019) entrenado en un corpus biomédico en español, el cual es recogido de varias fuentes que se explicarán a continuación. Al igual que mBERT el preentrenamiento de este modelo se basa en el modelado lingüístico enmascarado, siguiendo el enfoque empleado para el modelo base RoBERTa con los mismos parámetros que éste, en este caso: 12 capas de atención, 768 estados ocultos, 12 cabezas de atención, 125 millones de parámetros.

El corpus de entrenamiento está compuesto por varios corpus biomédicos en español recogidos de corpus disponibles públicamente, algunos de ellos son: Spanish Biomedical Crawled Corpus Carrino et al. (2021a), SciELO Spain[7] y artículos sobre ciencias de la vida en Wikipedia, entre otros.

[4] https://huggingface.co/dccuchile/bert-base-spanish-wwm-cased
[5] https://github.com/josecannete/spanish-corpora
[6] https://huggingface.co/PlanTL-GOB-ES/roberta-base-biomedical-es
[7] https://scielo.isciii.es/scielo.php

4.2.4. Biomedical-clinical language model for Spanish

El modelo RoBERTa$_{Clinical}$ se trata de un modelo biomédico y clínico en español[8]. Este modelo tiene la misma estructura que el modelo anterior, y ha sido entrenado con corpus biomédicos en español, pero en este caso además de esto, se utilizó un corpus con informes clínicos reales de más de 278 mil documentos y notas clínicas.

Además, estos modelos han sido evaluados para la tarea de NER con *datasets* biomédicos en español, como por ejemplo: PharmaCoNER Gonzalez-Agirre et al. (2019) o Cantemist Miranda-Escalada et al. (2020); y han obtenido muy buenos resultados.

4.2.5. Spanish RoBERTa-base biomedical for NER

Por último, el modelo RoBERTa$_{NER}$, está basado en RoBERTa y afinado para la tarea de NER[9]. Ha sido preentrenado utilizando el mayor corpus biomédico español conocido hasta la fecha, compuesto por documentos biomédicos, casos clínicos y documentos de historia clínica Electronic Health Records (EHR).

Este modelo es un afinado del modelo anterior, descrito en 4.2.3, para el *dataset* PharmaCoNER[10], anotado con entidades de sustancias, proteínas y compuestos. Por ello es interesante utilizarlo a pesar de no contar con los mismos tipos de entidades, ya que el modelo ha sido afinado para la tarea de NER.

4.3. Implementación de modelos NER

Tal y como se comentó en la propuesta, para abordar esta tarea se proponen dos arquitecturas de modelos distintas, las cuales se explicarán en las posteriores secciones junto con las técnicas de aumento de datos. En primer lugar, se explicará el preproceso de los datos antes de introducirlos en el modelo.

[8] https://huggingface.co/PlanTL-GOB-ES/roberta-base-biomedical-clinical-es
[9] https://huggingface.co/PlanTL-GOB-ES/bsc-bio-ehr-es-pharmaconer
[10] https://temu.bsc.es/pharmaconer/

4.3.1. Preprocesado de los datos

En primer lugar, se procede al preprocesado de los datos. Una vez leído el texto de los ficheros de entrada mediante el *script* proporcionado por el reto[11], se aplicó sobre las frases una serie de pasos, que se detallan a continuación:

- **Eliminación de caracteres no alfanuméricos**: se eliminaron los signos de puntuación (comas, puntos, signos de interrogación y exclamación, etc). Estos elementos, pueden provocar ruido y que el modelo no se centre tanto en las palabras.
- **Eliminación de tildes**: se eliminaron las tildes, y se dejó la palabra original sin tildes. Se eliminaron debido a que estos elementos son difíciles de procesar y también pueden causar ruido.

También se probó a aplicar tanto lematización como *stemming*, aunque los resultados no fueron satisfactorios, posiblemente debido a que había palabras con la misma raíz que podían pertenecer a entidades distintas, por lo que se dejaron las palabras originales.

Por último, se codificaron las etiquetas de las entidades siguiendo la codificación IOB explicadas en la sección 2.2.4. Esto es debido a que muchas entidades tienen más de una palabra y es necesario identificarlas todas. Cabe mencionar también, que se han codificado las etiquetas de clase (entidades) en valores numéricos del 0 a $N-1$, siendo N el número de clases (tipos de entidad).

4.3.2. Aumento de datos textuales

Tal y como se indicó la propuesta, para la tarea de NER se propone utilizar aumento de datos para tratar de mitigar el desbalance entre clases. En este caso, y dada la información obtenida durante el análisis de las entidades realizado en la sección 3.2.1, se plantea realizar un aumento de datos para las clases *Predicate* y *Reference* debido a que son las dos minoritarias.

La técnica de *Back Translation*, tal y como se explicó en la sección 2.2.6.1, se basa en utilizar un modelo de traducción para traducir la palabra que se pretende aumentar a otra lengua (en este caso inglés, ya que los modelos están más trabajados que en otras lenguas) y después volver a traducirla al

[11] https://github.com/ehealthkd/corpora/blob/master/scripts/anntools.py

castellano. El modelo utilizado para traducir del castellano al inglés, es el proporcionado por Language Technology Research Group at the University of Helsinki[12], un grupo a investigación PLN de la Universidad de Helsinki, disponible en la librería HuggingFace[13]. El modelo para traducir de nuevo del inglés al español es proporcionado también por este grupo[14].

Para la técnica de generación de sinónimos se utilizó una red Wordnet multi-idioma que soporta varias lenguas, incluido el castellano. Dicha red es Open Multilingual Wordnet[15]. La implementación fue proporcionada por la librería nlpaug[16].

En la tabla 4.1 se muestra la cantidad de datos en cada fase del proceso de aumento de datos. En la columna *Original* se muestran la cantidad de datos originales en el conjunto de entrenamiento, en *BackTranslation/Synonym* se muestra la cantidad de datos aumentada para esa clase utilizando una técnica u otra, ya que no se utilizan al mismo tiempo para poder estudiar las diferencias entre ambas. En la columna *Aumentado* se muestra la cantidad de datos en el conjunto de entrenamiento final para cada clase. También se puede observar que la frecuencia de las entidades *Predicate* y *Reference* en el conjunto aumentado es el doble que en el conjunto original.

	Cantidad de datos		
Entidad	Original	*Back Translation / Synonym*	Aumentado
Sin entidad	10972	0	10972
Concept	8083	0	8083
Action	2405	0	2405
Predicate	955	955	1910
Reference	274	274	548

Tabla 4.1: Cantidad de datos iniciales y aumentados para la tarea NER.

Para aumentar los datos se seleccionaron las palabras pertenecientes a dichas entidades y se generó por cada una de ellas una nueva palabra, es decir, se duplicaron las entidades de esas clases para aumentar su representatividad. En la Figura 4.1 se muestra la frecuencia de las entidades del conjunto de datos de entrenamientos una vez duplicadas.

Con esto se pretende aumentar la representación de los datos sin alterar

[12] https://blogs.helsinki.fi/language-technology/
[13] https://huggingface.co/Helsinki-NLP/opus-mt-es-en
[14] https://huggingface.co/Helsinki-NLP/opus-mt-en-es
[15] https://github.com/globalwordnet/OMW
[16] https://nlpaug.readthedocs.io/en/latest/augmenter/word/synonym.html

la predominancia de las clases más frecuentes, ya que lo más probable es que en los datos de test también exista mayor frecuencia de éstos. Este aumento se da con ambas técnicas de aumento de datos, tanto *Back Translation* como la generación de sinónimos. Esta acción se realiza antes de la extracción de características o de introducirse en el modelo.

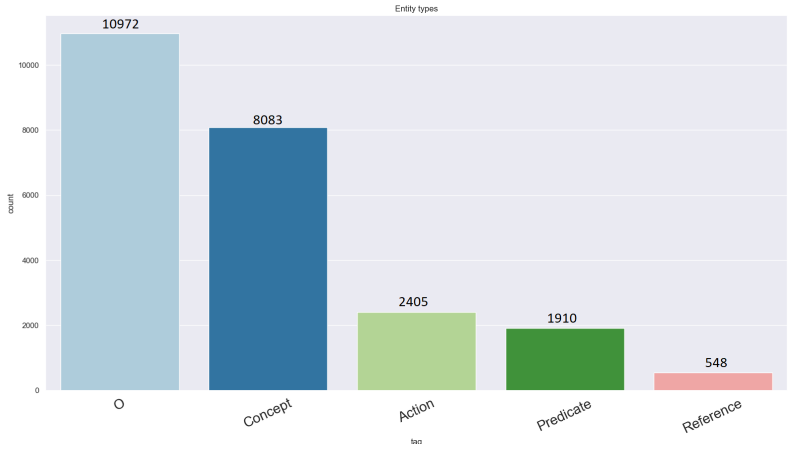


Figura 4.1: Frecuencia de aparición de las entidades en el conjunto de entrenamiento.

4.3.3. Extracción de características

Una vez se han preprocesado los datos es necesario aplicar técnicas de extracción de características, ya que los modelos de *Machine Learning* necesitan recibir vectores de características como entrada. Se seleccionaron una serie de técnicas de extracción de características, que se detallarán a continuación, pero los resultados de aplicar las mismas para cada modelo se verán en la sección 5.4.3. Todas las técnicas de extracción de características han sido explicadas previamente en la sección 2.2.2.

Hay que tener en cuenta que esta extracción de características solo aplica para clasificadores, ya que los modelos de *transformers* no requieren de este paso adicional, ya que calculan sus propias características.

4.3.3.1. *Frequency-Inverse Document Frequency*

La primera técnica de extracción de características es TF-IDF, la cual se explica en la sección 2.2.2. Esta técnica genera una matriz donde cada

fila representa cada palabra, y las columnas son el valor de la función TF-IDF para esa palabra en el conjunto de entrenamiento. Si se selecciona un subconjunto de frases de entrenamiento, tal que así:

> "La presencia del gen de células falciformes y otro normal se denomina rasgo drepanocítico.", "Entre los factores de riesgo están ser de raza afroamericana o el exceso de peso"

La matriz de características de estas dos frases quedaría tal y como se muestra a continuación, siendo las filas los tokens y las columnas el peso de los tokens en cada frase:

$$\begin{array}{l} \\ La \\ presencia \\ del \\ \vdots \\ exceso \\ de \\ peso \end{array} \begin{array}{c} \begin{array}{cc} phrase_1 & phrase_2 \end{array} \\ \begin{pmatrix} 0{,}272 & 0{,}000 \\ 0{,}272 & 0{,}000 \\ 0{,}272 & 0{,}000 \\ \\ 0{,}000 & 0{,}246 \\ 0{,}193 & 0{,}524 \\ 0{,}000 & 0{,}245 \end{pmatrix} \end{array}$$

4.3.3.2. *Word Embedding*

Otra de las técnicas utilizadas habitualmente para representar palabras son los *Word Embeddings*, un método que se basa en generar un vector para cada palabra del conjunto de datos basándose en los contextos en los que aparece en los textos. Dicha técnica se explicó en la sección 2.2.2. En esta ocasión se ha utilizado un modelo de *FastText*, entrenado previamente con textos biomédicos en español e inglés.

El modelo se ha obtenido del proyecto NLPMedTerm, el cual tiene como objetivo aportar a la comunidad investigadora recursos para el procesamiento del lenguaje natural en el dominio médico en español[17]. El proyecto fue finalizado por InterTalentum Posdoctoral Programme[18] en la Universidad Autónoma de Madrid, y proporciona varios recursos, entre ellos los vectores entrenados con el modelo *FastText*.

[17] http://www.lllf.uam.es/ESP/nlpmedterm_es.html

[18] https://congresos.fuam.es/fuamcongresos/intertalentum-posdoctoral-programme/home

El modelo ha sido entrenado siguiendo el algoritmo *Skip-Gram*, explicado anteriormente, generando vectores de 100 dimensiones. Los datos de entrenamiento del modelo provienen de dos fuentes, una de ellas es el corpus de textos de la Agencia Europea del Medicamento (European Medicines Agency) (EMEA) [19], con casi 14 millones de tokens, y la otra son artículos médicos del repositorio Scientific Electronic Library Online (SciELO)[20] con aproximadamente 25 millones de tokens.

Una vez se ha cargado el modelo, cada token es representado por su vector correspondiente del modelo, generando así vectores de características de 100 dimensiones.

También existen otros *embeddings* entrenados en texto biomédico en español Soares et al. (2019) generados a partir de los textos completos de artículos biomédicos de SciELO, los artículos de salud de Wikipedia y una concatenación de ambos corpus. Dichos *embeddings* fueron entrenados en *Word2Vec* y *FastText*[21] y no han sido utilizados en este trabajo.

4.3.4. Implementación de modelos de clasificación en NER

En primer lugar y a modo de *baseline*, se entrenarán una serie de clasificadores con las características explicadas en la sección anterior, para así tener unos primeros resultados a mejorar por los modelos *transformers*. Los clasificadores escogidos son SVM, *Random Forest* y Redes Neuronales, que se explicarán a continuación.

En segundo lugar, tal y como se comentó en la propuesta (sección 4.1), se implementarán estos mismos clasificadores, que recibirán como entrada los vectores de características extraídos de aplicar el modelo *transformer* preentrenado a las palabras de entrada.

Una vez se ha seleccionado el modelo con el que se va a realizar la clasificación, explicados en las posteriores secciones, se realizará un procesamiento del texto y una extracción de características utilizando el modelo de *transformer* preentrenado, y una vez obtenidos los *embeddings* que representen a cada palabra, pasárselos a un modelo o capa de clasificación. Por lo tanto, la arquitectura propuesta se muestra en la Figura 4.2. Es importante tener en cuenta que los modelos *transformer* tienen en cuenta el contexto de la frase,

[19] https://opus.nlpl.eu/EMEA.php
[20] https://scielo.org/
[21] https://plantl.mineco.gob.es/tecnologias-lenguaje/actividades/infraestructuras/Paginas/Word-embeddings.aspx

por lo que una misma palabra en distintos contextos no tendrá el mismo vector.

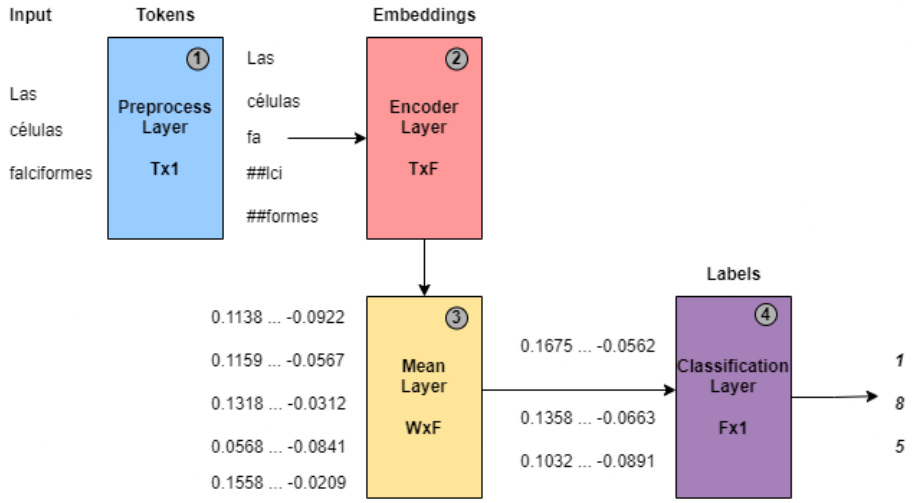


Figura 4.2: Arquitectura del modelo *transformer* con capa de clasificación para NER.

El comportamiento del modelo es el siguiente: en primer lugar, la primera capa del modelo recibe como entrada las frase del conjunto, tal y como se muestra con una frase de ejemplo en la Figura 4.2. Esta capa de preprocesamiento (punto 1) tokenizará las palabras, partiéndolas en unidades más pequeñas o *tokens* (la lista de *tokens* de la frase, sería un vector de T dimensiones) que luego el modelo utilizará para calcular un mapa de características en la capa de codificación (punto 2), donde cada *token* tendrá un vector de características de F dimensiones, por lo que la salida de la capa de codificación será una matriz de características de la forma TxF.

La problemática de esto, es que el modelo está clasificando los *tokens*, que no son necesariamente las palabras, que es lo que se necesita en este caso. Es por esto, por lo que el tercer paso consiste en pasar la matriz de características resultante por una capa adicional (punto 3). Esta capa calculará la media de los *tokens* pertenecientes a cada palabra (en caso de que exista más de un *token* para esa palabra), y generará un nuevo mapa de características de la forma WxF, siendo W la cantidad de palabras de la entrada.

Una vez se tienen los vectores de características para cada palabra de

entrada, por último, se le pasan a una capa de clasificación (punto 4) para que realice la predicción de entidades o relaciones para cada palabra o pares de palabras. Esta capa de clasificación será uno de los clasificadores que se explicarán a continuación.

4.3.4.1. Implementación del clasificador SVM

El primer modelo a implementar es un modelo de SVM. La razón de escoger este modelo es que se trata de un modelo sólido en cuanto a clasificación, con un coste computacional de entrenamiento no demasiado alto, y además ha sido bastante utilizado en la tarea de NER, como se comenta en la sección 2.2.4.1. Los hiperparámetros seleccionados para entrenar el modelo fueron los siguientes:

- **Función de penalización**: esta añade penalización al modelo a medida que aumenta su complejidad, en este caso se seleccionó la función de penalización L2. En la ecuación 4.1 se muestra dicha función, siendo λ parámetro de regularización que penaliza los parámetros para que el modelo generalice los datos y no se sobre-ajuste. Dicha penalización se suma a la función de coste.

$$J(\theta) = J(\theta) + \lambda \sum_{j=1}^{p} \theta_j^2 \tag{4.1}$$

- **Función de pérdida**: esta función es la encargada de indicar la calidad de las predicciones del modelo respecto a los datos. En este caso, se ha utilizado la función de pérdida Squared Hinge-Loss debido a que es altamente utilizada con el modelo SVM Lee y Lin (2013). En la ecuación 4.2 se muestra dicha función, siendo N el número de datos.

$$L(y, \hat{y}) = \sum_{i=0}^{N} (max(0, 1 - y_i, \hat{y}_i)^2) \tag{4.2}$$

- **Parámetro C**: se puede considerar como el grado de optimización que tiene que cumplir el modelo. Para este caso, se ha utilizado el parámetro por defecto, 100.

- **Estrategia multiclase**: la estrategia multiclase seguida es *one-to-rest*, tal y como se especifica en la sección 2.1.1.1 para los problemas

multiclase.

- **Pesado de clases**: el pesado de clases es equilibrado, es decir, utiliza la cantidad de datos de cada clase para ajustar automáticamente las ponderaciones de forma inversamente proporcional a las frecuencias de las clases en los datos de entrada. Lo cual sirve para combatir el desequilibrio entre clases. La función que calcula el peso de la clase i es la mostrada en la ecuación 4.3, siendo N el número total de muestras, C el número de clases, y N_i el número de muestras de la clase i.

$$W(i) = \frac{N}{C \cdot N_i} \tag{4.3}$$

4.3.4.2. Implementación del modelo *Random Forest*

El segundo modelo a implementar, es el modelo de ensamblado *Random Forest*. Con este modelo se pretendía ver que tal funcionan los modelos ensamblados para esta clase de problemas, ya que son ampliamente utilizados en tareas de clasificación. Los hiperparámetros utilizados con este modelo se detallan a continuación:

- **Número de estimadores**: este parámetro indica el número de Árboles de decisión que se integran en el modelo ensamblado, en este caso se han utilizado 100 estimadores.

- **Criterio de pureza**: este parámetro indica la función que mide la calidad de la división de los datos en los siguientes nodos. La calidad de una posible división del nodo es computada por la función de impureza. En este caso se ha utilizado la función Gini Breiman (1996) que se muestra en la ecuación 4.4, siendo k el número de clases, y p_i la probabilidad de que el un dato de D pertenezca a la clase i.

$$G(D) = 1 - \sum_{i=1}^{k} p_i^2 \tag{4.4}$$

- **Máximo número de características**: el número máximo de características a considerar para realizar la mejor división de nodos. En este caso se ha seleccionado la raíz cuadrada del número de características.

- **Pesado de clases**: como en el caso anterior, se utiliza un pesado de clases balanceado para mitigar el desbalance entre clases. La función de pesado es la que se explicó anteriormente en la ecuación 4.3.

4.3.4.3. Implementación de Redes Neuronales

El tercer y último modelo a implementar se trata de una red neuronal. En esta ocasión, se trata de una red neuronal sencilla, con pocas capas, encargada de realizar la tarea de clasificación. La estructura de la red propuesta es la siguiente:

- **Capa de entrada**: la primera capa del modelo, compuesta por 768 neuronas debido a las dimensiones de los vectores de características generados por el *transformer*.
- **Capa de *dropout***: esta capa es utilizada para evitar el sobreajuste aplicando un *dropout* del 25 %.
- **Capa oculta**: capa intermedia entre la capa de entrada y de salida, contiene la mitad de neuronas que la capa de entrada, 384 neuronas.
- **Capa de *dropout***: de nuevo otra capa intermedia para evitar el sobreajuste, en este caso aplicando un *dropout* del 50 %.
- **Capa de salida**: la última capa, una capa *softmax* para calcular las probabilidades de cada clases, y está compuesta por la misma cantidad de neuronas que clases hay para clasificar. Siendo cuatro tipos de entidades distintas, con la notación IOB pasarían a ser ocho, y añadiendo la clase sin entidad nueve clases, por lo tanto, nueve neuronas.

En la Figura 4.3 se puede ver el diagrama de dicha arquitectura. Respecto a los hiperparámetros de la red, se muestran a continuación:

- **Función de coste**: se ha utilizado la función *Cross-entropy* Zhang y Sabuncu (2018), ampliamente utilizada en tareas de clasificación.
- **Función de optimización**: se ha utilizado el optimizador Adam (*Adaptive Moment Estimation*) Ruder (2016) también muy utilizado, basado en tasas de aprendizaje adaptativas.
- **Tasa de aprendizaje**: se ha utilizado una tasa de aprendizaje de 0.001.

- **Épocas máximas**: se ha fijado un número máximo de épocas para el modelo, en este caso 80.
- ***Callbacks***: en caso de que el modelo deje de aprender se ha utilizado *Early Stopping* con paciencia 10, para que el aprendizaje se detenga cuando lleve más de 10 épocas sin mejorar los resultados de la evaluación.
- **Funciones de activación**: en las capas de entrada e intermedia se ha utilizado la función de activación ReLU (*Rectified Linear Unit*). En la capa de salida se ha utilizado la función *softmax*.
- **Pesado de clases**: como todos los casos, se utiliza un pesado de clases balanceado para mitigar el desbalanceo entre clase (ecuación 4.3).

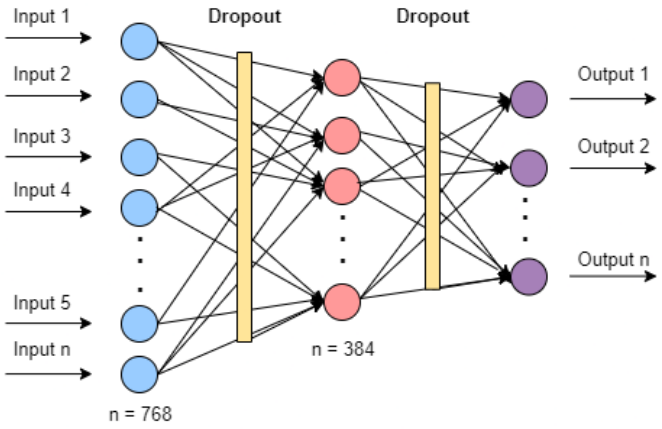


Figura 4.3: Aquitectura de la Red Neuronal.

Los resultados de la aplicación de este modelo, junto con los anteriores se verán más adelante en la sección de resultados 5.4.3.

4.3.5. Implementación del modelo *transformer*

El segundo enfoque que se propone, es utilizar como base uno de los modelos preentrenados descritos en la sección 4.2 y reentrenar la arquitectura de *transformer* completa con los datos de entrenamiento, para ajustar el modelo a los datos con los que se está trabajando. El modelo recibe como entrada una secuencia de tokens (palabras) equivalente a una frase, y obtiene

como resultado una secuencia de entidades, donde cada entidad corresponde a cada palabra de la frase de entrada.

La arquitectura del modelo base es el modelo BERT, en esta ocasión, entrenado para la tarea de NER. Ya que se trata de un modelo contextual lo interesante es pasarle como entrada la secuencia completa, es decir, la frase completa, y que éste prediga la secuencia de etiquetas.

En primer lugar, es necesario tokenizar la frase de entrada. Para ello, se utiliza un método llamado tokenización WordPiece Wu et al. (2016). Se trata de una estrategia de tokenización que trata de lograr un buen equilibrio entre el tamaño del vocabulario y las palabras fuera del vocabulario. El algoritmo segmenta las palabras en partes más pequeñas y construye el vocabulario utilizando la combinación de estas partes individuales. Cada una de estas partes o tokens será convertido a un vector de 768 dimensiones.

Una vez hecho esto, para un token dado, su representación de entrada se construye sumando los *embeddings* del token, los *embeddings* del segmento (en caso de que la entrada tenga más de una frase, en el caso de NER, no) y de posición correspondientes, tal y como se muestra en la Figura 4.4 Devlin et al. (2018). El cálculo de estos *embeddings* se realiza en las respectivas capas, que se explicarán a continuación.

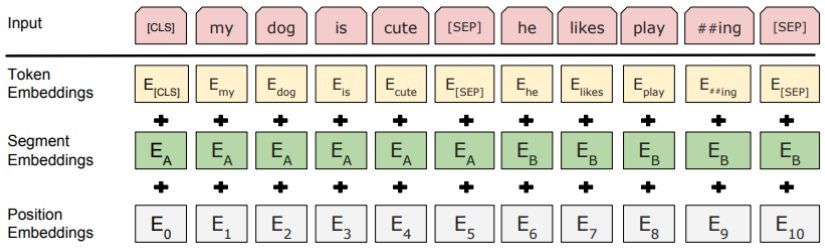


Figura 4.4: Representaciones de entrada en la arquitectura BERT base.

El modelo utilizado se compone de dos módulos: el módulo de *embeddings*, formado por una serie de capas de *embeddings*, proporciona una salida que posteriormente es utilizada como entrada a la segunda parte. Esta segunda parte o módulo es el *encoder*, que realizaría la clasificación. Así pues, el módulo de *embeddings* está compuesto por las siguientes capas:

- **Capa de *embedding***: la capa de entrada, que recibe la lista de tokens que se han tokenizado previamente con el algoritmo WordPiece, y contienen las etiquetas de inicio y fin de secuencia. Con un tamaño

de vocabulario de 52000 tokens y unas dimensiones de *embedding* de 768.

- **Capa de *embedding* posicional**: esta capa se necesita para identificar la posición de cada token dentro de la frase, ya que una misma palabra en distintas posiciones debe tener una representación distinta según el contexto.
- **Capa de *embedding* de segmentos**: esta capa se utiliza para saber que token en particular pertenece a cada secuencia. En el caso este trabajo, el modelo recibe sólo una secuencia (frase) por lo que el *embedding* siempre contendrá el mismo valor para cada uno de los tokens.
- **Capa de normalización**: hay que tener en cuenta que los cambios en la salida de una capa tenderán a causar cambios altamente correlacionados en las entradas sumadas de la siguiente capa. Por este motivo se realiza la normalización, para reducir esta influencia fijando la media y la varianza de las entradas sumadas dentro de cada capa.
- **Capa de *dropout***: se aplica un *dropout* del 10 % y es la última capa del módulo.

El módulo del *encoder* está formado por 12 capas de atención, donde se implementa el mecanismo de auto-atención, previamente explicado en la sección 2.1.1.5. Respecto a los hiperparámetros utilizados para el entrenamiento, se muestran a continuación:

- **Número de épocas**: se entrenaron los modelos con 25 y 40 épocas.
- **Tamaño del lote**: el tamaño de lote seleccionado fue 64, para ajustarlo a las prestaciones de las máquinas utilizadas para entrenar los modelos.
- **Tasa de aprendizaje**: se seleccionó una tase de aprendizaje de 0.00002 durante el ajustado.

4.4. Implementación de modelos RE

Tal y como se comentó en la propuesta (sección 4.1), para abordar esta tarea se propone utilizar en primer lugar un modelo de clasificación, encargado de

clasificar pares de palabras. Y en segundo lugar, entrenar otro clasificador secundario sólo con los datos de las entidades, esperando que éste ayude al primer modelo a discriminar mejor entre entidades.

El modelo principal recibirá como entrada los dos vectores concatenados, generados por el modelo de *transformer* aplicado a cada par de entidades del conjunto de datos. Mientras que el modelo secundario recibirá las muestras de datos con relación, que han sido pasadas por una serie de procesos, que se detallarán las posteriores secciones. También se explicará el preproceso de los datos antes de introducirlos en el modelo.

4.4.1. Preprocesado de los datos

Tal y como se explicó para la tarea de NER (sección 4.3.1), se realizan sobre el texto dos pasos, el primero es la eliminación de caracteres no alfanuméricos, y el segundo es la eliminación de tildes. Respecto a la codificación de las etiquetas de clase (relaciones) en valores numéricos del 0 a $N-1$, siendo N el número de clases (tipos de relación).

4.4.2. Aumento de muestras y reducción de dimensionalidad

Como se indicó la propuesta, para la tarea de RE se propone utilizar aumento de datos para tratar de mitigar el desbalance entre clases. Debido a la cantidad tan grande de datos con los que se pretende entrenar el modelo (aproximadamente 600 mil pares de palabras) resultó inviable aplicar técnicas de aumento de datos a los datos inicialmente, por lo que éstas, junto con la reducción de dimensionalidad se aplicarán a los datos con los que se entrena el modelo secundario (sólo con entidades).

Para realizar el aumento de muestras se ha utilizado el algoritmo SMOTE-NC, explicado en la sección 2.2.6.2, para aumentar los datos de entrada una vez ya se hayan extraído las características. Es decir, una vez se han obtenido los vectores de características de los pares entidades, se aplica el algoritmo para aumentar la cantidad de muestras de una determinada clase.

En primer lugar, se probó a utilizar el algoritmo SMOTE, pero se observó que aplicando SMOTE-NC e introduciendo ciertas de las características como categóricas, daba unos resultados ligeramente superiores. Para ello se seleccionaron las 15 primeras características como categóricas.

En segundo lugar, a la hora de seleccionar a que clases se les realiza el aumento, las clases candidatas eran las minoritarias, las cuales siguiendo el

análisis realizado en la sección 3.2.1, son *has-property*, *same-as*, *entails* y *part-of*. Por lo que se probó a aumentar las muestras de dichas clases. Sin embargo, tras analizar la matriz de confusión se observó que había casos en los que se producían confusiones entre las clases minoritarias, lo que daba lugar a falsos positivos de éstas. Finalmente la mejor combinación de clases para aumentar fueron las clases *in-time*, *entails* y *same-as*, dejando a un lado *part-of* y *has-property* y añadiendo *in-time*. Tal y como se realizó en la tarea de NER, el aumento se produjo duplicando el número de muestras de estas clases.

	Cantidad de datos		
Relación	Original	SMOTE-NC	Aumentado
Sin relación	602133	0	602133
causes	502	0	502
in-time	447	447	894
has-property	339	0	339
same-as	222	222	444
entails	211	211	422
part-of	172	0	172

Tabla 4.2: Cantidad de datos iniciales y aumentados para la tarea RE.

En la tabla 4.2 se muestra la cantidad de datos en cada fase del proceso de aumento de datos, por cuestiones de visualización se muestran sólo las seis relaciones menos frecuentes. En la columna *Original* se muestran la cantidad de datos originales en el conjunto de entrenamiento junto con el de desarrollo, en $SMOTE - NC$ se muestra la cantidad de datos aumentada para esa clase, y por último en la columna *Aumentado* se muestra la cantidad de datos en el conjunto de entrenamiento final para cada clase.

Una vez realizado este aumento, se aplica el algoritmo PCA para reducir las dimensiones de los datos, en este caso se redujeron a 12 características, siendo $min(N_{classes} - 1, N_{features})$) el mínimo de dimensiones posibles para el algoritmo, en este caso 14, ya que se trata de un problema con 15 clases.

4.4.3. Implementación de modelos de clasificación en RE

En la tarea de RE, al igual que en NER, se han implementado dos clasificadores, SVM y Redes Neuronales, ya que son ampliamente utilizados en la tarea de RE. La implementación de SVM es la misma que para la tarea de NER descrita en la sección 4.3.4.1.

Por otro lado, la implementación del modelo de Redes Neuronales también es la misma que fue utilizada en la tarea de NER, la cual se explica detalladamente en la sección 4.3.4.3. La única diferencia es en el pesado de las clases del modelo principal, en esta ocasión también se utiliza la función mostrada en la ecuación 4.3, pero se añade más peso en la clase sin entidad, ya que se observó que se producían muchos falsos positivos de entidades. El mejor peso encontrado, fue multiplicando por 18 el peso para la clase sin entidad obtenido por la fórmula 4.3.

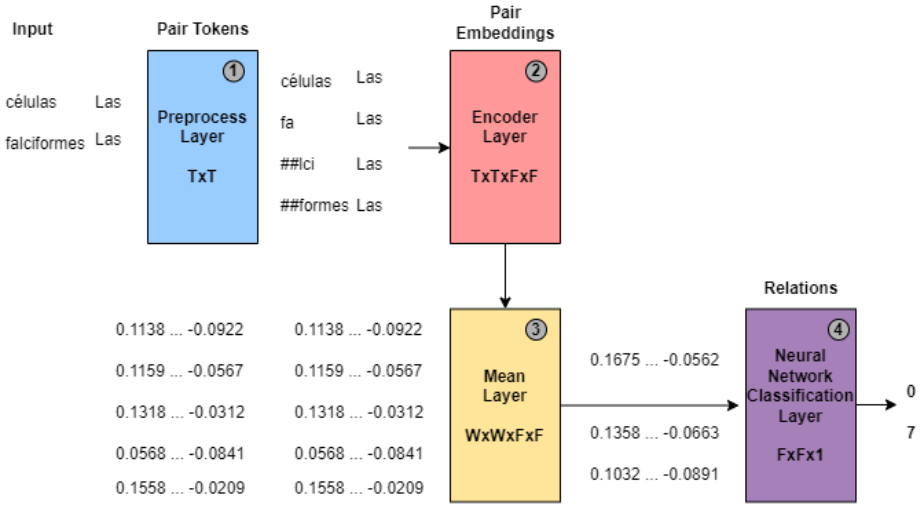


Figura 4.5: Arquitectura del modelo *transformer* con capa de clasificación para RE.

En la Figura 4.5 se muestra la arquitectura del modelo principal de Redes Neuronales, similar al modelo creado para NER. La diferencia en este caso, es que el modelo recibe los dos vectores concatenados, fruto de realizar la codificación de ambas entidades, en lugar de solo una entidad. El funcionamiento es el mismo que en la tarea de RE, explicado anteriormente en la sección 4.3.4.

Hasta ahora se ha explicado el funcionamiento del modelo de Redes Neuronales, pero la segunda propuesta es realizar dos modelos, uno principal encargado de realizar la clasificación de todos los pares de palabras, y otro secundario o de refuerzo, encargado de realizar la clasificación entre los pares de entidades con relación, para así repuntuar los pesos que el modelo inicial da a cada clase, por cada par de palabras. La arquitectura de este sistema se

muestra en la Figura 4.6. Nótese que en esta Figura los puntos 1 y 4, cada uno de ellos correspondería al modelo que se muestra en la Figura 4.5.

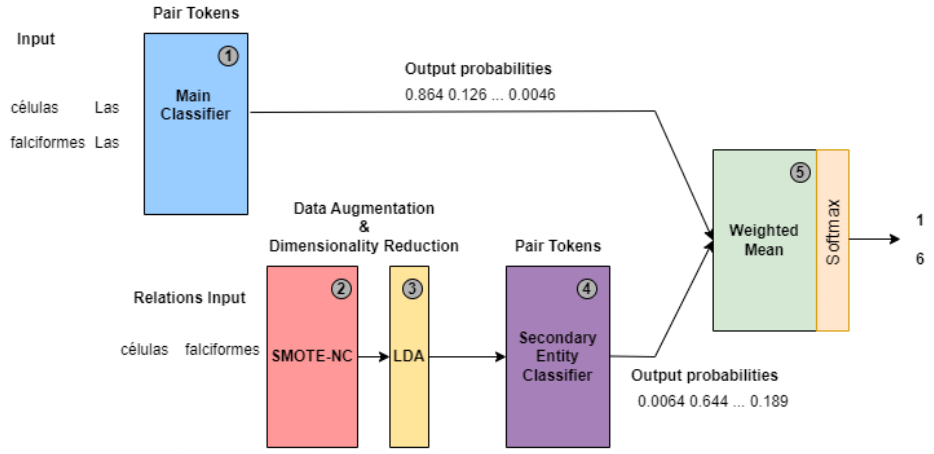


Figura 4.6: Arquitectura del modelo *transformer* con dos clasificadores.

El funcionamiento es el siguiente: en primer lugar se realiza la clasificación con el modelo explicado anteriormente (punto 1), pero en lugar de aplicar una función *softmax* a la salida del primer modelo, para obtener la clase seleccionada, se mantienen las probabilidades para cada una de las clases. Por otro lado, para entrenar al segundo modelo (punto 4), primero se seleccionó un subconjunto del conjunto de entrenamiento, este subconjunto está compuesto por las muestras que tienen una relación, es decir, se dejaron de lado las muestras sin relación. El objetivo de esto, es construir un modelo que esté entrenado exclusivamente con pares de entidades con relación, para así diferenciar entre las distintas relaciones. Después de esto, se pasan estos datos por el algoritmo SMOTE-NC y posteriormente LDA (puntos 2 y 3).

Una vez hecho esto, se introducen finalmente en el segundo modelo, el cual tiene la misma arquitectura que el primero. Para finalizar, se han obtenido las probabilidades de salida de ambos modelos y se combinan las probabilidades para aquellas muestras que el modelo principal ha detectado como entidad, con una media ponderada, mediante la ecuación 4.5, donde $\hat{y}$ sería la clase final que el sistema ha predicho, $\hat{y}_{main}$ sería el vector de probabilidades del modelo principal, y $\hat{y}_{relation}$ el vector del modelo de relaciones.

$$\hat{y} = \arg\max(\frac{\hat{y}_{main} \cdot 0{,}70 + \hat{y}_{relation} \cdot 0{,}3}{2}) \tag{4.5}$$

Finalmente, tras el análisis de la influencia de los distintos tipos de entidad en las relaciones realizado en la sección 3.2.3, se pretende realizar una serie de reglas, con las cuales se aumente la probabilidad de que ciertas relaciones se den para determinados pares de relaciones. En la Tabla 4.3 se muestran los pesos utilizados para ayudar reforzar ciertas relaciones para según que entidades. Dichos pesos son sumados a la predicción del clasificador principal.

Tabla 4.3: Pesado para cada relación según el tipo de entidades.

	Relaciones				
Entidades	Sin relación	*arg*	*domain*	*has-property*	*subject*
Sin alguna entidad	0.5	0	0	0	0
Concep-Concept	0.35	0	0	0.15	0
Action-Concept	0.35	0	0	0.15	0.15
Predicate-Concept	0.2	0.15	0.15	0	0.15

El cálculo de las probabilidades de cada relación para cada par de entidades se muestra en la ecuación 4.6, la cual es la misma que la anterior 4.5, pero en este caso añadiendo los pesos ($\hat{y}_{entities}$) para las relaciones que se muestran en la Tabla 4.3. No obstante, añadir estas reglas apenas obtuvo efecto en los resultados, posiblemente debido a que eran redundantes y el modelo fue capaz de extraerlas de los propios datos. Por este motivo no se mostrarán los resultados de aplicar esto en el capítulo de evaluación 5.

$$\hat{y} = \arg\max(\frac{(\hat{y}_{main} + \hat{y}_{entities}) \cdot 0{,}70 + \hat{y}_{relation} \cdot 0{,}3}{2}) \tag{4.6}$$

Capítulo 5

Evaluación

En este capítulo se describe la metodología utilizada para evaluar el caso de estudio propuesto, a la vez que presenta los resultados obtenidos en las diferentes tareas y sobre la colección de evaluación.

5.1. Colecciones de evaluación

Para la edición de eHealth-KD 2021, los sistemas se evaluarán con el conjunto de test. Como ya se vio en la sección 3.2, los datos de entrenamiento y de desarrollo, así como los de test contendrán frases extraídas de MedlinePlus[4], Wikinews[5] y el corpus CORD-19, todas ellas relacionadas con temas de salud, pero mostrando una importante variedad en cuanto a formato y estructura. Todas ellas fueron anotadas manualmente. Los ficheros con las frases de test anotadas se encuentran en el repositorio de eHealth[1]. El conjunto de test de las tareas de NER y RE está compuesto por 50 frases en español y 50 frases en inglés.

5.2. Metodología de evaluación

Una vez el modelo propuesto ha sido entrenado es necesario realizar la predicción del conjunto de test para poder evaluarlo. El reto eHealth-KD proporciona una colección de desarrollo y de test. Los archivos de entrada de los conjuntos de desarrollo y test son archivos de texto plano en formato UTF-8 con una frase por línea. Las frases no han sido preprocesadas en ningún sentido. Un ejemplo de esto se puede ver en la Figura 5.1.

[1] https://github.com/ehealthkd/corpora/tree/master/2021/eval/testing

```
Sony formed a joint venture with Ericsson, a mobile phone
company based in Sweden.
Sony announced today that ...
```

Figura 5.1: Fichero de texto de ejemplo.

La salida de modelo tiene que ser un archivo en formato BRAT standoff[2], donde cada línea representa una entidad o una relación. En la Figura 5.2 se muestra un ejemplo de las entidades extraídas para el texto de ejemplo de la Figura 5.1, donde la primera columna representa el identificador de la entidad, la segunda columna representa el tipo de entidad, junto con las posiciones inicial y final respecto al texto completo donde se encuentra la entidad; y la última columna muestra el texto de la entidad. Para el caso de las entidades con más de una palabra, se deben separar las posiciones mediante ";".

```
T1          Location 0 5;16 23              North America
T2          Location 10 23                  South America
```

Figura 5.2: Fichero de ejemplo en formato BRAT para las entidades.

Para el caso de las relaciones, se muestra un ejemplo en la Figura 5.3 donde de nuevo, la primera columna representa el identificador y la segunda columna el tipo de entidad, en este caso *Origin*. Los elemenos *Arg1* y *Arg2* hacen referencia a los identificadores de las dos entidades que participan en la relación.

```
R1          Origin Arg1:T3 Arg2:T4
```

Figura 5.3: Fichero de ejemplo en formato BRAT para las relaciones.

A continuación, se muestra un ejemplo real del conjunto de frases de test de este trabajo. Para la frase de ejemplo: "La espina bífida es un defecto del tubo neural, un tipo de defecto congénito del cerebro, la columna vertebral o de la médula espinal.", las entidades extraídas, se pueden ver en la Figura 5.4. Las entidades junto con las relaciones entre éstas, se pueden observar

[2] https://brat.nlplab.org/standoff.html

en la Figura 5.5.

```
T1      Concept 3 9;10 16       espina bífida
T2      Concept 23 30   defecto
T3      Concept 35 39;40 46     tubo neural
T4      Concept 59 66   defecto
T5      Concept 67 76   congénito
T6      Concept 81 88   cerebro
T7      Concept 93 100;101 110  columna vertebral
T8      Concept 119 125;126 133 médula espinal
```

Figura 5.4: Entidades extraídas en formato BRAT para frase de ejemplo.

```
T1      Concept 3 9;10 16       espina bífida
T2      Concept 23 30   defecto
T3      Concept 35 39;40 46     tubo neural
T4      Concept 59 66   defecto
T5      Concept 67 76   congénito
T6      Concept 81 88   cerebro
T7      Concept 93 100;101 110  columna vertebral
T8      Concept 119 125;126 133 médula espinal
R0      in-place Arg1:T2 Arg2:T3
R1      is-a Arg1:T1 Arg2:T2
R2      in-context Arg1:T4 Arg2:T5
R3      in-place Arg1:T4 Arg2:T6
R4      in-place Arg1:T4 Arg2:T7
R5      in-place Arg1:T4 Arg2:T8
R6      is-a Arg1:T2 Arg2:T4
```

Figura 5.5: Entidades y relaciones extraídas en formato BRAT para frase de ejemplo.

Para agilizar la entrada y salida de datos, el reto proporciona *scripts* de Python para leer y escribir este formato en el repositorio de eHealth-KD 2021[3].

El orden en que aparecen las entidades y las relaciones en el archivo de salida es irrelevante. Sólo es importante ser coherente con respecto a los identificadores. Cada entidad tiene un identificador único que se utiliza en las anotaciones de las relaciones para referirse a ella. Su archivo de salida puede tener identificadores diferentes a los de la salida *gold*, y los *scripts* de evaluación podrán encontrar correctamente las anotaciones que coincidan.

Una vez se ha obtenido la salida del modelo en el formato BRAT necesario, se deberá ejecutar el *script* de evaluación proporcionado por el reto[4], que calculará las métricas necesarias para la evaluación, éstas métricas están detalladas en la sección siguiente 5.3.

[3] https://github.com/ehealthkd/corpora/tree/master/scripts
[4] https://github.com/ehealthkd/corpora/blob/master/scripts/score.py

5.3. Métricas de evaluación

La evaluación final de los sistemas se realiza evaluando cada uno de los escenarios que componen la tarea de forma independiente. Las métricas de evaluación son la precisión, el *recall* y el *F1Score*, siendo el *F1Score* la métrica que determina el *ranking* de los participantes del reto. Dichas métricas están definidas de forma específica para cada tarea, las cuales se describen en las posteriores secciones. A continuación se definen las métricas de evaluación para cada escenario.

5.3.1. Evaluación NER

La evaluación aislada de la tarea NER evalúa el escenario 2. Para calcular las puntuaciones a continuación se definen las coincidencias correctas, parciales, ausentes, incorrectas y espurias para este problema:

- **Coincidencias correctas**: se notifican cuando un texto en el archivo proporcionado por el modelo coincide exactamente con un tramo de texto correspondiente en el archivo *gold* en los valores de inicio y final y también el tipo de entidad. Sólo puede haber una coincidencia correcta por entrada en el archivo *gold*. Por lo tanto, las entradas duplicadas contarán como espurias.
- **Coincidencias incorrectas**: se notifican cuando los valores inicial y final coinciden, pero no el tipo.
- **Coincidencias parciales**: se notifican cuando dos intervalos [inicio, fin] tienen una intersección no vacía. Una frase parcial sólo se comparará con una única frase correcta. Por ejemplo, “tipo de cáncer” podría ser una coincidencia parcial tanto para “tipo” como para “cáncer”, pero sólo se cuenta una vez como coincidencia parcial con la palabra “tipo”. La palabra “cáncer” se cuenta entonces como desaparecida. El objetivo es evitar que unos pocos tramos de texto grandes que abarcan la mayor parte del documento obtengan una puntuación muy alta.
- **Coincidencias ausentes**: se notifican cuando las entidades no aparecen en el fichero de salida del modelo, pero sí en el fichero *gold*.
- **Coincidencias espurias**: se notifican cuando las entidades no aparecen en el fichero *gold*, pero sí en el fichero de salida del modelo.

Con estas definiciones, se define la precisión, *recall* y *F1Score* de la tarea NER en las ecuaciones 5.1, 5.2 y 5.3, respectivamente, siendo C, P, I, S y M las entidades correctas, parciales, incorrectas, espurias y ausentes, respectivamente.

$$Prec_{NER} = \frac{C + \frac{1}{2}P}{C + I + P + S} \tag{5.1}$$

$$Rec_{NER} = \frac{C + \frac{1}{2}P}{C + I + P + M} \tag{5.2}$$

$$F1_{NER} = 2 \cdot \frac{Prec_{NER} \cdot Rec_{NER}}{Prec_{NER} + Rec_{NER}} \tag{5.3}$$

Una mayor precisión significa que el número de identificaciones espurias es menor en comparación con el número de identificaciones ausentes, y un mayor *recall* significa lo contrario. Las coincidencias parciales reciben la mitad de la puntuación de las coincidencias correctas, mientras que las identificaciones erróneas y espurias no reciben ninguna puntuación.

5.3.2. Evaluación RE

La evaluación aislada de la tarea RE evalúa el escenario 3. Para evaluar esta tarea, el fichero de salida debe contener las entidades proporcionadas por el *gold standard* y las relaciones reconocidas por el modelo. Para calcular las puntuaciones, a continuación se definen las coincidencias correctas, las ausentes y las espurias:

- **Coincidencias correctas**: se notifican cuando las relaciones coinciden exactamente con el archivo *gold*, incluyendo el tipo y los identificadores correspondientes a cada una de las entidades participantes en la relación.

- **Coincidencias ausentes**: se notifican cuando las relaciones no aparecen en el fichero de salida del modelo, pero sí en el fichero *gold*, ya sea por que el tipo de relación es incorrecto o por que alguno de los identificadores de las entidades no coincide.

- **Coincidencias espurias**: se notifican cuando las relaciones no aparecen en el fichero *gold*, pero sí en el fichero de salida del modelo, ya

sea por que el tipo de relación es incorrecto o por que alguno de los identificadores de las entidades no coincide.

Una vez definidas las métricas, se define la precisión, *recall* y *F1Score* la tarea RE en las ecuaciones 5.4, 5.5 y 5.6, respectivamente, siendo C, S y M las entidades correctas, espurias, y ausentes, respectivamente.

$$Prec_{RE} = \frac{C}{C+S} \tag{5.4}$$

$$Rec_{RE} = \frac{C}{C+M} \tag{5.5}$$

$$F1_{RE} = 2 \cdot \frac{Prec_{RE} \cdot Rec_{RE}}{Prec_{RE} + Rec_{RE}} \tag{5.6}$$

5.4. Resultados

En esta sección, se presentarán los resultados obtenidos durante el desarrollo de este trabajo para cada uno de los modelos implementados, junto con las características utilizadas para entrenar a cada uno de ellos, si lo requiere. También se comentarán los resultados obtenidos por el resto de participantes del reto, para poder comparar las técnicas utilizadas con las realizadas en este trabajo.

5.4.1. Resultados de los participantes de eHealth-KD

En primer lugar, en esta sección se expondrán los resultados obtenidos por los participantes del reto eHealth-KD. Estos resultados sirven como base para saber a que resultados aspirar con la propuesta de este trabajo, así como también descubrir la dificultad del mismo. También es interesante comparar estos resultados con los obtenidos durante el trabajo, para ver en que medida se ha conseguido mejorar, o en caso de no hacerlo, por que motivos.

En la Tabla 5.1 se muestran los resultados de los cinco mejores participantes de la tarea de NER, los cuales son bastante buenos de forma general. Los mejores resultados para cada tabla se señalan con un asterisco. También se muestran los resultados de una *baseline* sencilla computacionalmente, proporcionada por el reto Piad-Morfis et al. (2021).

Tabla 5.1: Resultados de los participantes de eHealth-KD en la evaluación de NER con mejores resultados.

Equipo	*F1Score*	Precisión	*Recall*
PUCRJ-PUCPR-UFMG	0.706*	0.715*	0.697
Vicomtech	0.684	0.699	0.747*
IXA	0.653	0.614	0.698
UH-MMM	0.608	0.546	0.685
uhKD4	0.527	0.517	0.537
baseline	0.306	0.350	0.272

Por otro lado, la Tabla 5.2 muestra los resultados de los cinco mejores participantes de la tarea de RE. En esta tarea se han obtenido resultados más bajos, debido a su dificultad. Una vez se tiene conocimiento de los resultados obtenidos por el resto de participantes, en las posteriores secciones se muestran los obtenidos en este trabajo.

Tabla 5.2: Resultados de los participantes de eHealth-KD en la evaluación de RE con mejores resultados.

Equipo	*F1Score*	Precisión	*Recall*
IXA	0.430*	0.453	0.409*
Vicomtech	0.372	0.542	0.283
uhKD4	0.318	0.556*	0.222
PUCRJ-PUCPR-UFMG	0.263	0.366	0.205
UH-MMM	0.054	0.077	0.041
baseline	0.033	0.437	0.017

5.4.2. Resultados de la propuesta para la tarea NER

En primer lugar, se presentan los resultados obtenidos para la tarea de NER. Los resultados obtenidos se mostraran junto con los resultados de los anteriores participantes, ordenados de mayor a menor por la métrica *F1Score*, y mostrando en negrita los resultados de este trabajo.

Como primeros resultados, en la Tabla 5.3 se muestras los obtenidos por los modelos de clasificación propuestos en la sección 4.3.4, utilizando las características que se explicaron en la sección 4.3.3, vectores TF-IDF y los *embeddings* de *FastText* del dominio biomédico 4.3.3.2. También se muestran los participantes que obtuvieron los 3 primeros puestos. Tal y como era de

esperar, los resultados no son demasiado buenos, aún así consiguen superar a la *baseline* propuesta por el reto. El modelo que mejor parece funcionar es SVM junto con las características extraídas de la representación TF-IDF.

Tabla 5.3: Resultados obtenidos por los modelos de clasificación en la evaluación de NER.

Modelo	Características	*F1Score*	Precisión	*Recall*
PUCRJ-PUCPR-UFMG	-	0.706*	0.715*	0.697
Vicomtech	-	0.684	0.699	0.747*
IXA	-	0.653	0.614	0.698
SVM	**TF-IDF**	0.571	0.493	0.678
NN	***FastText***	0.478	0.308	0.643
SVM	***FastText***	0.470	0.419	0.532
Random Forest	**TF-IDF**	0.460	0.572	0.384
NN	**TF-IDF**	0.389	0.560	0.298
Random Forest	***FastText***	0.374	0.316	0.458
baseline	-	0.306	0.350	0.272

En segundo lugar, en la Tabla 5.4 se muestran los resultados de aplicar los modelos *transformers* descritos en la sección 4.2, para cada modelo de clasificador planteado para esta tarea, explicados en la sección 4.3.4.

Como se puede ver, los resultados mejoran respecto a los modelos anteriores en algunos casos mejorando significativamente, rozando el segundo puesto para el caso del modelo SVM con el *transformer* RoBERTa$_{Bio}$, lo cual indica que el modelo entrenado en ámbito médico proporciona una ventaja respecto al resto de modelos. A continuación, se muestran los resultados de aplicar el ajustado a los modelos de *transformer* explicada en la sección 4.3.5. Los resultados para cada modelo escogido, para 25 y 40 épocas, se muestran en la Tabla 5.5.

Como se puede ver, los resultados son mucho mejores, llegando a superar a los resultados de los participantes y alcanzando el podio en el reto. Los mejores resultados los ha obtenido el ajustado del modelo RoBERTa$_{Clinical}$ entrenado con texto biomédico, lo que confirma la hipótesis anterior. Respecto al número de épocas, no parece tener demasiados efectos.

Tal y como se comentó en la sección 4.3, que explica las propuestas de NER, se aplicaron dos técnicas de aumento de datos. Los resultados de aplicar la primera técnica, *Back Translation*, se muestran en la Tabla 5.6. Se puede observar que existen diferencias al aplicar dicha técnica en algunos

Tabla 5.4: Resultados obtenidos por la arquitectura *transformer* + clasificación en la evaluación de NER.

Modelo	Clasificador	*F1Score*	Precisión	*Recall*
PUCRJ-PUCPR-UFMG	-	0.706*	0.715*	0.697
Vicomtech	-	0.684	0.699	0.747*
RoBERTa$_{Bio}$	**SVM**	0.683	0.676	0.690
mBERT	**SVM**	0.666	0.656	0.675
RoBERTa$_{Clinical}$	**SVM**	0.665	0.645	0.685
BETO	**SVM**	0.662	0.665	0.660
IXA	-	0.653	0.614	0.698
RoBERTa$_{NER}$	**SVM**	0.648	0.621	0.677
RoBERTa$_{Bio}$	**NN**	0.646	0.653	0.640
RoBERTa$_{Clinical}$	**NN**	0.624	0.620	0.630
BETO	**NN**	0.607	0.591	0.623
BETO	***Random Forest***	0.594	0.606	0.581
RoBERTa$_{Bio}$	***Random Forest***	0.585	0.605	0.566
mBERT	***Random Forest***	0.573	0.628	0.526
RoBERTa$_{Clinical}$	***Random Forest***	0.568	0.605	0.535
mBERT	**NN**	0.562	0.605	0.524
RoBERTa$_{NER}$	**NN**	0.540	0.478	0.622
baseline	-	0.306	0.350	0.272
RoBERTa$_{NER}$	***Random Forest***	0.243	0.411	0.172

Tabla 5.5: Resultados obtenidos de los modelos *transformer* ajustados.

Modelo	Épocas	*F1Score*	Precisión	*Recall*
RoBERTa$_{Clinical}$	25	0.736*	0.731*	0.740
mBERT	40	0.730	0.720	0.741
RoBERTa$_{Clinical}$	40	0.728	0.727	0.728
mBERT	25	0.727	0.706	0.750*
RoBERTa$_{Bio}$	40	0.724	0.712	0.736
RoBERTa$_{Bio}$	25	0.723	0.710	0.736
RoBERTa$_{NER}$	25	0.718	0.716	0.719
RoBERTa$_{NER}$	40	0.716	0.711	0.721
PUCRJ-PUCPR-UFMG	-	0.706	0.715	0.697
BETO	25	0.698	0.682	0.713
BETO	40	0.693	0.674	0.712
Vicomtech	-	0.684	0.699	0.747
IXA	-	0.653	0.614	0.698
baseline	-	0.306	0.350	0.272

Tabla 5.6: Resultados obtenidos de los modelos *transformer* ajustados con *Back Translation*.

Modelo	Épocas	*F1Score*	Precisión	*Recall*
RoBERTa$_{Bio}$	40	0.739*	0.730	0.750*
RoBERTa$_{Bio}$	25	0.738	0.731*	0.747
mBERT	40	0.723	0.717	0.730
RoBERTa$_{Clinical}$	40	0.720	0.711	0.728
RoBERTa$_{Clinical}$	25	0.717	0.726	0.708
mBERT	25	0.715	0.713	0.716
RoBERTa$_{NER}$	40	0.707	0.712	0.701
PUCRJ-PUCPR-UFMG	-	0.706	0.715	0.697
BETO	40	0.701	0.692	0.710
BETO	25	0.697	0.681	0.713
RoBERTa$_{NER}$	25	0.696	0.706	0.687
Vicomtech	-	0.684	0.699	0.747
IXA	-	0.653	0.614	0.698
baseline	-	0.306	0.350	0.272

casos mejoran los resultados y en otros empeoran. En el caso del modelo RoBERTa$_{Bio}$ mejoran ligeramente, obteniendo un nuevo mejor resultado de un 0.739 de *F1Score*. En BETO, también parece tener resultados positivos a diferencia del resto de modelos.

La otra técnica de aumento de datos utilizada es la generación de sinónimos de las palabras. Los resultados de aplicar esta técnica se muestran en la Tabla 5.7. Como se puede observar, los resultados son ligeramente peores que en el caso de no utilizar la técnica, a excepción del modelo RoBERTa$_{Bio}$, en el cual se han mejorado ligeramente los resultados. También se puede ver que en este caso el aumento de épocas ha empeorado los resultados, posiblemente debido a un sobre-ajuste.

Por lo tanto, los mejores resultados obtenidos en este tarea se han conseguido reentrenando el modelo RoBERTa$_{Bio}$ con los datos de entrenamiento de la tarea, además de haber aplicado la técnica de *Back Translation* para el aumento de datos, obteniendo así un primer puesto en el reto, con un *F1Score* de 0.739. El análisis de estos resultados se verá en el capítulo 6.

Tabla 5.7: Resultados obtenidos de los modelos *transformer* ajustados con sinónimos.

Modelo	Épocas	*F1Score*	Precisión	*Recall*
RoBERTa$_{Bio}$	25	0.726*	0.724*	0.728
RoBERTa$_{Clinical}$	25	0.720	0.724	0.716
RoBERTa$_{Clinical}$	40	0.720	0.716	0.724
mBERT	25	0.718	0.717	0.718
mBERT	40	0.711	0.714	0.707
RoBERTa$_{Bio}$	40	0.708	0.718	0.698
PUCRJ-PUCPR-UFMG	-	0.706	0.715	0.697
BETO	25	0.696	0.692	0.699
BETO	40	0.694	0.691	0.698
RoBERTa$_{NER}$	40	0.691	0.706	0.677
RoBERTa$_{NER}$	25	0.690	0.710	0.673
Vicomtech	-	0.684	0.699	0.747
IXA	-	0.653	0.614	0.698
baseline	-	0.306	0.350	0.272

5.4.3. Resultados de la propuesta para la tarea RE

Una vez vistos los resultados de NER, en esta sección se presentan los los resultados de la tarea RE. La primera propuesta para esta tarea es aplicar un modelo *transformer* como codificador, junto a los modelos explicados en la sección 4.4.3, un clasificador SVM y una Red Neuronal, que se encargarán de realizar la clasificación. Los resultados de esto se muestra en la Tabla 5.8. Como se puede observar, los resultados no son demasiado buenos, aún así se logra superar la *baseline* propuesta por el reto, e incluso alcanzar un cuarto puesto.

El modelo que mejores resultados parece proporcionar es RoBERTa$_{Clinical}$, junto con la clasificación de Redes Neuronales, por este motivo, en el segundo enfoque se utilizó este clasificador. El segundo enfoque propuesto se basa en utilizar un segundo clasificador de Redes Neuronales de refuerzo, para que ayude a discriminar mejor entre las entidades. Los resultados de aplicar este segundo clasificador se muestran en la figura 5.9. Se puede observar que los resultados han mejorado ligeramente tras aplicar este segundo clasificador.

Tabla 5.8: Resultados obtenidos por la arquitectura *transformer* + clasificación en la evaluación de RE.

Modelo	Clasificador	*F1Score*	Precisión	*Recall*
IXA	-	0.430*	0.453	0.409*
Vicomtech	-	0.372	0.542	0.283
uhKD4	-	0.318	0.556*	0.222
PUCRJ-PUCPR-UFMG	-	0.263	0.366	0.205
RoBERTa$_{Clinical}$	**NN**	0.267	0.228	0.322
PUCRJ-PUCPR-UFMG	-	0.263	0.366	0.205
RoBERTa$_{Bio}$	**NN**	0.225	0.183	0.290
BETO	**NN**	0.223	0.257	0.197
RoBERTa$_{Clinical}$	**SVM**	0.203	0.165	0.262
RoBERTa$_{Bio}$	**SVM**	0.200	0.166	0.253
mBERT	**NN**	0.174	0.182	0.166
BETO	**SVM**	0.190	0.159	0.237
RoBERTa$_{NER}$	**SVM**	0.172	0.173	0.172
mBERT	**SVM**	0.166	0.148	0.191
RoBERTa$_{NER}$	**NN**	0.112	0.108	0.125
baseline	-	0.033	0.437	0.017

Tabla 5.9: Resultados obtenidos por la arquitectura *transformer* + 2 clasificadores en la evaluación de RE.

Modelo	*F1Score*	Precisión	*Recall*
IXA	0.430*	0.453	0.409*
Vicomtech	0.372	0.542	0.283
uhKD4	0.318	0.556*	0.222
RoBERTa$_{Clinical}$	0.270	0.231	0.326
PUCRJ-PUCPR-UFMG	0.263	0.366	0.205
RoBERTa$_{Bio}$	0.231	0.189	0.289
BETO	0.226	0.260	0.199
mBERT	0.175	0.183	0.168
RoBERTa$_{NER}$	0.112	0.108	0.125
baseline	0.033	0.437	0.017

Por último, se aplicaron técnicas de aumento de datos (SMOTE-NC) y reducción de dimensionalidad (LDA) a los datos con los que se entrena el segundo clasificador. Los resultados de aplicar estas técnicas se muestran en la Tabla 5.10. Como se puede observar, los datos han mejorado ligeramente al aplicar estas técnicas, llegando a mejorar los resultados originales utilizando

un sólo clasificador (Tabla 5.8).

Tabla 5.10: Resultados obtenidos por la arquitectura *transformer* + 2 clasificadores en la evaluación de RE + SMOTE-NC + LDA.

Modelo	*F1Score*	Precisión	*Recall*
IXA	0.430*	0.453	0.409*
Vicomtech	0.372	0.542	0.283
uhKD4	0.318	0.556*	0.222
RoBERTa$_{Clinical}$	0.273	0.233	0.330
PUCRJ-PUCPR-UFMG	0.263	0.366	0.205
RoBERTa$_{Bio}$	0.236	0.193	0.305
BETO	0.223	0.257	0.197
mBERT	0.180	0.188	0.173
RoBERTa$_{NER}$	0.114	0.111	0.126
baseline	0.033	0.437	0.017

Por lo tanto, los mejores resultados obtenidos en este tarea se han conseguido aplicando dos clasificadores de Redes Neuronales, además de aplicar SMOTE-NC y LDA, obteniendo un cuarto puesto en el reto con un *F1Score* de 0.273. El análisis de estos resultados se realizará en el capítulo 6, dedicado a ello.

5.4.4. Comparativa de resultados

Por último, en esta sección se va a visualizar los mejores resultados obtenidos en este trabajo para cada uno de los escenarios y se mostrarán junto con los resultados de los otros participantes del reto. En la Tabla 5.11 se muestran los resultados de la tarea de NER, en los cuales se ha conseguido superar a los demás participantes y quedar en primer posición en las tres métricas de evaluación.

El modelo que ha obtenido el primer puesto es el modelo de *transformer* RoBERTa$_{Bio}$ ajustado con los datos de entrenamiento y desarrollo de la tarea, aumentados con la técnica *Back Translation* y entrenado con 40 épocas. El nombre del sistema es RoBERTa$_{Bio}$-BT40.

Por último, la tabla 5.12 muestra los cuatro mejores resultados de la tarea de RE. Este escenario es más dificultoso, como se puede observar en los resultados. En esta tarea se ha conseguido obtener un cuarto puesto con el enfoque de dos clasificadores de Redes Neuronales, recibiendo como

entrada las características extraidas del modelo RoBERTa$_{Clinical}$ y aplicando aumento de datos con SMOTE-NC y LDA. El nombre del sistema es RoBERTa$_{Clinical}$-SN-LDA2.

Tabla 5.11: Mejores resultados obtenidos en la evaluación de NER.

Equipo / Modelo	*F1Score*	Precisión	*Recall*
RoBERTa$_{Bio}$-BT40	0.739*	0.730*	0.750*
PUCRJ-PUCPR-UFMG	0.706	0.715	0.697
Vicomtech	0.684	0.699	0.747
IXA	0.653	0.614	0.698
baseline	0.306	0.350	0.272

Tabla 5.12: Mejores resultados obtenidos en la evaluación de RE.

Equipo / Modelo	*F1Score*	Precisión	*Recall*
IXA	0.430*	0.453	0.409*
Vicomtech	0.372	0.542	0.283
uhKD4	0.318	0.556*	0.222
RoBERTa$_{Clinical}$-SN-LDA2	0.273	0.233	0.330
baseline	0.033	0.437	0.017

En el siguiente capítulo 6 se llevará a cabo el análisis de los resultados obtenidos.

Capítulo 6

Discusión

Este capítulo analiza y discute en profundidad los resultados obtenidos en la evaluación presentada en el capítulo anterior. Se hablará de los modelos propuestos y se analizarán los errores obtenidos, para poder proponer mejoras en trabajos futuros.

A nivel general los resultados obtenidos han sido buenos, consiguiendo mejorar ampliamente los resultados de los anteriores participantes en la tarea de NER y resultados competentes en la tarea de RE, alcanzando un primero y cuarto puesto, respectivamente. A continuación se van a analizar los posibles errores cometidos, así como también los resultados obtenidos para cada modelo.

6.1. Análisis de errores

Para analizar los errores en ambas tareas, primero se va a presentar una tabla mostrando las métricas de evaluación para cada entidad y cada relación, para observar las clases en las cuales haya habido más problemas y tratar de analizarlo. También se mostrará la matriz de confusión del mejor modelo obtenido en cada tarea para ver que clases son las más problemáticas o confusas.

En la Tabla 6.1 se muestran la métricas de evaluación del mejor modelo obtenido con NER (RoBERTa$_{Bio}$-BT40). Las entidades se han mostrado con la notación IOB utilizada durante el entrenamiento (sección 4.3.1), donde la *B-* representa a una entidad de una sola palabra o la primera palabra de la entidad, mientras que *I-* representa las siguientes palabras de la entidad. La columna Muestras indica el número de muestras de test que tienen esa

clase.

Tabla 6.1: Métricas de evaluación para la entidades.

Entidad	Precisión	*Recall*	*F1Score*	Muestras
B-Action	0.63	0.80	0.71	137
B-Concept	0.81	0.87	0.84	678
B-Predicate	0.65	0.34	0.44	98
B-Reference	0.43	0.30	0.35	10
I-Action	0.00	0.00	0.00	8
I-Concept	0.81	0.61	0.70	227
I-Predicate	0.00	0.00	0.00	20
Sin entidad	0.89	0.92	0.90	1110
Media macro	0.53	0.48	0.49	2288
Media ponderada	0.82	0.83	0.82	2288

La media macro se calcula mediante la ecuación 6.1, siendo C el número total de clases y $Metric$ la métrica en cuestión, e indica las métricas globales calculadas por clase, sin tener en cuenta la cantidad de muestras de cada una de ellas. Por otro lado, la media ponderada si tiene en cuenta las muestras de cada clase y se calcula mediante la ecuación 6.2, siendo N el número total de muestras y N_i el número de muestras de la clase i.

$$Metric_{macro} = \frac{\sum_{i=0}^{C} Metric(i)}{C} \tag{6.1}$$

$$Metric_{weighted} = \sum_{i=0}^{C} Metric(i) \cdot \frac{N_i}{N} \tag{6.2}$$

A la vista de los resultados se hace evidente que el modelo funciona mejor en el caso de las clases con mayor número de muestras, y en los casos con menor número, como *I-Action* e *I-Predicate*, no llega a acertar ningún caso, pero al tratarse de una muestra tan pequeña, apenas penaliza los resultados. En el caso de la clase *B-Reference* si se ha llegado a acertar en alguna predicción a pesar de que es una muestra pequeña, y esto puede ser debido a que es una de las clases a las que se le aplicó el aumento de datos, lo que ha podido suponer esta mejora. La clase *I-Reference* no aparece en el conjunto de test.

Otra de las métricas para analizar los errores es la matriz de confusión, donde cada fila de la matriz representa las instancias de una clase real,

mientras que cada columna representa las instancias de una clase predicha por el modelo, para así poder ver con que clases pueden producirse falsos positivos y negativos. La diagonal de izquierda a derecha representa los datos que han sido predichos correctamente. La matriz de confusión para el modelo RoBERTa$_{Bio}$-BT40 de la tarea NER se muestra en la Figura 6.1.

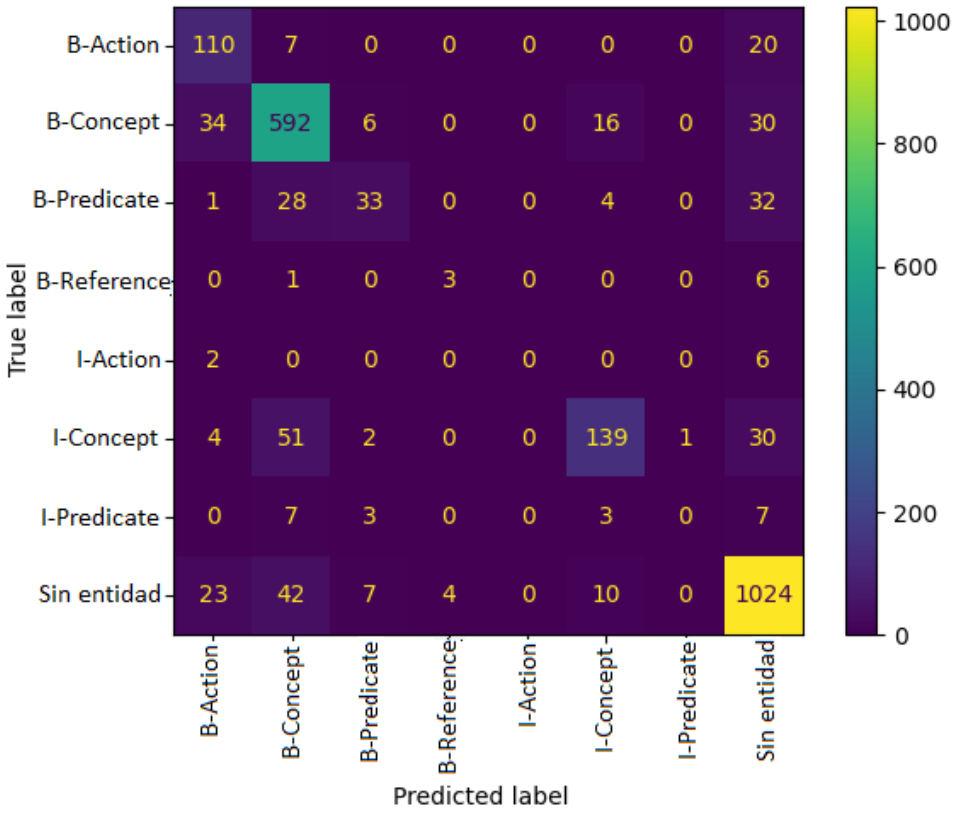


Figura 6.1: Matriz de confusión del modelo de la tarea NER.

Como se puede ver en la imagen, hay una gran parte de las muestras clasificadas en la clase correcta. Sin embargo, la clase que más confusión ha ocasionado es la clase sin entidad, la cual tiene falsos negativos (como con *B-Action* y *B-Concept*), pero sobre todo falsos positivos (última columna), es razonable que se confunda con clases más frecuentes, como *B-Concept*, pero es peligroso que se confunda en el caso de *B-Predicate*, ya que es prácticamente un tercio de los datos, los que se han clasificado erróneamente como sin entidad. En general se puede ver una tendencia de las clases minoritarias a etiquetarse como sin entidad, lo cual es esperable ya que se trata de la clase más representada.

También existen confusiones entre entidades, como *I-Concept* y *B-Concept*, las cuales representan a dos palabras de una misma entidad, por lo que la confusión probablemente esté relacionada con eso, ya que al fin y al cabo son parte de una misma entidad, sólo que el modelo tiene mayor tendencia a etiquetar la palabra como si esta fuese la primera o única de la entidad (*B-Concept*). Esto es normal ya que las entidades etiquetadas con *B-* están más representadas, debido a que muchas entidades son de una única palabra.

También hay una fuerte confusión entre *B-Predicate* y *B-Concept*, así como también entre *B-Concept* y *B-Action*, lo cual si que indica un problema del modelo para distinguir estas clases entre sí, lo que puede estar influenciado por que las palabras de estas entidades pueden aparecer en contextos similares, ya que el modelo *transformer* tiene una fuerte influencia de dicho contexto.

Respecto a la tarea de RE, en la Tabla 6.2 se muestran la métricas de evaluación del mejor modelo obtenido con RoBERTa$_{Clinical}$-SN-LDA2, con las mismas métricas que se han explicado para la tarea de NER.

Tabla 6.2: Métricas de evaluación para las relaciones.

Relación	Precisión	*Recall*	*F1Score*	Muestras
Sin relación	0.99	0.95	0.97	47438
arg	0.10	0.38	0.15	29
causes	0.26	0.18	0.21	34
domain	0.26	0.44	0.33	43
entails	0.25	0.05	0.09	19
has-part	0.00	0.00	0.00	1
has-property	0.06	0.13	0.08	71
in-context	0.09	0.24	0.13	196
in-place	0.10	0.28	0.15	78
in-time	0.23	0.17	0.19	42
is-a	0.10	0.48	0.16	52
part-of	0.00	0.00	0.00	3
same-as	0.00	0.00	0.00	16
subject	0.09	0.52	0.16	100
target	0.10	0.60	0.17	130
Media macro	0.18	0.29	0.19	48252
Media ponderada	0.98	0.94	0.96	48252

En la tabla se puede apreciar que la mayoría de clases tienen unas métricas muy bajas y que la clase sin relación es la que obtiene unas métricas

muy altas, debido a que la mayoría de muestras pertenecen a ésta. De forma general se puede observar que la métrica más alta es el *recall*, lo que indica que se está detectando una buena parte de las relaciones, pero a la vez se existe confusión entre las clases, ya que la precisión es muy baja.

Como en el caso de NER las relaciones con muy pocas muestras no han sido detectadas por el modelo, como *has-part*, *part-of* y *same-as*, lo cual es razonable teniendo en cuenta la cantidad de muestras de otras clases. Sin embargo las clases mayoritarias, como *in-context* y *target*, no han tenido mucho mejores métricas que otras clases menos representadas, lo que puede significar que seguramente no sean tan separables del resto, como por ejemplo, la clase *domain* o *causes*.

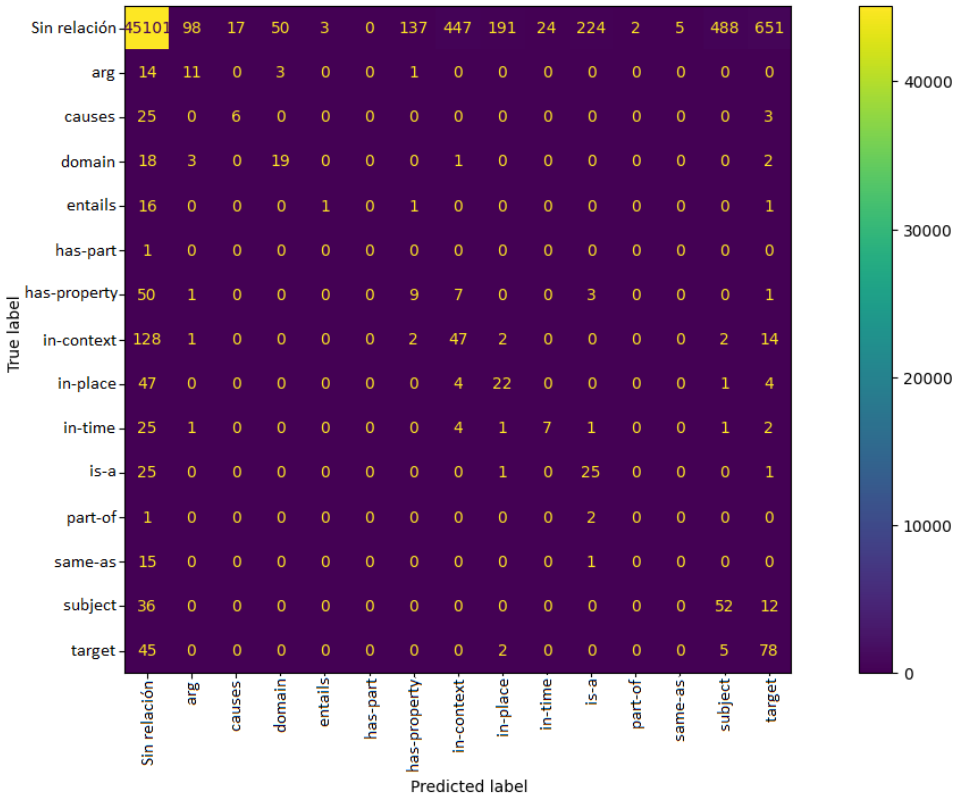


Figura 6.2: Matriz de confusión del modelo de la tarea RE.

Para visualizar mejor los errores cometidos por el modelo, en la Figura 6.2 se muestra la matriz de confusión de esta tarea. Como era de esperar la mayor confusión se da entre la clase sin relación y el resto de clases (primera fila y primera columna). Los mayores errores parecen ser los falsos

positivos de las distintas relaciones, clasificadas como relaciones cuando no lo son, especialmente en las relaciones *target*, *subject* e *in-context*, que son las relaciones mayoritarias. Esto puede ser debido a que se dio poco peso a la clase sin relación, pero se decidió así por que al dar mucho peso a la misma, no se identificaban apenas relaciones, por lo que se asumió el número de falsos positivos para que el modelo fuera capaz de reconocer más relaciones.

También existen casos de confusión entre entidades, pero estos son minoría, ya que para mitigar estos se entrenó un modelo específico para realizar esta distinción, pero al haber tenido errores con la clase sin relación, estos errores no se han podido corregir con el segundo modelo, ya que sólo se repuntuan los pesos de las relaciones que el primer modelo ha identificado como relación.

6.2. Análisis de los modelos utilizados

Dicho esto, también es importante analizar que modelos o aproximaciones han proporcionado mejores resultados. Los mejores resultados en la tarea de NER han sido proporcionados por el modelo de *transformer* entrenado con texto biomédico y reentrenado con los datos de entrenamiento, lo que indica que el *Transfer Learning* entre modelos previamente entrenados con textos del ámbito, resulta positivo. También es visible que utilizar un modelo completo de *transformer* (en lugar de utilizar sólo los *embeddings*) es más conveniente que utilizar un clasificador aparte.

Respecto a los resultados de los distintos modelos de *transformers* utilizados, los mejores resultados han sido los entrenados con texto biomédico y clínico (RoBERTa$_{Bio}$ y RoBERTa$_{Clinical}$). Por lo tanto, indica que el uso de modelos ya preentrenados en el ámbito, sirve para poder utilizarlos como base, ya que están más adaptados a ese tipo de textos, lo cual es muy útil tanto para utilizarlo como codificadores, como para reentrenar el modelo completo, como se observa en las tablas de resultados de la tarea NER.

También el modelo general de multilingual BERT (mBERT) ha obtenido muy buenos resultados, similares a los obtenidos por los modelos biomédicos, lo cual indica que también es factible ajustar un modelo general para el ámbito biomédico, ya que se trata de un modelo entrenado con una gran cantidad de texto.

Por último, es necesario mencionar que el modelo RoBERTa$_{NER}$ ajus-

tado para la tarea de NER, no ha funcionado tan bien respecto a los otros dos modelos preentrenados. Esto posiblemente sea debido a que las etiquetas para las que fue ajustado al modelo difieren de las utilizadas en este trabajo, lo que puede que desajuste las representaciones para estas etiquetas.

Por otro lado, los mejores resultados en RE se han obtenido utilizando el modelo *transformer* como codificador, junto con dos clasificadores de Redes Neuronales, además de utilizar algoritmos para aumentar el número de muestras y realizar una reducción de dimensionalidad que ayude a que los datos sean más separables. En esta tarea también queda claro que los modelos *transformers* entrenados con texto biomédico, favorecen la resolución de tareas en este área, ya que el modelo que mejores resultados ha obtenido ha sido RoBERTa$_{Clinical}$, y en segundo lugar RoBERTa$_{Bio}$.

De la misma forma que NER, los modelos generales como mBERT o BETO no han dado tan buenos resultados, especialmente en esta tarea, posiblemente debido a que las relaciones se dan entre términos relacionados con la medicina y los modelos médicos ayudan en esta tarea.

Capítulo 7

Conclusiones y trabajo futuro

7.1. Conclusiones

El reconocimiento de entidades y relaciones sigue siendo un reto para el análisis y clasificación de textos biomédicos. En este trabajo se han realizado varias propuestas para dichas tareas en textos biomédicos en español, como el uso de modelos de *transformers* generales y preentrenados en este dominio.

En este trabajo se han presentado varias propuestas para el desarrollo de la tarea de NER y RE. Tras la investigación del estado del arte de ambas tareas en el ámbito biomédico en español, se generaron lineas de trabajo atendiendo a las necesidades encontradas, como utilizar modelos previamente entrenados en texto biomédico en español y realizar un aumento de datos mediante varias técnicas para tratar el desbalanceo propio de estas tareas, lo cual ha resultado satisfactorio, especialmente en la tarea de NER.

Como conclusión general del trabajo, queda de manifiesto que el uso de modelos de *transformers* es una de las mejores opciones a la hora de desarrollar tareas de reconocimiento de entidades y extracción de relaciones, así como también, que es importante explorar la gran variedad que existe de estos modelos.

El uso de modelos adaptados al tipo de textos con los que se va a realizar también juega un papel importante a la hora de la selección de los modelos, aunque como se ha visto en la sección anterior 6.2, también se puede concluir que los modelos generales ajustados a la tarea también dan buenos

resultados.

Por lo tanto, la contribución de este trabajo es el uso de modelos de *transformers* preentrenados con textos biomédicos en español, para realizar la tarea de NER y RE en texto biomédico en español.

Durante el desarrollo del trabajo, se probaron una serie de modelos más tradicionales, como SVM o Redes Neuronal, que a pesar de dar buenos resultados no llegaron a superar los obtenidos por el uso del *transformer* en la tarea de NER. También se vio que el aumento de datos puede ayudar a los modelos a generalizar mejor y a ser capaces de detectar clases menos frecuentes, como se ha visto en el desarrollo de ambas tareas. El hecho de utilizar modelos entrenados con textos en español y en el área biomédica y clínica, ha supuesto una mejora respecto a modelos generales como mBERT o BETO en todas las pruebas realizadas, dejando claro que el uso de estos modelos específicos es una buena opción.

7.2. Trabajo futuro

Como se ha mencionado en la sección dedicada a analizar los errores 6.1, el mayor problema encontrado tras analizar los resultados, es la dificultad de los modelos en clasificar los datos cuando existe una clase con una frecuencia de aparición muchísimo mayor que el resto de clases, como se ha observado especialmente en la tarea RE. Es por esto, por lo que se propone una linea de trabajo futuro dedicada a tratar de solventar este problema, ya que, a pesar de observar mejoras en los resultados con el uso de *Data Augmentation*, se siguen observando problemas a la hora de discriminar entre las distintas clases menos comunes.

Otra posible linea de investigación podría ser el uso de *Transfer Learning* con texto biomédico en otras lenguas, como por ejemplo el inglés, ya que existe mucho mas material en esta lengua que se podría utilizar junto con modelos plurilingües como mBERT, y así poder transferir la información del inglés para tener disponible más material de entrenamiento para los modelos. O bien utilizar modelos entrenados en texto biomédico en múltiples lenguas con modelos multilingües, para así abarcar más cantidad de textos.

Referencias

Aberdeen, J., Bayer, S., Yeniterzi, R., Wellner, B., Clark, C., Hanauer, D., Malin, B., y Hirschman, L. (2010). The mitre identification scrubber toolkit: design, training, and assessment. *International journal of medical informatics*, 79(12):849–859.

Abney, S. (2002). Bootstrapping. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 360–367.

Alfaro-González, D., Pérez-Perera, D., González-Rodríguez, G., Otaño-Barrera, A. J., y Linares, R. C. (2021). uhkd4 at ehealth-kd challenge 2021: Deep learning approaches for knowledge discovery from spanish biomedical documents. In *IberLEF@ SEPLN*, pages 703–711.

Andrés, E. (2021). Ixa at ehealth-kd challenge 2021: Generic sequence labelling as relation extraction approach. In *IberLEF@ SEPLN*, pages 668–674.

Antons, D., Grünwald, E., Cichy, P., y Salge, T. O. (2020). The application of text mining methods in innovation research: current state, evolution patterns, and development priorities. *R&D Management*, 50(3):329–351.

Aone, C., Halverson, L., Hampton, T., y Ramos-Santacruz, M. (1998). Sra: Description of the ie2 system used for muc-7. In *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29-May 1, 1998*.

Arkhipov, M., Trofimova, M., Kuratov, Y., y Sorokin, A. (2019). Tuning multilingual transformers for language-specific named entity recognition. In *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing*, pages 89–93.

Bach, N. y Badaskar, S. (2007). A review of relation extraction. *Literature review for Language and Statistics II*, 2:1–15.

Balakrishnama, S. y Ganapathiraju, A. (1998). Linear discriminant analysis-a brief tutorial. *Institute for Signal and information Processing*, 18(1998):1–8.

Black, W. J., Rinaldi, F., y Mowatt, D. (1998). Facile: Description of the ne system used for muc-7. In *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29-May 1, 1998*.

Bojanowski, P., Grave, E., Joulin, A., y Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Breiman, L. (1996). Some properties of splitting criteria. *Machine learning*, 24(1):41–47.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Brin, S. (1998). Extracting patterns and relations from the world wide web. In *International workshop on the world wide web and databases*, pages 172–183. Springer.

Canete, J., Chaperon, G., Fuentes, R., Ho, J.-H., Kang, H., y Pérez, J. (2020). Spanish pre-trained bert model and evaluation data. *Pml4dc at iclr*, 2020:1–10.

Carrino, C. P., Armengol-Estapé, J., de Gibert Bonet, O., Gutiérrez-Fandiño, A., Gonzalez-Agirre, A., Krallinger, M., y Villegas, M. (2021a). Spanish biomedical crawled corpus: A large, diverse dataset for spanish biomedical language models.

Carrino, C. P., Armengol-Estapé, J., Gutiérrez-Fandiño, A., Llop-Palao, J., Pàmies, M., Gonzalez-Agirre, A., y Villegas, M. (2021b). Biomedical and clinical language models for spanish: On the benefits of domain-specific pretraining in a mid-resource scenario.

Chawla, N. V., Bowyer, K. W., Hall, L. O., y Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

Chieu, H. L. y Ng, H. T. (2003). Named entity recognition with a maximum entropy approach. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, pages 160–163.

Cho, H. y Lee, H. (2019). Biomedical named entity recognition using deep neural networks with contextual information. *BMC bioinformatics*, 20(1):1–11.

Cho, K., Van Merriënboer, B., Bahdanau, D., y Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., y Stoyanov, V. (2019). Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Cover, T. y Hart, P. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27.

Cuadros, M. (2021). Vicomtech at ehealth-kd challenge 2021: Deep learning approaches to model health-related text in spanish.

Devlin, J., Chang, M.-W., Lee, K., y Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Doğan, R. I., Leaman, R., y Lu, Z. (2014). Ncbi disease corpus: a resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47:1–10.

Edunov, S., Ott, M., Auli, M., y Grangier, D. (2018). Understanding back-translation at scale. *arXiv preprint arXiv:1808.09381*.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

Ethayarajh, K. (2019). How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. *arXiv preprint arXiv:1909.00512*.

Friedman, C., Rindflesch, T. C., y Corn, M. (2013). Natural language processing: state of the art and prospects for significant progress, a workshop sponsored by the national library of medicine. *Journal of biomedical informatics*, 46(5):765–773.

Gado, N. E. I., Grall-Maës, E., y Kharouf, M. (2017). Linear discriminant analysis based on fast approximate svd. In *International Conference on Pattern Recognition Applications and Methods*, volume 2, pages 359–365. SCITEPRESS.

García García, M. E. (2021). Desambiguación de acrónimos en literatura médica española.

Giachanou, A., Gonzalo, J., Mele, I., y Crestani, F. (2017). Sentiment propagation for predicting reputation polarity. In *European Conference on Information Retrieval*, pages 226–238. Springer.

Gonzalez-Agirre, A., Marimon, M., Intxaurrondo, A., Rabal, O., Villegas, M., y Krallinger, M. (2019). Pharmaconer: Pharmacological substances, compounds and proteins named entity recognition track. In *Proceedings of The 5th Workshop on BioNLP Open Shared Tasks*, pages 1–10.

Grishman, R. y Sundheim, B. M. (1996). Message understanding conference-6: A brief history. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.

Hasegawa, T., Sekine, S., y Grishman, R. (2004). Discovering relations among named entities from large corpora. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 415–422.

Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., y Scholkopf, B. (1998). Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28.

Hochreiter, S. y Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Huang, C.-C. y Lu, Z. (2016). Community challenges in biomedical text mining over 10 years: success, failure and the future. *Briefings in bioinformatics*, 17(1):132–144.

Kafle, K., Yousefhussien, M., y Kanan, C. (2017). Data augmentation for visual question answering. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 198–202.

Kambhatla, N. (2006). Minority vote: at-least-n voting improves recall for extracting relations. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 460–466.

Khan, P., Kader, M. F., Islam, S. R., Rahman, A. B., Kamal, M. S., Toha, M. U., y Kwak, K.-S. (2021). Machine learning and deep learning approaches for brain disease diagnosis: principles and recent advances. *IEEE Access*, 9:37622–37655.

Khurana, D., Koli, A., Khatter, K., y Singh, S. (2017). Natural language processing: State of the art, current trends and challenges. *arXiv preprint arXiv:1708.05148*.

Kim, J.-D., Ohta, T., Tsuruoka, Y., Tateisi, Y., y Collier, N. (2004). Introduction to the bio-entity recognition task at jnlpba. In *Proceedings of the international joint workshop on natural language processing in biomedicine and its applications*, pages 70–75. Citeseer.

Krallinger, M., Leitner, F., Rodriguez-Penagos, C., y Valencia, A. (2008). Overview of the protein-protein interaction annotation extraction task of biocreative ii. *Genome biology*, 9(2):1–19.

LeCun, Y., Bottou, L., Bengio, Y., y Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lee, C.-P. y Lin, C.-J. (2013). A study on l2-loss (squared hinge-loss) multiclass svm. *Neural computation*, 25(5):1302–1323.

Li, J., Cheng, J.-h., Shi, J.-y., y Huang, F. (2012). Brief introduction of back propagation (bp) neural network algorithm and its improvement. In *Advances in computer science and information engineering*, pages 553–558. Springer.

Li, J., Sun, A., Han, J., y Li, C. (2020). A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):50–70.

Li, J., Sun, Y., Johnson, R. J., Sciaky, D., Wei, C.-H., Leaman, R., Davis, A. P., Mattingly, C. J., Wiegers, T. C., y Lu, Z. (2016). Biocreative v cdr task corpus: a resource for chemical disease relation extraction. *Database*, 2016.

Lin, Y., Shen, S., Liu, Z., Luan, H., y Sun, M. (2016). Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2124–2133.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., y Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Loh, W.-Y. (2011). Classification and regression trees. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 1(1):14–23.

Machado, G., Mendoza, M. R., y Corbellini, L. G. (2015). What variables are important in predicting bovine viral diarrhea virus? a random forest approach. *Veterinary research*, 46(1):1–15.

Mejova, Y. y Srinivasan, P. (2011). Exploring feature definition and selection for sentiment classifiers. In *Fifth international AAAI conference on weblogs and social media*.

Mikolov, T., Le, Q. V., y Sutskever, I. (2013a). Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., y Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

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

Miranda-Escalada, A., Gascó, L., Lima-López, S., Farré-Maduell, E., Estrada, D., Nentidis, A., Krithara, A., Katsimpras, G., Paliouras, G., y

Krallinger, M. (2022). Overview of distemist at bioasq: Automatic detection and normalization of diseases from clinical texts: results, methods, evaluation and multilingual resources.

Monteagudo-Garcıa, L., Marrero-Santos, A., Fernández-Arias, M., y Canizares-Dıaz, H. (2021). Uh-mmm at ehealth-kd challenge 2021. In *Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2021)*.

Morwal, S., Jahan, N., y Chopra, D. (2012). Named entity recognition using hidden markov model (hmm). *International Journal on Natural Language Computing (IJNLC) Vol*, 1.

Nadeau, D. y Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Nadkarni, P. M., Ohno-Machado, L., y Chapman, W. W. (2011). Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, 18(5):544–551.

Naseem, U., Razzak, I., Khan, S. K., y Prasad, M. (2021). A comprehensive survey on word representation models: From classical to state-of-the-art word representation language models. *Transactions on Asian and Low-Resource Language Information Processing*, 20(5):1–35.

Nédellec, C. (2005). Learning language in logic-genic interaction extraction challenge. In *4. Learning language in logic workshop (LLL05)*. ACM-Association for Computing Machinery.

Otegi, A., Agirre, A., Campos, J. A., Soroa, A., y Agirre, E. (2020). Conversational question answering in low resource scenarios: A dataset and case study for basque. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 436–442.

Patle, A. y Chouhan, D. S. (2013). Svm kernel functions for classification. In *2013 International Conference on Advances in Technology and Engineering (ICATE)*, pages 1–9. IEEE.

Pavanelli, L., Schneider, E. T. R., Gumiel, Y. B., Ferreira, T. C., de Oliveira, L. F. A., de Souza, J. V. A., Paiva, G. P. M., e Oliveira, L. E. S., Moro, C.

M. C., Paraiso, E. C., et al. (2021). Pucrj-pucpr-ufmg at ehealth-kd challenge 2021: A multilingual bert-based system for joint entity recognition and relation extraction. In *IberLEF@ SEPLN*.

Pawar, S., Palshikar, G. K., y Bhattacharyya, P. (2017). Relation extraction: A survey. *arXiv preprint arXiv:1712.05191*.

Pennington, J., Socher, R., y Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., y Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Piad-Morfis, A., Estevez-Velarde, S., Gutierrez, Y., Almeida-Cruz, Y., Montoyo, A., y Muñoz, R. (2021). Overview of the ehealth knowledge discovery challenge at iberlef 2021. *Procesamiento del Lenguaje Natural*, 67(0):233–242.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P. J., et al. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

Ramshaw, L. A. y Marcus, M. P. (1999). Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.

Refaeilzadeh, P., Tang, L., y Liu, H. (2009). Cross-validation. *Encyclopedia of database systems*, 5:532–538.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Sachan, D. S., Xie, P., Sachan, M., y Xing, E. P. (2018). Effective use of bidirectional language modeling for transfer learning in biomedical named entity recognition. In *Machine learning for healthcare conference*, pages 383–402. PMLR.

Sagi, O. y Rokach, L. (2018). Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249.

Sanh, V., Debut, L., Chaumond, J., y Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61:85–117.

Segura-Bedmar, I., Martínez Fernández, P., y Sánchez Cisneros, D. (2011). The 1st ddiextraction-2011 challenge task: Extraction of drug-drug interactions from biomedical texts.

Sharma, R., Sharma, K., y Khanna, A. (2020). Study of supervised learning and unsupervised learning. *International Journal for Research in Applied Science and Engineering Technology*, 8(6):588–593.

Sharnagat, R. (2014). Named entity recognition: A literature survey. *Center For Indian Language Technology*, pages 1–27.

Smith, L. y Tanabe, L. (2008). nee ando rj, kuo cj, chung if, hsu cn, lin ys, klinger r, friedrich cm, ganchev k, et al. overview of biocreative ii gene mention recognition. *Genome Biol*, 9(2):2.

Soares, F., Villegas, M., Gonzalez-Agirre, A., Krallinger, M., y Armengol-Estapé, J. (2019). Medical word embeddings for Spanish: Development and evaluation. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 124–133, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Song, Y.-Y. y Ying, L. (2015). Decision tree methods: applications for classification and prediction. *Shanghai archives of psychiatry*, 27(2):130.

Sun, A. y Grishman, R. (2012). Active learning for relation type extension with local and global data views. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1105–1112.

Sutton, R. S. y Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Uc-Cetina, V., Navarro-Guerrero, N., Martin-Gonzalez, A., Weber, C., y Wermter, S. (2022). Survey on reinforcement learning for language processing. *Artificial Intelligence Review*, pages 1–33.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., y Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, L. L., Lo, K., Chandrasekhar, Y., Reas, R., Yang, J., Eide, D., Funk, K., Kinney, R., Liu, Z., Merrill, W., et al. (2020). Cord-19: The covid-19 open research dataset. *ArXiv*.

Wei, C.-H., Peng, Y., Leaman, R., Davis, A. P., Mattingly, C. J., Li, J., Wiegers, T. C., y Lu, Z. (2016). Assessing the state of the art in biomedical relation extraction: overview of the biocreative v chemical-disease relation (cdr) task. *Database*, 2016.

Willett, P. (1988). *Document retrieval systems*. Taylor Graham Publishing.

Willett, P. (2006). The porter stemming algorithm: then and now. *Program*.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Yan, H., Deng, B., Li, X., y Qiu, X. (2019). Tener: adapting transformer encoder for named entity recognition. *arXiv preprint arXiv:1911.04474*.

Yang, X., Song, Z., King, I., y Xu, Z. (2022). A survey on deep semi-supervised learning. *IEEE Transactions on Knowledge and Data Engineering*.

Yao, L., Liu, H., Liu, Y., Li, X., y Anwar, M. W. (2015). Biomedical named entity recognition based on deep neutral network. *Int. J. Hybrid Inf. Technol*, 8(8):279–288.

Yin, W., Kann, K., Yu, M., y Schütze, H. (2017). Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*.

Zhang, S. y Elhadad, N. (2013). Unsupervised biomedical named entity recognition: Experiments with clinical and biological texts. *Journal of biomedical informatics*, 46(6):1088–1098.

Zhang, Z. y Sabuncu, M. (2018). Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in neural information processing systems*, 31.

Zhong, Z. y Chen, D. (2020). A frustratingly easy approach for entity and relation extraction. *arXiv preprint arXiv:2010.12812*.

Zhou, G., Su, J., Zhang, J., y Zhang, M. (2005). Exploring various knowledge in relation extraction. In *Proceedings of the 43rd annual meeting of the association for computational linguistics (acl'05)*, pages 427–434.

Zhu, X. y Goldberg, A. B. (2009). Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130.