# Desarrollo de un sistema de clasificación multi-etiqueta basado en Transformers para la clasificación de códigos eCIE-O-3.1


**Trabajo Fin de Máster**

**Laura Valencia Gracia**

Trabajo de investigación para el

Máster en Tecnologías del Lenguaje

Universidad Nacional de Educación a Distancia

Dirigido por la

**Prof. Dr. Da. Laura Plaza Morales**

Octubre 2021

# Agradecimientos

El presente trabajo de investigación fue realizado bajo la supervisión de Laura Plaza Morales a quien me gustaría mostrar mi más profundo agradecimiento por su guía, sus consejos, su dedicación y paciencia para que esto saliera adelante de manera exitosa.

A mis padres, a mi hermana y a toda mi familia, gracias a quienes soy quien soy y hacia quienes sólo puedo expresar mi sincero agradecimiento por apoyarme durante toda esta etapa académica. En especial quiero agradecer a mi madre todo apoyo y un abrazo reconfortante para renovar energías.

A Rodolfo, mi novio, por todo el cariño que me has mostrado cuando las cosas me superaban. Gracias por enseñarme a ser más fuerte y feliz y por saber cómo alegrarme un día malo. Gracias por estar a mi lado siempre.

A mis amigos, que siempre han estado ahí con palabras de apoyo, sacándome una sonrisa en días en los que todo cuesta más y ofreciendo ideas para este trabajo. Sin vosotros, este trabajo sería mucho peor.

Muchas gracias a todos.

# Resumen

En el presente trabajo se proponen diferentes arquitecturas basadas en *Tranformers* que pretenden resolver un problema de clasificación multi-etiqueta de códigos morfológicos eCIE-O-3.1, que son los dedicados a las neoplasias. Para ello, se ha utilizado el conjunto de datos facilitados en la tarea competitiva CANTEMIST cuyo objetivo era presentar un sistema capaz de hacer una clasificación multi-etiqueta de códigos morfológicos eCIE-O-3.1 en informes médicos. Los datos de CANTEMIST se han utilizado para entrenar los diferentes modelos propuestos, junto con diferentes experimentos que se han realizado para tratar de mejorar el rendimiento de los modelos base. Los modelos base consisten en diferentes modelos BERT pre-entrenados con distintos conjuntos de datos y diferentes idiomas: algunos modelos son específicamente empleados en español mientras que otros son multi-idioma, algunos modelos han sido pre-entrenados con textos médicos y otros con textos de ámbito general.

Se ha realizado un preprocesamiento de los textos médicos para que puedan ser entrenados por el modelo, ya que BERT necesita un tipo específico de datos de entrada.

Finalmente, se ha realizado una evaluación exhaustiva del sistema de clasificación sobre el conjunto de test de CANTEMIST para determinar su rendimiento. Se han comparado los resultados obtenidos con los de los sistemas que presentaron los participantes de CANTEMIST en 2020. Los resultados obtenidos muestran que el procedimiento empleado es capaz de realizar una clasificación multi-etiqueta con un buen acierto, aunque con limitaciones y problemas debido a la aproximación empleada.

# Abstract

The present work proposes different architectures based on *Tranformers* that intend to solve a problem of multi-label classification of morphological codes eCIE-O-3.1, which are those dedicated to neoplasms. For this purpose, the dataset provided in the competitive task CANTEMIST was used. The objective was to present a system capable of making a multi-label classification of morphological codes eCIE-O-3.1 in medical reports. The different proposed models will be trained with this dataset, together with different experiments that have been carried out to try to improve the performance of the base models. The base models consist of different pre-trained BERT models with different data sets and different languages: some models are specifically used in Spanish while others are multi-language, some models have been pre-trained with medical texts and others with medical texts. general scope.

Medical texts have been preprocessed so that they can be trained by the model, since BERT needs a specific type of input data.

Finally, an extensive evaluation is performed on the CANTEMIST test set to determine the performance of our approach in the multi-label classification task. The results obtained have been compared with the systems presented by the CANTEMIST participants in 2020. The results obtained show that this procedure used is able to perform a multi-label classification with good results, although with limitations and problems due to the approach used.

# Índice general

# Índice de Figuras

# Índice de Tablas

# Capítulo 1

# Introducción

El presente capítulo tiene como misión presentar al lector la motivación y los objetivos del presente trabajo, así como la definición del problema de investigación que nos ocupa. Además, se indica la estructura y los principales puntos abordados en este trabajo de final de máster.

## 1.1. Motivación

La codificación clínica se puede definir como la traducción de la terminología médica escrita por un médico para describir la queja, el problema, el diagnóstico, el tratamiento o el motivo de un paciente para buscar atención médica, en un formato codificado. La necesidad de codificar la información clínica de un paciente es reconocida a nivel nacional e internacional y una de las codificaciones más ampliamente utilizadas es la codificación CIE o ICD (Ministerio de Sanidad, 2020) (ICD son las siglas en inglés de *International Statistical Classification of Diseases and Related Health Problems*, en español corresponde a la Clasificación Internacional de Enfermedades). Esta codificación, aunque existen otras, permite el almacenamiento, recuperación y análisis de información de salud para la toma de decisiones basada en evidencia, así como compartir y comparar información sanitaria entre hospitales, regiones, entornos y países. Esta codificación permite, además, hacer comparaciones de datos en la misma ubicación en diferentes períodos de tiempo. Cada enfermedad tiene asignado un código CIE, y si un paciente tiene una enfermedad crónica, como es el caso de padecer diabetes o una enfermedad cardíaca, su código CIE normalmente seguirá sus registros médicos.

Los datos clínicos se deben registrar de forma precisa y coherente según

normas nacionales bien definidas para permitir su uso en análisis estadísticos (Organization, 2021). La información extraída de una codificación clínica precisa refleja mejor el patrón de práctica de los médicos y proporciona una base sólida para el proceso de toma de decisiones. El uso de herramientas como los indicadores clínicos impulsa la necesidad de contar con información clínica registrada de forma precisa y coherente. Es por eso que una buena calidad en la codificación es imprescindible para realizar una valoración adecuada de la asistencia prestada en los centros asistenciales. Ésta permite optimizar los beneficios y oportunidades que ofrecen los sistemas de clasificación de pacientes para la gestión sanitaria, tanto a nivel clínico como a nivel económico-financiero (Bowman, 2008).

La Clasificación Internacional de Enfermedades (CIE) es una herramienta de diagnóstico estándar creada por la Organización Mundial de la Salud (OMS) para monitorear la incidencia y prevalencia de enfermedades y afecciones relacionadas. La CIE sirve para clasificar las distintas enfermedades, patologías y trastornos conocidos en el mundo de acuerdo a sus síntomas, las causas, los tipos de personas afectadas por la enfermedad, etc. El objetivo de esta clasificación es poder comparar y compartir datos de manera consistente y estándar entre hospitales, regiones y países en distintos períodos de tiempo. Actualmente, la codificacón CIE que está vigente desde 1990 es el CIE-10 (Ministerio de Sanidad, 2020). La siguiente edición, que será la CIE-11, ya está aprobada y entrará en vigor en 2022. En la edición actual de CIE-10, los códigos topográficos están compuestos por una letra en la primera posición seguida de números; es decir, se trata de una codificación alfanumérica, mientras que los códigos morfológicos, que son los que utilizaremos en este trabajo, se componen siete caracteres de los cuales 6 son dígitos, con una barra / que separa el 4ª del 5º dígito. Por ejemplo, el código morfológico 8000/6 hace referencia a la neoplasia metastásica. En la Tabla 1.1 se muestran todos los códigos CIE-10 ordenados por tipos de afecciones. Cada afección puede ser asignada a una categoría y cada una de estas categorías puede incluir un grupo de enfermedades similares. Esta estructura permitió incluir un mayor número de códigos y términos diagnósticos en comparación con las ediciones anteriores, cuyos códigos eran sólo numéricos.

La tarea de codificación de registros clínicos es llevada a cabo en hospitales y centros sanitarios por profesionales con años de experiencia y con un amplio conocimiento de la terminología médica, habilidades analíticas y una

| Código | Título |
|---|---|
| A00-B99 | Ciertas enfermedades infecciosas y parasitarias. |
| C00-D49 | Neoplasias. |
| D50-D89 | Enfermedades de la sangre y de los órganos hematopoyéticos y otros trastornos que afectan el mecanismo de la inmunidad. |
| E00-E90 | Enfermedades endocrinas, nutricionales y metabólicas. |
| F00-F99 | Trastornos mentales y del comportamiento. |
| G00-G99 | Enfermedades del sistema nervioso. |
| H00-H59 | Enfermedades del ojo y sus anexos. |
| H60-H95 | Enfermedades del oído y de la apófisis mastoides. |
| I00-I99 | Enfermedades del sistema circulatorio. |
| J00-J99 | Enfermedades del sistema respiratorio. |
| K00-K93 | Enfermedades del aparato digestivo. |
| L00-L99 | Enfermedades de la piel y el tejido subcutáneo. |
| M00-M99 | Enfermedades del sistema osteomuscular y del tejido conectivo. |
| N00-N99 | Enfermedades del aparato genitourinario. |
| O00-O99 | Embarazo, parto y puerperio. |
| P00-P96 | Ciertas afecciones originadas en el periodo perinatal. |
| Q00-Q99 | Malformaciones congénitas, deformidades y anomalías cromosómicas. |
| R00-R99 | Síntomas, signos y hallazgos anormales clínicos y de laboratorio, no clasificados en otra parte. |
| S00-T98 | Traumatismos, envenenamientos y algunas otras consecuencias de causa externa. |
| V01-Y98 | Causas externas de morbilidad y de mortalidad. |
| Z00-Z99 | Factores que influyen en el estado de salud y contacto con los servicios de salud. |
| U00-U99 | Códigos para situaciones especiales. |

Tabla 1.1: Códigos CIE-10

excelente atención a los detalles para producir altos niveles de precisión. Los codificadores clínicos experimentados tendrán conocimientos avanzados y un historial impresionante de trabajo dentro de los sistemas de atención médica relevantes y en diferentes marcos de codificación, y serán capaces de revisar los expedientes en detalle y convertir los procedimientos médicos documentados, el diagnóstico y los recursos en códigos (AAPC, 2021). Sin embargo, es una práctica muy común externalizar la labor de un codificador por la escasez prolongada de codificadores clínicos a tiempo completo en los hospitales. Por ello, la contratación de un equipo experimentado que sea capaz de administrar la transcripción, facturación, programación de citas y codificación se traduce en una mayor comodidad y una reducción de costes. Estos profesionales pueden averiguar, a partir de notas médicas, por ejemplo, la sala en la que permaneció cada paciente, cuánto tiempo duró su operación, el tiempo de recuperación y cualquier otro tratamiento que recibió después de una cirugía. Con esta información, se elige el código alfanumérico correspondiente y todo queda registrado en el sistema informático de tal manera que cualquier hospital pueda entender las notas médicas sin tener que leer todo el historial y poder planificar la atención futura del paciente. La codificación es laboriosa y requiere mucho tiempo porque un codificador experimentado suele necesitar unos 20 minutos por caso clínico de media (Carla Smith y Dooling, 2019), aunque este promedio necesitaría un ajuste dependiendo de otros factores tales como la complejidad organizacional, combinación de

casos y otras tareas asignadas a cada codificador. Por lo tanto, un sistema de clasificación de código automático puede reducir significativamente el esfuerzo humano.

En este trabajo, nuestro objetivo es automatizar la codificación de historiales clínicos. En concreto, nos vamos a centrar en los códigos C00-D48 que son los que se corresponden con las neoplasias, o tumores. Para ello, vamos a utilizar la CIE-O que se encuentra en la tercera revisión, la CIE-O-3 (ECI, 2020). La CIE-O se trata de una extensión de la *International Statistical Classification of Diseases and Related Health Problems* aplicada al dominio específico de las enfermedades tumorales, y es la codificación estándar para el diagnóstico de neoplasias. La CIE-O consiste en dos sistemas de codificación que describen el tumor:

- El código topográfico, que describe el sitio anatómico de origen (o sistema de órganos) del tumor. El código siempre tiene un prefijo "C", seguido de un número de tres dígitos que indica el sitio (dos dígitos) y el subsitio (un dígito), separados por un punto decimal. Por ejemplo, los códigos C15-C26 hacen referencia a las neoplasias malignas de órganos digestivos. Los códigos de C43-C44 son las neoplasias malignas de piel. Sin embargo, en este trabajo nos vamos a centrar más en los códigos que describimos a continuación.

- El código morfológico/histológico, que describe el tipo celular (o histología) del tumor, junto con el comportamiento (maligno o benigno). Este es el código que veremos a continuación.

En la figura 1.1 muestra el formato de los códigos morfológicos CIE-O. El código morfológico de eCIE-O-3.1, la versión en español de la *International Classification of Diseases for Oncology (ICD-O)*, consta de siete caracteres que son 6 dígitos y una barra "/" que separa los dígitos 4º y 5º. Los cuatro primeros dígitos identifican el tipo histológico de la neoplasia, el quinto número indica su comportamiento y el sexto el grado de diferenciación de la neoplasia o la designación del inmunofenotipo en las leucemias y linfomas. Solamente son obligatorios los caracteres para identificar el tipo histológico del tumor y su comportamiento. El sexto dígito es opcional y su uso será exclusivo para neoplasias malignas primarias (/3) y neoplasias "in situ" (/2). Algunos ejemplos de código CIE-O serían los siguientes:

- 8070/2: Carcinoma epidermoide in situ, SAI.

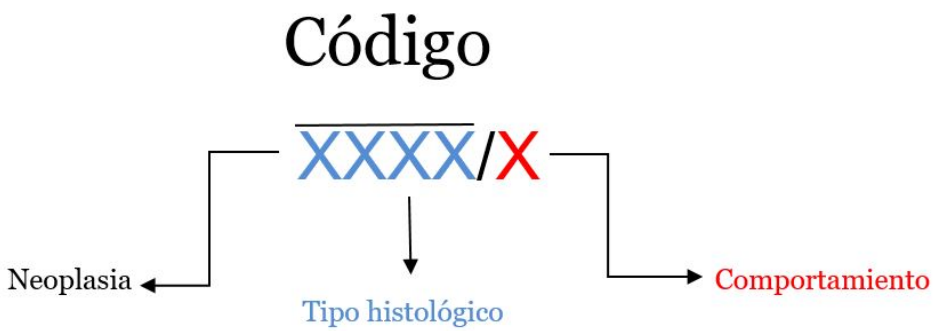



Figura 1.1: Ejemplo de formato de código morfológico CIE-O

- 9261/6: Adamantinoma de huesos largos, metástasis.
- 9150/3: Hemangiopericitoma maligno.
- 8097/6: Carcinoma basocelular, tipo nodular, metástasis.
- 8740/6: Melanoma maligno en nevus de unión, metástasis.

Para realizar una correcta codificación de una neoplasia se necesita determinar el comportamiento de ésta a partir de la morfología o histología obtenida del historial clínico, para determinar cuál es el código correspondiente. De esta forma, es posible comprobar si la neoplasia es maligna primaria, secundaria, in-situ, benigna, de evolución incierta o no especificada. Esto viene indicado con el quinto dígito, tal y como se muestra en la tabla 1.2.

| Codificación | Tipo de neoplasia |
|---|---|
| 0 | 6 Benigno |
| 1 | Evolución incierta o sin especificar |
| 2 | In situ |
| 3 | Maligno, localización primaria |
| 6 | Maligno, localización secundaria (metástasis) |
| 9 | Maligno, sin certeza de si se trata de sitio primario o metastásico.(No se utilizará en hospitalización) |

Tabla 1.2: Tabla de valores para el quinto dígito de CIE-O-3.1

Toda esta información la tienen que ganar los profesionales con estudio y experiencia. Un sistema de clasificación automático les podría facilitar mucho la labor, no solo a los codificadores, también a los profesionales de la salud, porque de esta manera tienen más y mejores datos para la gestión

de la formación de salud de los pacientes (Shachak Aviv, 2009). En el punto siguiente, se expone el problema que se pretende resolver en este trabajo.

## 1.2. Definición del problema

En la actualidad tenemos a nuestra disposición inmensos volúmenes de información en forma de lenguaje natural, es decir, información desestructurada, presentada como texto libre. Debido a esto, ha surgido como necesidad el estudio de técnicas para procesar toda esta información de dichos volúmenes. Esta tarea se lleva a cabo mediante el uso de equipos de cómputo, siendo el principal problema la dificultad, por parte de las máquinas, de comprender el lenguaje humano. Con el fin de dar solución a este problema surge una disciplina que relaciona directamente la informática y la lingüística, denominada Procesamiento del Lenguaje Natural (PLN). Este trabajo va a centrarse en el creciente interés por el procesamiento de documentos clínicos utilizando técnicas de minería de texto y PLN. En concreto, en una de las áreas de investigación más activas en PLN aplicado a datos clínicos, que es el desarrollo de herramientas que realicen la codificación automática de historias clínicas, es decir, la tarea de extraer de forma autónoma información estructurada valiosa contenida en las notas médicas no estructuradas, siguiendo normas estandarizadas y utilizando terminologías de codificación.

El PLN puede facilitar el uso de información de la literatura y de los registros médicos electrónicos (EHR) en el análisis de datos biomédicos. Nuestro interés en el presente trabajo se centra en esta segunda categoría de documentos: los registros médicos. Un registro de salud o registro médico electrónico es una versión digital de la historia clínica de un paciente. Si bien un EHR contiene los antecedentes médicos y de tratamiento de los pacientes, un sistema EHR está diseñado para ir más allá de los datos clínicos estándar recopilados en el consultorio de un proveedor y puede incluir una visión más amplia de la atención de un paciente. Sin embargo, debido a que el acceso a las EHR es complicado por problemas de privacidad, no siempre es posible la creación de un corpus de entrenamiento que permita automatizar ciertos procesos clínicos.

Los codificadores clínicos traducen la información de los registros médicos de los pacientes a códigos alfanuméricos. La codificación clínica es, como ya hemos comentado, una habilidad especializada que requiere un excelente

conocimiento de la terminología médica, procesos de enfermedades y reglas de codificación, así como atención a los detalles y habilidades analíticas. Como se ha mencionado, el proceso de codificación se basa en inspecciones manuales y juicios basados en la experiencia de los codificadores clínicos, y el esfuerzo requerido para la abstracción de información es extremadamente laborioso, requiere mucho tiempo y es propenso a errores humanos. Las imprecisiones en la codificación pueden resultar en importantes pérdidas de ingresos. Como resultado, existe una demanda creciente de codificadores clínicos precisos y experimentados, pero al mismo tiempo una creciente falta de datos clínicos de calidad (reportes mal redactados, por ejemplo).

Por todo ello, y debido a la gran relevancia del cáncer como una de las principales causas de muerte y al creciente gasto sanitario para tratamientos oncológicos, la OMS ha elaborado un recurso de clasificación específico para oncología conocido como Clasificación Internacional de Enfermedades para Oncología en español, el eCIE-O-3.1 (o ICD-O por sus siglas en inglés) que lleva en uso 25 años.

Nuestra investigación se enmarca en la tarea competitiva CANTEMIST (CANTEMIST, 2021), cuyo objetivo es extraer información crítica de informes clínicos mediante la aplicación de técnicas de procesamiento del lenguaje natural y minería de textos. El concurso internacional CANTEMIST (*Cancer Text Mining Shared Task*) ha demostrado la capacidad de la nueva generación de herramientas de inteligencia artificial para extraer y catalogar información clínica para ayudar a los médicos a realizar diagnósticos más precisos de los casos de cáncer. El concurso fue organizado por el Barcelona Supercomputing Center (BSC) en el marco del plan de promoción de las Tecnologías del Lenguaje (plan TL) de la Secretaría de Estado de Digitalización e Inteligencia Artificial (SEDIA)), del Ministerio de Economía y Transformación Digital. En CANTEMIST han participado 25 equipos de 16 países, entre ellos grupos académicos y empresas como Bosch Center for Artificial Intelligence, Siemens Healthineers y Vicomtech Foundation. Las herramientas que se han mostrado han sido evaluadas por un comité de expertos internacionales, y el resultado del concurso se ha presentado durante el congreso Iberlef, organizado por la Sociedad Española para el Procesamiento del Lenguaje Natural. Los resultados del concurso y la descripción de los diferentes sistemas se pueden consultar en su página web (CANTEMIST, 2021).

El conjunto de datos que recoge CANTEMIST utilizado para esta tarea

está compuesto por informes de casos clínicos, que son bastante parecidos a un EHR y no presentan problemas de privacidad. CANTEMIST consiste en un corpus anotado manualmente generado por expertos en oncología con el objetivo de detectar automáticamente las menciones de términos de morfología tumoral y vincularlas a sus códigos eCIE-O correspondientes.

A continuación se puede ver un ejemplo de un caso clínico de CANTEMIST y sus códigos correspondientes:

Anamnesis

Varón de 75 años Exfumador con antecedentes de dislipidemia, DM, resección de Ca epidermoide de pabellón auricular izquierdo y antecedentes oncológicos familiares Padre con hepatocarcinoma por el cual falleció, hermana y prima con cáncer de ovario e hijo adenocarcinoma de páncreas el cual falleció a los 46 años y a quien se le realizo la secuenciación de BCRA tras la consulta de consejo genético dando como resultado que era portador de una mutación deletérea c.6275_6276delTT; p.Leu2092fs en el exón 11 del gen BRCA2.
Nuestro paciente fue diagnosticado de adenocarcinoma de próstata el cual fue tratado con prostatectomía radical en el 2000. En 2008 tiene una recidiva bioquímica manejada con radioterapia. En Octubre de 2013 presenta una nueva recidiva bioquímica pero sin evidencia de enfermedad a distancia hasta que en Diciembre de 2013 cuando se evidencio progresión ganglionar y se inicio bloqueo hormonal simple.
Durante los respectivos controles posteriores se evidenció en TAC de enero 2016 masa en cola de páncreas de 5.6 cm con múltiples adenopatías retroperitoneales, peripancreáticas, celiacas, en hilio hepático y en ligamento gastro-hepático diagnosticándose tras la realización de una biopsia de la lesión de adenocarcinoma pobremente diferenciado de páncreas.

Exploración física

Pruebas complementarias

Diagnóstico

Tratamiento

Evolución

Al conocerse el nuevo diagnostico en su familia; acude a nuestra consulta de consejo genético su hija a quien se solicito el estudio genético concluyendo que también es portadora de la misma mutación que su hermano y padre.

El historial anterior tiene asociados los siguientes códigos CIE-O:

- cc_onco266 8140/3 ⇒ Adenocarcinoma, SAI
- cc_onco266 8000/3 ⇒ Neoplasia maligna
- cc_onco266 8000/6 ⇒ Neoplasia metastásica
- cc_onco266 8170/3 ⇒ Carcinoma hepatocelular, SAI
- cc_onco266 8070/3 ⇒ Carcinoma epidermoide, SAI

En este caso, tenemos un caso clínico con múltiples etiquetas asignadas, cada una correspondiente a un código CIE-O que describe cada neoplasia según su localización y su morfología. La mayoría de los casos son similares, son multi-etiqueta, solo unos pocos textos tienen una sola etiqueta asignada. Un clasificador automático conseguiría extraer una lista ordenada de los códigos eCIE-O correspondientes a cada caso clínico.

La aplicación de técnicas PLN para esta tarea presenta las dificultades generales de un problema de clasificación multi-etiqueta. Teniendo este punto en cuenta, el principal objetivo de este trabajo es desarrollar un sistema para participar en CANTEMIST tal y como se explica en el apartado de Propuesta y Objetivos 1.3.

Para desarrollar el sistema de clasificación, CANTEMIST facilita un conjunto de datos de entrenamiento con el que trabajar. Este corpus consiste en un conjunto de 3.000 casos clínicos en español, donde cada caso clínico quedaría almacenado como un único archivo en texto plano en codificación UTF-8. El corpus fue anotado por expertos siguiendo unas directrices de

anotación bien definidas adaptadas de las recomendaciones de codificación clínica publicadas por el Ministerio de Sanidad español. Tras el etiquetado, el corpus tuvo que pasar varios ciclos de control de calidad y análisis de consistencia. Las anotaciones se han escrito en un archivo con el nombre de documento separado por un espacio de tabulador de sus correspondientes anotaciones de código eCIE-O-3.1 en formato BRAT. Solo hay una anotación por fila, de modo que si hubiera dos anotaciones por documento, habría dos filas para indicar la anotación.

Un ejemplo de etiquetado en el corpus sería el siguiente:

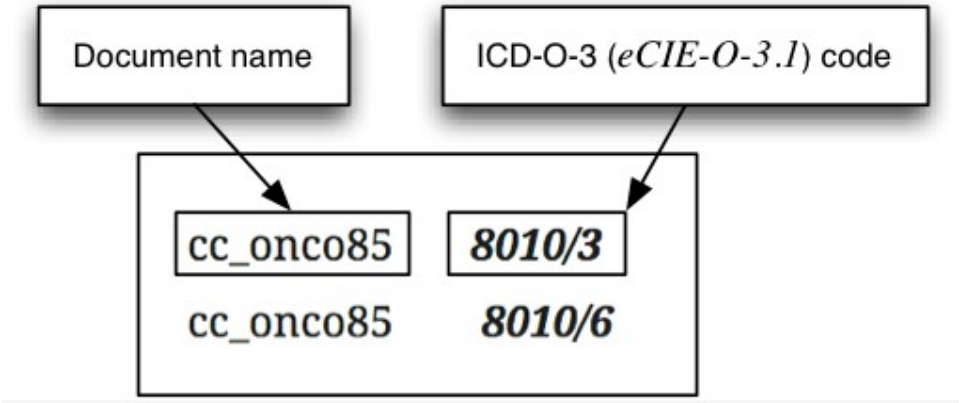



Figura 1.2: Ejemplo de etiquetado de CANTEMIST

Los códigos de morfología de la eCIE-O-3.1 tienen una estructura muy específica: constan de al menos 5 dígitos, donde los primeros cuatro dígitos indican el tipo de tumor o célula y el quinto dígito indica el comportamiento del tumor; un dígito opcional codifica la clasificación o diferenciación histológica. Además, los anotadores de CANTEMIST introdujeron una extensión de código específica de la tarea que es /H, aunque está presente en un porcentaje muy bajo del total de los códigos. Esta extensión se utiliza cuando eCIE-O-3.1 no ofrece un código suficientemente específico para codificar la mención de morfología del tumor. En la imagen de la figura 1.2 vemos un ejemplo de etiquetado de un historial clínico de CANTEMIST. A la izquierda se puede observar el nombre del texto de referencia y a la derecha los códigos morfológicos asociados a ese historial clínico. En este caso, al historial clínico con nombre cc_onco85 se le han etiquetado dos códigos: el 8010/3 y el 8010/6, códigos que corresponden a Carcinoma SAI y a Carcinoma metastásico SAI,

respectivamente.

## 1.3. Propuesta y objetivos

Nuestro problema a abordar, por tanto, es el desarrollo de un sistema automático de codificación clínica capaz de asignar códigos eCIE-O-3.1 a historiales clínicos de pacientes oncológicos en español, de acuerdo a la definición y los datos dados en CANTEMIST. Además, compararemos nuestros resultados con los resultados obtenidos por los participantes de CANTEMIST para esta tarea.

Para el correcto funcionamiento de estos sistemas de clasificación supervisados se necesitan ejemplos previamente etiquetados con los que entrenar el algoritmo. Por eso, se utilizará como entrenamiento el conjunto de datos de CANTEMIST. Este corpus se ha dividido aleatoriamente en tres subconjuntos de datos: el conjunto de datos de entrenamiento, de desarrollo y de test para entrenar, ajustar los parámetros y para testear, respectivamente, el algoritmo que se va a entrenar. Vamos a entrenar distintos modelos preentrenados de BERT disponibles en (**?**) junto con una capa Dropout y una capa final Dense. Los modelos BERT entrenados han sido el Multilingua BERT, RoBERTa, BETO y BERT-SciELO.

Por último, se emplean algoritmos de clasificación supervisada con la intención de obtener, para cada documento, una lista ordenada de sus correspondientes códigos eCIE-O-3.1. A lo largo del trabajo, se presenta el ciclo completo de preprocesamiento de los conjuntos de datos y la construcción del sistema de clasificación.

## 1.4. Estructura del documento

El presente trabajo se compone de los siguientes capítulos:

**Capítulo 1. Introducción.** Este capítulo introduce los principales motivos que han llevado a la realización de este trabajo y las diferentes problemáticas encontradas en los diferentes sistemas de codificación automática, así como sus limitaciones. Por último, se presentan las diferentes contribuciones del trabajo realizado.

**Capítulo 2. Estado del arte.** Este capítulo describe en mayor detalle la disciplina que nos ocupa, presentando su origen y su historia hasta el

presente. Se muestran las técnicas actuales más utilizadas para resolver las tareas más relevantes del tema abordado, así como sus debilidades.

**Capítulo 3. Sistema propuesto.** En este capítulo se describe en profundidad el método propuesto para realizar un clasificador multi-etiqueta capaz de resolver la tarea propuesta en este trabajo.

**Capítulo 4. Evaluación.** Este capítulo describe la metodología utilizada para evaluar el sistema propuesto, a la vez que presenta los resultados obtenidos al evaluar el método propuesto.

**Capítulo 5. Discusión.** Este capítulo se analizan y discuten los resultados obtenidos en la evaluación presentada en el capítulo anterior.

**Capítulo 6. Conclusiones y trabajo futuro.** Este capítulo recopila las diferentes conclusiones extraídas del trabajo realizado, y propone algunas líneas de trabajo futuro.

# Capítulo 2

# Estado del arte

En este segundo capítulo, en primer lugar, se hace una descripción de los sistemas de clasificación generales. Posteriormente, se describen en detalle los métodos que los demás participantes de CANTEMIST han utilizado para resolver la tarea. Por último, se muestran las técnicas actuales más utilizadas para resolver los problemas de clasificación multi-etiqueta, así como las debilidades de cada técnica propuesta.

## 2.1. Técnicas de clasificación

Una de las áreas de investigación de PLN que es, al fin y al cabo, el eje de este trabajo es la clasificación automática de textos o documentos, que consiste en colocar un documento dentro de un conjunto de clases o etiquetas previamente definidas en función de su contenido. Un mismo documento puede pertenecer a una, varias, todas o ninguna de las clases dadas. Cuando se utiliza aprendizaje automático, el objetivo es aprender a clasificar a partir de ejemplos que permitan hacer la asignación a la categoría automáticamente. Este es un trabajo ideal para las máquinas, que son capaces de procesar sin mayores esfuerzos cantidades enormes de textos, tarea que a un humano le llevaría mucho tiempo y esfuerzo. No obstante, dado que la máquina no sabe al principio catalogar un texto en función de ningún criterio, porque no son capaces de interpretar el texto, requiere realizar previamente un proceso de aprendizaje (Kotsiantis, 2007).

Para llevar a cabo esta tarea se han utilizado diversas técnicas a lo largo de los años. Lllamamos enfoques tradicionales a los que se han utilizado más hasta la llegada y explotación de técnicas de deep learning en el campo del

PLN. Los enfoques tradicionales incluyen los denominados sistemas basados en reglas, que son la forma más simple de inteligencia artificial y consiste en crear reglas codificadas basadas en el conocimiento de un experto humano para resolver un problema, en este caso, clasificar textos. Los sistemas basados en Machine Learning tradicional funcionan mediante aprendizaje supervisado, es decir, que se entrena un modelo con datos de entrada y salida conocidos para que pueda predecir salidas futuras. Posteriormente, con la llegada del Deep Learning, se pudo entrenar modelos mediante el uso de redes neuronales, lo que permitía conseguir más precisión en los modelos al utilizar más datos de entrenamiento.

Para construir cualquier clasificador de textos o documentos, ya sea empleando técnicas tradicionales o de DL, es necesario seguir los siguientes pasos (M. Ikonomakis, 2005):

- Extraer los atributos o *features* necesarias para realizar una representación fiel del texto y que permita la utilización de un algoritmo de clasificación, ya que los algoritmos no entienden las palabras como tal, entienden una representación matemática del texto con el que se quiere trabajar.
- Desarrollar procedimientos por los cuales los documentos puedan ser clasificados automáticamente dentro de categorías.
- Evaluar la calidad de la clasificación mediante métricas.

En los próximos apartados, revisamos el estado del arte en clasificación de textos multi-etiqueta, centrándonos en aquellos documentos que tratan de clasificar documentos según códigos o categorías médicas.

### 2.1.1. Clasificación basada en reglas

Desde hace bastantes años, se ha tratado de crear sistemas que puedan clasificar de forma automática la codificación médica. La aproximación tradicional para resolver esta tarea ha consistido en la creación a mano de reglas para encontrar palabras clave que puedan ser o estar cerca de un contexto que denote que exista una cierta enfermedad o un síntoma en el caso clínico del paciente. Los sistemas basados en reglas utilizan patrones predefinidos por un experto o grupo de expertos para crear un conjunto de pautas mediante la combinación de palabras u otros atributos (Farkas y Szarvas, 2008). En este

tipo de aproximaciones, la precisión puede ser considerablemente alta siempre y cuando estas reglas estén confeccionadas y revisadas por un experto en el dominio donde se quieran aplicar las reglas, además de que los textos sean más o menos uniformes y no presenten demasiada variabilidad lingüística (Waltl, 2018). Esta aproximación tiene los inconvenientes de que es costoso tanto de construir como de mantener, además de que solo se pueden utilizar en el ámbito para el que se crearon. Siguiendo el esquema de clasificación, en esta aproximación se hace toda la extracción de features a mano con el apoyo de técnicas de pre-procesamiento. El pre-procesamiento es básicamente el tratamiento básico del texto original para que sea más sencillo trabajar con él y encontrar información relevante. Por ejemplo: se transcribe todo a mayúsculas o minúsculas, se eliminan los signos de puntuación, se eliminan los espacios redundantes entre palabras así como los del inicio y final del texto, se eliminan las letras repetidas contiguas de una palabra dejando un máximo de dos (Salazar et al., 2013), se divide el texto en secuencias más pequeñas, se realiza una revisión ortográfica o tokenización y se eliminan las palabras poco frecuentes (Wang et al., 2020).

En trabajos anteriores para la clasificación automática de codificación médica (Farkas y Szarvas, 2008) lo habitual es crear una serie de reglas para la detección de síntomas y enfermedades en los textos y asignar el código CIE correspondiente. Para las reglas, se crea una lista de sinónimos/abreviaturas de diferentes síntomas/enfermedades. Además, sus reglas eliminan los códigos de los síntomas si ha encontrado el diagnóstico de la enfermedad que provoca dichos síntomas. El contexto es importante porque en un informe se puede encontrar muchas veces la situación en la que un paciente sufra una "posible" enfermedad, o que "sea sospechoso" de padecer determinada dolencia. No se deben codificar casos clínicos con probables enfermedades. Por tanto, es necesario crear ciertas reglas que puedan identificar estos contextos y no tener en cuenta esos códigos (Crammer et al., 2007). Los diccionarios se han utilizado como apoyo para guardar en ellos palabras o grupos de palabras que son sinónimos entre sí. Si se encuentra un grupo en el texto, se reemplaza por su correspondiente sinónimo. La creación de una aproximación de reglas junto con un diccionario ha probado ser bastante eficiente (Crammer et al., 2007). Además, es muy importante el tratamiento de la negación de síntomas o enfermedades. No es lo mismo afirmar que un paciente tiene un síntoma concreto que indicar en el informe que el paciente no presenta cierto

síntoma. Por esto, (Crammer et al., 2007) propuso no tener en cuenta los códigos de los síntomas o enfermedades cuando tuvieran una negación cerca.

Se pueden hacer reglas para comparar las enfermedades y síntomas, o secuencias de n-gramas, entre los informes médicos de un paciente con su historial médico completo (Crammer et al., 2007). Por ejemplo, puede aparecer el síntoma de tos en la historia clínica, pero neumonía en el informe. Estas reglas se pueden complementar mediante la creación de archivos de vocabulario con las etiquetas del conjunto de datos y archivos de mapeo de etiquetas que contienen la correspondencia entre el código CIE y su identificador alfanumérico, además de su identificador alfanumérico en el archivo de vocabulario.

En cuanto a la evaluación de estos modelos, la precisión, el recall y el F1-Score son las métricas más utilizadas. Los métodos basados reglas dan buenos resultados en general, pero cuando se ha comparado su rendimiento con la aplicación de algoritmos de Machine Learning (como el algoritmo MIRA *Margin Infused Relaxed Algorithm* (Koby Crammer, 2003)) o técnicas mixtas (basado en reglas + algoritmo) su rendimiento no llega a los resultados de otras técnicas más modernas (Crammer et al., 2007). Algoritmos y técnicas como las del Machine Learning (XGBoost, Naive Bayes, algoritmo de SVM...) ofrecían resultados prometedores (Miranda-Escalada et al., 2020). Las técnicas de Machine Learning que se dejaron de aplicar con la llegada del Deep Learning, se revisan en el siguiente apartado.

### 2.1.2. Clasificación basada en Machine Learning tradicional

El aprendizaje automático, o Machine Learning (ML), es una rama de la inteligencia artificial y la informática que se centra en el uso de datos y algoritmos para imitar la forma en que aprenden los humanos, mejorando gradualmente su precisión. Se basa en utilizar modelos matemáticos de datos para ayudar a una máquina a aprender sin instrucción directa, como se hace en los sistemas basados en reglas. La gran ventaja del aprendizaje automático con respecto de los sistemas basados en reglas es que no se tienen que escribir ni codificar las reglas en el código, lo cual es un gran ahorro en esfuerzo y tiempo si nos enfrentamos a secuencias de lenguaje natural con mucha variabilidad lingüística (Waltl, 2018). Los ruidos en gran parte inevitables y las variaciones del lenguaje plantean obstáculos para la aplicación de las reglas. Esto significa que, en tales casos, los modelos estadísticos de

aprendizaje automático tienen la ventaja de generalizar mejor, lo que resulta en unos resultados de recall mejores (Waltl, 2018).

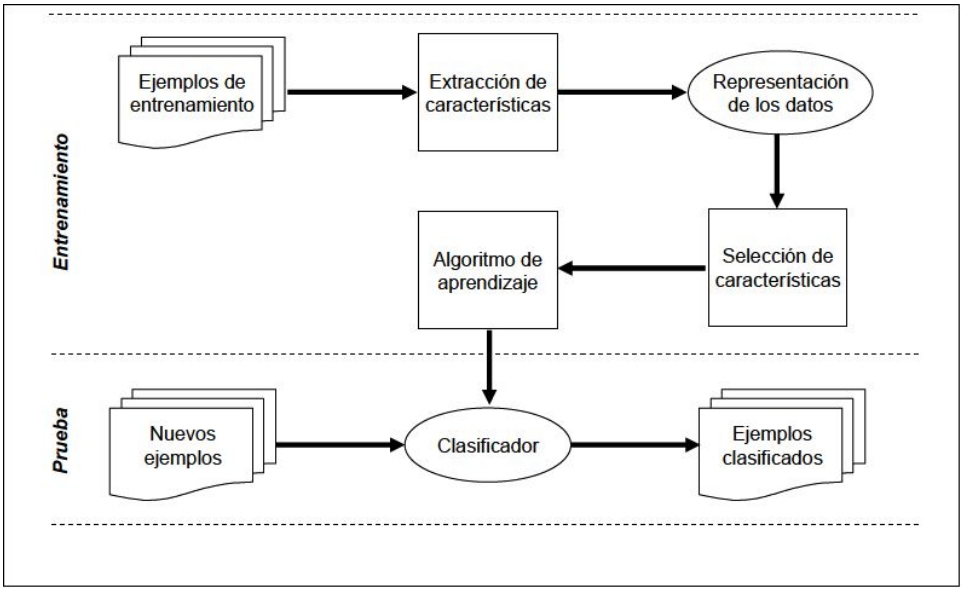



Figura 2.1: Un ejemplo de paradigma de aprendizaje de Machine Learning

En la imagen 2.1 se muestra el ciclo de entrenamiento de un algoritmo de ML, el cual intenta aprender conceptos a través de un conjunto de datos etiquetado. El clasificador construído utiliza los conceptos aprendidos para clasificar nuevos documentos que no ha visto durante el entrenamiento. El aprender de ejemplos es un problema conocido como aprendizaje supervisado, ya que parte de una serie de clases o categorías diseñadas a priori, en las cuales hay que distribuir a cada uno de los documentos.

Pero en el aprendizaje automático no solo hay un tipo de categoría de aprendizaje. Según cómo sea el entrenamiento, podemos encontrar diferentes categorías de aprendizaje automático (Dey, 2016):

- Los modelos de aprendizaje automático supervisados se entrenan con conjuntos de datos etiquetados, que permiten que los modelos aprendan y se vuelvan más precisos con el tiempo. Por ejemplo, un algoritmo se entrenaría con imágenes de perros y otras cosas, todas etiquetadas por humanos, y la máquina aprendería formas de identificar imágenes de perros por sí misma. El aprendizaje automático supervisado es el tipo más común que se usa en la actualidad.

- En el aprendizaje automático no supervisado, un programa busca patrones en datos sin etiquetar. El aprendizaje automático sin supervisión puede encontrar patrones o tendencias que las personas no buscan explícitamente. Por ejemplo, un programa de aprendizaje automático no supervisado podría examinar los datos de ventas en línea e identificar diferentes tipos de clientes que realizan compras.

- El aprendizaje automático de refuerzo entrena a las máquinas a través de prueba y error para tomar la mejor acción al establecer un sistema de recompensa. El aprendizaje por refuerzo puede entrenar modelos para jugar o entrenar vehículos autónomos para conducir al decirle a la máquina cuándo tomó las decisiones correctas, lo que le ayuda a aprender con el tiempo qué acciones debe tomar.

Éstas son solo algunas de las subcategorías más conocidas, pero hay muchas más, como el aprendizaje reforzado o el aprendizaje semi-supervisado.

Para resolver una tarea de clasificación de documentos donde contamos con un conjunto de datos etiquetado, se recurriría al aprendizaje automático supervisado. Como vemos en la imagen 2.1 un ciclo de entrenamiento consiste en recopilar una serie de ejemplos de entrenamiento etiquetados, hacer una serie de extracción y selección de características y con esos datos, se alimenta al algoritmo elegido (M. Ikonomakis, 2005). Sin embargo, la mayor dificultad de la clasificación de textos es la representación de un documento debido a que se trata de una secuencia de palabras. Los algoritmos no son capaces de entender valores categóricos como entrada y por eso deben someterse a diferentes transformaciones (M. Ikonomakis, 2005) (tokenización, lematización, eliminado de stopwords, conseguir una representación vectorial del texto y realizar una selección o transformación de características a partir de ese vector).

Uno de los algoritmos más utilizados para el cálculo de estos patrones es el denominado *Naive Bayes*. Este clasificador es de tipo probabilístico, el cual se basa en el cálculo de distribuciones de probabilidad en función de datos observados. Este clasificador se basa en el Teorema de Bayes con la suposición ingenua de que las características son independientes entre sí.

$$P(C\_k|X) = \frac{P(X|C_k)P(C_k)}{P(X)} \tag{2.1}$$

para k=1,2,...,K

donde los elementos de la ecuación del Teorema de Bayes 2.1 son:

- $P(C_k \mid X)$ es la probabilidad posterior.
- $P(X \mid C_k)$ es la probabilidad del predictor X dada la probabilidad de la clase $C_k$.
- $P(C_k)$ Es la probabilidad de la clase $C_k$.
- $P(X)$ probabilidad previa del predictor X.

Lo que se pretende es calcular la probabilidad posterior a partir de la probabilidad y las probabilidades previas. Usando la regla de la cadena, la probabilidad de $P(X \mid C_k)$ se puede descomponer de la siguiente manera:

$$P(X \mid C_k) = P(x_1, ..., x_n \mid C_k) = P(x_1 \mid x_2, ...x_n, C_k)$$
$$P(x_2 \mid x_3, ...x_n, C_k) \cdots P(x_{n-1} \mid x_n, C_k)P(x_n \mid C_k)$$

Sin embargo, el cálculo de los conjuntos de probabilidades anteriores 2.1.2 pueden ser complicados y tediosos de calcular. Afortunadamente, con la ayuda del supuesto independencia condicional de la que parte *Naive Bayes*, que se expresa matemáticamente de la siguiente manera:

$$P(x_i \mid x_{i+1}, ..., x_n \mid C_k) = P(x_i \mid C_k) \tag{2.2}$$

Se puede obtener lo siguiente:

$$P(X \mid C_k) = P(x_1, ..., x_n \mid C_k) = \prod_{i=1}^{n} P(x_i \mid C_k) \tag{2.3}$$

Con la anterior ecuación 2.4 se puede expresar matemáticamente la probabilidad posterior bayesiana de la siguiente manera:

$$P(C_k|X) = \frac{P(C_k) \prod_{i=1}^{n} P(x_i \mid C_k)}{P(X)} \tag{2.4}$$

Con estas ecuaciones y teniendo en cuenta que P(X) es constante dada la entrada y que $\propto$ significa positivo proporcional a, se obtiene la siguiente ecuación:

$$P(C_k|X) \propto P(C_k) \prod_{i=1}^{n} P(x_i \mid C_k) \tag{2.5}$$

Como el objetivo es para cada valor de $C_k$ calcular su máximo, que haría $P(C_k) \prod_{i=1}^{n} P(x_i \mid C_k)$, se puede reformular la ecuación anterior 2.5 como sigue:

$$\hat{C} = \arg \max_{C_k} P(C_k) \prod_{i=1}^{n} P(x_i \mid C_k) \tag{2.6}$$

*Naive Bayes* es uno de los algoritmos más utilizados, pero también se enfrenta a los típicos problemas de ML que es que si una a variable categórica le corresponde una etiqueta en el conjunto de datos de prueba, que no se observó en el conjunto de datos de entrenamiento, el modelo asignará una probabilidad de 0 y no podrá hacer una predicción correctamente. Además, la suposición subyacente de *Naive Bayes* de que los las variables son independientes entre sí, apenas sucede cuando trabajamos con datos textuales.

Otro de los algoritmos más conocidos aplicados a PLN es SVM (por sus siglas en inglés *Support Vector Machines*). Este algoritmo se basa en el principio de minimización de riesgos estructurales (Burges, 1998) y es un método de aprendizaje automático popular para analizar datos y reconocer patrones. Un SVM realiza la clasificación mediante la construcción de un hiperplano N-dimensional, es decir, un plano generalizado en N-dimensiones, de forma que separa de manera óptima los datos en dos o más categorías. El número de dimensiones depende del número de categorías del conjunto de entrenamiento.

Matemáticamente, un problema de clasificación con el algoritmo SVM se resolvería mediante la ecuación 2.7 donde para cada punto de datos $z$ que se quiere clasificar, se calcularía de la siguiente manera:

$$\hat{z} = \operatorname{sgn} \left\{ b + \sum_{i=1}^{l} y_i \alpha_i \Phi \left( x_i, z \right) \right\} \tag{2.7}$$

donde $z \in R$ es un punto que necesita ser clasificado.

En pocas palabras, así es como funciona un modelo de algoritmo de máquina de vectores de soporte:

- En primer lugar, encuentra las líneas o límites que clasifican correctamente el conjunto de datos de entrenamiento.
- Después, a partir de esas líneas o límites, el algoritmo elige la que tiene la distancia máxima desde los puntos de datos más cercanos.

Hay muchos otros algoritmos que pueden resolver un problema de cla-

sificación de textos. En función de las características de cada algoritmo, hay que elegir el que mejor pueda resolver la tarea que se presenta.

Como es de suponer, se pueden aplicar técnicas de Machine Learning para realizar una clasificación automática de textos bajo una temática biomédica. Se puede recopilar un conjunto de datos, como casos clínicos, y alimentar un algoritmo para que los pueda clasificar automáticamente. En hospitales y clínicas de atención médica se ha recopilado una enorme cantidad de información clínica de texto libre, como informes de patología, informes de progreso, notas clínicas y resúmenes de alta. Estos datos brindan la oportunidad de desarrollar muchas aplicaciones útiles de aprendizaje automático si los datos pudieran transferirse a una estructura apta para el aprendizaje con etiquetas adecuadas para el aprendizaje supervisado. La anotación de estos datos debe ser realizada por expertos clínicos calificados, por lo que se limita el uso de estos datos debido al alto costo de la anotación. Sin embargo, hay disponibles algunos corpus ya etiquetados en internet (Information, 2021). Una vez se dispone de un corpus etiquetado, se pueden aplicar diferentes pre-procesados y transformaciones a los textos y, posteriormente, se le puede aplicar un algoritmo a esos datos. Como se ha mencionado en el capítulo 1, la motivación de aplicar estas técnicas es poder ejecutar una clasificación automática que permita agilizar los trámites y abaratar los costes de tener a una persona especializada monitorizando cada documento a mano.

Hay varios enfoques pueden servir cuando queremos enfrentarnos a un problema de clasificación de textos clínicos. Por ejemplo, el trabajo de (Liting Du, 2018) en el que basándose en un total de 14.719 resúmenes de alta de diferentes centros de salud regionales en la ciudad de Yaán, provincia de Sichuan, China, construyeron un modelo de campos aleatorios condicionales (CRF) para identificar la PHI (siglas en ingés de información de salud protegida) en el texto clínico, y luego usaron expresiones regulares, que definieron manualmente, para optimizar los resultados de reconocimiento de las categorías de PHI que contaban con menos muestras en el conjunto de entrenamiento. La evaluación mostró con un valor alto de *F-measure* de 0.9878, que el modelo basado en CRF tuvo un buen desempeño.

Otro ejemplo de aplicación de Machine Learning sería (Pilar López Úbeda, 2021) donde resuelven una tarea de clasificación multiclase. Utilizaron un corpus real proporcionado por el centro médico privado "HT medica" compuesto por casi 700.000 exámenes de tomografía computarizada (TC) y

resonancia magnética (RM) obtenidos durante el uso clínico de rutina. El objetivo era asignar informes radiológicos escritos en lenguaje natural y protocolos de imágenes radiológicas. Mediante la aplicación de modelos SVM y Random Forest consiguieron un 92.2 % de precisión en el conjunto de datos de TC y un 86,9 % en el conjunto de datos de resonancia magnética, los cuales son buenos resultados.

Una dificultad adicional a la que nos podemos encontrar cuando estamos construyendo un modelo de clasificación de ML sería enfrentarnos a una clasificación con muchas categorías. (Hasan, 2016) desarrollaron una solución ML para clasificar entrevistas clínicas en 17 categorías diferentes. Hicieron pruebas con diversos algoritmos ML, pero el que mejor resultados arrojó fue el SVM. Los parámetros óptimos fueron un kernel de función de base radial (RBF) con los parámetros C y $\gamma$ asignados a 4.0 y 0.1, respectivamente. También observaron que la función de pérdida L1 funciona mejor que la función de pérdida L2 para SVM lineal. Sus resultados demuestran el potencial de utilizar métodos de aprendizaje automático junto con características léxicas, semánticas y contextuales para la anotación automática de transcripciones de entrevistas clínicas con una precisión casi humana (un 70,8 % de precisión con SVM).

### 2.1.3. Clasificación basada en Deep Learning

El aprendizaje profundo, una técnica avanzada de inteligencia artificial, se ha vuelto cada vez más popular en los últimos años, gracias a la abundancia de datos y al aumento de la potencia informática. Es la tecnología principal detrás de muchas de las aplicaciones que usamos todos los días, incluida la traducción de idiomas online y el etiquetado automático de rostros en las redes sociales, por poner algunos ejemplos. Básicamente, el Deep Learning es una red neuronal con tres o más capas. Estas redes neuronales intentan simular el comportamiento del cerebro humano, aunque lejos de igualar su capacidad, lo que le permite “aprender"de grandes cantidades de datos. Si bien una red neuronal con una sola capa aún puede realizar predicciones aproximadas, las capas ocultas adicionales pueden ayudar a optimizar y refinar la precisión. Las redes neuronales son especialmente buenas para encontrar de forma independiente patrones comunes en datos no estructurados, como son los documentos escritos con lenguaje natural.

Para aplicar técnicas de Deep Learning se puede tratar los casos clíni-

cos como una secuencia de caracteres, donde funcionan muy bien las redes neuronales recurrentes (RNN). Las RNN (Rumelhart, 1985) son un tipo de red neuronal donde la salida de la capa anterior se alimenta como entrada a la siguiente capa. En las redes neuronales tradicionales, todas las entradas y salidas son independientes entre sí, pero en casos como cuando se requiere predecir la siguiente palabra de una oración, se requieren las palabras anteriores y, por lo tanto, es necesario recordar dichas palabras. Es decir, las RNN se distinguen por su "memoria", ya que toman información de entradas anteriores para influir en la entrada y salida actuales. Aunque este tipo de redes tienen dos inconvenientes: no capturan las dependencias entre palabras si están muy separadas entre sí en la secuencia, además de que el problema del desvanecimiento de gradiente resurge y, debido a esto, no son capaces de capturar las dependencias en una sola dirección del lenguaje. Básicamente, en el caso del procesamiento del lenguaje natural, se asume que la palabra que viene después no tiene ningún efecto sobre el significado de la palabra que viene antes. Con nuestra experiencia con los idiomas sabemos que esto no es cierto. Para resolver parcialmente estos problemas se crearon las GRU (*Gated Recurrent Units*) (Chun et al., 2014) y las LSTM (*Long short-term memory*) (Hochreiter y Schmidhuber, 1997). Los principios de las LSTM y GRU son comunes en términos de modelado de secuencias largas. Primero, se puede procesar un número arbitrario de pasos de tiempo, y además, se puede eliminar la información redundante e incorporar un componente de memoria almacenado en los pesos. La memoria es introducida en la red por el vector de estado oculto que es único para cada secuencia de entrada, siempre comenzando desde un vector de elemento cero para t = 0.

Las GRU (Chun et al., 2014) son una modificación de la red neuronal recurrente básica que, con mecanismos de activación, ayudan a capturar dependencias de largo alcance en la secuencia y pueden resolver el problema de desvanecimiento de gradiente que presenta la red neuronal recurrente estándar. GRU admite la activación de puertas y un estado oculto para controlar el flujo de información. Para resolver el problema que surge en RNN, GRU usa dos puertas, como se muestra en la figura 2.2: la puerta de actualización y la puerta de reinicio. Se pueden considerar como dos entradas vectoriales (0,1) que pueden realizar una combinación convexa. Estas combinaciones deciden qué información de estado oculto se debe actualizar (pasar) o restablecer el estado oculto cuando sea necesario. Asimismo, la red también

aprende a omitir observaciones temporales irrelevantes. Además, a diferencia de las LSTM, GRU no tiene una puerta de salida y combina la entrada y la puerta de olvido en una única puerta de actualización. En este sentido, GRU se considera más eficiente en términos de estructura más simple, lo que puede ser una gran ventaja si se busca un modelo rápido y compacto.

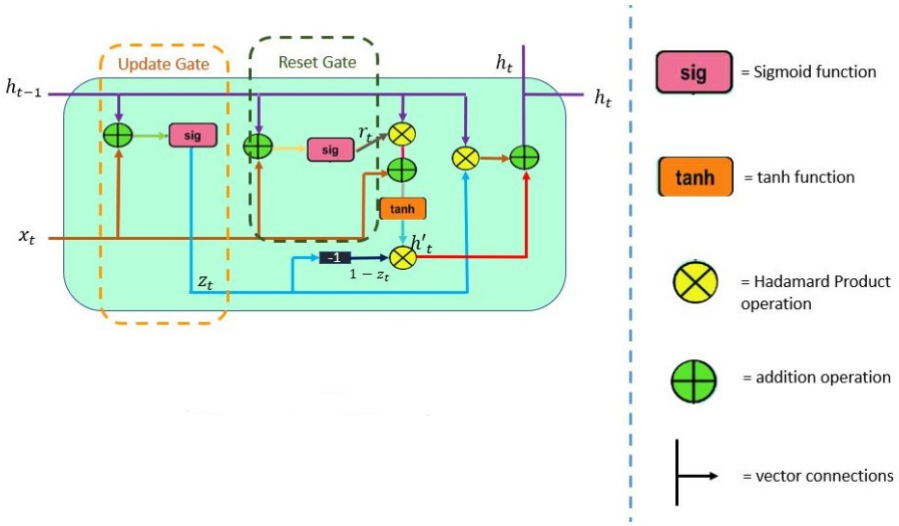



Figura 2.2: Gated Recurrent Network (GRU)

Las LSTM (Hochreiter y Schmidhuber, 1997) constan de tres puertas, una más que GRU: la puerta de entrada, la puerta de olvido y la puerta de salida. En teoría, si se tiene que lidiar con grandes cantidades de datos, las LSTM deberían ser capaces de recordar secuencias más largas que las GRU y superarlas en tareas que requieran modelar correlaciones de largo alcance.

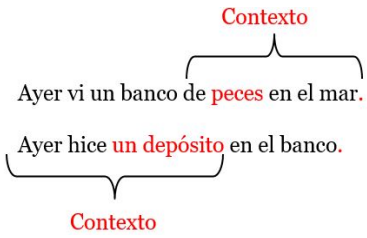
Pero esto no implica que no se puedan combinar las fuerzas de estas dos celdas. Por ejemplo, la estructura del modelo de (Wang et al., 2020) se trata de una red neuronal de cuatro capas. La primera capa es la capa de inserción de palabras, que transforma la entrada de texto libre en vectores de palabras. La segunda capa es una capa GRU que consume menos tiempo para el cálculo de los pesos que la LSTM. Las capas restantes son dos capas densas con funciones de activación de ReLU y sigmoid por separado, donde las capas densas finales deben generar el vector con la dimensión que esperamos predecir. En los casos de clasificación multi-etiqueta, el tamaño de la capa salida debe ser igual a la cantidad de etiquetas.

La opción más utilizada en los últimos años para resolver tareas de clasificación de textos, tanto en CANTEMIST como en otros ámbitos, ha sido la implementación de BERT. BERT (Devlin et al., 2018) es la técnica desarrollada por Google basada en redes neuronales para el pre-entrenamiento del procesamiento del lenguaje natural publicado por primera vez en octubre de 2018 en el trabajo *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* (Devlin et al., 2018) donde se explican los dos pasos esenciales que tomaron para el desarrollo de BERT: el pre-entrenamiento y el ajuste. Durante el pre-entrenamiento, el modelo se entrena con un corpus masivo de datos sin etiquetar. El modelo BERT original está pre-entrenado con dos corpus: BooksCorpus (800M palabras aprox.) y una versión de Wikipedia en inglés (2.500M palabras aprox.). Sin embargo, el modelo BERT pre-entrenado puede no ser adecuado para aplicaciones de dominios específicos como, en este caso, la medicina. Por lo tanto, es aconsejable pre-entrenar BERT con conjuntos de datos personalizados. Para el paso de ajuste, el modelo BERT se inicializa primero con los parámetros pre-entrenados, y todos los parámetros se ajustan utilizando datos etiquetados de las tareas posteriores. Cada tarea posterior tiene modelos de ajuste separados, aunque se inicializan con los mismos parámetros pre-entrenados. Estos dos pasos permiten que BERT tenga un rendimiento significativamente más alto que los modelos que solo cuentan con solo un pequeño conjunto de datos.

BERT ha sido pre-entrenado para dos tareas principales: modelado de lenguaje enmascarado y predicción de la siguiente oración. En lugar de que, a partir de una secuencia de entrada, se haga una predicción de la siguiente palabra en una secuencia como se hacía tradicionalmente, BERT utiliza una técnica novedosa llamada Modelado de lenguaje enmascarado o MLM (Devlin et al., 2018) (*Masked Language Modeling*) por sus siglas en inglés: enmascara al azar un porcentaje de las palabras en la oración y luego las intenta predecir. El enmascaramiento significa que el modelo mira en ambas direcciones y utiliza el contexto completo de la oración, tanto a la izquierda como a la derecha, para predecir la palabra enmascarada, es decir, oculta. A diferencia de los modelos de lenguaje anteriores, tiene en cuenta tanto los tokens anteriores como los siguientes a la vez. Los modelos existentes basados en LSTM (*Long Short-Term Memory*) combinados de izquierda a derecha y de derecha a izquierda (bidireccionales) no tenían la capacidad de

fijarse en los tokens anteriores y posteriores "a la vez". Es por eso que BERT es tan potente y por lo que se considera a BERT como un modelo bidireccional profundo. Cabe destacar además que los modelos bidireccionales son más poderosos que los modelos de lenguaje unidireccionales. Sin embargo, en un modelo de varias capas los modelos bidireccionales no funcionan. Esto es debido a que las capas inferiores filtran información y permiten que un token se vea a sí mismo en capas posteriores.

Un ejemplo ilustrativo de que la bidireccionalidad de un modelo es importante para comprender verdaderamente el significado de un idioma sería el siguiente:




Como hemos mencionado previamente, BERT es capaz de capturar el significado del contexto tanto el de la izquierda como el de la derecha y, por tanto, puede predecir la palabra banco en los dos casos sin cometer errores.

Las palabras enmascaradas no siempre se reemplazan con el token enmascarado [MASK] porque entonces los tokens enmascarados nunca se verían antes del proceso de ajuste. Por lo tanto, el 15 % de los tokens se eligen al azar y:

- El 80 % de las veces, los *tokens* son reemplazados por el token [MASK].
- El 10 % de las veces, los *tokens* se reemplazan por una ficha aleatoria.
- El 10 % de las veces, los *tokens* no se modifican.

Aunque el modelado de lenguaje enmascarado puede codificar el contexto bidireccional para representar palabras, no modela explícitamente la relación lógica entre pares de texto. Para ayudar a comprender la relación entre dos secuencias de texto, BERT considera una tarea de clasificación binaria, la predicción de la siguiente oración, en su entrenamiento previo. Al generar pares de oraciones para el pre-entrenamiento, la mitad de las veces son oraciones consecutivas con la etiqueta "Verdadero"; mientras que durante la otra mitad de las veces la segunda oración se extrae al azar del corpus con

la etiqueta “Falso”. Entonces se requiere que BERT realice una predicción indicando si la segunda oración es aleatoria o no, con el supuesto de que la oración aleatoria se desconectará de la primera oración.

El modelo se entrena con MLM y predicción de la siguiente oración. Esto se hace con el objetivo de minimizar la función de pérdida combinada de las dos estrategias.

Para el entrenamiento, BERT (Devlin et al., 2018) se basa en un transformador (el mecanismo de atención que aprende las relaciones contextuales entre las palabras de un texto). Un transformador básico consta de un codificador para leer la entrada de texto y un decodificador para producir una predicción para la tarea. Dado que el objetivo de BERT es generar un modelo de representación del lenguaje, solo necesita la parte del codificador. La entrada al codificador para BERT es una secuencia de tokens que primero se convierten en vectores y luego se procesan en la red neuronal. Pero antes de que pueda comenzar el procesamiento, BERT necesita como entrada los siguientes *embeddings*, como se muestra en la figura 2.3:

- *Embeddings* de *tokens*: se agrega un token [CLS] a los *tokens* de palabra de entrada al principio de la primera oración y un token [SEP] al final de cada oración.
- *Embeddings* de segmentación: se agrega un marcador que indica la oración A o la oración B a cada ficha. Esto permite que el codificador distinga entre oraciones.
- *Embeddings* de posición: se agrega una incrustación posicional a cada *token* para indicar su posición en la oración.

### 2.1.4. Sistemas de clasificación con Deep Learning

En este capítulo vamos a presentar algunos de los trabajos publicados donde aplican sistemas de clasificación Deep Learning para resolver tareas de PNL tanto clínicas como generales.

(Minaee, 2020) hicieron una revisión de un total de 150 modelos diferentes que aplicaron a 40 conjuntos de datos populares ampliamente utilizados para la clasificación de texto, análisis de sentimiento y demás tareas de PLN, entre los cuales están los conjuntos de datos de WikiQA, IMDB, AG News, 20-NEWS, DBpedia, Yelp y Amazon. No solo aplicaron modelos de Deep

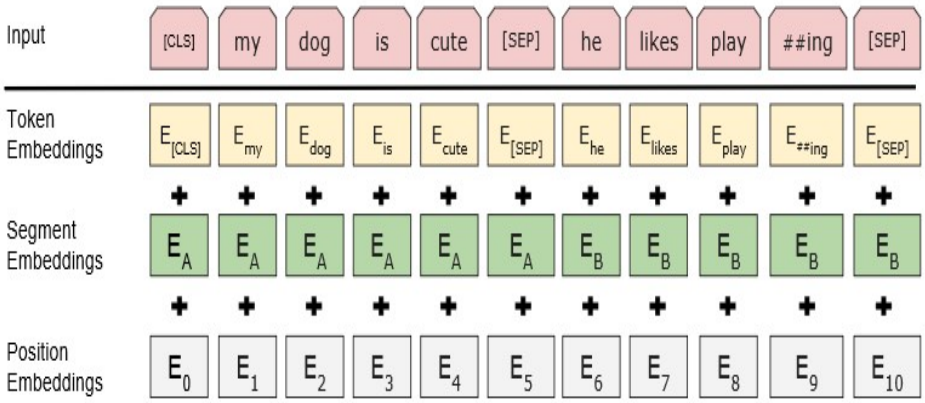



Figura 2.3: La representación de entrada para BERT: las incrustaciones de entrada son la suma de las incrustaciones de tokens, las incrustaciones de segmentación y las incrustaciones de posición. (Devlin et al., 2018)

Learning en este trabajo, también aplicaron otros modelos como el *Naive Bayes*, BoW+SVM o LDA. Podemos ver que en todos estos conjuntos de datos, el uso de modelos Deep Learning lleva a mejoras significativas en el rendimiento de los modelos, sobre todo en los conjuntos de datos de WikiQA y SQuAD donde el aumento significativo del rendimiento se atribuye a la aplicación de BERT.

El problema que encontraron (Minaee, 2020) fue la falta de interpretabilidad de los modelos de DL. Es decir, no tienen claro qué es lo que aprende un modelo para destacar en el rendimiento de clasificación de un conjunto de datos, pero acabar fallando con otro conjunto. Aunque los mecanismos de atención y autoatención brindan una idea para responder a estas preguntas, aún falta un estudio detallado del comportamiento subyacente y la dinámica de estos modelos DL. Una mejor comprensión de los aspectos teóricos de estos modelos puede ayudar a desarrollar mejores modelos seleccionados para varios escenarios de análisis de texto.

(Wei, 2020) construyeron una red CNN para resolver una tarea de clasificación binaria de textos. Utilizaron 4 conjuntos de datos de asuntos legales, con millones de registros cada uno. Para cada conjunto de datos, primero reservaron un conjunto seleccionado al azar de registros etiquetados con un tamaño de alrededor de 25.000 registros como conjunto de test. Después, generaron cuatro conjuntos de entrenamiento incrementales seleccionando

aleatoriamente del conjunto de registros etiquetados restantes. Por lo tanto, las cantidades de conjuntos de entrenamiento son diferentes debido a los volúmenes originales de datos en diferentes proyectos. Para resolver esta tarea, usaron una CNN y un algoritmo SVM para poder comparar resultados. La red CNN cuenta con una primera capa conteniendo los Embeddings, una capa *Dropout* seguida de una capa Conv1D y MaxPooling1 para terminar con una capa densa. Para ambos modelos establecieron un valor de threshold de 0.5. La precisión de la clasificación del modelo de CNN en los mismos datos de prueba con diferentes datos de entrenamiento cambia en cada proyecto. Los resultados demostraron que el mayor volumen de datos de entrenamiento produce mejores rendimientos con este modelo de CNN, mientras que el volumen más pequeño de datos de entrenamiento puede tener un peor rendimiento. Además, las precisiones con el modelo de CNN son más altas que las precisiones que se obtuvieron con el algoritmo de SVM cuando se usa un mayor volumen de datos de entrenamiento. Estos hallazgos establecen claramente que CNN supera los enfoques tradicionales de minería de texto para la clasificación de texto con grandes volúmenes de datos.

En el campo clínico, (Oleynik, 2019) participaron en el *2018 National PLN Clinical Challenges (n2c2) Shared Task* donde, a partir de un conjunto de datos anotado con narrativas médicas de 202 pacientes para la clasificación de texto binario de múltiples etiquetas, aplicaron una solución a partir de un modelo LSTM. Su red tenía una sola capa LSTM con 64 células y una capa de salida RNN con activación sigmoidea y función de pérdida de Cross-entropy. Para la optimización, usaron Adam basada en gradientes con una tasa de aprendizaje de 0.02 y entrenaron la red durante 25 épocas con una tasa de abandono del 50 % para evitar el sobreajuste del modelo. Sin embargo, su modelo LSTM no superó a la aplicación del algoritmo SVM, donde obtuvieron una precisión de 80,35 %. Su modelo LSTM solo consiguió un 73,77 % de precisión. Probablemente, la adición de más capas al modelos LSTM pueda mejorar ese porcentaje. Aquí un sistema basado en Deep Learning todavía no puede superar a un sistema basado en reglas o que utilice métodos Machine Learning cuando el conjunto de datos presenta un problema de desequilibrio de clases, es decir, que haya más muestras de entrenamiento en una clase que en otra.

### 2.1.5. Sistemas de clasificación utilizados en CANTEMIST

En esta tarea nos enfrentamos a un problema multi-etiqueta, puesto que necesitamos obtener todos los códigos eCIE-O-3.1 que contenga cada documento. La clasificación de etiquetas múltiples es el problema de aprendizaje supervisado donde una instancia puede estar asociada con multi-etiqueta. Es un problema que contrasta con la tarea tradicional en la que una instancia solo puede estar asociada a una sola etiqueta.

La mayoría de los participantes de CANTEMIST han utilizado BERT, que explicaremos más detalladamente en el siguiente apartado, para la clasificación de los códigos, como hizo (García-Pablos, Pérez, y Cuadros, 2020) al utilizar diferentes modelos de BERT, entre ellos SciBERT (modelo BERT entrenado con un corpus de textos científicos), BETO, multilanguage BERT...

La segunda opción más utilizada, fueron redes neuronales, tanto convolucionales como recurrentes, en concreto las redes recurrentes LSTM. (Hassan, Sánchez, y Domingo-Ferrer, 2020) usaron CNN porque este tipo de redes reducen el cálculo al explotar la correlación local de los datos de entrada. En este caso, las entidades con nombre de tumor contienen en promedio tres palabras, por lo que usaron CNN con 512 filtros (kernel) de tamaño tres. Hicieron un agrupamiento de 2-max para obtener el máximo de dos números para cada respuesta de filtro, lo que nos da dos vectores de tamaño 512. Estos dos vectores se pasaron al LSTM para combinarlos. Finalmente, la salida se pasó al clasificador, que es una capa completamente conectada.

(Ruas et al., 2020) aplicaron X-Transformer con algunas modificaciones. X-Transformer es una propuesta de (Wei-Cheng et al., 2020) en la que la indexación semántica de etiquetas reduce el gran espacio de salida. Luego, los transformadores se ajustan al problema de clasificación de texto de etiquetas multiples extremo que asigna instancias a grupos de etiquetas. Y finalmente, los rankers lineales se entrenan condicionalmente en los clústeres y la salida del Transformer para volver a clasificar las etiquetas dentro de los clústeres predichos. Para resolver esta tarea, (Ruas et al., 2020) crearon el pipeline de la figura 2.4: primero procesa los datos de CANTEMIST antes de ejecutarlos a través de X-Transformer. Finalmente, las predicciones se convierten al formato BRAT requerido.

Las modificaciones propuestas por (Ruas et al., 2020) fueron las que se enumeran a continuación:

- Vectorización de todas las etiquetas disponibles de los conjuntos de

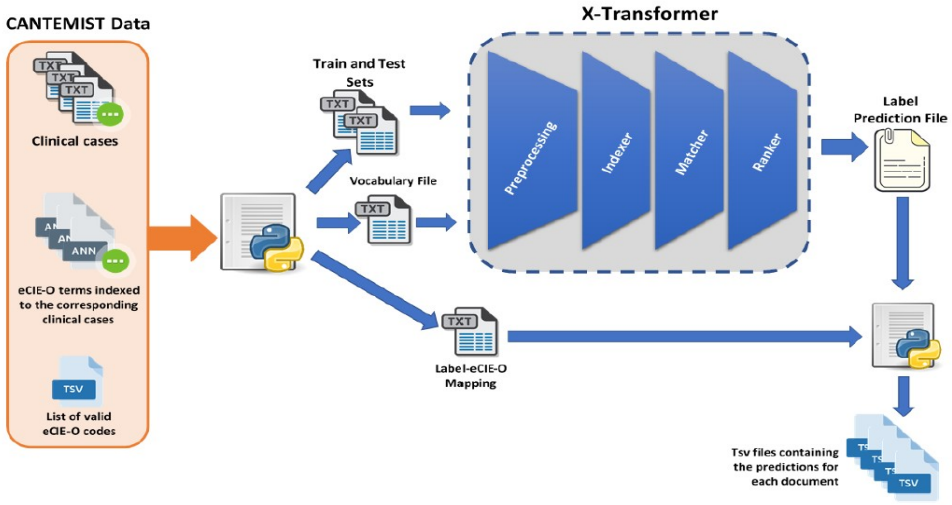



Figura 2.4: Pipeline de X-Transformer propuesto.

entrenamiento y prueba, debido a que había etiquetas que no estaban presentes en los conjuntos de prueba, para que evitar que no funcionara correctamente si el número de etiquetas entre conjuntos no coincidiera.

- Inclusión del modelo preentrenado BETO en las elecciones de modelos a entrenar.
- Adaptación del algoritmo para que fuera posible procesar datos de entrada que contengan signos diacríticos.

Aun aplicando las más novedosas técnicas de Deep Learning, la red no presentará un rendimiento óptimo si no se le suministran suficientes datos para que el modelo pueda inferir una generalización sobre los datos de la tarea. Es decir, el aprendizaje supervisado se construye sobre un conocimiento a priori y se debe disponer de un conjunto de documentos suficiente de ejemplo para cada uno de los códigos. Algunos participantes encontraron el conjunto de datos de CANTEMIST insuficiente para algunos códigos eCIE-3.0 y han recogido un corpora que puedan alimentar a la red, como BERT-scielo con los corpora de Galén oncology information system y la base de datos de MIMIC-III con un corpus multilingüe (López-García et al., 2020), o incluso los propios participantes han recopilado datos de los diferentes códigos y los han etiquetado a mano.

La evaluación se realizará comparando los resultados generados automáticamente con los resultados generados por la anotación manual de expertos. Los participantes han utilizado diferentes métricas (precisión, Recall, F1-Score...) para calcular el rendimiento del modelo. Aunque son métricas válidas, la métrica aplicada por CANTEMIST para fines de evaluación es la Precisión Media Ponderada (MAP), dada por la fórmula:

$$\mathbf{MAP} = \frac{\sum_{\mathbf{q=1}}^{\mathbf{Q}} \mathbf{AveP(q)}}{\mathbf{Q}} \tag{2.8}$$

Expresión matemática de la métrica MAP

donde Q es el número de consultas del conjunto y AveP (q) es la precisión media (AP) para una consulta concreta q. Lo que la fórmula nos indica es que, para una consulta dada, q, calculamos su AP correspondiente, y luego la media de todos estos puntajes AP nos daría un solo número, llamado mAP, que cuantifica cómo de bueno es el modelo entrenado cuando realiza la consulta.

### 2.1.6. Clasificación de documentos clínicos utilizando redes neuronales

A lo largo de los años y, sobre todo, recientemente, este tipo de técnicas son las más utilizadas. No solo se han utilizado redes neuronales LSTM y GRU básicas, también se han combinado para conseguir mejores resultados en la clasificación del texto de informes de imágenes (Li Qiang, 2020).

Además de estas técnicas de clasificación, recientemente el uso de BERT ha aumentado para resolver tareas de clasificación, no solo general, sino también enfocadas al ámbito clínico para la clasificación de textos. Para ello, se recurre a los distintos modelos preentrenados de BERT y se comparan los resultados de los modelos propuestos (López García, 2021).

Aunque el uso de BERT no siempre garantiza buenos resultados cuando se trata de clasificación multi-etiqueta de textos clínicos. (Shang Gao, 2020) encontraron en su comparación de rendimiento de diferentes modelos que el rendimiento de BERT rara vez era mejor que otros modelos con arquitecturas más simples como un algoritmo de Machine Learning base de HiSAN (*Hierarchical Self-attention Network*)(S. Gao y Ramanathan, 2019) o que una red CNN, aun cuando se limitan los tokens que pueden aceptar estas arquitecturas, para simular la limitación de tokens de BERT, que es

de 512 tokens. Ni siquiera el modelo de BERT preentrenado con textos de Pubmed preprocesados y las notas clínicas del corpus MIMIC-III, el modelo BlueBERT, pudo superarlos. (Y. Gu y Poon, 2020) muestran que una debilidad significativa de los modelos preentrenados de BioBERT y BlueBERT es que utilizan el vocabulario original de WordPiece de BERT, que ha sido generado a partir del corpus de Wikipedia y BooksCorpus. Por otra parte, se puede construir el vocabulario de WordPiece directamente en el dominio de interés, el clínico en nuestro caso, para evitar que las palabras clave importantes se segmenten en múltiples subtokens y esto conduce a una mayor precisión en las tareas posteriores.

## 2.2. Limitaciones de los sistemas de codificación de CANTEMIST

En esta sección vamos a presentar y describir las limitaciones que han encontrado los participantes al aplicar sus diferentes métodos propuestos, junto con las soluciones que propusieron. El objetivo de este apartado es mostrar los problemas que se puedan presentar y con ese conocimiento poder diseñar un sistema que pueda resolverlos.

### 2.2.1. Problema de la longitud máxima

Como principal desventaja, una de las características de BERT es que su arquitectura basada en Transformer está diseñada para procesar una secuencia de entrada de sub-tokens con una longitud limitada de N= 512, en la implementación original de BERT. Esto presenta un problema para esta tarea porque todos los archivos que componen el corpus contienen secuencias de más de 750 de sub-tokens, muy por encima de la longitud máxima admitida por BERT. En el caso de GPT-2 la longitud máxima admitida es de 1024 tokens, que permitiría procesar los archivos más cortos. Aun así, el problema continúa con los archivos más grandes que pueden tener hasta 1.800 tokens aproximadamente. Esta desventaja se ha tenido en cuenta entre los participantes de CANTEMIST porque la solución no es tan simple como segmentar el texto de forma aleatoria. Hay que elegir una aproximación óptima que nos permita mantener el contexto de las palabras en cada subsecuencia.

Una solución común para realizar tareas de etiquetado de secuencias en documentos largos es definir una unidad de procesamiento más granular,

como las oraciones o los párrafos. Es probable que una oración quede dentro de 512 tokens, aunque es menos probable que esto ocurra a nivel de párrafo, por lo que la tarea se puede realizar sin recortar ninguna parte potencialmente relevante de un documento de entrada. Además, este enfoque presenta varios riesgos: los divisores de oraciones pueden introducir errores; las oraciones aisladas pueden carecer de información relevante para la tarea objetivo, y por último, las longitudes de oraciones desequilibradas pueden conducir a un uso ineficiente de los recursos computacionales (García-Pablos, Pérez, y Cuadros, 2020). Para solucionar este inconveniente, las diferentes propuestas de los anteriores participantes de CANTEMIST fueron las siguientes:

Los participantes (López-García et al., 2020) aplicaron un enfoque de clasificación basado en fragmentos desarrollado inicialmente para abordar la tarea CodiEsp-D. Para adaptarlo a la tarea de CANTEMIST primero se divide cada documento clínico del corpus en pequeños fragmentos de texto. Utilizando las anotaciones de códigos eCIE-O-3.1 disponibles en otra subtarea de CANTEMIST (CANTEMIST-NORM), se anotó cada fragmento con los códigos de oncología que ocurren exclusivamente dentro del fragmento. Para finalizar, se usan los fragmentos anotados para realizar el ajuste supervisado del modelo BERT en una tarea de clasificación de múltiples etiquetas a nivel de fragmento.

Sin embargo, el enfoque anterior conlleva mucho trabajo manual, lo que puede llevar a errores humanos y a emplear demasiado tiempo en la tarea. Una solución muy atractiva y muy utilizada por los participantes de CANTEMIST es crear una "ventana deslizante" que pueda permitir conservar el contexto de las secuencias. De esta manera, no se pierde información ni posibles códigos por el camino, además de que evita tener que realizar trabajo manual. Por ejemplo, la solución propuesta por (Chapman y Neumann, 2020) fue desarrollar un componente de modelo al que llamaron "agrupación de documentos" que encaja cada documento en lotes de distintos tamaños, de modo que cada lote es solo un documento, y utiliza una "ventana deslizante" para que haya una superposición entre las secuencias de un lote de tal manera que se pueda conservar el contexto. Para ello, utilizaron un tamaño máximo de lote por GPU de 10 tokens y un tamaño de superposición de 50, lo que significa que cada secuencia siguiente en un lote contenía los últimos 50 tokens de la secuencia anterior en el lote. Finalmente, realizaron un agrupamiento máximo sobre la dimensión de lote de los logits.

Se utilizó una aproximación de ventanas deslizantes en secuencias de 502 tokens, tras un pre-procesamiento mediante el tokenizador de BERT pre-entrenado, con secuencias de sub-palabras divididas en ventanas de una longitud fija de 300. Posteriormente, se añadieron los contextos circundantes de tamaño 100, rellenando según fuera necesario para obtener subsecuencias de tamaño C + W + C. Finalmente, los tokens [CLS] y [SEP] de BERT se agregan a cada subsecuencia. Una máscara indica qué posiciones de secuencia son parte de la ventana y cuáles forman el contexto. Se atienden tanto los contextos como las posiciones de la ventana para construir las incrustaciones contextuales de BERT, pero la función de pérdida solo se calcula para las posiciones dentro de la ventana.

Una vez se ha elegido una técnica, se pueden hacer predicciones con las subsecuencias resultantes.

En nuestro caso, hemos optado por realizar una segmentación de los textos del conjunto de datos en secuencias de 200 tokens (secuencia máxima que se ha establecido a la hora de tokenizar los fragmentos de texto) con una superposición de 50 tokens, lo que nos permite no perder el contexto de cada palabra.

### 2.2.2. Problema del conjunto de datos desbalanceado

En el subconjunto de anotaciones perdidas, el 8 % de los códigos contienen una "H". Este porcentaje es tan bajo como el 2 % en todo el conjunto de pruebas. Además, el 13,2 % de las anotaciones perdidas incluyen el sexto dígito de diferenciación en su código (el sexto dígito en eCIE-O indica la diferenciación del tumor). Por el contrario, este porcentaje es del 5,6 % en todo el conjunto de test. Además, los códigos de prueba perdidos son menos frecuentes en los conjuntos de entrenamiento y desarrollo. La mediana de apariciones de los códigos perdidos en el conjunto de entrenamiento y desarrollo es de 1, mientras que para los códigos del conjunto de test es 3. Por último, el 20,8 % de las anotaciones perdidas tienen el código de metástasis (8000/6), mientras que este código representa el 34,6 % del conjunto de prueba completo. Se puede afirmar que los códigos perdidos son más específicos y menos frecuentes. En la imagen 2.5 podemos ver la incidencia por código.

En el conjunto de datos normalmente encontramos una o más etiquetas eCIE-O-3.1 por documento. (Chapman y Neumann, 2020) observaron que muchos de los códigos solo estaban en el conjunto de test o en el de desarrollo

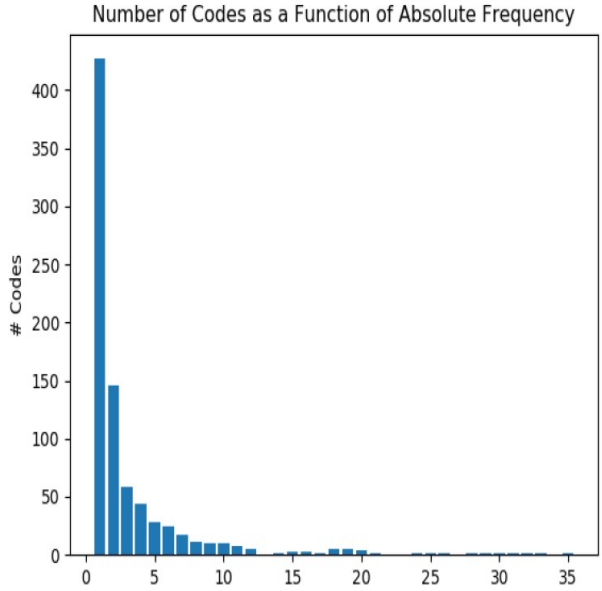



Figura 2.5: Frecuencia de los códigos en el corpus de CANTEMIST (Chapman y Neumann, 2020)

(167 y 208, respectivamente). También observaron que más de la mitad de todos los códigos presentes en la totalidad del conjunto de datos solo tienen una única instancia de entrenamiento positivo, y el 67 % de los 850 códigos tienen solo 1-2 instancias de entrenamiento positivo, etc. Mientras tanto, el código más frecuente, 8000/6, se asigna a más de 1000 documentos en todas las divisiones de datos. Esto es indicativo de un gran desequilibrio en los conjuntos de datos.

Debido a esta extraña morfología (el uso de /H) y a lo desequilibrado que se encuentra el conjunto de datos de CANTEMIST, (García-Pablos, Pérez, y Cuadros, 2020) lo han abordado como un problema de etiquetado de secuencia multitarea. Es decir, han dividido los códigos eCIE-O-3.1 en varias partes, cada una de las cuales comprende un objetivo de clasificación. Los objetivos de clasificación que seleccionaron fueron: los primeros 3 dígitos del código, el cuarto dígito, y los dígitos de "Comportamiento y Calificación" junto con el indicador H, como una sola variable. Además, añadieron la etiqueta BIO que indica si un token es parte de una entidad o no lo es, además de ayudar reconocer entidades contiguas. Esta etiqueta ayuda a discernir si un token es el comienzo de una entidad de morfología tumoral (B -, "Begin"), si el token está dentro de una entidad (I-, "In") o si el token no es parte de

una entidad (O, "Fuera"). La desventaja de este método es que hay que hacer un post-procesamiento de las 4 predicciones por token para transformarlas al formato de BRAT, donde cada entidad de morfología tumoral detectada, ya sea uno o varios tokens, se asocia con un código eCIE-O-3.1 válido.

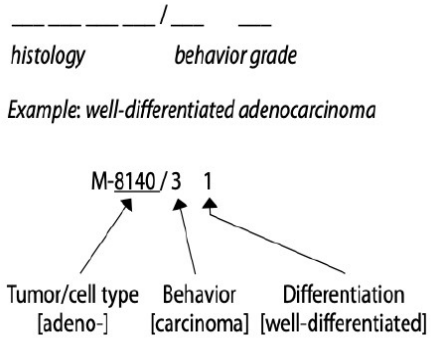



Figura 2.6: Estructura de un código morfológico (Chapman y Neumann, 2020)

En (López-García et al., 2020), para adaptar los modelos Multilingual BERT y BERT- ScieLO a las características distintivas del dominio de los textos clínicos en español, se ha utilizado como corpus para el pre-entrenamiento el "Sistema de Información Oncológica de Galén" (Ribelles et al., ), que consta de 30.9 mil documentos sin etiquetar de notas clínicas oncológicas redactadas por médicos de los Servicios de Oncología del Hospital Universitario Virgen de la Victoria (HUVV) y del Hospital Regional Universitario (HRU) de Málaga, España, aunque también utilizaron un corpus en inglés que fueron los resúmenes de alta de la base de datos MIMIC-III (57.4 mil documentos) (Johnson et al., 2020). Se utilizaron los dos corpora como un solo corpus clínico bilingüe, utilizado para realizar el pre-entrenamiento no supervisado del modelo multilingüe.

Los investigadores de (Chapman y Neumann, 2020) buscaron una fuente externa de datos para completar las descripciones de los códigos eCIE-O-3.1 (ECI, 2020), pero faltaron algunos para algunos códigos que especifican el tipo de celda, el comportamiento y el grado. Se encargaron de generar descripciones propias tomando la descripción correspondiente de los primeros cinco dígitos de ese código (tipo de celda + comportamiento) y simplemente agregando el texto que corresponde al dígito final, o la calificación. Como solo hay cuatro dígitos posibles para la calificación, solo hay cuatro descripciones

posibles para la calificación.

# Capítulo 3

# Sistema automático de clasificación de códigos CIE

En este capítulo del presente trabajo vamos a presentar la solución que hemos planteado para resolver la tarea de asignar automáticamente códigos eCIE-O-3.1 a historiales de pacientes oncológicos. En primer lugar, vamos a describir en qué consiste un sistema de clasificación multietiqueta y presentaremos algún ejemplo de etiquetado del corpus que vamos a usar. Posteriormente, mostraremos paso a paso el método que hemos propuesto para resolver esta tarea.

## 3.1. Sistema de clasificación multietiqueta

Como hemos mencionado, el objetivo del presente trabajo de fin de máster es desarrollar sistemas automáticos de codificación clínica capaces de asignar códigos eCIE-O-3.1 a historiales clínicos de pacientes oncológicos en español, siguiendo las directrices de la tarea CANTEMIST, así como comparar nuestros resultados con los participantes de CANTEMIST para dicha tarea.

Como hemos indicado, para el correcto funcionamiento de los sistemas de clasificación supervisados se necesitan ejemplos previamente etiquetados con los que entrenar el algoritmo, es decir, un conjunto de entrenamiento, en nuestro caso el conjunto de entrenamiento anotado por expertos en codificación clínica que facilita CANTEMIST.

En la imagen de la figura 1.2, pudimos ver un ejemplo de etiquetado de un historial clínico de CANTEMIST. Donde encontramos el nombre del

informe médico de referencia a la izquierda, junto con sus respectivos códigos morfológicos a la derecha. Recordemos, que solo hay un código por fila, por tanto, un mismo informe puede aparecer en varias filas, correspondiendo ese número de filas al número de códigos a los que está asociado.

Este corpus se ha dividido aleatoriamente en tres subconjuntos de datos: el conjunto de datos de entrenamiento, de desarrollo y de test para entrenar, ajustar los parámetros y para testear, respectivamente, el algoritmo que se va a entrenar. Vamos a entrenar distintos modelos pre-entrenados de BERT disponibles en (**?**) junto con una capa Dropout y una capa final Dense. Los modelos BERT entrenados han sido el Multilingua BERT, RoBERTa, BETO y BERT-SciELO.

Por último, se emplean algoritmos de clasificación supervisada con la intención de obtener para cada documento una lista ordenada de sus correspondientes códigos eCIE-O-3.1. A lo largo del trabajo, se presenta el ciclo completo de preprocesamiento de los conjuntos de datos y la construcción del sistema de clasificación.

Para resolver esta tarea de clasificación multi-etiqueta, se ha optado por comparar el rendimiento de los modelos multilingual BERT, BETO (José Cañete y Fuentes, 2020) (modelo basado en BERT entrenado con un corpus exclusivamente en español), RoBERTa (Yinhan Liu, 2019) y BERT-SciELO (Liliya Akhtyamova y Cardiff, 2019) (modelo basado en BERT ajustado con el corpus de SciELO (SciELO, 2020)). SciELO consiste en un corpus de mil millones de frases extraídas de la web de *Scientific Electronic Library Online*.

A lo largo del siguiente apartado, se van a describir los pasos que se han seguido además de explicar cómo se han resuelto los problemas que han ido surgiendo. El código se ha ejecutado en Google Colab, que proporciona una GPU gratuita.

## 3.2. Etapas del desarrollo del clasificador

Como guía, los siguientes apartados muestran el proceso que se va a seguir en este trabajo, desde el preprocesamiento de los textos de entrenamiento y de test, la definición de los modelos que se han desarrollado, el entrenamiento de dichos modelos, algunos experimentos de parametrización y su posterior evaluación, como se muestra en la figura 3.1.

Todos los pasos que hemos seguido en el desarrollo son comunes salvo

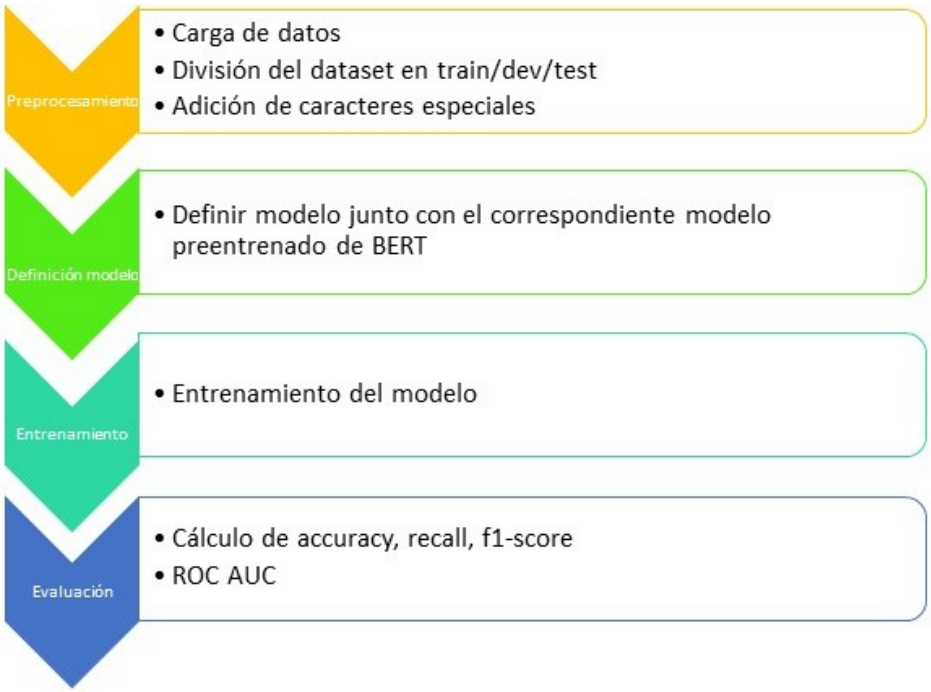



Figura 3.1: Diagrama del trabajo

el segundo, el paso de "definición del modelo", donde la carga del modelo base es diferente según el modelo pre-entrenado que se escoja. Por ejemplo, el modelo de RoBERTa se cargará de una manera diferente al modelo Multilingual de BERT, porque RoBERTa no hace uso de *token_type_ids*. Es en este mismo paso donde también se asignarán las modificaciones a los diferentes hiperparámetros. Los valores de los hiperparámetros pueden ser claves cuando se busca un rendimiento óptimo de un modelo. Estas modificaciones las definiremos y explicaremos más adelante, en el apartado 3.3.

### 3.2.1. Preprocesamiento del texto

Como en todas las tareas de *Deep Learning* o *Machine Learning*, antes de entrenar un modelo se debe hacer un preprocesamiento de los datos, en este caso texto. Para alimentar a la arquitectura de BERT, como ya hemos visto anteriormente, se deben obtener los *embeddings* de posición, de segmentación y de token. Para ello, el primer paso es usar el tokenizador BERT para dividir primero la oración en tokens. Para ello, se ha utilizado AutoTokenizer que se

trata de una clase de tokenizador genérico que se instancia como una de las clases de tokenizador que hay disponibles en la biblioteca de transformers de huggingface (**?**). En los argumentos del tokenizador se han truncado los segmentos a una longitud máxima de 200 especificada con el argumento *max_ length* o a la longitud de entrada máxima aceptable para el modelo si no se proporciona ese argumento. Esto truncará token por token, eliminando un token de la secuencia más larga del par si se proporciona un par de secuencias (o un lote de pares). Todos los argumentos del tokenizador son comunes a todos los modelos de BERT utilizados a excepción de RoBERTa, que no cuenta con *token_ type_ ids* y, por tanto, se le ha asignado el valor de False. Es decir, con RoBERTa no es necesario que se indique qué token pertenece a qué segmento. Simplemente separa sus segmentos con el token de separación *tokenizer.sep_ token*. En el proceso de tokenización se agregan los tokens especiales necesarios para las clasificaciones de oraciones (estos son [CLS] en la primera posición y [SEP] al final de la oración).

Al tratarse de una tarea que pretende resolver un problema de clasificación multi-etiqueta, se han codificado las etiquetas de los conjuntos de datos de entrenamiento, test y validación mediante el uso del MultiLabelBinarizer, disponible en la librería *open-source* scikit-learn. Con esto se consigue un array de longitud del número total de etiquetas (códigos eCIE-O) con valores 0 y 1 para cada informe médico. El valor 1 indica la presencia de un cierto código eCIE-O en el informe médico, mientras que el valor 0 indica su ausencia.

Para resolver el problema de la longitud de las oraciones se ha segmentado el texto en secuencias de 200 tokens, que es la secuencia máxima que se ha establecido a la hora de tokenizar los fragmentos de texto, con una superposición de 50 tokens para no perder el contexto de cada palabra (Raghavendra Pappagari, 2019).

Con todos los fragmentos de texto tokenizados, con los *embedding* y con las etiquetas en su codificación correcta, se obtienen los tensores de todos esos valores y se alimenta a un *DataLoader* con un *batch_ size* con un valor de 32. Para el conjunto de entrenamiento se utiliza un *Sampler* aleatorio que elige textos del conjunto de entrenamiento de forma aleatoria para entrenar en cada *epoch*. La razón es que al utilizar un *Sampler* aleatorio podemos evitar el sobreajuste del modelo que vamos a entrenar. Esto no se hace para el conjunto de test porque no es necesario, ya que no influye nada el orden

que tengan los textos de test. Y, por tanto, se usa un *Sampler* secuencial, que su función es seguir los textos del conjunto de test de forma secuencial, como su nombre indica.

### 3.2.2. Definición de los modelos y entrenamiento

Se ha creado una arquitectura base común para los modelos. Los hiperparámetros de los modelos base también son comunes. Solo hay algunos pequeños cambios, como en el caso de RoBERTa, que no cuenta con *token_type_ids* como se ha mencionado anteriormente. Los *token_type_ids* se representan como una máscara binaria que identifica los dos tipos de secuencia en el modelo. La primera secuencia, el "contexto" utilizado para la pregunta, tiene todos sus valores representados por un 0, mientras que la segunda secuencia, correspondiente a la "pregunta", tiene todos sus valores representados por un 1.

Lo que se muestra en la imagen 3.2, que podemos ver más abajo, es nuestra arquitectura propuesta del modelo base multilingüe de BERT con la capa de Dropout y la capa densa de salida. Los demás modelos tienen una arquitectura similar con el modelo preentrenado de BERT correspondiente:

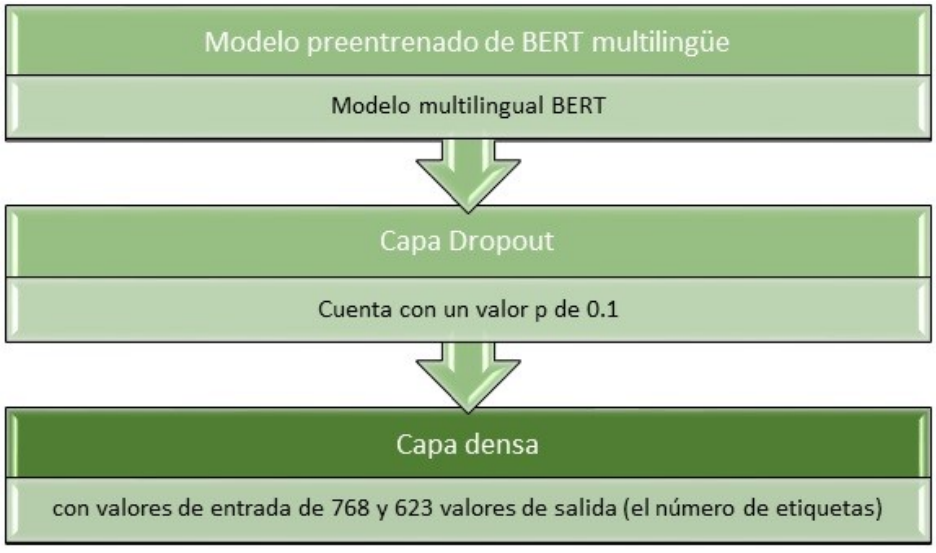



Figura 3.2: Arquitectura del modelo de BERT multilingüe

Entre la arquitectura del modelo de BERT y la capa densa completa-

mente conectada hay una capa de *Dropout* (Nitish Srivastava, 2014), una técnica de regularización que se utiliza para reducir el sobreajuste en redes neuronales artificiales mediante la prevención de coadaptaciones complejas en los datos de entrenamiento. Es decir, durante el entrenamiento, la capa de *Dropout* asigna a cero aleatoriamente algunos de los elementos del tensor de entrada con probabilidad p utilizando muestras de una distribución de Bernoulli. Cada nodo será asignado a cero de forma independiente en cada *epoch*, es decir, en cada *epoch* habrá diferentes nodos elegidos aleatoriamente que tendrán un valor asignado de cero. Esto ha demostrado ser una técnica eficaz para la regularización y la prevención de la coadaptación de neuronas (Geoffrey E. Hinton, 2012). A la capa de *Dropout* se le ha asignado un valor del parámetro p de 0,1. Esto quiere decir que desde el modelo preentrenado BERT hasta la capa densa, cada nodo tiene un 10 % de probabilidades de ser asignado a un valor de 0. En la imagen 3.3 se puede ver cómo algunos nodos aleatorios quedan eliminados (con valor 0) al aplicar una capa de *Dropout*

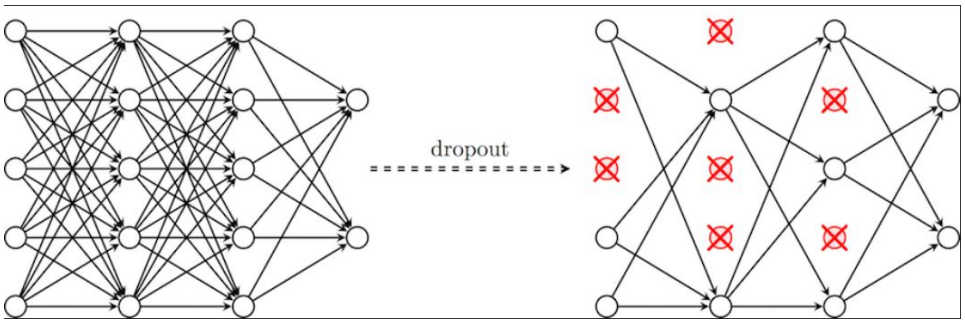



Figura 3.3: Dropout

La última capa de clasificación de etiquetas múltiples es una capa densa completamente conectada de tamaño 623, que son las 623 etiquetas posibles de los códigos eCIE-O-3.1 disponibles en el conjunto de entrenamiento, y usamos la función de activación sigmoidea para obtener probabilidades independientes de cada clase. La función sigmoidea se trata de una función matemática que tiene la característica de poder tomar cualquier valor real y asignarlo entre 0 y 1 con la forma de la letra "S", que podemos observar en la imagen 3.4. La función sigmoidea también se denomina función logística. La función sigmoidea se representa matemáticamente de la siguiente forma:

$$\phi(z) = \frac{1}{1 + e^{-z}} \tag{3.1}$$

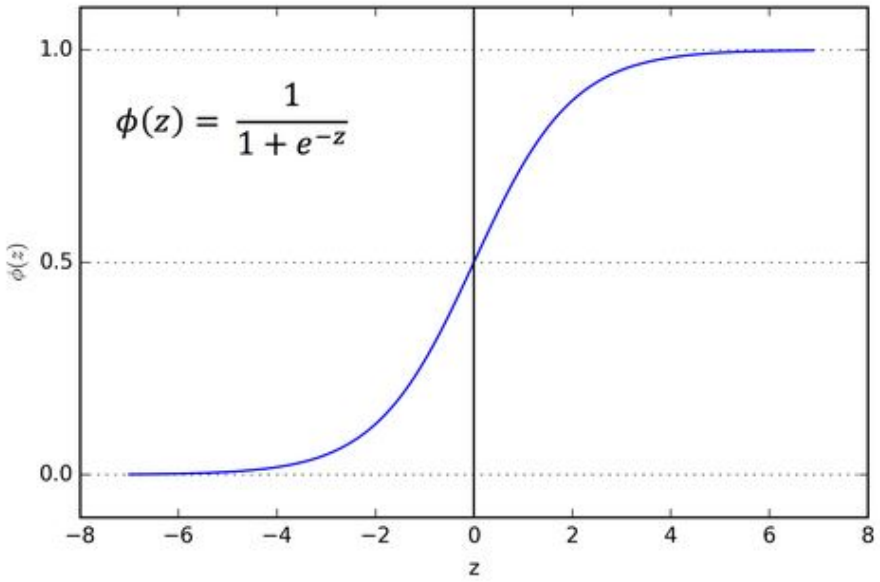



Figura 3.4: Función Sigmoid

La función sigmoidea actúa como una función de activación en el aprendizaje automático que se utiliza para agregar no linealidad en un modelo de aprendizaje automático, en palabras simples, decide qué valor pasa como salida y qué valor no pasa. Es decir, en la función sigmoidea $\phi(z) = \frac{1}{1+e^{-z}}$ si el valor de $z$ va a infinito positivo, entonces el valor predicho de $y$ se convertirá en 1 y si va a infinito negativo, entonces el valor predicho de $y$ se convertirá en 0. Por otra parte, si el resultado de la función sigmoidea es mayor que 0.5 entonces clasificamos esa etiqueta como clase 1 o clase positiva y si es menor que 0.5 entonces podemos clasificarla en clase negativa o etiqueta como clase 0.

El número de epochs elegidos para el entrenamiento han sido 4, siendo cada epoch la cantidad de veces que se desea pasar el conjunto de datos completo. De hecho, los creadores de BERT recomiendan solo entre 2-4 epochs de entrenamiento (Devlin et al., 2018) para ajustar BERT en una tarea específica de PLN, en comparación con los cientos de horas de GPU necesarias para entrenar el modelo BERT original o un LSTM desde cero.

En cuanto a la función de pérdida, se ha utilizado la entropía cruzada binaria con la clase de *BCELoss()* de PyTorch. La clase *BCELoss()* se utiliza para medir el error de una reconstrucción, en nuestro caso, un codificador automático. BCELoss se basa en la entropía cruzada binaria. La entropía cruzada binaria compara cada una de las probabilidades que ha predecido el modelo con la etiqueta real de la clase, que puede ser 0 o 1. Luego calcula la puntuación que penaliza las probabilidades basándose en la distancia del valor esperado. Eso significa cómo de cerca o de lejos está dicha probabilidad del valor real. La única función de activación compatible con la entropía cruzada binaria es la función sigmoidea, que es la que hemos fijado en la capa densa final. La razón de por qué esto es así se debe a que la entropía cruzada binaria necesita calcular los logaritmos de $\hat{y}_i$ y $(1 - \hat{y}_i)$, que solo pueden existir si el valor de $\hat{y}_i$ se encuentra entre 0 y 1. A diferencia de Softmax, es independiente para cada componente de vector (clase), lo que significa que la pérdida calculada para cada componente de vector no se ve afectada por los valores de otros componentes.

En cuanto al optimizador, se ha utilizado AdamW con los parámetros por defecto. De esta forma, se implementa el algoritmo de Adam con corrección de caída de peso (Ilya Loshchilov, 2019), es decir, con regularizador L2. Al optimizador se le ha aplicado un *learning rate schedule* para que el *learning rate* vaya disminuyendo por cada inicio de *epoch*. El mejor *learning rate* depende del conjunto de datos y la tarea que está aprendiendo su modelo. En muchos casos, también depende de lo cerca que esté su modelo de la solución óptima. Un *learning rate schedule* ayuda con ambos aspectos, ya que abarca un rango de *learning rate* que se vuelve más pequeño a medida que se espera que el modelo alcance su solución óptima.

### 3.2.3. Evaluación de los modelos desarrollados

Como hemos mencionado en capítulos anteriores, para poder entrenar y evaluar los modelos que hemos desarrollado, se ha segmentado el conjunto de datos de CANTEMIST en 3 subconjuntos: el de entrenamiento, de desarrollo y de test que sirven para entrenar, ajustar los parámetros y para testear, respectivamente. Hemos utilizado los conjuntos de entrenamiento y desarrollo para entrenar y afinar el modelo, como sus nombres indican. El conjunto de test, lo hemos utilizado para extraer los resultados que hemos plasmado en la tabla 4.2.

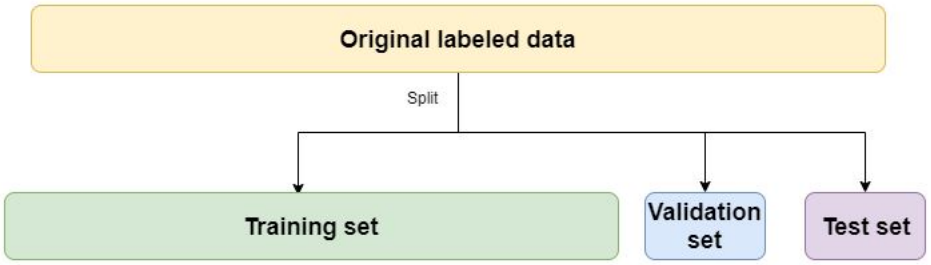



Figura 3.5: Segmentación de los subconjuntos de entrenamiento, desarrollo y prueba

CANTEMIST proporciona 4 conjuntos de datos, uno de entrenamiento, dos de desarrollo (dev1 y dev2) y uno de test. El conjunto de test, servirá para comprobar la calidad de inferencia del modelo frente a informes médicos que el modelo no ha visto previamente. Para entrenar y afinar se combinaron los conjuntos de entrenamiento y de desarrollo dev1 y dev2, y de ese conjunto se eligió un 25 % de textos aleatorios para usarlos como subconjunto de validación. Esto es, cada vez que en el entrenamiento se completa un *epoch*, se utilizará ese conjunto de validación para evitar el sobreajuste del modelo y para evaluar el rendimiento de modelos con diferentes valores de hiperparámetros. Por lo tanto, se ha utilizado la mayor parte del conjunto de datos para entrenamiento y dos subconjuntos notablemente más pequeños para desarrollo y test, como se puede ver en la imagen 3.5.

Una vez entrenado el modelo, se ha evaluado los modelos con ciertas métricas que hemos utilizado para comparar nuestros resultados con los que obtuvieron los participantes de CANTEMIST. Como (Miranda-Escalada et al., 2020) recopilaron los resultados de todos los participantes de CANTEMIST para todas las subtareas, incluyendo la de codificación (los resultados de los participantes con mejores resultados en codificación están plasmados en la tabla 4.1), y todos usaban las métricas de precisión, exactitud, *recall* y el valor-F, aunque en CANTEMIST también usaban MAP (*Mean Average Precision*), nosotros hemos optado por usar las métricas de precisión, *recall*, valor-F y la ROC Auc. Las métricas que hemos utilizado para evaluar los modelos se explican más exhaustivamente en el capítulo. 4.1

Básicamente, la definición general de *Average Precision* (AP) es tratar de encontrar el área bajo la curva de precisión-*recall* anterior. Por tanto,

el MAP consiste en la media de varias AP. La precisión promedio (AP) es una forma de resumir la curva de *recall* de precisión en un solo valor que representa el promedio de todas las precisiones. Según la ecuación 3.2, el AP se calcula utilizando un bucle que pasa por todas las precisiones/*recall*, se halla la diferencia entre el valor de *recall* actual y el siguiente y finalmente se multiplica por la precisión actual. Con ese resultado, se halla la media para obtener el valor de MAP.

$$AP = \sum_{k=0}^{k=n-1} [Recalls(k) - Recalls(k+1)] * Precisiones(k) \tag{3.2}$$

Donde:

$Recalls(n) = 0$
$Precisiones(n) = 1$
n = Número de umbrales

## 3.3. Experimentos y pruebas de parametrización de los modelos

A los modelos base anteriormente descritos, se les han realizado pequeñas modificaciones en los valores de los parámetros con el fin de poder obtener un mejor resultado. Uno de los cambios ha sido disminuir el número de *epochs* de entrenamiento a 3 para todos los experimentos. También se ha probado a aumentar el número de *epochs* a 9 y 10. El objetivo de estos cambios era comprobar el rendimiento de los distintos modelos obtenidos con diferentes *epochs*. Esperamos obtener mejores resultados según se aumentan los *epochs*.

También se ha modificado el valor del *learning rate*, o tasa de aprendizaje, de 1e-3, que es el valor asignado por defecto, a dos valores que son 3e-5, que es un valor recomendado de huggingface para tareas de PLN, y 5e-5. El *learning rate* es un hiperparámetro que controla cuánto estamos ajustando los pesos de nuestra red con respecto a la pérdida de gradiente. Cuanto menor sea el valor, más lento se podrá desplazar a lo largo de la pendiente descendente. Si bien esto podría parecer una buena idea (usar un *learning rate* bajo) para asegurarnos de que no estamos perdiendo ningún mínimo

local, lo que se busca es llegar al mínimo global, que es el punto en el que una función toma el valor mínimo. En la imagen 3.6 se pueden ver gráficamente un punto mínimo global, el valor óptimo que buscamos para tener un modelo con un rendimiento óptimo, y varios puntos mínimos locales que queremos evitar porque para el modelo pueden parecen ser mínimos globales, pero no lo son, por lo que el entrenamiento se puede encallar y no alcanzar nunca sus valores óptimos. También podría significar que tardaremos mucho en converger, especialmente si nos atascamos en una región de meseta. El objetivo de este experimento era comprobar si el tamaño del *step* en cada iteración mientras se mueve hacia un mínimo de una función de pérdida era el óptimo. Continuando con las modificaciones al optimizador, se ha aplicado un valor de 0.01 al parámetro *weight_ decay_ rate* de AdamW. El objetivo de este cambio es mantener los pesos pequeños y evitar en lo posible la explosión del gradiente. Esto se debe a que se le van aplicar a los pesos una penalización (la *l2 norm*) y ese valor se agrega a la función de pédida de manera que cada iteración de la red intentará optimizar los pesos del modelo además de la función de pérdida.

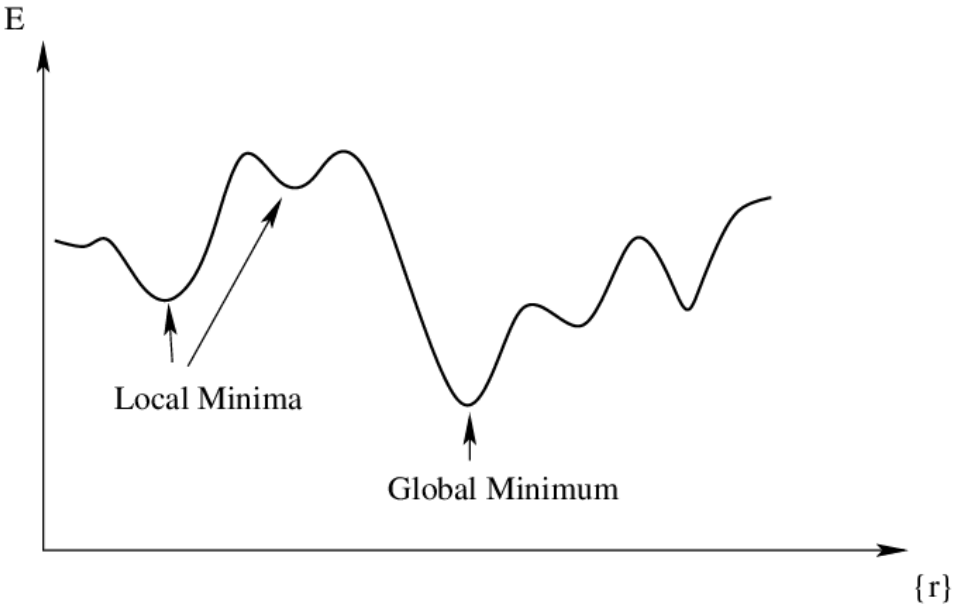



Figura 3.6: Mínimo global versus mínimo local

Por último, se ha probado a aplicar el RMSProp (*Root Mean Square Propogation*), un optimizador de *learning rate* adaptativo propuesto por

Geoffrey Hinton. Rmsprop se desarrolló como una técnica estocástica para el aprendizaje por mini lotes, ya que gradientes de funciones muy complejas tienden a desaparecer o explotar a medida que los datos se propagan a través de la función. RMSprop se ocupa de este problema mediante el uso de un promedio móvil de gradientes cuadrados para normalizar el gradiente. Esta normalización equilibra el tamaño del paso (impulso), disminuyendo el paso para gradientes grandes para evitar la explosión y aumentando el paso para gradientes pequeños para evitar la desaparición. Entonces, el objetivo de cambiar el optimizador RMSProp en vez del AdamW era comprobar cómo respondía el modelo al utilizar un optimizador que hace que el *learning rate* no disminuya demasiado rápido evitando la convergencia. La teoría es que RMSProp capaz de reducir continuamente la pérdida a lo largo del proceso de formación hasta alcanzar el mínimo local. Junto con RMSProp, se ha aumentado el valor del parámetro p de la capa de Dropout de 0.1 a 0.25 porque al contrario que el optimizador AdamW, RMSProp es más propenso al *overfitting*.

# Capítulo 4

# Evaluación y resultados

Este capítulo describe la metodología utilizada para evaluar el sistema de clasificación multietiqueta, a la vez que presenta los resultados obtenidos en la evaluación de la tarea sobre el conjunto de evaluación que ofrece de CANTEMIST.

## 4.1. Metodología de evaluación

Se han elegido 4 métricas para evaluar el rendimiento de los modelos entrenados. Estas métricas también permitirán comparar los resultados conseguidos en este trabajo con los resultados de los participantes de CANTEMIST. En el siguiente apartado, se van a definir las métricas con las que se han evaluado los modelos.

### 4.1.1. Métricas de evaluación

La evaluación de un modelo es una parte importante de la creación de un modelo de aprendizaje automático eficaz. La métrica de evaluación de clasificación más frecuente y más conocida es la precisión, o *accuracy*. Sin embargo, utilizar esta métrica puede ser engañoso en algunas situaciones. Es por eso que en este trabajo se ha decidido utilizar las métricas de precisión, el recall, el valor-f y la curva ROC.

En una tarea de clasificación multi-etiqueta, las métricas pueden calcularse entre clases o entre muestras, lo que da como resultado muchos números. Resumir los resultados en pocos números para fines de comparación es complicado y puede hacerse de varias maneras. La elección correcta de pro-

mediar para cada experimento depende de la cantidad de clases, la cantidad promedio de temas asignados a los documentos, la cantidad de documentos en cada clase y la sensibilidad de las distintas clases a las decisiones equivocadas tomadas por el clasificador.

A continuación se describen las distintas métricas mencionadas:

- Precisión: La precisión es una métrica de evaluación que determina el número de predicciones correctas realizadas por el modelo.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{4.1}$$

- Exactitud o *Accuracy*: se refiere a cuán cerca del valor real se encuentra el valor medido. En términos estadísticos, la exactitud está relacionada con el sesgo de una estimación. Cuanto menor es el sesgo más exacta es una estimación. Cuando se expresa la exactitud de un resultado, se expresa mediante el error absoluto que es la diferencia entre el valor experimental y el valor verdadero.

$$Acc = \frac{TruePositive + TrueNegative}{TruePositive + FalsePositive + TrueNegative + FalseNegative} \tag{4.2}$$

- Recall: esta métrica mide si todo lo que debería predecirse se predice. La recall debe utilizarse como una métrica de rendimiento cuando la importancia de los falsos negativos es alta y la importancia de los falsos positivos es baja.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{4.3}$$

- *F-measure*: es la media armónica entre la precisión y el recall. El valor-F debe usarse como una métrica de rendimiento si el número de muestras en la clase positiva es muy inferior y si el costo de los falsos positivos y el costo de los falsos negativos son muy altos.

$$F\text{-}measure = \frac{2}{1/Precision + 1/Recall} \tag{4.4}$$

La curva ROC (*Receiver Operating Characteristic*) presenta la sensibilidad de una prueba diagnóstica que produce resultados contínuos, en función de los falsos positivos (complementario de la especificidad), para distintos puntos de corte.

Los micro y macro promedios (para cualquier métrica) calcularán cosas ligeramente diferentes y, por lo tanto, su interpretación es diferente. Un macro promedio calculará la métrica de forma independiente para cada clase y luego tomará el promedio (por lo tanto, tratará a todas las clases por igual), mientras que un micro promedio agregará las contribuciones de todas las clases para calcular la métrica promedio. En una configuración de clasificación de clases múltiples, es preferible el micro promedio si se sospecha que puede haber un desequilibrio de clases (es decir, puede tener muchos más ejemplos de una clase que de otras clases). Se ha elegido el micro promedio de la curva ROC como métrica para comparar los resultados de los modelos porque aplana las etiquetas verdaderas binarizadas y las etiquetas predichas y evalúa el modelo como un clasificador binario único.

Aunque no hubiera habido una mejora considerable en el rendimiento de los modelos, es remarcable observar el acercamiento del modelo multilingual + RMSprop a los resultados óptimos que ha conseguido modelo base de BERT-SciELO. RMSprop también consigue mejorar el rendimiento de BERT-SciELO en un 3 % en la métrica de *accuracy*. En todos los modelos AdamW supera con creces al optimizador RMSProp.

La tabla 4.1 muestra los resultados obtenidos por los participantes de CANTEMIST en la tarea específica de clasificación multi-etiqueta de códigos morfológicos eCIE-O de informes médicos. Estos son los resultados que obtuvieron con el conjunto de test que facilitó CANTEMIST para la evaluación de los modelos.

Vicomtech (García-Pablos, Pérez, y Cuadros, 2020) es el equipo que mejores resultados obtiene en todas las métricas menos en Recall, que es superado por ICB-UMA (López-García et al., 2020). Fadi es el equipo que tiene los segundos mejores resultados en todas las métricas.

La estrategia que consigue los mejores resultados es la que ha seguido (Hassan, Sánchez, y Domingo-Ferrer, 2020) con una red neuronal LSTM bidireccional.

A continuación, se exponen en la tabla 4.2 los resultados del entrenamiento obtenidos de los diferentes modelos de BERT desarrollados en este

| Equipo | Accuracy | f1-score | Recall | mAP |
|---|---|---|---|---|
| ICB-UMA | 0.182 | 0.013 | 0.928 | 0.85 |
| kathrync | 0.007 | 0.268 | 0.51 | 0.39 |
| lasigeBioTM | 0.211 | 0.312 | 0.601 | 0.51 |
| Vicomtech | 0.875 | 0.855 | 0.836 | 0.85 |
| Fadi | 0.826 | 0.832 | 0.838 | 0.79 |
| Bigbyte | 0.794 | 0.861 | 0.73 | 0.68 |
| NLNDE | 0.77 | 0.77 | 0.771 | 0.75 |
| mhjabreel | 0.797 | 0.805 | 0.812 | 0.74 |
| episource | 0.68 | 0.681 | 0.681 | 0.57 |

Tabla 4.1: Tabla de resultados de los participantes de CANTEMIST en la tarea de clasificación de códigos eCIE-O

trabajo, junto con los resultados de los experimientos, para poder comparar y evaluar su eficiencia a la hora de extraer los diferentes códigos eCIE-O de los reportes médicos. En esta tabla no se han añadido los resultados de los experimentos de los cambios de número de epochs y *weight_ decay_ rate*, por no considerarlos relevantes. Cuando se ha usado el optimizador RMSprop y se ha utilizado un valor de *learning rate* de 3e-5, en vez del valor predefinido de 1e-3, los resultados obtenidos han sido los mismos.

Tal y como se observa en la tabla 4.2, no hay grandes diferencias entre los modelos. Los modelos de BETO y Multilingual BERT arrojan los mismos resultados en todas las métricas menos en la curva ROC, que en el caso de BETO es más bajo. El modelo que mejor rendimiento tiene es BERT-SciELO, seguramente debido al corpus con contenido biomédico con el que ha sido pre-entrenado. RoBERTa es el modelo que peor ha respondido al ajuste al corpus de CANTEMIST, lo cual era de esperar porque no es un modelo específicamente entrenado para clasificar modelos en español y tampoco es un modelo que se haya entrenado específicamente para tratar textos clínicos. En cuanto a los resultados de los experimentos, los que mejor se han comportado son los modelos que han utilizado un optimizador RMSprop. También se observa una ligera mejora en los modelos cuando se ha utilizado un valor de 3e-5 para el *learning rate*. Por otra parte, aumentar los *epochs* a 10 no ha supuesto una mejora al modelo, de hecho este cambio produce el efecto contrario, y el valor de 5e-5 para el *learning rate* ha empeorado el rendimiento de los modelos.

Aunque no hubiera habido una mejora considerable, es remarcable ob-

| Modelo | Accuracy | f1-score | Recall | mAP | ROC-Auc |
|---|---|---|---|---|---|
| Multilingual BERT | 0.77 | 0.71 | 0.65 | 0.67 | 0.92 |
| Multilingual BERT 10 epochs | 0.69 | 0.64 | 0.6 | 0.64 | 0.85 |
| Multilingual BERT lr=5e-5 | 0.66 | 0.63 | 0.58 | 0.61 | 0.8 |
| Multilingual BERT+lr=3e-5 | 0.77 | 0.71 | 0.65 | 0.68 | 0.91 |
| Multilingual BERT+RMSprop | 0.77 | 0.71 | 0.65 | 0.65 | 0.91 |
| RoBERTa | 0.83 | 0.73 | 0.66 | 0.73 | 0.93 |
| RoBERTa 10 epochs | 0.72 | 0.63 | 0.59 | 0.63 | 0.8 |
| RoBERTa+lr=5e-5 | 0.78 | 0.7 | 0.63 | 0.65 | 0.88 |
| RoBERTa+lr=3e-5 | 0.81 | 0.73 | 0.66 | 0.71 | 0.92 |
| RoBERTa + RMSprop | 0.83 | 0.73 | 0.65 | 0.74 | 0.92 |
| BETO | 0.81 | 0.73 | 0.66 | 0.69 | 0.82 |
| BETO 10 epochs | 0.71 | 0.63 | 0.59 | 0.63 | 0.8 |
| BETO+lr=5e-5 | 0.76 | 0.68 | 0.62 | 0.66 | 0.77 |
| BETO+lr=3e-5 | 0.81 | 0.73 | 0.66 | 0.68 | 0.82 |
| BETO+RMSprop | 0.81 | 0.68 | 0.68 | 0.7 | 0.91 |
| BERT-SciELO | 0.77 | 0.72 | 0.65 | 0.67 | 0.89 |
| BERT-SciELO 10 epochs | 0.69 | 0.64 | 0.59 | 0.63 | 0.82 |
| BERT-SciELO+lr=5e-5 | 0.72 | 0.67 | 0.61 | 0.64 | 0.82 |
| BERT-SciELO+lr=3e-5 | 0.77 | 0.72 | 0.65 | 0.68 | 0.89 |
| BERT-SciELO+RMSprop | 0.8 | 0.73 | 0.66 | 0.69 | 0.9 |

Tabla 4.2: Tabla de resultados. La fila resaltada en azul corresponde a las puntuaciones más altas para cada métrica de un modelo.

servar el acercamiento del modelo multilingual + RMSprop a los resultados óptimos del modelo base de BERT-SciELO. RMSprop también consigue mejorar el rendimiento de BERT-SciELO en un 3 % en la métrica de *accuracy*. En todos los modelos, RMSProp supera con creces al optimizador AdamW. Además, se ha comprobado que se puede entrenar con 3 epochs y se obtienen exactamente los mismos resultados, con lo que se puede ahorrar en tiempo de computación.

Aunque los resultados que hemos obtenido con los modelos que hemos mostrado en el presente trabajo son algo inferiores a los resultados de los mejores participantes de CANTEMIST en la tarea de clasificación de códigos eCIE-O, con nuestro sistema propuesto hemos conseguido tener mejores resultados que muchos de los participantes.

En el siguinte apartado discutiremos con más detalle los resultados obtenidos y los compararemos con los que obtuvieron los participantes de CANTEMIST.

# Capítulo 5

# Discusión

Este capítulo analiza y discute en profundidad los resultados obtenidos en la evaluación presentada en el capítulo anterior y se comparan con los resultados que obtuvieron los anteriores participantes de CANTEMIST. Se comentan, además, algunas técnicas que podrían ser puntos de partida de futuros trabajos para mejorar el rendimiento de modelos que quieran resolver esta tarea o tareas de clasificación multi-etiqueta similares.

## 5.1. Discusión de los resultados

En este apartado se van a discutir los resultados obtenidos con los modelos de BERT entrenados y a compararlos con los resultados de los demás participantes de CANTEMIST, además de describir los problemas del modelo presentado y sus posibles soluciones.

El hecho de añadir más de 4 *epochs* de entrenamiento no ha aportado ningún valor. Lo único que se ha logrado con esto ha sido sobre entrenar el modelo, con lo que los resultados no han mejorado y, de hecho, han empeorado según los resultados que hemos obtenido en el conjunto de test. Algo similar ha ocurrido con el valor de *learning rate* 5e-5. Una vez pasamos el umbral y tenemos un valor de *learning rate* demasiado bajo, nos encontramos con que el modelo no consigue mejorar. Esto se puede comprobar en los valores que obtenemos en las funciones de pérdida en cada *epoch* porque se puede observar que la función de pérdida apenas varía de la primera *epoch* hasta la última. Sin embargo, el valor 3e-5 sí ha conseguido una mejora en los modelos, aunque estas mejoras no hayan sido significativas.

El mejor resultado de los modelos entrenados y presentados en el pre-

sente trabajo ha sido el modelo de RoBERTa, que aunque no consigue superar a los mejores resultados de los participantes, obtiene unos resultados aceptables. Como se ha mencionado en el capítulo de donde se describen los modelos sección:3.2.2, se ha utilizado el modelo RoBERTa preentrenado junto con una capa de Dropout y una capa densa final para realizar la clasificación multi-etiqueta. RoBERTa es un transformer basado en BERT y modifica algunos de sus hiperparámetros además de eliminar el preentrenamiento de la siguiente oración de BERT, lo que hace que el modelo se pueda entrenar con mini-batches y más grandes, y valores de *learning rate* más altos. Le sigue el modelo de BETO base, el modelo de BERT entrenado con textos en español. Aunque de entre todos los modelos entrenados no hay un salto de calidad muy grande, se puede comprobar que la diferencia más grande de accuracy entre modelos es de un 17 % y si comparamos el recall, la diferencia es de un 8 %.

El mejor resultado obtenido por los participantes de CANTEMIST ha sido el de (Hassan, Sánchez, y Domingo-Ferrer, 2020) mediante el entrenamiento de una red neuronal LSTM bidireccional y una CNN junto con una capa Max-pooling 2D, que sirve para reducir la dimensionalidad de los datos de entrenamiento. Otros participantes como Vicomtech (García-Pablos, Pérez, y Cuadros, 2020) o ICB-UMA (López-García et al., 2020) han obtenido puntuaciones equivalentes para la métrica principal (MAP), 0,847. El equipo de Vicomtech, ha desarrollado un sistema con precisión equilibrada (0,875), recall (0,836) y valor f1 (0,855), es el equipo que mejor rendimiento ha conseguido con su propuesta de un modelo conjunto compuesto por dos modelos base con unos hiperparámetros conmunes: la tasa de aprendizaje base de 2e-5 con un *linear warm-up scheduling* que alcanza su máximo durante las primeras 5.000 iteraciones, un *batch size* de 6, un valor de *dropout* de 0.1 y 200 *epochs*, con una paciencia de 50 *epochs*. Por otro lado, el equipo de ICB-UMA ha maximizado la métrica MAP, ya que su sistema tiene una precisión realmente baja (0.007) y una recuperación realmente alta (0.928).

Sin embargo, el problema principal que nos hemos encontrado tanto nosotros como los participantes de CANTEMIST ha sido el sesgo producido por el valor de la clase mayoritaria, es decir, el código que aparece en la mayoría de los casos clínicos (8000/6, código que hace referencia a la metástatis, aparece en 1.000 de los 3.000 documentos etiquetados que proporciona CANTEMIST, lo que es un claro indicativo de un desequilibrio). Los códigos

que solo aparecen en el conjunto de test, y no en el de entrenamiento, no han sido clasificados correctamente, tal y como era de esperar puesto que el modelo no ha podido generalizar a partir de datos que no ha visto nunca. En la figura 2.5 se observa que la mayoría de los códigos de CANTEMIST no son comunes. Para solventar este desequilibrio, además de buscar nuevos datos de entrenamiento para las clases minoritarias, existen técnicas que permiten crear observaciones sintéticas de las clases minoritarias en función de los datos disponibles. Algunos algoritmos que permiten esto incluyen codificadores automáticos variacionales (VAE), SMOTE (técnica de sobremuestreo de minorías sintéticas) o MSMOTE (técnica de sobremuestreo de minorías sintéticas modificadas).


Las técnicas de aumento de datos se utilizan para generar datos sintéticos adicionales utilizando los datos que tienen. Los métodos de aumento son muy populares en las aplicaciones de visión por computadora, pero son igual de poderosos para PLN. Algunos ejemplos de aumento de datos para PLN se describen a continuación:

Uno de los métodos más utilizados es el reemplazo de sinónimos en el que se elige aleatoriamente n número de palabras de la oración, que no sean *stopwords*, y estas palabras se reemplazan con uno de sus sinónimos elegidos al azar. Por ejemplo, tenemos esta oración:

> Este **artículo** se centrará en resumir las **técnicas** de aumento de datos en PLN.

El método selecciona aleatoriamente n número de palabras palabras, en este ejemplo seleccionamos dos, las palabras “artículo” y “técnicas”, y las reemplaza con ensayo y métodos, respectivamente.

Este **ensayo** se centrará en resumir los **métodos** de aumento de datos en PLN.

También se puede recurrir a la eliminación de palabras aleatorias:

Este **artículo** se centrará en resumir el aumento de datos en PLN.

La inserción de palabras también es un buen recurso. Sin embargo, hay que ser cuidadoso ya que tienen que ser palabras que encajen en el ámbito en el que nos encontremos:

Este **artículo** se centrará en resumir las **técnicas** de aumento de datos en PLN.

El método selecciona aleatoriamente n número de palabras. En este caso, volvemos a escoger las palabras "artículo" y "técnicas" y las intercambiamos por sinónimos como redacción y métodos, respectivamente. Por último, estos sinónimos se insertan en una posición aleatoria en la oración.

Este **artículo** se centrará en la **redacción** que resume las técnicas de aumento de datos en los **métodos** de PLN.

El objetivo de estos métodos es conseguir más observaciones por cada código, o clase, y así tener un conjunto de clases más equilibrado.

# Capítulo 6

# Conclusiones y trabajo futuro

Este capítulo recopila las diferentes conclusiones extraídas del trabajo realizado. Asimismo, se indican algunas líneas de trabajo futuras a considerar con el fin de mejorar la solución propuesta.

## 6.1. Conclusiones

En este trabajo se ha presentado un sistema de clasificación de códigos eCIE-O-3.1 mediante el uso de Transformers. En concreto, el Transformer BERT ha demostrado ser un gran avance en el campo del procesamiento del lenguaje natural y la comprensión del lenguaje y esto se debe a que BERT permite crear modelos de alto rendimiento con un esfuerzo mínimo en una variedad de tareas de PNL, entre ellas tareas de clasificación multi-label como la que hemos estudiado en este trabajo.

Si con pocos datos disponibles se pueden conseguir resultados bastante decentes como se demuestra en los resultados obtenidos por los participantes de CANTEMIST y los resultados del presente trabajo, la aplicación de este tipo de arquitecturas alimentadas con una gran cantidad de datos puede arrojar resultados excelentes que pueden ser de ayuda en el campo de la medicina, en este caso, pero también en muchos otros campos y ámbitos.

Sin embargo, el uso de técnicas de estado del arte no implica que se vayan a evitar los habituales problemas del ML y DL como puede ser el desbalanceo de las clases. El corpus de CANTEMIST, que se ha utilizado en el presente trabajo, cuenta con 3.000 casos clínicos en español anotados por expertos. El número de clases del corpus hace que sea muy probable que haya algunas clases que aparezcan más que otras en el conjunto de

datos. (Chapman y Neumann, 2020) observaron que ciertos códigos solo se encontraban en el conjunto de entrenamiento y otros solo en el conjunto de test. Además de haber clases con solo 1 o 2 instancias con las que el modelo no es capaz de generalizar y, por tanto, de aprender. Ciertos códigos aparecen en la gran mayoría de los datos que se han utilizado para entrenar nuestro modelo y son los que el modelo va a inferir con más frecuencia en futuras predicciones.

Aun así, tras el preprocesado y entrenamiento, los modelos se han comportado bestante bien y se han obtenido resultados que pueden mejorarse con el tiempo y la recopilación de datos de entrenamiento.

## 6.2. Trabajo futuro

Como propuesta para un trabajo futuro, se podrían intentar desarrollar modelos adicionales con otros tipos de arquitecturas transformer, como pueden modelos SciBERT (Iz Beltagy, 2019), XLM-RoBERTa (Alexis Conneau, 2019), BioBERT (Jinhyuk Lee, 2019) (entrenado con los corpus biomédicos específicos del dominio, incluidos PubMed y PubMed Central) o Biomedical X-Transformer, y utilizarlos para entrenar otros modelos X-Transformer que pueden ser entrenados con otros artículos científicos biomédicos en español. Estos modelos, al estar entrenados específicamente en español o haber usado un corpus científico muy parecido a los textos que provee CANTEMIST, pueden arrojar mejores resultados de clasificación.

Además de utilizar modelos transformers más especializados y técnicas de estado del arte para resolver esta tarea, se pueden buscar datos adicionales para alimentar al modelo y resolver el problema del desbalanceo de clases, posiblemente utilizando alguna técnica de creación de observaciones sintéticas o buscando formas de alimentar los datos de las distintas clases, o códigos. Para los códigos que tengan pocos datos, bastaría con utilizar algún algoritmo que pueda crear datos sintéticos.

Otra propuesta de trabajo futuro consiste en ampliar el modelo de clasificación de los códigos eCIE-O y, no solo hacer una clasificación de neoplasias, también se puede investigar formas y algoritmos para desarrollar clasificadores multi-etiqueta para cualquiera de los códigos que se muestran en la tabla 1.1.

# Bibliografía

## Bibliografía


[AAPC2021] AAPC. 2021. All about medical coding, Mayo. https://www.aapc.com/medical-coding/medical-coding.aspx.

[Alexis Conneau2019] Alexis Conneau, Kartikay Khandelwal, Naman Goyal Vishrav Chaudhary Guillaume Wenzek Francisco Guzmán Edouard Grave Myle Ott Luke Zettlemoyer Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *Facebook AI.*

[Bowman2008] Bowman, Sue. 2008. Why icd-10 is worth the trouble. *Journal of AHIMA 79, no.3.*

[Burges1998] Burges, Christopher J.C. 1998. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery volume 2, pages 121–167.*

[CANTEMIST2021] CANTEMIST. 2021. Cancer text mining shared task – tumor named entity recognition, Mayo. https://temu.bsc.es/cantemist/.

[Carla Smith y Dooling2019] Carla Smith, Sue Bowman y Julie A. Dooling. 2019. Measuring and benchmarking coding productivity: A decade of ahima leadership. *A Decade of AHIMA Leadership.*

[Chapman y Neumann2020] Chapman, Kathryn y Günter Neumann. 2020. Automatic icd code classification with label description attention mechanism. *Saarland University, Saarbrücken, Germany.*

[Chun et al.2014] Chun, Junyoung, Caglar Gulcehr, yungHyun Cho, y Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *Università de Montrèal CIFAR Senior Fellow.*

[Crammer et al.2007] Crammer, Koby, Mark Dredze, Kuzman Ganchev, Kuzman, y Partha Pratim Talukdar. 2007. Automatic code assignment to medical text. *Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA*.

[Devlin et al.2018] Devlin, Jacob, Ming-Wei Chang, Kenton Lee, y Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *Google AI Language*.

[Dey2016] Dey, Ayon. 2016. Machine learning algorithms: A review. *Department of CSE, Gautam Buddha University*.

[ECI2020] ECI. 2020. Documentación de los códigos ecie. https://eciemaps.mscbs.gob.es/ecieMaps/documentation/documentation.html.

[Farkas y Szarvas2008] Farkas, Richard y György Szarvas. 2008. Automatic construction of rule-based icd-9-cm coding systems. *The Second International Symposium on Languages in Biology and Medicine (LBM)*.

[García-Pablos, Pérez, y Cuadros2020] García-Pablos, Aitor, Naiara Pérez, y Montse Cuadros. 2020. Vicomtech at cantemist 2020. *SNLT group at Vicomtech Foundation, Basque Research and Technology Alliance (BRTA), Mikeletegi Pasealekua 57,*.

[Geoffrey E. Hinton2012] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky Ilya Sutskever Ruslan R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *Department of Computer Science, University of Toronto*.

[Hasan2016] Hasan, Mehedi, Alexander Kotov April Idalski Carcon Ming Dong Sylvie Naar Kathryn Brogan Hartlieb. 2016. A study of the effectiveness of machine learning methods for classification of clinical interview fragments into a large number of categories. *Journal of Biomedical Informatics Volume 62, August 2016, Pages 21-31*.

[Hassan, Sánchez, y Domingo-Ferrer2020] Hassan, Fadi, David Sánchez, y Josep Domingo-Ferrer. 2020. Tumor entity recognition and coding for spanish electronic health records. *CYBERCAT-Center for Cybersecurity Research of Catalonia*.

[Hochreiter y Schmidhuber1997] Hochreiter, Sp. y J. Schmidhuber. 1997. Long short-term memory. *Neural Computation (1997) 9 (8): 1735–1780.*

[Ilya Loshchilov2019] Ilya Loshchilov, Frank Hutter. 2019. Decoupled weight decay regularization. *University of Freiburg.*

[Information2021] Information, National Center Biotechnology. 2021. Medicine corpora. https://www.ncbi.nlm.nih.gov/research/bionlp/Data/.

[Iz Beltagy2019] Iz Beltagy, Kyle Lo, Arman Cohan. 2019. Scibert: A pretrained language model for scientific text. *Allen Institute for Artificial Intelligence, Seattle, WA, USA.*

[Jinhyuk Lee2019] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim Donghyeon Kim Sunkyu Kim Chan Ho So Jaewoo Kang. 2019. Biobert: a pretrained biomedical language representation model for biomedical text mining. *Department of Computer Science and Engineering, Korea University, Seoul 02841, Korea.*

[Johnson et al.2020] Johnson, A. E. W., T. J. Pollard, L. Shen, L. H. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, y R. G. Mark. 2020. Mimic-iii, a freely accessible critical care database.

[José Cañete y Fuentes2020] José Cañete, Gabriel Chaperon y Rodrigo Fuentes. 2020. Spanish pre-traiend bert model and evaluation data. *Department of Computer Science, Universidad de Chile.*

[Koby Crammer2003] Koby Crammer, Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *School of Computer Science Engineering.*

[Kotsiantis2007] Kotsiantis, S. B. 2007. Supervised machine learning: A review of classification techniques. *Department of Computer Science and Technology, University of Peloponnese, Greece.*

[Li Qiang2020] Li Qiang, Li Yao-kun, Xia Shu-yue Kang Yan. 2020. An improved medical text classification model: Ls-gru. *Journal of Northeastern University Natural Science, 2020, 41(7): 938-943.*

[Liliya Akhtyamova y Cardiff2019] Liliya Akhtyamova, Paloma Martínez, Karin Verspoor y John Cardiff. 2019. Bmc medical informatics and decision making testing contextualized word embeddings to improve ner in spanish clinical case narratives. *Technological University Dublin.*

[Liting Du2018] Liting Du, Chenxi Xia, Zhaohua Deng Gary Lu Shuxu Xia Jingdong Ma. 2018. A machine learning based approach to identify protected health information in chinese clinical text. *International Journal of Medical Informatics Volume 116, August 2018, Pages 24-32.*

[López-García et al.2020] López-García, Guillermo, José M. Jerez, Nuria Ribelles, Emilio Alba, y Francisco J. Veredas. 2020. Icb-uma at cantemist 2020: Automatic icd-o coding in spanish with bert. *Departamento de Lenguajes y Ciencias de la Computación.*

[López García2021] López García, Guillermo José M. Jerez, Nuria Ribelles Emilio Alba Francisco J. Veredas. 2021. Transformers for clinical coding in spanish. *IEEE Access ( Volume: 9).*

[M. Ikonomakis2005] M. Ikonomakis, S. Kotsiantis, V. Tampakas. 2005. Text classification using machine learning techniques. *Department of Mathematics University of Patras, GREECE.*

[Minaee2020] Minaee, Shervin; KalchbrennerNal; Erik Cambria; Narjes Nikzad; Meysam Chenaghlu; Jianfeng Gao. 2020. Deep learning based text classification: A comprehensive review.

[Ministerio de Sanidad2020] Ministerio de Sanidad, Servicios Sociales e Igualdad, Unidad Técnica de Codificación CIE-10-ES. 2020. Manual de codificación cie-10-es diagnósticos. https://www.mscbs.gob.es/estadEstudios/estadisticas/normalizacion/CIE10/UT_MANUAL_DIAG_2016_prov1.pdf.

[Miranda-Escalada et al.2020] Miranda-Escalada, Antonio, Aitor Gonzalez-Agirre, Jordi Armengol-Estapé, y Martin Krallinger. 2020. Overview of automatic clinical codingannotations, guidelines, and solutions for non-english clinical cases at codiesp track ofclef ehealth 2020. *Barcelona Supercomputing Center, Spain.*

[Nitish Srivastava2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky Ilya Sutskever Ruslan Salakhutdinov. 2014. Dropout: A simple way

to prevent neural networks from overfitting. *Department of Computer Science, University of Toronto.*

[Oleynik2019] Oleynik, Michel, Amila Kugic Zdenko Kasáč Markus Kreuzthaler. 2019. Evaluating shallow and deep learning strategies for the 2018 n2c2 shared task on clinical text classification. *Journal of the American Medical Informatics Association, Volume 26, Issue 11, November 2019, Pages 1247–1254.*

[Organization2021] Organization, World Health. 2021. International statistical classification of diseases and related health problems (icd), Mayo. https://www.who.int/standards/classifications/classification-of-diseases.

[Pilar López Úbeda2021] Pilar López Úbeda, Manuel Carlos Díaz Galiano, Teodor Martín Noguerol Antonio Luna-Alfonso Ureña López M. Teresa Martín Valdivia. 2021. Automatic medical protocol classification using machine learning approaches. *Computer Methods and Programs in Biomedicine Volume 200, March 2021, 105939.*

[Raghavendra Pappagari2019] Raghavendra Pappagari, Piotr Zelasko, Jesús Villalba Yishay Carmiel Najim Dehak. 2019. Hierarchical transformers for long document classification. *Automatic Speech Recognition and Understanding Workshop.*

[Ribelles et al.] Ribelles, N., J. M. Jerez, D. Urda, J. L. Subirats, A. Márquez, C. Quero, E. Torres, L. Franco, y E. Alba. Galeón: Sistema de información para la gestión y coordinación de procesos.

[Ruas et al.2020] Ruas, Pedro, Andre Neves, Vitor D. T. Andrade, y Francisco M. Couto. 2020. Lasigebiotm at cantemist: Named entity recognition and normalization of tumour morphology entities and clinical coding of spanish health-related documents. *Faculdade de Ciencias da Universidade de Lisboa, Portugal.*

[Rumelhart1985] Rumelhart, David E, Geoffrey E Hinton Ronald J Williams. 1985. Neural networks and physical systems with emergent collective computational abilities. *Institute for Cognitive Science University of California, San Diego.*

[S. Gao y Ramanathan2019] S. Gao, J. X. Qiu, M. Alawad J. D. Hinkle N. Schaefferkoetter-H.-J. Yoon B. Christian P. A. Fearn L. Penberthy X.-C. Wu L. Coyle G. Tourassi y A. Ramanathan. 2019. Classifying cancer pathology reports with hierarchical self-attention networks. *Artificial Intelligence in Medicine, vol. 101, p. 101726, 2019.*

[Salazar et al.2013] Salazar, Juan Antonio Goicoechea, María Adoración Nieto García, Antonio Laguna Téllez, Canto. Vicente David Casasola, Juliana RodrÃguez Herrera, y Francisco Murillo. Cabezas. 2013. Desarrollo de un sistema de codificación automática para recuperar y analizar textos diagnósticosde los registros de servicios de urgencias hospitalarios. *Departamento de Medicina Preventiva y Salud Pública, Facultad de Medicina de la Universidad de Sevilla, España.*

[SciELO2020] SciELO. 2020. Scientific electronic library online. www.scielo.org.

[Shachak Aviv2009] Shachak Aviv, Reis Shmuel. 2009. The impact of electronic medical records on patient–doctor communication during consultation: a narrative literature review. *Journal of Evaluation in Clinical Practice.*

[Shang Gao2020] Shang Gao, Mohammed Alawad1, M. Todd Young John Gounley Noah Schaefferkoetter Hong Jun Yoon Xiao-Cheng Wu Eric B. Durbin Jennifer Doherty Antoinette Stroup Linda Coyle Georgia Tourassi. 2020. Limitations of transformers on clinical text classification. *IEEE Journal of Biomedical and Health Informatics.*

[Waltl2018] Waltl, B., Bonczek G. Matthes F. 2018. Rule-based information extraction: Advantages, limitations and perspective. *Jusletter IT (02 2018).*

[Wang et al.2020] Wang, Ssu-Ming, Yu-Hsuan Chang, Lu-Cheng Kuo, Feipei Lai, Yun-Nung Chen, Fei-Yun Yu, Chih-Wei Chen, Chung-Wei Lee, y Yufang Chung. 2020. Using deep learning for automatic icd-10 classification from free text data. *Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan.*

[Wei2020] Wei, Fusheng, Han Qin Shi Ye Haozhen Zhao. 2020. Empirical study of deep learning for text classification in legal document review.

[Wei-Cheng et al.2020] Wei-Cheng, Chang, Kai Zhong, Yu Hsiang-Fu, Yiming Yang, y Inderjit Dhillon. 2020. Taming pretrained transformers for extreme multi-label text classification. *KDD*.

[Y. Gu y Poon2020] Y. Gu, R. Tinn, H. Cheng M. Lucas N. Usuyama X. Liu T. Naumann J. Gao y H. Poon. 2020. Domain-specific language model pretraining for biomedical natural language processing. *arXiv preprint arXiv:2007.15779, 2020*.

[Yinhan Liu2019] Yinhan Liu, Myle Ott, Naman Goyal Jingfei Du Mandar Joshi Danqi Chen Omer Levy Mike Lewis Luke Zettlemoyer Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *University of Washington, Seattle, WA*.

# Apéndice A

# Publicaciones

Publicaciones derivadas del trabajo realizado.

## Todo list