Trabajo de Final de Grado

Grado en Ingeniería Informática

Mención en Computación

Codificación automática de entidades médicas en informes clínicos

Autor: **Marco Terral Rodríguez**

Director: **Lluis Padro Cirera**

CoDirector: **Jordi Turmo Borrás**

30 Abril 2021

2020 - 2021 Q1/Q2

Abstract

The objective of this final degree project is to research the problem of medical entity normalization, on the ICD-10 standard, in primary care clinical reports written in Catalan and Spanish. To our knowledge, this is the first study on this problem applied to this typology of clinical documents. The proposed approach is divided into two steps, the first inspired in Information Retrieval techniques for initial filtering, followed by a second based on Google's new neural network for modeling languages, BERT, for the final assignment.

Resumen

Este trabajo de final de grado tiene como objetivo realizar una investigación sobre la codificación de entidades médicas, mediante el estándar CIE-10, en informes clínicos de atención primaria elaborados en catalán y castellano. Se trata del primer estudio sobre este problema, aplicado a dicha tipología de documentos clínicos, del que tengamos conocimiento. El acercamiento planteado se realiza en dos etapas, la primera inspirada en técnicas de *Information Retrieval* para un filtrado inicial de posibles entidades, seguido de una segunda basada en la nueva red neuronal de modelado de idiomas de Google, BERT, para la asignación final.

Resum

Aquest treball de final de grau té com a objectiu realitzar una recerca sobre la codificació d'entitats mèdiques, mitjançant l'estàndard CIM-10, en informes clínics d'atenció primària elaborats en català i castellà. Es tracta del primer estudi sobre aquest problema, aplicat a aquesta tipologia de documents clínics, del qual tinguem coneixement. L'apropament plantejat es realitza en dues etapes, la primera inspirada en tècniques de *Information Retrieval* per a un filtratge inicial de possibles entitats, seguit d'una segona basada en la nova xarxa neuronal de modelatge d'idiomes de Google, BERT, per a l'assignació final.

Índice

Índice de figuras

Índice de cuadros

1. Introducción y contextualización

Este trabajo se realiza dentro del marco A de la UPC, supeditado al proyecto nacional TADIA-MED (PID2019-106942RB-C33) cuyos IPs son Jordi Turmo y Lluis Padró, profesores del departamento de Ciencias de la Computación e integrantes de los centros de investigación TALP e IDEAI de la UPC.

Con el paso de los años y las mejoras sustanciales de los sistemas médicos, se incrementan el número de documentos clínicos generados. Con la llegada de nuevas tecnologías muchos, si no todos, se pasan a formatos electrónicos si no son ya redactados de esta manera [1, 2]. Considerando también la llegada de nuevos estándares más complejos para la codificación de entidades médicas y su adopción por parte de diversos gobiernos [3], se hace más acuciante la necesidad de clasificar estos documentos con ellos de forma eficiente y rápida. Además de todo lo anterior la accesibilidad de estos datos se encuentra a la orden del día, tanto para el paciente, en consonancia con las leyes sobre el acceso a los datos médicos, como para los profesionales, bajo la necesidad de acceso a mayor información para estudios más ambiciosos. Requiriendo enriquecer estos documentos con mayor información, para facilitar dichas aplicaciones.

1.1. Conceptos

Para poder entender el problema que nos planteamos resolver, y como queremos hacerlo, es pertinente entender las partes que lo componen, por ello dedicaremos este apartado a explicar los conceptos más importantes.

1.1.1. Documentos clínicos

Por documentos clínicos entendemos cualquier clase de documento que se genera en la atención sanitaria al paciente y los derivados de gestionar dicha atención. Como podemos ver esto engloba una cantidad importante de información. Por ello es necesario diferenciar diversos tipos de documentos, pues no solo contienen diversas terminologías e informaciones sino que la forma de redacción, e incluso creación, varían sustancialmente. Como podemos intuir las anotaciones realizadas por un médico en medio de un diagnóstico no son lo mismo que una alta hospitalaria.

De toda la variedad documental existente, son los informes clínicos, y más concretamente los informes de atención primaria, los que más nos interesan pues son la base de nuestro proyecto. No sólo debido a la complejidad que representan para nuestro problema sino que son los documentos a los cuales tenemos acceso gracias al SIDIAP(Sistema d'informació per al Desenvolupament de Investigació en Atenció Primària), entidad que nos aporta los datos con los que trabajar. Con informes clínicos nos referimos a los diversos documentos que se generan durante la atención sanitaria a un paciente, elaborados por diversos personales sanitarios. Nosotros nos centraremos en los generados durante la atención primaria dada a un paciente, pues son los disponibles.

Estos informes tienen el siguiente conjunto de características que hacen su análisis más difícil, de ahí la complejidad que suman a nuestro problema [1, 4]:

- **Acrónimos**
- **Jergas**
- **Abreviaciones**
- **Omisiones de palabras**
- **Sinónimos**
- **Errores ortográficos**
- **Uso de diversas acepciones ortográficas**

1.1.2. Entidades médicas

Entendemos por entidades médicas cualquier término o conjunto de términos que hacen referencia a un concepto biosanitario. No hablamos solo de enfermedades, síntomas o medicamentos sino también de procedimientos, situaciones sociales e incluso regiones anatómicas; en definitiva, cualquier concepto susceptible de ser considerado para una estandarización de la documentación clínica a nivel internacional. En general, cada estándar propuesto suele partir de la misma base, aunque puede diferir en ciertos aspectos. Es por ello que cuando hablemos de entidades médicas será en el contexto del estándar usado.

1.1.3. Estándares de codificación de entidades médicas

Con estándares de codificación nos referimos a sistemas elaborados por distintas entidades con el objetivo de proporcionar una clasificación estándar de conceptos biosanitarios, principalmente suelen ser estructuras en forma de esquemas conceptuales u ontologías y con codificaciones numéricas únicas que identifican cada uno de esos conceptos.

Existen múltiples estándares, creados por diversas organizaciones con objetivos distintos. Un ejemplo de esto es el ICD (*International Statistical Classification of Diseases and Related Health Problems*) desarrollado por la OMS (Organización Mundial de la Salud) con la intención de definir un estándar en las documentaciones clínicas de todo el mundo, para facilitar la identificación e investigación de enfermedades y tendencias de salud a nivel global [5]. Este estándar consta de diversas ediciones y variaciones a lo largo de los años, pero nos centraremos en el CIE-10 y CIM-10, que son la modificación española y catalana del ICD de décima edición.

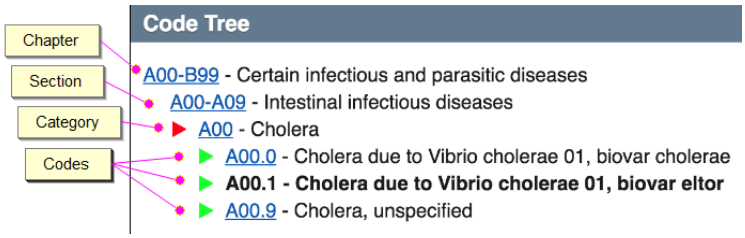


Figura 1: Estructura del ICD-10

Fuente:
https://www.unboundmedicine.com/icd/view/ICD-10-CM/860000/all/About_ICD_10_CM_Coding_Gui

Otro de estos estándares a tener en cuenta es Snomed-CT (*Systematized Nomenclature of Medicine – Clinical Terms*) creado por la IHTSDO (*International Health Terminology Standards Development Organisation*), a la cual está adscrita España entre otros países. Cuyo objetivo fundamental es crear un estándar médico global, se trata de uno de los estándares más amplios y precisos con enfoque multilingüe[6].

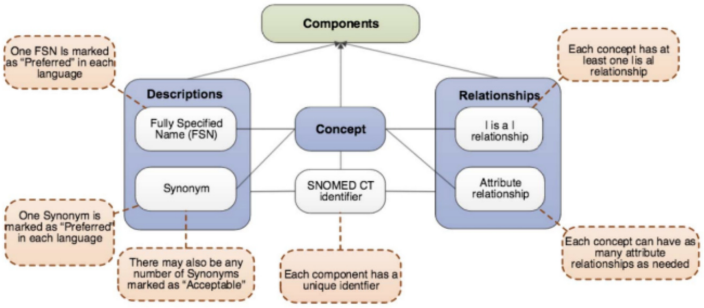


Figura 2: Estructura de Snomed-CT

Fuente: http://www.snomed.org/snomed-ct/five-step-briefing

Como podemos ver en la figura 1, pese a partir de un mismo objetivo estos dos estándares difieren bastante en funcionamiento. En ICD los propios códigos representan un árbol conceptual donde cada nivel de profundidad corresponde a más especificidad y un código concreto hace referencia a una enfermedad mediante un término correcto de la literatura científica (denominado descriptor) [7]. Mientras que en la figura 2 vemos que Snomed-CT tiene en cuenta las diversas formas de hacer referencia a un término (sinónimos) y basa su estructura en una ontología, donde se definen relaciones mucho más complejo aparte de la jerárquica [6]. Cabe destacar que las identificaciones numéricas de Snomed-CT no representan significado alguno por si mismos, a diferencia de los de ICD-10. Cada dígito representa una posición dentro de la jerarquía conceptual, determinando el tipo de enfermedad y las circunstancias que la rodean.

1.1.4. Anotación de documentos clínicos

La anotación de documentos clínicos es una tarea de enriquecimiento de estos, especialmente informes. Cualquier centro que generé documentos médicos debe por ley anotarlos[3], tarea que

se realiza por los denominados anotadores médicos. A día de hoy, se realiza por ordenador con ayuda de programas, pero sigue siendo una tarea principalmente manual.

Siendo más concretos, consiste en identificar, mediante un estándar, las entidades correspondientes así como categorizar el documento según su diagnóstico, si existe uno. Muchas veces se realiza mediante diversos estándares para cubrir los distintos tipos de entidades, un ejemplo de esto puede ser el uso conjunto de CIE-10 y Snomed-CT para codificar enfermedades y regiones anatómicas respectivamente.

Esta actividad se encuentra con diversos tipos de problemas durante su desarrollo, ya que el anotador requiere de un conocimiento especifico que permita identificar cualquier mención codificable y asignarle su identificador con suficiente nivel de especificidad. Este último resulta ser uno de los problemas más acuciantes, pues mucha veces se hallan identificadores más genéricos de los que se podrían asignar. No es raro encontrar una mención de carcinoma de pulmón inferior derecho identificada simplemente como cáncer de pulmón.

1.2. El problema

Actualmente, en los CAPs (Centros de Atención Primaria) de toda Cataluña se generan a diario cientos de informes clínicos. Con la llegada de las nuevas tecnologías, las formas de redactar y archivar dichos informes han cambiando radicalmente. Permitiendo generar una base de datos extensa sobre el diagnóstico de pacientes, siguiendo con una tendencia global del uso de los registros médicos electrónicos [8, 9]. Hablamos de una gran cantidad de información hecha mucho más accesible, facilitando la explotación del potencial de dichos datos.

Por ello existen muchas iniciativas e investigaciones que tienen como objetivo facilitar dicha explotación, entre ellas las centradas en la extracción de información clínica [10, 4]. La extracción de información clínica tiene el objetivo de analizar documentos clínicos y extraer información de estos. Podemos definir dos subtareas mínimas para poder cumplirlo: identificar segmentos de texto que consideremos relevantes (principalmente términos que hagan referencia a entidades médicas) y asignar a cada uno de esos fragmentos de texto una entidad dentro de un estándar de codificación. Estos dos pasos, conocidos respectivamente en el campo del Procesamiento del Lenguaje Natural (PLN o NLP) cómo NER(*Named Entity Recognition*) y NEN(*Named Entity Normalization*), permiten extraer la información mínima que en otros pasos se refinará en función del objetivo que se persiga. Como puede ser la deducción de distintas relaciones entre medicamentos y sintomatologías.

Dentro de este TFG, el problema que intentamos resolver es NEN aplicado a informes clínicos: "Dado un término, o conjunto de términos, debemos asignarle una entidad médica perteneciente a una ontología o estándar de codificación". Como parte de un estudio preliminar a un proyecto mayor, para la elaboración de un sistema de extracción de información en informes clínicos. Un ejemplo de esta tarea se encuentra en la anotación de documentos clínicos, donde una vez seleccionados los fragmentos que hablan de enfermedades el anotador debe asignarles sus respectivos códigos. Que supone una gran diferencia respecto a la codificación de este mismo documento, donde se debe asignar un único identificador de enfermedad a este. Normalmente se trata del diagnóstico y servirá para categorizar dicho informe. Como podemos observar se tratan de dos tareas similares, pero distintas en objetivos y acercamientos, por ello debemos recalcar que nuestro trabajo se centra en codificar los fragmentos de texto referentes a

enfermedades y no todo el documento.

Más concretamente, estamos resolviendo este problema como parte de un estudio para encontrar un método que resuelva este problema aplicado a los documentos dados por el SIDIAP, y cuyas menciones a enfermedades ya han sido destacadas previamente. Encontrado así un posible método que resuelva el paso de NEN para el proyecto base, del que forma parte este TFG. Debemos considerar que estamos tratando informes con unas propiedades muy concretas pues, además de lo ya comentado, deberemos lidiar con dos idiomas, como son el catalán y el castellano, y las nuevas clases de características que derivan de este bilingüismo particular:

- **Expresiones del otro idioma**
- **Vocabulario alternado**
- **Normas ortográficas del otro idioma**
- **Neologismos nacidos del bilingüismo**

1.3. Actores implicados

Como hemos visto, este proyecto se encuentra supeditado a uno mayor, siendo sus investigadores principales beneficiarios así como sus patrocinadores. Podemos considerar también a las entidades que facilitan este proyecto base como patrocinadores indirectos del nuestro.

Ciertamente este problema no solo existe dentro de los sistemas de extracción de información, sino que tiene otras aplicaciones. Como el enriquecimiento de documentos clínicos para facilitar la gestión, análisis estadísticos, investigaciones(pues facilita la selección de grupos de un determinado perfil) o un mayor entendimiento por parte los pacientes de su historial médico.

Bajo estas premisas los beneficiarios no serían solo los investigadores en el campo de la EIDC (Extracción de Información de Documentos Clínicos) por un estudio sobre rendimiento de una técnica para resolver NEN en una tipología de datos poco estudiados. Sino también investigadores y profesionales del campo de la medicina, pues facilitaría su trabajo. También hemos de mentar a los pacientes que verán facilitada la lectura de su historial clínico y otros documentos a los que tienen acceso. Como últimos beneficiarios podemos hablar de hospitales y organizaciones sanitarias, pues la anotación de documentos clínicos es una actividad obligatoria y muy costosa que aún se realiza a mano, cualquier porcentaje de automatización puede generar enormes ahorros [1].

2. Alcance del proyecto

Para definir el alcance que pretendemos obtener con este proyecto debemos primero definirlo bien, así como las circunstancias que lo rodean. Una vez establecido esto en el capítulo anterior podemos pasar a determinar su alcance en forma de objetivos y subobjetivos, así como los posibles riesgos y obstáculos que puedan surgir durante su desarrollo.

2.1. Objetivos genéricos y subobjetivos

Como sabemos el objetivo principal de este proyecto es obtener un método para la asignación de una entidad médica a un fragmento de texto. Este método debe cumplir el siguiente conjunto de características:

- **Exactitud**: Obviamente lo primero que buscamos es que el concepto asignado sea correcto.
- **Especificidad**: Ya que el método tiene que ser exacto, y teniendo en cuenta la estructura de CIE-10, sería relativamente trivial elegir un código muy general (como enfermedad infecciosa) por lo que queremos un nivel de especificación lo suficientemente alto.
- **Independencia del lenguaje**: Uno de los factores a considerar es que tratamos informes tanto en castellano como en catalán, por la naturaleza del conjunto de datos usados. Por lo que necesitamos que se cumplan los requisitos anteriores independientemente del idioma.

2.1.1. Subobjetivos

Una vez hemos definido nuestros objetivos debemos considerar que subobjetivos necesitamos cumplir, obteniendo los siguientes:

- **Estudio preliminar**: La primera subtarea es hacer un estudio exhaustivo sobre todas las investigaciones al respecto, para no solo obtener un buen estado del arte sino un buen conjunto de todas las posibles soluciones existentes. Así como los puntos flacos y distintas formas de suplirlos.
- **Establecer métricas**: Pese a que tengamos claro las características del método que buscamos debemos ser capaces de evaluarlo de forma objetiva, para ello definiremos un conjunto de métricas para estimar la eficacia del método usado así como de las distintas mejoras que aplicaremos.
- **Elección y aplicación de un método y sus mejoras**: Una vez hecho un estudio, debemos elegir que método implementar. Así como decidir el conjunto de mejoras que vamos a tener que aplicar a este para adaptarlo a nuestras circunstancias particulares.
- **Estudio y comparación de las mejoras del método**: Una vez elaborado el método y sus mejoras debemos comparar su rendimiento de acuerdo con nuestros criterios y métricas, y así poder determinar cuáles son los más adecuadas para nuestro proyecto, eligiendo así las mejoras que más nos benefician.

2.2. Obstáculos

Un obstáculo importante es la necesidad de adaptar la mayoría de las implementaciones de los artículos, como veremos en las siguientes secciones la mayoría de investigaciones no tienen las mismas circunstancias que nuestro problema. Por ello algunos de estos métodos, si no la mayoría, pueden necesitar una revisión y adaptación, sumando además las mejoras que seguramente debamos elaborar. Si quisiéramos usar el método del artículo *Automatic Generation of a Qualified Medical Knowledge Graph and its Usage for Retrieving Patient Cohorts from Electronic Medical Records* [9] deberíamos modificar su acercamiento para adaptar esas tres clases genéricas finales a nuestro CIE-10 con cientos de ellas.

2.3. Riesgos

El principal riesgo es el no poder desarrollar todas las mejoras a nuestro método, por cuestiones temporales, y que no acabemos con un método óptimo para nuestro problema. Pese a que los artículos pueden ser muy ilustrativos ninguno explica su implementación exacta, y eso dando por supuesto que no necesitamos adaptar las soluciones para nuestro problema. Lo anterior sumado a que deberemos implementar mejoras no especificadas puede aumentar este riesgo. Tampoco hay que olvidar que uno de los métodos que hayamos descartado sea el más efectivo, lo cual incrementa aun más este riesgo.

3. Estado del arte

Como hemos mentado anteriormente, las técnicas de NLP tienen cabida en muchas facetas y aplicaciones de la medicina moderna. Incluso las más sencillas, como el enriquecer documentos clínicos de forma automática, pueden suponer una mejoría enorme no solo en aspectos económicos sino sociales.

Es por ello que existen proyectos y organizaciones que tienen como objetivo el desarrollo y avance de las técnicas de NLP en medicina y biomedicina, mediante la creación de retos y conjunto de datos. Un ejemplo de lo anterior es la *CLEF eHealth Lab Series*, la cual organiza anualmente competiciones proponiendo dos tareas relacionadas con EIDC dentro de estas áreas [8]. Entre la enorme variedad de estas actividades podemos hallar algunas creadas con el propósito de desarrollar la investigación en torno a NEN y/o automatización de la anotación de documentos médicos.

También encontramos productos comerciales que asisten en estas tareas de NER y NEN, tanto en el análisis de documentos clínicos[11] como en cualquier ámbito más general. Aunque es cierto que se encuentran más enfocados a ser herramientas de asistencia, que procesos totalmente automatizados.

Existen múltiples ejemplos de tareas asociables a nuestro problema: como en CLEF 2018 y 2020, que consiste en asociar el diagnóstico, usando el estándar ICD-10 y CIE-10 respectivamente, a un documento [8, 12, 13, 1]. O en CodiEsp [14], en el que dado un conjunto de documentos clínicos debemos asignarles todos los códigos de CIE-10 relevantes. Otra clase de actividad es Cantemist [15] dónde se propone dada una lista de menciones a entidades, pertenecientes a informes clínicos en Español, se deben relacionar cada una con un código dentro de CIE-O-3, una variante de CIE especializada en Oncología.

La razón por la que mencionamos estas variaciones, junto a aplicaciones más cercanas, es que suponen una fuente de información interesante[1]. Como veremos, muchos de los artículos que resuelven nuestro problema lo hacen en otros idiomas y tipologías de documentos clínicos. Necesitamos, por lo tanto, investigaciones que lidien con datos como los nuestros y hayan de prepararlos para aplicaciones, si no iguales, similares. Cabe destacar que algunas de las soluciones pasan primero por resolver NEN sobre esos documentos, ofreciéndonos implementaciones aplicables.

La mayoría de investigaciones punteras, que se realizan respecto a NEN, tienden a usar un conjunto de métodos mixtos. Se emplean estructuras de conocimientos[2] y técnicas de aprendizaje autónomo, incluyendo algunas reglas basadas en conocimiento experto. Principalmente, veremos cuatro acercamientos a la hora de resolver este problema:

- **Una clasificación multiclase**[9, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26];
- **Una traducción de un lenguaje informal a uno formal**[27, 28]

[1] Como pueden ser los métodos de preprocesamientos de los términos.
[2] Como ontologías, diccionarios y bases de conocimiento.

- **Un sistema de IR "local"**[29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39]
- **Un sistema de IR "colectivo"**[40, 41, 42, 43, 44, 45, 46]

A continuación discutiremos cada uno de ellos en mayor profundidad, así como sus ventajas y desventajas.

3.1. Clasificación multiclase

El primer método que queremos discutir es el más intuitivo: dado una instancia de una clase averiguar de que clase es, una clasificación multiclase. Este método puede llegar a ser muy potente, los sistemas desarrollados bajo esta premisa pueden llegar a resultados de elevada *accuracy* con relativa facilidad. Cabe recordar que hablamos de uno de los problemas fundacionales del aprendizaje autónomo, dotándonos de una gran cantidad de herramientas que utilizar.

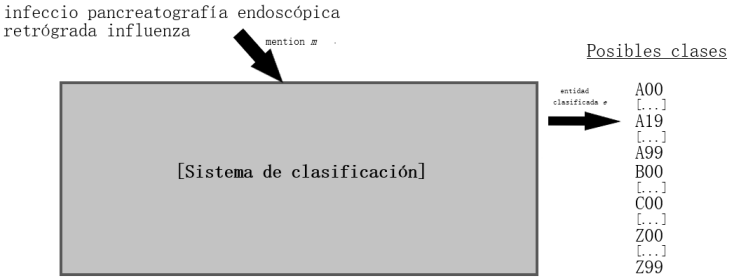


Figura 3: Esquema conceptual de la clasificación

Fuente: Propia

Este método presenta tres desventajas mayores. Primero, se requiere de un gran conjunto de datos y cuánto más balanceado mejor. Muy pocos artículos consideran el desbalance de clases, y si lo hacen es considerándolo para la clase *None* [23]. Para usar métodos de este tipo tendríamos que tener un número parecido de ejemplos para cada clase, lo que puede ser muy complejo de obtener si consideramos la naturaleza de nuestro conjunto de datos.

El rendimiento de muchos de estos sistemas se ven reducido si no se combina con NER [16, 17, 18, 19]. Dicho de otra manera, los mejores sistemas realizan la tarea de NER y NEN conjuntamente, dado un texto encuentran y clasifican menciones a una entidad. La idea detrás de esto último es la siguiente: lo que permite entender que un fragmento de texto realiza una mención a una entidad(como el contexto) es aprovechable para clasificar esa misma.

Por último, estos sistemas clasifican en un conjunto de clases muy reducido. Hablamos de entre 5 y 15 categorías, muy alejadas de las cientos disponibles en CIE-10. Por lo que para usar estos se requerirá de adaptar la solución [9, 16, 17].

3.2. Traducción de registro

El método de traducción de un registro informal a formal es un acercamiento novedoso, pues existen pocas investigaciones al respecto. Se basa en que las menciones de entidades en informes clínicos suelen ser de tono más informal que en literatura médica. Proponiendo que la tarea a realizar sea la traducción de estas a la definición formal. Una vez traducida esta mención es trivial asignarle el identificador correspondiente.

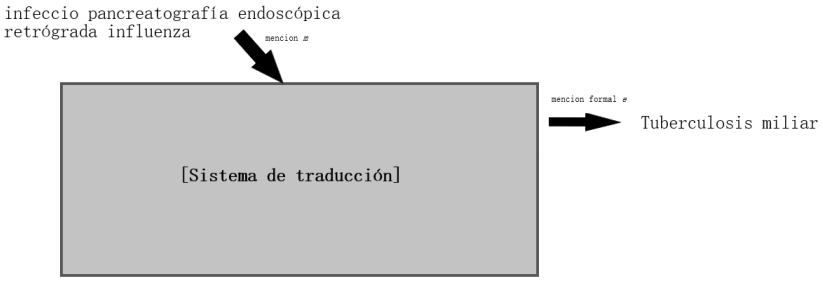


Figura 4: Esquema conceptual de la traducción

Fuente: Propia

Las ventajas no son muy claras, ya que el rendimiento de estos sistemas no es muy bueno. El único añadido que podemos considerar es que todas las investigaciones tenían como objetivo codificar empleando ICD-10 en lenguas latinas (Español, Francés e Italiano)[27, 28].

Pese a que sea una idea interesante, la poca información disponible dificulta la implementación de esta solución. Ya que las dos investigaciones que las realizan tienen acercamientos técnicos diametralmente opuestos, sobre los que apenas profundizan. Se requeriría de una cuantiosa inversión temporal en desarrollo con pocas garantías de éxito. Debemos recordar que los sistemas aún tienen que verse aplicados a otros conjuntos de datos más generalistas, y que, pese a usarse en datos especializados, ninguna de las investigaciones obtuvo resultados favorables. Otra desventaja, aunque menor, es el uso de estructuras de datos externas para complementar el sistema de traducción[3] que pueden resultar complejas de encontrar, principalmente en catalán.

3.3. *Information retrieval* local y global

Por último, hablaremos de los métodos que denominaremos de *Information Retrieval*(IR), ya que abordan el problema como uno. Dada una mención en un documento podríamos considerarla como una búsqueda dentro de la base de conocimiento que estructura las entidades. Por lo que debemos devolver la entidad más relevante para dicha búsqueda. El objetivo sería computar para cada entidad su relevancia y elegir la mayor (o bien las K mayores).

El principal problema radica en que el cálculo de la relevancia de una entidad a determinada mención no es trivial, y es computacionalmente costoso. Pero estos métodos aprovechan una

[3] En la resolución de abreviaciones.

realidad de las entidades: la gran mayoría son trivialmente descartables. Un ejemplo de esto sería como a una mención con referencias a enfermedades cardiacas le resulta completamente irrelevante las entidades referentes a enfermedades mentales.

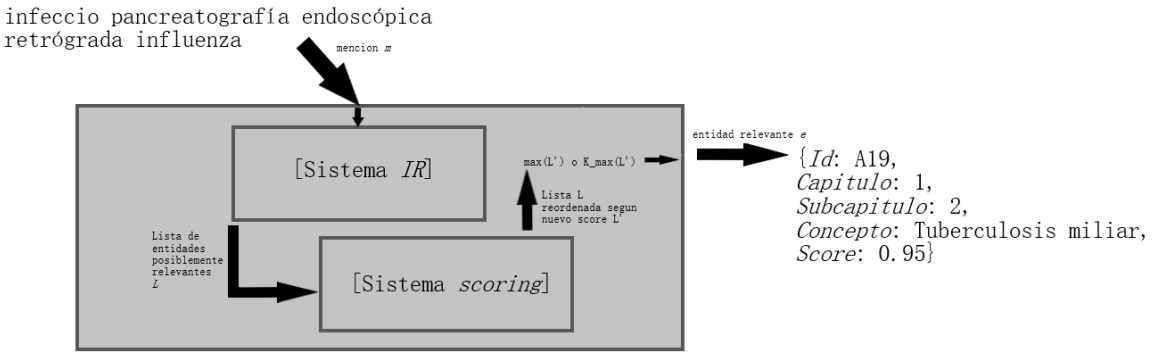


Figura 5: Esquema conceptual del sistema IR

Fuente: Propia

Por ello proponen elaborar un sistema a dos partes: una primera que recoja todas las entidades mínimamente relevantes mediante un sistema tradicional de *Information Retrieval*. Y un sistema de puntuaciones posterior mucho más costoso y preciso, pero con un número menor de entidades que procesar. Al final del cual se elegirá la mejor puntuada (o los de K mejores). Dónde difieren las dos tipologías de métodos, que disertaremos a continuación, es en el acercamiento a este segundo sistema.

3.3.1. Sistema de relevancia local

El primer acercamiento plantea calcular la relevancia de una entidad a una mención de forma individual, o bien por cada par (*mención, entidad*) [29, 30, 31, 33, 34, 37, 38, 39] o de forma colectiva entre todas las posibles entidades de una mención[32, 35, 37]. Este acercamiento se realiza mediante redes neuronales, desde distintas vertientes como son *Convolutional Neural Networks*(CNN) o *Bidirectional Encoder Representations from Transformers*(BERT).

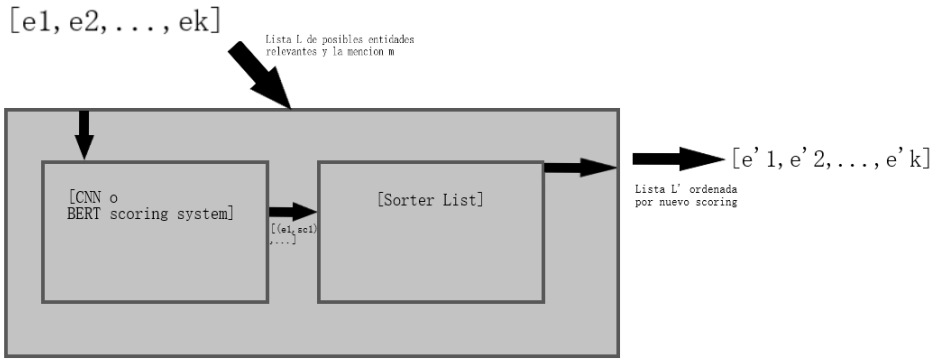


Figura 6: Esquema conceptual del *scoring system* local

Fuente: Propia

La ventaja principal de este método es la rapidez y facilidad de entreno que tienen estas soluciones. Permiten, además, el uso de datos desbalanceados y no se ven limitadas por la cantidad de clases disponibles. Una vez acabado el entreno, la ejecución es mucho más rápida con respecto a su otra variante.

Por otro lado, podemos encontrar ciertas desventajas. La primera y principal es la dependencia del rendimiento del sistema al funcionamiento del filtrado primario[4], también atribuible a la otra variante. Existen también dependencias a estructuras externas de conocimiento que presentan ciertas dificultades ya comentadas (sección 3.2).

3.3.2. Sistema de relevancia global

Por último debemos hablar sobre la variante del segundo sistema. En esta se calcula la relevancia de cada par (*mención, entidad*) basándose en una coherencia global del documento. Existen dos formas de abordarlas, pero ambas pasan por generar un grafo de relaciones entre las diversas entidades propuestas para las menciones. Posteriormente, este grafo es procesado para obtener la relevancia de cada entidad en función de su importancia en este, computado mediante un algoritmo de *PageRank* [40, 41, 42, 43, 45] o una red neuronal[44, 46].

[4]Si ningún concepto relevante es devuelto poco podrá hacer este.

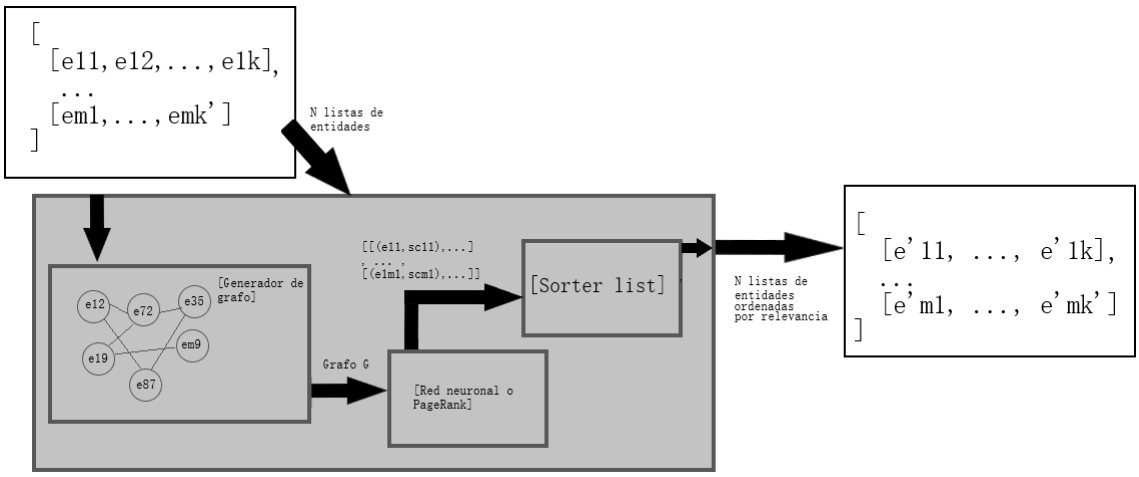


Figura 7: Esquema conceptual del *scoring system* global

Fuente: Propia

La principal ventaja que aporta este método es tener en cuenta el contexto de las entidades, lo cual es algo que ningún otro método considera.

Pero tiene dos desventajas considerables, aparte de las ya asociadas con el primer sistema. La primera es el bajo rendimiento en *accuracy* de estos sistemas, respecto al resto. Y el segundo, un tiempo de ejecución más largo que las alternativas. Ya que generar y procesar el grafo[5] tiene un coste nada deleznable, lo que limita sus aplicaciones.

3.4. Tipología de los conjuntos de datos

Una vez discutidos los métodos, debemos abordar cuáles son las clases de documentos e idiomas cubiertos por estas investigaciones. Así como los estándares en los que se encuentran anotados. Considerando los datos sobre los que se aplicaron las investigaciones podremos entender, en mayor medida, como responderán los métodos a la naturaleza de nuestros datos.

El alcance de muchas de las competiciones mencionadas anteriormente van más allá del período en las que se organizan. Muchos de los datos usados se mantienen accesibles para facilitar la reproducibilidad de los ganadores así como el desarrollo posterior de métodos. No es extrañar que dichas organizaciones tiendan a actualizar y mejorar estos conjuntos, llamados también *Corpus*, a lo largo de los años. Pues suponen una buena base para una investigación, dando un estándar común desde el que comparar resultados. Quitando, además, la dificultad de recolectar y preparar datos propios, los cuáles raramente podrían llegar al nivel de calidad de estos conjuntos. Por lo que no es extrañar que muchas de las investigaciones y soluciones de las que hemos hablado tiendan a usarlos.

La mayoría de estos conjunto de datos son elaborados sobre lengua Inglesa [47, 48, 49, 50,

[5] Sobretodo con *PageRank*.

51, 52, 53, 54, 55, 56], aunque también existen, en menor medida, de otras lenguas. Como la Francesa [57, 58, 12], Española[15, 27] o Húngara [12]. Pese a todos los puntos positivos de estos datos "públicos" existen también métodos que cubren otros idiomas mediante conjuntos de documentos privados como son el Chino[37], Danés[59] o Búlgaro[60].

Como podemos deducir, al ser la mayoría de *Corpus* en inglés se dota de cierto favoritismo a este, así como a una tipología de documentos y estándar de codificación. Gran parte de estos conjuntos están formados por *abstracts*, patentes y artículos de investigación de múltiples campos de especialización; los cuáles tienen un carácter más formal [48, 49, 50, 51, 54, 52, 55]. Es cierto que también existen, en menor medida, *Corpus* que contienen documentos más informales. Como Mimic [47], formado exclusivamente por informes clínicos de atención primaria. O el conjunto empleado en la tarea de Clef 2013 [56], que contiene toda clase de documentos relacionados con la atención sanitaria desde informes de diagnósticos hasta altas hospitalarias. Debemos destacar, también, la existencia de un conjunto llamado AMIA [53]. Este recopila textos libres de usuarios en redes sociales, donde se hace mención a entidades médicas de forma más implícita, una clase de texto totalmente informal y con una gran ausencia de terminología médica.

La mayoría de *corpus* ingleses, por no decir todos, se encuentran anotados en UMLS(Unified Medical Language System)[61]. Se trata una base de datos constituida de múltiples estándares y ontologías especializadas y generalistas[6], dotándole de gran profundidad y amplitud de conocimiento. Como contraparte, tenemos que la mayoría de datos no ingleses emplean sus propias bases de conocimiento para el mapeo de entidades, normalmente se tratan de variantes del estándar ICD-10 en ese idioma.

Con respecto a la lengua española encontramos dos conjuntos:

- ***Corpus* de Osakidetza:** Una serie documentos clínicos de emergencias dado por un servicio hospitalario Vasco[27].
- **Cantemist:** Informes clínicos dentro de la especialidad de Oncología[15].

Como vemos estos se acercan bastante a nuestros datos, pues tratan con documentos médicos de escritura más informal, pero aun así no aplican del todo. Lo primero, y más obvio, es que estos datos se encuentran en castellano. Por otra parte, el *corpus* de Osakidetza no es del todo público[7], lo cual dificulta comprobar la similitud de sus datos. Mientras que el conjunto de Cantemist se basa en informes de una rama médica muy especializada, y nada garantiza que los métodos que mejor rinden en él lo hagan en datos más generalistas. Respecto a los estándares de codificación usados, como mencionamos, se tratan de la variante española de ICD-10 y, más concretamente para Cantemist, de CIE-O-3, la variante oncológica española.

Por lo que podemos observar en este estado del arte, este el primer estudio (del que tengamos conocimiento) de técnicas de NEN sobre un conjunto de datos como el nuestro. Hablamos de informes clínicos de atención primaria en castellano y catalán anotados en CIE-10.

[6]Snomed-CT e ICD-10 entre otros muchos.

[7]No tenemos acceso al él debido a que requiere realizar trámites burocráticos relacionados con la protección de los datos de los pacientes.

4. Justificación de la alternativa escogida

Una vez estudiados los diversos acercamientos a nuestro problema, en el estado del arte, debemos plantearnos que alternativas escoger y porqué. Para ello discutiremos qué método emplearemos, las métricas usadas para medir sus resultados y el estándar de codificación usado. Pero antes de discutir detalles de implementación haremos un análisis inicial de los documentos dados por el SIDIAP. Cabe destacar que este manuscrito no ha sido preparado en colaboración con este registro y, por lo tanto, no refleja necesariamente sus opiniones ni puntos de vista. La calidad y exactitud de esta memoria es exclusivamente de nuestra responsabilidad.

4.1. Estudio de los datos

Antes de poder explicar que solución hemos escogido, y las razones para ello, debemos entender los datos con los que debe tratar nuestro problema. Es por ello que antes de optar por un método, conjunto de métricas y estándar de codificación realizaremos un análisis de los informes clínicos disponibles; para poder elegir con conocimiento de causa.

4.1.1. Descripción del conjunto

Actualmente, la base de archivos a la que tenemos acceso se encuentra formada por informes clínicos de atención primaria de toda clase de intervenciones. Desde diagnósticos clínicos a resúmenes de procedimientos, pasando por reportes de atención domiciliaria y equipos de apoyo a enfermos crónicos. Dichos documentos pueden estar redactados en castellano o catalán y se encuentran anotados en diversos estándares, cada uno cubriendo una serie de conceptos diferentes, de los cuáles nos interesan los que hagan referencia a enfermedades. De los estándares usados solo dos cubren esas entidades por diseño: Snomed-CT y CIE-10. De hecho, hallamos que el uso dado a Snomed-CT es como estándar para cubrir la anatomía corporal de las enfermedades anotadas por CIE-10. Por lo que, con respecto a nuestro proyecto, nos centraremos en CIE-10 y las anotaciones hechas con él en estos informes.

Debemos destacar también que los conjunto de datos usados aún son experimentales, a día de hoy seguimos encontrando diversos errores de anotación en ellos[8]. Por lo que será un factor a considerar no solo a la hora de analizarlos, sino de seleccionar un método y medir su rendimiento.

Antes de continuar con un estudio más profundo, debemos aclarar qué estándar de CIE-10 empleamos y como se estructura, ya que existen múltiples versiones para cada idioma. Además con el paso de los años, CIE ha ido quitando ramas de su estándar para crear otras. Un ejemplo de esto es la sustitución de la rama de códigos E13 (asociado a la diabetes mellitus) por la E14. A día de hoy no existen formas triviales de actualizar los códigos al estándar más reciente, por lo que debemos asegurarnos de usar la versión correcta. El estándar bajo el cual se anotaron los documentos es la versión de ICD-10 catalana: CIM-10 [62], del 2010, siendo del 2008 su correspondiente en español [63]. Ambas versiones tienen una correspondencia casi total,

[8]Comunes debido a la ingente cantidad de datos procesados por los profesionales.

exceptuando códigos de enfermedades temporales que no aparecen en nuestros documentos. Por lo que podemos usar estos dos estándares de forma totalmente intercambiable. De aquí en adelante cualquier mención a CIE hará referencia tanto a CIM-10 del 2010 como a CIE-10 del 2008, pues a efectos prácticos son lo mismo.

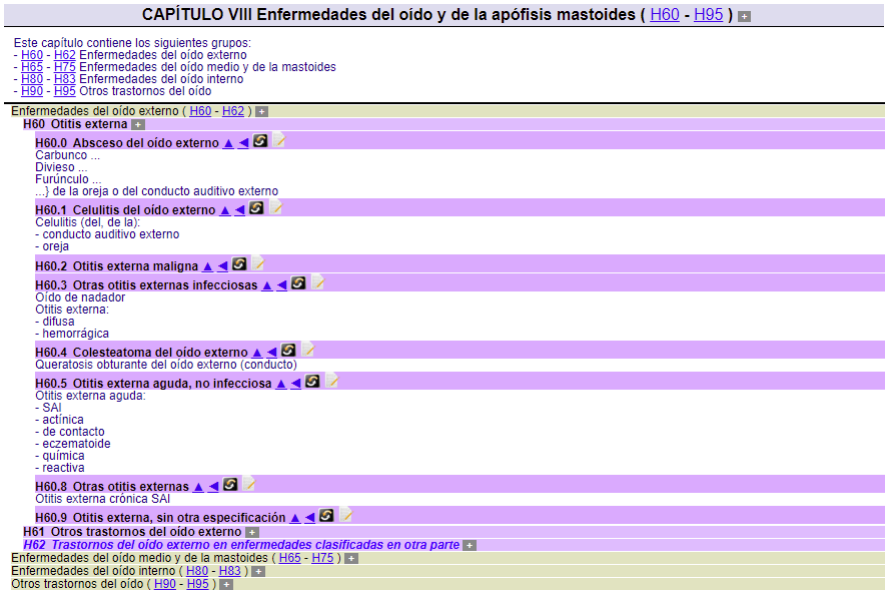


Figura 8: Estructura del séptimo capítulo de CIE-10

Fuente: https://eciemaps.mscbs.gob.es/ecieMaps/browser/index_10_2008.html

Como hemos explicado anteriormente, CIE es un estándar jerárquico: a cada dígito añadido, o nivel de profundidad, más específico se vuelve un identificador. Tratamos en esta versión con una longitud máxima de 5 dígitos y 5 niveles que denominaremos: Capítulo, Subcapítulo, Código, Subcódigo e Infracódigo[9].

Un último apunte a considerar sobre cómo está creado CIE. Son necesarios los 2 primeros dígitos para identificar un capítulo y los 3 dígitos de un código para un subcapítulo (en lugar de usar el 1º y 2º dígito, como podríamos intuir). Se establece también que la longitud mínima siempre es el código de 3 dígitos, pudiendo tener subcódigos e infracódigos si necesario. Podemos observar estos comportamientos en las figuras 8 y 9 que representan las estructuras del capítulo VII y X respectivamente.

[9]Con tal de evitar confusiones Código como medida de 3 dígitos siempre será referido de forma explícita con sus 3 dígitos o bien como código completo a diferencia de código como sinónimo de identificador.

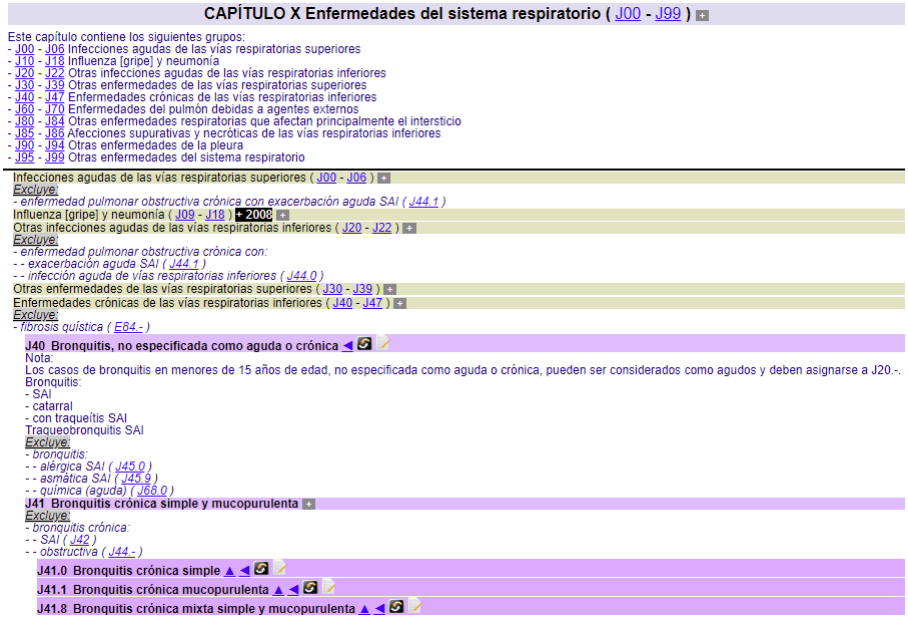


Figura 9: Estructura del décimo capítulo de CIE-10

Fuente: https://eciemaps.mscbs.gob.es/ecieMaps/browser/index_10_2008.html

4.1.2. Estudio de la profundidad y distribución de los códigos

Como podemos ver en la tabla 1 contamos con un buen número de documentos y menciones, a la altura de muchos *corpus*[29]. Pero un análisis tan simple no nos basta, sabemos que esto datos representan codificaciones de enfermedades tratadas por los CAPs. Por lo que es muy probable que tratemos con un conjunto de datos descompensados. Sabemos que hay enfermedades más frecuentes que otras e incluso se codifican en CIE algunas erradicadas en España, si no lo están en toda Europa. Por lo que es necesario un análisis de la distribución de los identificadores que estamos tratando.

Idioma	Documentos	Menciones codificadas	Códigos distintos cubiertos
Castellano	13076	7143	843
Catalán	19412	8449	912
Total	32488	15592	1181

Tabla 1: Vista preliminar de los datos

Pese a que las normas de los anotadores médicos obligan a usar siempre el identificador más específico posible (si tiene subcódigos o infracódigos deben usarse), denominado código válido, la realidad es muy distinta. Después de un análisis superficial, hemos observado que tenemos identificadores más genéricos de lo que deberían ser.

Lo anterior puede suponer un problema, ya que hacer más específico un código no resulta una tarea trivial sin el conocimiento médico pertinente. Mientras que por la otra parte, la ge-

neralización consiste en recortar dígitos finales. Es por ello que hemos encontrado necesario investigar también la profundidad a usar. Para poder resolver satisfactoriamente este proyecto debemos establecer un marco común y entendible desde el que partir. Al concluir con una profundidad recortaremos los identificadores más largos y descartaremos todos los que deberían ser más específicos, pero no lo son. Nuestro objetivo, por lo tanto, es elegir una profundidad que descarte el mínimo número de códigos posibles y maximice la especificidad dotándonos de este marco común del que hablamos.

Para poder hacer un análisis de la profundidad y distribución de nuestros datos, hemos elaborado tablas de frecuencia de los códigos válidos para cada nivel de profundidad. Hemos hecho esto no solo para el conjunto total, sino también separando por idiomas. De esta manera podremos saber si existen desequilibrios particulares respecto a estos. Las tablas elaboradas son muy largas y densas para observarlas y deducir algo sobre ellas, por eso hemos procesado los datos en gráficos más fáciles de digerir que comentaremos a continuación.

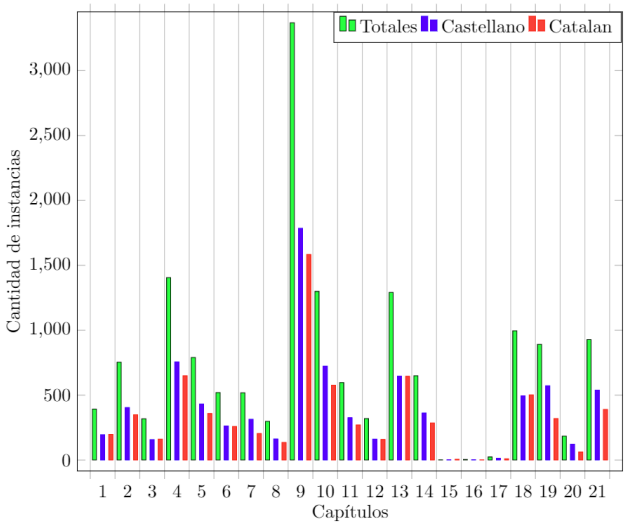
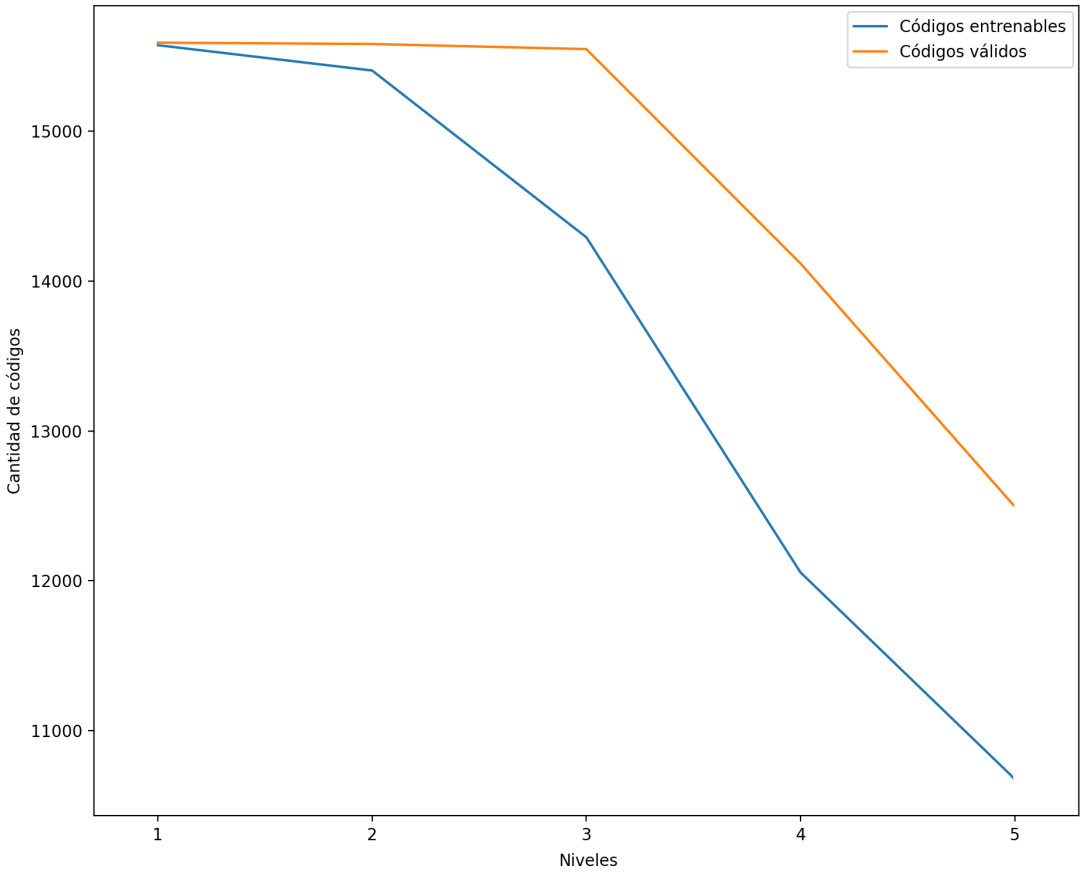


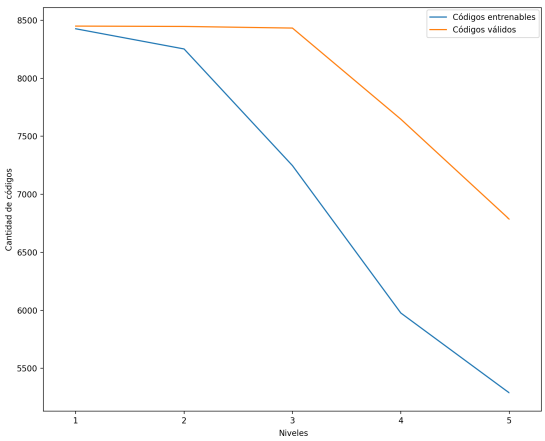
Figura 10: Distribución por capítulos de las instancias

Fuente: Propia

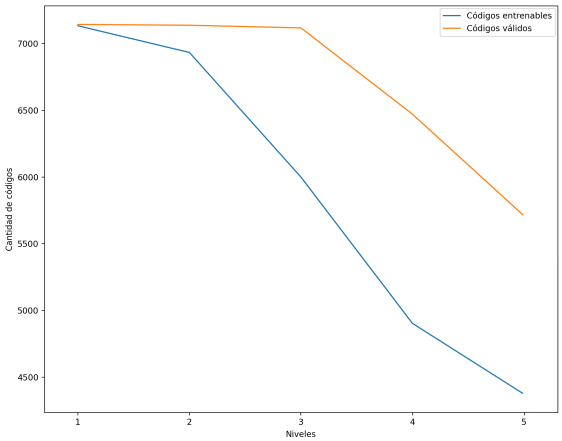
Como vemos en la figura 10 las distribuciones por capítulos nos pronostican lo que habíamos discutido anteriormente, estamos tratando con un conjunto de datos completamente desbalanceado. Esto no es algo propio de alguna lengua sino del conjunto de datos en sí, pues tanto en catalán como castellano se reproducen distribuciones similares. Además, los capítulos que apenas tienen instancias son los relacionados con enfermedades durante el embarazo, el período perinatal y de malformaciones de nacimiento que son casos que los CAPS no atienden demasiado. Obviando que son un tipo de problemas que no afectan mucho a la población actual. Si por el otro lado miramos los capítulos más poblados; podemos observar que son enfermedades del sistema circulatorio, del respiratorio y endocrinas; la clase de enfermedades más cronificables y susceptibles de ser vistas en Atención Primaria. Lo que respalda aún más esta idea de que las cantidades responden a la naturaleza de obtención de este *corpus*. Cabe destacar que originalmente CIE tiene 22 capítulos, siendo el último destinado a uso temporal y explicando porqué no tenemos datos de este.



(a) Evolución de los códigos bilingües por profundidad



(b) Evolución de los códigos en castellano por profundidad



(c) Evolución de los códigos en catalán por profundidad

Figura 11: Evolución de los códigos por profundidad

Fuente: Propia

Como podemos deducir a mayor profundidad menor número de códigos válidos, pues a mayor especificad más grupos de identificadores genéricos dejan de ser utilizables, como observamos en los gráficos de la figura 11. Por las cantidades que se gestionan, vemos que el total de códigos válidos perdidos (o de instancias pérdidas), en el peor caso, es relativamente pequeño e insignificante, si solo nos fijamos en la validez. El problema de fijarnos solo en este dato es que no enseñan el otro factor que discutimos anteriormente: una distribución desigual. A mayor profundidad también aumenta la cantidad de códigos válidos posibles (o clases), por ejemplo ”A00” deriva en ”A00.1”, ”A00.2” y ”A00.9”. La cantidad de clases posibles crece exponencialmente a cada dígito, por lo que debemos ver cuantas de estas clases se cubren a cada nivel. Ya que igual de 100 instancias, 95 van a una rama mientras que el resto recibe 2 y 3. A efectos prácticos se pierde información, 2 o 3 instancia de una clase no son suficientes para un buen entreno.

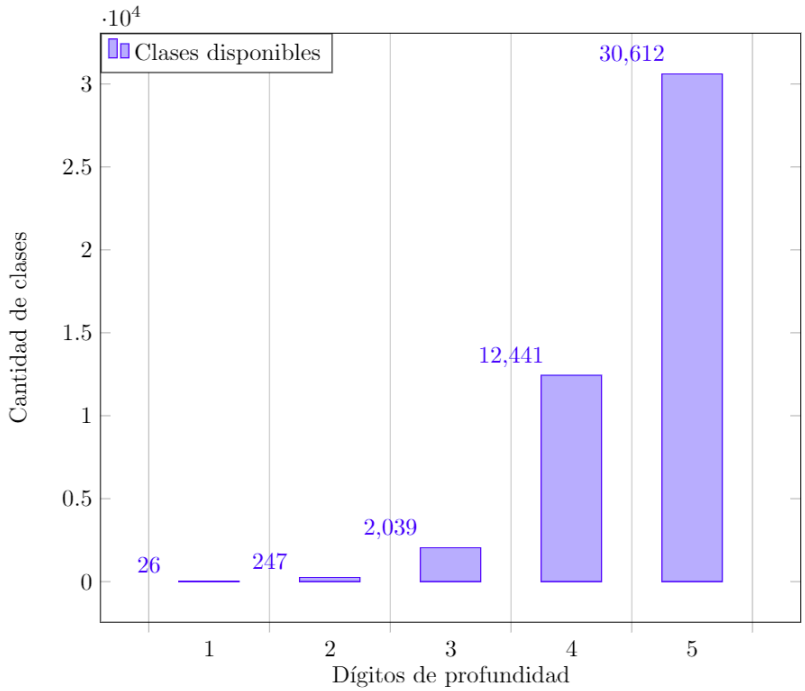
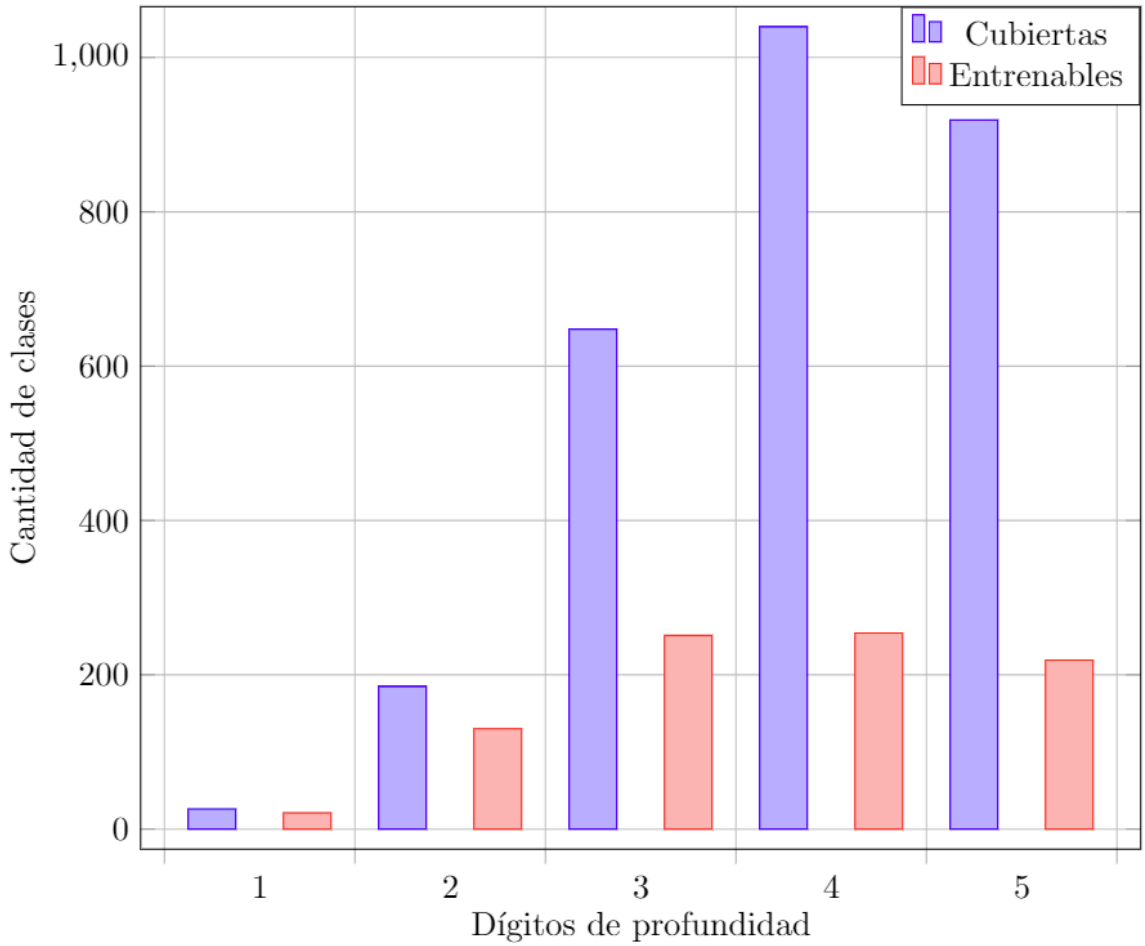


Figura 12: Evolución de las clases existentes por profundidad

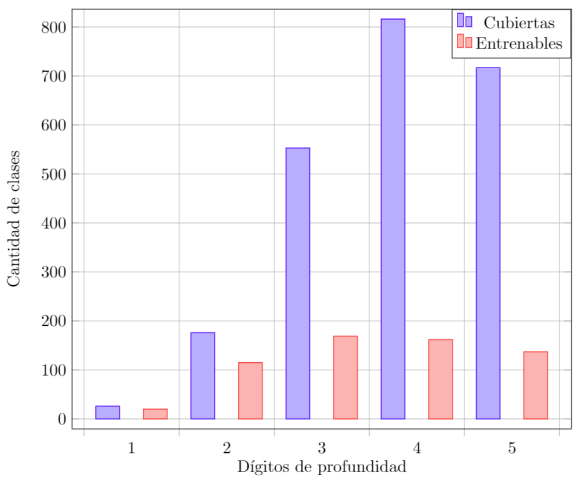
Fuente: Propia

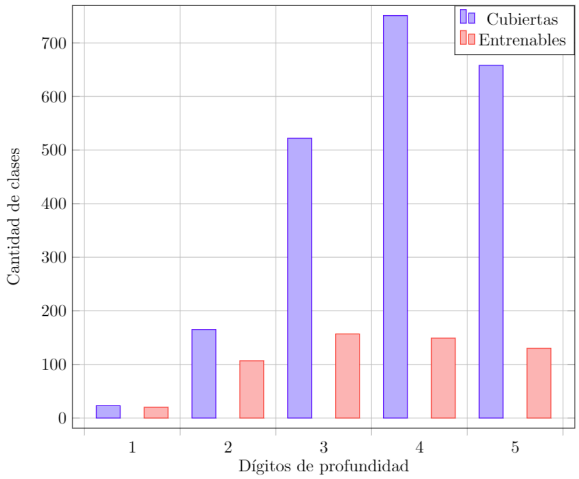
Observando la figura 12 comprobamos que a mayor profundidad mayor número de clases disponibles. Pero el número de estos cubiertos por suficientes instancias no lo hace al mismo ritmo. Si contamos ahora con las métricas de cuantos códigos entrenables hay por profundidad en las figuras de la tabla 11[10], vemos como al incrementarse la especificidad se produce un mayor decrecimiento de la cantidad de instancias disponibles.

[10]No solo quitamos los códigos que no sean válidos en esa profundidad, sino también quitamos el conjunto de códigos que no cubren una clase en suficiente número. Por ejemplo si la clase ”A02.2” no tiene el suficiente número de instancias todo estas se descartan.



(a) Evolución de las clases bilingües por profundidad





(b) Evolución de clases en castellano por profundidad (c) Evolución de clases en catalán por profundidad

Figura 13: Evolución de las clases por profundidad

Fuente: Propia

Podemos ver en la figura 13, como se respalda este comportamiento que mencionábamos de la distribución. A mayor profundidad mayor número de clases cubiertas hasta llegar a la profundidad 5[11]. Pero aunque haya un crecimiento de clases cubiertas, las cubiertas con un número ”suficiente” de instancias crecen más paulatinamente y se estancan a partir de la profundidad 4. La razón de esto como comentábamos es por la distribución desigual.

Entendemos por ”suficientes” o ”entrenables” al menos diez instancias, este número está lejos de ser ideal. Pero es uno modulable en caso de hacer K fold crossing (de K valor 5 u 10) y divisible en proporciones de 80/20 para entrenamiento y prueba. Pese a ser un número tan pequeño, vemos que ya afecta significativamente a las clases cubiertas y al decrecimiento de códigos que nos sean útiles. Lo que nos permite hacernos una idea de como afecta a la distribución el profundizar.

Antes de empezar a explicar nuestras elecciones, y nuestros razonamientos detrás de estas, creemos necesario mencionar como, pese a desbalance inherente de los datos, estos siguen la misma distribución entre lenguas. Como hemos observado en las figuras 10, 11 y 13. Trasladándose así la cuestión de obtener un rendimiento independiente del idioma al método. Ya que los datos no muestran una parcialidad hacía ninguno de los dos.

4.2. Estándar de codificación de entidades médicas y la profundidad usada

La razón principal para adoptar el estándar CIE-10, en nuestro proyecto, es debido a tres factores clave. Siendo el primero que el gobierno español ya exige que todos los documentos oficiales se clasifiquen mediante este estándar [3]. El segundo es el valor añadido de investigar normalizar entidades médicas en CIE, pues como hemos visto existen pocas que lo usen de forma exclusiva, y más si consideramos que es tanto sobre la versión española como catalana. Y siendo el último por una necesidad dentro del proyecto base, como ya sabemos es el usado en nuestros datos. Si tenemos en cuenta que la alternativa es Snomed-CT, que cubre las mismas enfermedades que CIE [6], no estamos perdiendo demasiada información. Y cabe la posibilidad de aplicar métodos que emplean Snomed-CT, ya que podremos traducir los resultados a CIE.

Como ya hemos mencionado, en la sección 4.1.1, nuestros datos se componen de diversas tipologías de informes clínicos de atención primaria. Nosotros emplearemos toda esta variedad documental, pues nos dota de una gran variedad de formas de expresarse, relativas a las circunstancias de creación así como de enfermedades. Ya que un documento de atención domiciliaria versará más sobre ciertos tipos de enfermedades que de otros, diferentes a los que se pueden encontrar en informes de diagnóstico.

Por último debemos hablar sobre el nivel de profundidad que usaremos. Como hemos visto tenemos diversa cantidad de identificadores para cada nivel de profundidad. Elegir una profundidad limitará el acceso a los códigos disponibles, así como determinar la clase de asignación que estamos realizando. Recordemos que una longitud muy pequeña puede ser trivial y una demasiado larga puede ser demasiado específica. Por lo que no cubriríamos de forma suficiente el entrenamiento. Nuestro objetivo es determinar cuál es la más adecuada, y para ello usaremos

[11] Recordemos que era la profundidad original dónde había muchos generales que no podíamos contar como válidos.

las figuras comentadas en el estudio anterior.

Nos parece obvio que las longitudes más adecuadas son la 3 y la 4. Pese a que 2 y 1 tienen mayor número de códigos y una mejor relación entre las clases cubiertas y las existentes, como vimos en 11. Estos factores no son tan altos como para poder suponer una ventaja respecto a la especificidad dada por los otros dos. Tampoco perderíamos datos si empleásemos dichas longitudes, pues podríamos elaborar una asignación por capítulo o subcapítulo a partir de estas. De las dos opciones que nos quedan, la profundidad de 3 dígitos nos pareció la más adecuada. Ya que, como vemos en las figuras 11 y 13, pese a añadir especificidad con longitud 4 esta apenas significa nada. Ya que el número de códigos que cubrimos de forma "suficiente" apenas aumenta, y el número de clases disponibles se sextuplica. Hay que tomar también en consideración, la caída nada deleznable de 2000 códigos válidos, al pasar de 3 a 4 dígitos. Por lo que para nuestro problema emplearemos CIE-10 con una longitud de tres dígitos, también llamada código completo.

4.3. Método

El método que nos hemos propuesto seguir es el de IR *Retrieval* con la variante de relevancia local. Más concretamente, nos basaremos en un conjunto de investigaciones que usan BERT, un nuevo modelo de redes neuronales, como segundo sistema. Aunque ya profundizaremos en ello en siguientes secciones.

Las razones principales para elegir este método respecto a otros son diversas y las explicaremos a continuación. Obviamente un acercamiento por producto que resuelva NER y NEN es inviable pues no son completamente automáticos y siguen requiriendo asistencia manual, además de los costes añadidos de tener que adquirir una licencia de un producto para su uso en otro proyecto. Uno de los primeros factores que tenemos en cuenta es el rendimiento de los sistemas y su tiempo de ejecución, con esto en mente es trivial ver porque métodos como la traducción de registro y el IR global fueron descartados.

Como sabemos el objetivo de este proyecto es implementar una solución al problema de NEN, y necesitamos que rinda de forma consistente sobre las diversas clases desbalanceadas. Este problema no se encuentra abordado por investigaciones de clasificación. También necesitamos un método que nos ofrezca ciertas garantías de rendimiento sobre nuestra clase de documentos. Los métodos de clasificación tampoco ofrecen dichas garantías, ya que en su mayoría son de lengua inglesa. Debemos también tener en cuenta el problema del número inferior de clases (en muchas de las soluciones) que requeriría de una modificación crítica de la arquitectura de estos métodos.

Pese a que los problemas anteriores se pueden solucionar, debemos considerarlos si tenemos en cuenta la otra alternativa. Esta nos permite no sólo jugar con datos desbalanceados y con un gran número de clases, sino que representa el estado del arte actual de este problema. Durante varios años ya ha supuesto el *Gold Standard* para diversos *corpus* (que incluyen documentos más informales). Con el añadido de que la solución que pretendemos seguir incorpora las tecnologías más innovadoras del campo del Procesamiento de Lenguaje Natural.

Aunque es cierto que deberemos lidiar con problemas derivados de este método, consideramos a estos más abordables. Ya que las soluciones para estas, a nuestro parecer, no incrementan

en tanto el grado de incertidumbre sobre el rendimiento. A diferencia de las posibles soluciones a las desventajas de la clasificación.

4.4. Métricas

Respecto a las métricas empleadas, usaremos *accuracy* para medir el rendimiento de todo el sistema. Aplicaremos esta medición de métricas al método aplicado a documentos mixtos así como separado por idioma. De esta manera observaremos si existen parcialidades respecto al catalán o castellano.

La razón para emplear la *accuracy* es por un ejercicio de comparación, creemos importante el poder comparar nuestros resultados con los de métodos similares, los cuáles usan esta métrica. También es la métrica que más sentido tiene, pues trabajaremos asignando una entidad por mención sin usar una clase *None*. Por lo que siempre tendremos una asignación, y lo que nos interesa es ver cómo asigna este método.

Como sabemos se usan dos sistemas independientes entre si. Con el objetivo de facilitar el desarrollo, también mediremos el rendimiento de esto sistemas independientemente. Cabe mencionar que estas métricas son más un heurístico de análisis durante el desarrollo que una medición precisa del rendimiento del método. Para eso nos centraremos en la *accuracy* del sistema entero como tal.

5. Metodología y rigor

En esta sección debatiremos el cómo y el porqué de la elección del desarrollo en cascada y cómo se mantendrá el rigor con este método. También se mentarán las principales herramientas de desarrollo que usaremos.

5.1. Desarrollo en cascada

De todas las metodologías existentes hemos escogido el desarrollo en cascada. Como podemos ver todos lo subojetivos son muy dependientes del orden ejecutado, lo positivo es que estas dependencias temporales son debidas al tercer subobjetivo: para complementar el método desarrollado debemos seleccionar, implementar y comparar las mejoras sobre las anteriores. La metodología de cascada nos permite realizar el primer, segundo y la primera parte del tercer subojetivo (desarrollar el método base) de forma que podamos elaborar mejoras, y compararlas de forma iterativa, con la posibilidad de volver a reconsiderar métricas, si es necesario.

El ciclo de trabajo ideal sería el siguiente: realizar una búsqueda intensiva del estado del arte y decidir las métricas a usar para evaluar el método y sus mejoras, hasta este punto tendríamos un desarrollo más secuencial. Y luego entraríamos en la fase más iterativa, desarrollar varias mejoras, eligiéndolas en función de que clase de soluciones están dando mejores resultados, compararlos posteriormente entre todos y con la posibilidad de volver a la segunda, o incluso primera, etapa si al acabar esta iteración se detectan problemas a solventar. Con esta forma de trabajo podemos minimizar el mayor riesgo de este proyecto, con el valor añadido de facilitar la documentación de todo el proceso debido a la naturaleza de esta metodología.

5.2. Herramientas para el desarrollo

Debido a la sensibilidad de los datos gestionados por los CAPs no podemos tener una copia de ellos en local, por ello todas las pruebas y ejecuciones se harán sobre la plataforma del departamento de Ciencias de la Computación, más concretamente el HPC de rdlab. Usaremos un repositorio GIT para facilitar la gestión de cada iteración así como el desarrollo base del método. Para la planificación de cada etapa, desarrollo y verificación del correcto desempeño de cada iteración usaremos Google Calendar y Trello.

5.3. Métodos de validación

La metodología en cascada nos ayudará no solo al desarrollo sino también a la validación de este. Al final de cada iteración, junto a la evaluación de los nuevas mejoras implementadas y la posible ampliación de métricas, se concertará una reunión con los directores para discutir los resultados y asegurar un desarrollo correcto. También se realizarán reuniones de seguimiento a final de cada etapa dentro de la iteración, con el objetivo de verificaciones menos intensas y más enfocadas a reforzar la constancia del avance del proyecto.

6. Planificación temporal

Esta entrega tiene como objetivo elaborar una planificación realista y práctica para llevar a término nuestro TFG, el cual está previsto que ocupe 18 créditos o 540 horas de trabajo, siguiendo la normativa de la UPC y la FIB [64] en la que se establece que cada crédito equivale a 30 horas de trabajo. Para ello desglosaremos el proyecto en las diversas tareas que necesitamos realizar, incluyendo sus estimaciones temporales; una planificación de la realización de estas, mediante un Gantt; así como un plan de gestión para los riesgos y obstáculos previstos.

Para una planificación correcta del proyecto debemos poder identificar y entender las tareas que necesitamos cumplir, así como sus dependencias, y es lo que haremos a continuación.

6.1. Desglose de tareas

6.1.1. Tareas relacionadas con la gestión del proyecto [T1-T8]

Bajo este título hablaremos de las diversas tareas que cubren la gestión de nuestro proyecto. Como primera tarea [T2] nos referiremos a la planificación general de todo el proyecto, por razones obvias es la primera que debemos considerar pues sin ella nos se podrá llevar a buen término el resto de tareas. También debemos destacar: la definición del alcance del proyecto [T1], pues debemos saber hasta donde puede y queremos que este llegue; elaborar un presupuesto[T3], sobre lo que disponemos y/o necesitamos; redactar el informe de sostenibilidad [T4], para considerar los impactos sociales, medioambientales y económicos de nuestro trabajo.

Una vez hemos hablado de las tareas con una realización más clásica, hemos de abordar las tareas de gestión dentro de las iteraciones de nuestro desarrollo en cascada. Dentro de estas podemos ver una planificación inicial de la iteración[T5], esta tarea tiene como objetivo asegurar el desarrollo correcto de los algoritmos elegidos para poder compaginar no solo tiempo de desarrollo sino de entrenamiento de modelos y medición de resultados. Al final de cada iteración realizaremos una reunión de seguimiento con nuestros directores, esta tarea [T6] nos permitirá asegurar un desarrollo correcto de la siguiente iteración, así como llevar un control de lo que ya se ha hecho y con qué ritmo.

Después de cada implementación, y durante la depuración, haremos una breve reunión con los directores [T7], para favorecer que se cumplan los plazos asignados a cada paso y que la iteración se haga cumpliendo el ritmo inicial propuesto.

También debemos tener en cuenta la documentación del proyecto[T8], esta tarea se desarrollará en paralelo a todas las discutidas a continuación.

6.1.2. Estudio del estado del arte [T9]

Como ya hemos mencionado, este proyecto se basa en la investigación y desarrollo de métodos y mejoras para resolver el problema de codificar entidades médicas en informes clínicos. Una de las bases de este TFG es la investigación misma, por ello debemos dedicar una buena

cantidad de tiempo a la realización de un buen estado del arte y entender qué métodos tenemos a nuestro alcance, los puntos fuertes y débiles de cada uno así como entender como suplir estos últimos. Consideramos vital el completar esta tarea antes de pasar al desarrollo iterativo de las soluciones, aunque es posible volver si necesitamos ampliar la investigación sería deseable no hacerlo, pues supondría un coste temporal importante.

6.1.3. Selección de métricas para la medición de resultados[T10]

Necesitamos investigar qué métricas expresan mejor lo que buscamos en el método y los suplementos para perfeccionarlo que implementaremos. Lo ideal sería definir un conjunto estático de métricas, pero existe la posibilidad de que, con nuevos componentes, veamos que las usadas hasta ahora no sirven y debamos buscar alguna nueva. Es por ello que ya hemos considerado realizar la mayor parte de esta tarea al principio del proyecto, y volver a ella parcialmente al inicio de cada iteración, basándonos en los nuevos resultados. Esta investigación inicial se puede compaginar, en cierta medida, con la investigación del estado del arte.

6.1.4. Preparación del sistema [T11]

Para poder trabajar en nuestros métodos debemos tener un entorno listo para el desarrollo, prueba y medición de las soluciones. Principalmente, hablamos de configurar un repositorio GIT, instalación de *software* y librerías, elaborar un conjunto de pruebas para los métodos así como un sistema de medición de resultados que permita añadir nuevas métricas de forma relativamente trivial. Esta tarea es inicial al proyecto, pues una vez creado este sistema ya solo debemos añadir las métricas nuevas, si necesario.

6.1.5. Selección de los algoritmos [T12]

Esta tarea consiste en elegir que algoritmos implementaremos junto con el método que mejoraran; debido a la cantidad de soluciones que podemos encontrar, a como respondan estas en nuestra implementación y nuestros datos. Es muy posible dejarse una mejora muy relevante, por ello realizaremos esta tarea en diversas iteraciones. Elegiremos en la primera iteración el método sobre el cual construiremos nuestra solución y en cada iteración posterior las mejoras serán seleccionadas en función de cómo rinde el método fundacional con ellas y que puntos flacos muestra que se puedan solucionar.

6.1.6. Implementación de los algoritmos [T13][T14][T15][T16]

Una vez elegido el método o las mejoras de la iteración debemos implementarlos correctamente, estas tareas se dedicarán a ello. Podemos descomponer este proceso en cuatro tareas: recopilar y adaptar lo necesario para implementar [T13], programar los algoritmos como tal [T14], aplicar un conjunto de pruebas [T15] y depurar estos métodos [T16]. Obviamente esta es uno de los procesos más importante, y por su naturaleza, también se puede alargar ampliamente, por lo que le dedicaremos un número importante de horas.

6.1.7. Entrenamiento de los modelos [T17]

Este entrenamiento puede compaginarse perfectamente con las tareas de probar y depurar los métodos (entenderemos cada conjunto de mejoras junto al método fundacional como uno propio) ([T15] y [T16]), pues para poder probar las soluciones de este tipo necesitamos entrenarlas primero y además se trata de un proceso totalmente automático, pero ocupa tiempo y puede incrementarse gravemente según el modelo y el tipo de entrenamiento, por lo que se debe considerar su coste.

6.1.8. Medición de resultados [T18]

Una vez implementados los métodos debemos medir los resultados mediante las métricas que hemos decidido. Pese a que esta tarea no es costosa temporalmente, es crítica para un desarrollo correcto del proyecto. Es en esta fase donde podríamos ver carencias en las métricas así como la clase de métodos que mejor están rindiendo.

6.1.9. Comparación de resultados y conclusiones [T19]

Al acabar todas las iteraciones deberemos comparar los resultados de todos los métodos, concluir sobre ellos y elaborar la defensa ante el tribunal. Así pues, en esta tarea redactaremos nuestras conclusiones finales; acabaremos la memoria, junto al final de la tarea de documentación, y prepararemos la defensa de esta.

6.1.10. Selección del personal [T20]

Para poder realizar el proyecto necesitaremos buscar y contratar a los recursos humanos, necesarios para realizar las tareas.

6.2. Recursos

Una vez hemos hablado de las distintas tareas, debemos tener en cuenta los recursos usados para llevarlas a cabo.

Como ya comentamos en apartados anteriores, los datos que usaremos son muy sensibles por ello emplearemos la plataforma del Departamento de Ciencias de la Computación, al cual accederemos mediante SSH desde nuestro portátil, para la elaboración y prueba de los métodos.

Usaremos el lenguaje de programación Python para el desarrollo de los métodos así como el sistema para medir el rendimiento de estos. Python es un lenguaje muy popular lo que nos da acceso a múltiples librerías, Tensorflow entre otras, y un amplio catalogo de recursos que nos facilitarán el desarrollo.

Con respecto de los recursos humanos para la realización de las tareas mentadas, serán necesarios tres clases de perfiles profesionales distintivos: un ingeniero informático, un director

de proyecto y un reclutador de recursos humanos. En la siguiente figura,14, podemos ver una estructura de los roles del proyecto.

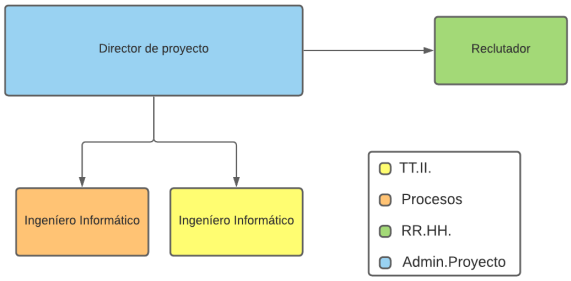


Figura 14: Estructura de roles del proyecto

Fuente: Propia

Para la realización de las tareas más relacionadas con la gestión y la planificación usaremos Trello y Google Calendar.

Por último, todas las actividades relacionadas con elaborar la documentación y memoria se harán mediante el uso Latex.

6.3. Estimación temporal y diagrama de Gantt

6.3.1. Estimación temporal

A continuación detallamos en las siguientes tablas los costes temporales de cada tarea, en horas, así como sus dependencias y los responsable de llevarla a término. Como vemos aparecen un total de 640 horas, esto es porque estamos contando las horas de entrenamiento de modelos y la selección del personal. Pese a no considerar que el entrenar modelos sea parte del trabajo si es importante considerar sus horas, por las dependencias que tienen de ella otras tareas. Estamos considerando la tarea de selección del personal, pues aún siendo inherente al proyecto y afectando a su presupuesto, se realiza fuera de este por lo que se consideran horas añadidas a la 540 horas de desarrollo. Considerando el inicio del TFG en el 21/09/20, con 5 horas diarias de trabajo hasta el 5 de octubre y 8 de media de ahí en adelante, sin contar fines de semana, finalizaríamos el 14/01/20, pero eso no fue así. El proyecto se finalizo el 24/03/2021, este salto de casi tres meses se produjo debido a retrasos fuera de nuestro control que nos impidió el trabajo y que explicaremos, más concretamente, en la sección posterior a la gestión económica del proyecto. A continuación explicaremos la división de tareas, la gestión de riesgos y el diagrama final de Gantt.

Tareas	Dependencias	Coste temporal
Definición del alcance [T1]	T20	25
Planificación general [T2]	T1	30
Elaborar presupuesto [T3]	T2	15
Informe de sostenibilidad [T4]	T1	20
Planificación de la iteración [T5]	T2	10
Reunión de control [T6]	T18	4
Reunión de seguimiento [T7]	T15	1
Documentar [T8]	T1	80
Estudio del estado del arte [T9]	T3, T4 y T5	90
Selección de métricas [T10]	T9	70
Preparación del sistema [T11]	T10	15
Selección de los algoritmos [T12]	T11	20
Preparar la implementación [T13]	T12	20
Implementación de los algoritmos [T14]	T13	18
Pruebas [T15]	T17	25
Depuración [T16]	T15	27
Entrenamiento de los modelos [T17]	T14	50
Medición de resultados [T18]	T16 y T17	30
Comparación de resultados y conclusiones [T19]	T8	40
Selección del personal [T20]	-	50
Total	**-**	**640**

Tabla 2: Estimación temporal y dependencias de las tareas

Tareas	Responsable	Soporte	Ejecutor
Definición del alcance [T1]	D	-	D
Planificación general [T2]	D	I	D
Elaborar presupuesto [T3]	D	-	D
Informe de sostenibilidad [T4]	D	-	D
Planificación de la iteración [T5]	D	I	D/I
Reunión de control [T6]	D	I	D/I
Reunión de seguimiento [T7]	D	I	D/I
Documentar [T8]	D	I	D/I
Estudio del estado del arte [T9]	I	-	I
Selección de métricas [T10]	I	-	I
Preparación del sistema [T11]	I	-	I
Selección de los algoritmos [T12]	I	-	I
Preparar la implementación [T13]	I	-	I
Implementación de los algoritmos [T14]	I	-	I
Pruebas [T15]	I	-	I
Depuración [T16]	I	-	I
Entrenamiento de los modelos [T17]	I	-	I
Medición de resultados [T18]	I	-	I
Comparación de resultados y conclusiones [T19]	D	I	D
Selección del personal [T20]	R	-	R

Tabla 3: Tabla de responsabilidades

6.4. Diagrama de Gantt sobre el desarollo de las tareas

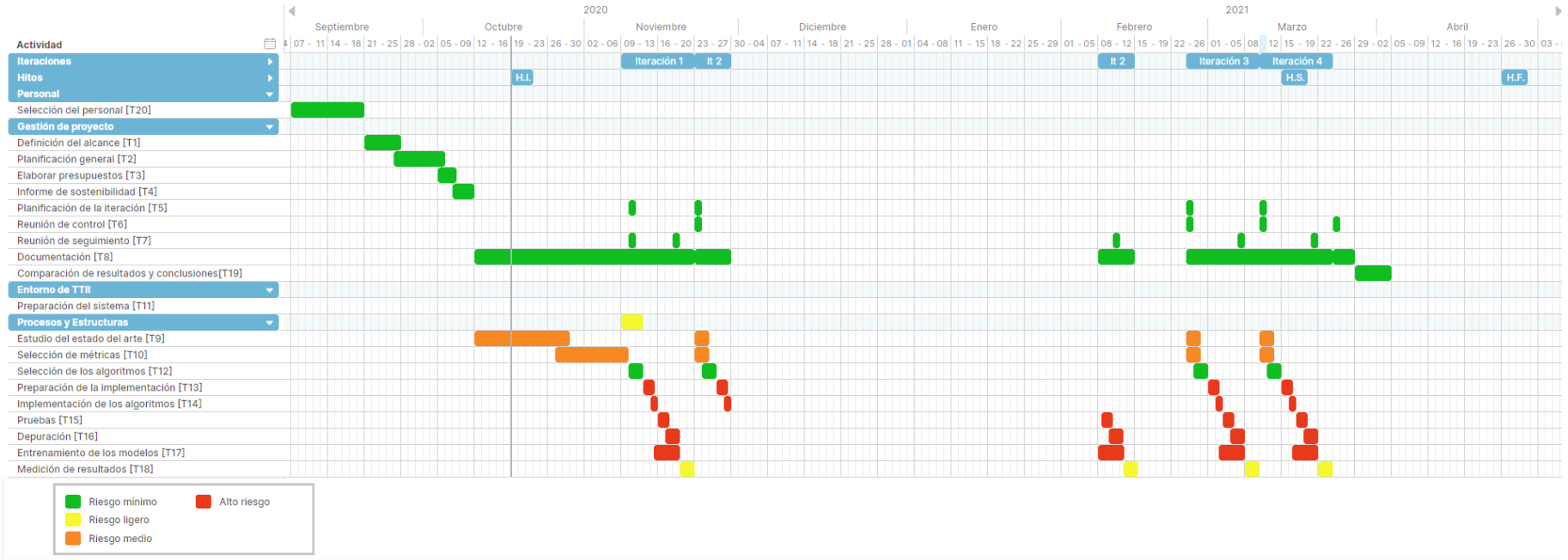


Figura 15: Diagrama de Gantt actualizado

6.5. Gestión del riesgo: Planes alternativos y obstáculos

Como podemos ver, en el diagrama de Gantt anterior(15), nos hemos decantado por un desarrollo con 4 iteraciones, cada una se constituye de tareas de gestión así como desarrollo. También hemos incluido, al principio de cada una, las tareas [T9] y [T10]. Lo anterior es debido a dos riesgos que hemos previsto: el volver a revisar métricas, al desarrollar soluciones, y el otro riesgo, mucho menos probable, de tener que volver a investigar el estado del arte.

Hemos sobre-estimado ciertas tareas, como puede ser la selección de algoritmos o la medición de los resultados, esto es debido a que, pese a ser tareas relativamente poco largas, son críticas para el desarrollo de nuestro proyecto y pueden requerir de más tiempo en circunstancias adversas.

Si los riesgos anteriores no se cumplen esto nos dará tiempo adicional que tenemos para el proceso de implementación de soluciones que, como habíamos considerado en entregas anteriores, puede alargarse debido a los obstáculos que presenta esta tarea; o bien para el entreno de modelos, pues como ya hemos discutido en los apartados anteriores puede suponer la tarea con mayor coste y variabilidad temporal.

Además, no estamos contando con los fines de semanas en esta planificación, por dos motivos: un descanso necesario así como contar con dos días comodín a la semana, en caso de que alguna tarea se alargue mucho más de la cuenta.

Como podemos ver, hemos previsto el final del TFG el 24 marzo del 2021 (considerando los retrasos que ya explicaremos), siendo la defensa de la memoria en Abril, esto nos da un margen algo menor de tres semanas para poder solucionar cualquier problema o riesgo imprevisto que pueda surgir y retrasar el desarrollo del proyecto.

7. Gestión económica

A continuación elaboraremos unos presupuestos bajo un análisis económico del proyecto, en el cual identificaremos los costes, una estimación cuantitativa de estos y la descripción de planes de control para asegurar el cumplimiento con la desviación mínima.

7.1. Identificación de los costes

7.1.1. Costes de las actividades y personal

Para poder estimar los costes monetarios de cada actividad debemos considerar no solo su coste temporal sino también quien las realizara, pues cada empleado tendrá un sueldo asignado. Para este proyecto, las tareas no tienen costes adicionales propios que no dependan directamente del personal que las realice.

El primer personal que resulta obvio contratar es un ingeniero informático para las tareas de investigación y desarrollo del proyecto, así como la elaboración de la documentación más relacionada con las implementaciones. También necesitamos contar con un perfil de ingeniero orientado hacía las tareas de gestión así como documentación y presentación de la memoria, preferiblemente informático, para un entendimiento amplio del proyecto. Por último, necesitaremos un personal con perfil enfocado a recursos humanos, para la contratación de los anteriores.

Como hemos dicho antes, el coste de las actividades vendrá dado por el tiempo de realización así como el sueldo del personal contratado; por ello podremos ver a continuación el coste bruto, por horas, de cada empleado [65, 66, 67].

Perfil	**Sueldo bruto (€/h)**
Ingeniero informático	15
Director de proyecto	20
Reclutador	18

Tabla 4: Sueldo bruto de los empleados.

En la tabla 5, vemos las diversas tareas planificadas así como su relación, en horas, con los empleados. La mayoría de tareas se realizan solas; excepto la documentación, las reuniones de seguimiento del proyecto y la planificación; pues necesitan la coordinación del ingeniero y director del proyecto. El director deberá realizar en su mayoría la tarea de planificación general, así como discutir lo previsto con el ingeniero para poder asegurar su desarrollo correcto y replantear ciertas cuestiones (si es necesario). La planificación de la iteración se hará de forma conjunta, para hacer cuadrar la general con las necesidades de la iteración. En su mayoría el ingeniero realizará la documentación y asistirá al director durante el redactado de las conclusiones.

Tarea	Informático	Director	Reclutador
Selección del personal [T20]	0	0	50
Definición del alcance [T1]	0	25	0
Planificación general [T2]	2	30	0
Elaborar presupuesto [T3]	0	15	0
Informe de sostenibilidad [T4]	0	20	0
Planificación de la iteración [T5]	10	10	0
Reunión de control [T6]	1	1	0
Reunión de seguimiento [T7]	4	4	0
Documentar [T8]	60	20	0
Estudio de estado del arte [T9]	90	0	0
Selección de métricas [T10]	70	0	0
Preparación del sistema [T11]	15	0	0
Selección de los algoritmos [T12]	20	0	0
Preparar la implementación [T13]	20	0	0
Implementación de los algoritmos [T14]	18	0	0
Pruebas [T15]	25	0	0
Depuración [T16]	27	0	0
Entrenamiento de los modelos [T17]	50	0	0
Medición de resultados [T18]	30	0	0
Comparación de resultados y conclusiones [T19]	5	40	0

Tabla 5: Horas dedicadas a cada tarea por los empleados

7.1.2. Costes genéricos

A continuación hablaremos de los costes inherentes al proyecto, que no dependen de ninguna actividad en concreto.

- **Despacho Coworking**: Debido a las condiciones actuales, por la pandemia del Covid-19, el uso de espacios físicos es peligroso. Pero también hemos de considerar la salud mental de nuestros trabajadores. Empleados como el ingeniero informático, que tienen muchas horas de trabajos por delante, pueden necesitar un espacio de trabajo exterior a su casa. Por ello alquilaremos un despacho de Coworking durante los cuatro meses de desarrollo cercano al hogar del programador para facilitar su desempeño. Como a priori no sabemos donde será, tomaremos como referencia el precio de un espacio en Barcelona. El resto de tareas se realizarán siguiendo las directivas sobre el teletrabajo, y las tareas que requieran de comunicación se harán mediante herramientas de comunicación Online y telefónicas.

- **Portátiles**: Debido a que todas las actividades son poco exigentes, con un portátil de gama media/baja podremos realizar todas las tareas. Ya que es necesaria una coordinación entres los dos empleados adquiriremos dos.

- **Plataforma de desarrollo**: El desarrollo y medición de los métodos deberá hacerse desde una plataforma que contenga los datos sensibles, en este caso la provista por el departamento de ciencias de la computación de la FIB, a la cual tendremos acceso de manera gratuita dentro del marco de cooperación de este proyecto.

- **Datos para el entrenamiento de modelos**: Para el estudio de este problema necesitamos informes clínicos así como un conjunto de términos destacados y etiquetados, para el entrenamiento de modelos. Estos datos ya los tenemos dados por el SIDIAP y el proyecto base respectivamente.
- **Software**: Usaremos Latex, Python, Trello, Google Calendar y Skype. Como sabemos todo estos servicios son gratuitos u ofrecen una versión de prueba. [68][69][70][71][72].

7.1.3. Costes de contingencias

Hemos decidido incluir un coste de contingencias del 10 % del total de costes genéricos y de las actividades. Los principales riesgos que pretendemos cubrir con estas contingencias son: por una parte solventar la subestimación en la contratación del personal, pues se trata de una simplificación; considerar pequeños alargamientos temporales así como algún coste superior puntualmente en los genéricos.

7.1.4. Costes de riesgos previstos

El riesgo principal, con el que ya estamos familiarizados, es un alargamiento excesivo del desarrollo. Pese a que ya hemos sobre-estimado las tareas críticas, existe el riesgo de tener que realizar una última iteración, para poder desarrollar alguna solución nueva o arreglar y/o mejorar alguna existente, por ello tenemos dos semanas como colchón. Esto nos da tiempo, pero debemos considerar los costes adicionales, que serían los del sueldo del ingeniero y del director, el espacio del Coworking no será renovado, pues ya estará pagado hasta inicios de febrero, y los portátiles ya están adquiridos. Consideramos unas 80 horas para el desarrollo de esta iteración extra, 20 para el director y 60 para el ingeniero; que se usaran para planificar, coordinar, desarrollar y medir los nuevos métodos. Este riesgo, denominado I1, estimamos que tiene un 30 % de posibilidades de ocurrir.

El segundo riesgo, I2, es el de que alguna iteración se alargue más de la cuenta, como ya sabemos esto pasaría sobre todo debido la implementación de soluciones o entrenamiento de modelos. Para cumplir con los plazos previstos debemos pagar las horas extraordinarias para llevar a cabo esta iteración, costando un 75 % de más que la hora normal [73], vamos a suponer un incremento de 10 horas, en tareas del informático, en cada iteración. Este riesgo lo estimamos en un 50 % por cada iteración, por lo que lo tenemos en cuenta este coste cuatro veces.

7.1.5. Costes de amortización

Para la amortización consideramos principalmente las de los dos portátiles adquiridos para este proyecto.

7.2. Estimación de los costes

Una vez discutida la tipología de los costes les daremos un valor cuantitativo en esta sección.

7.2.1. Costes genéricos y sus amortizaciones

En la tabla 6 podemos ver el precio de los costes, la vida útil y amortización de estos [74][75]. Para los precios sin IVA hemos considerado uno del 21 %.

Concepto	Coste (€)	Vida útil (meses)	Coste amortizado (€)
Despacho de Coworking	225/mes	∞	0
Portátil	600	60	40
LaTex	0	∞	0
Python	0	∞	0
Trello	0	∞	0
Google Calendar	0	∞	0
Skype	0	∞	0
Plataforma de desarollo	0	∞	0
Datos para el entrenamiento	0	∞	0

Tabla 6: Costes genéricos

7.2.2. Costes del personal

En la tabla 7 podemos ver los sueldos con un incremento del 30 %, respecto a la tabla 4, para tener en cuenta los gastos de la seguridad social así como otros impuestos y costes adicionales relacionados con la contratación de personal.

Perfil	Sueldo con 30 % (€/h)
Ingeníero informático	19.5
Director de proyecto	26
Reclutador	23.4

Tabla 7: Sueldo de los empleados con un 30 %.

7.2.3. Presupuesto final

Una vez considerado lo anterior, podemos definir nuestro presupuesto final (8).

Tareas	**Coste (€)**
Selección del personal [T20]	1170
Definición del alcance [T1]	650
Planificación general [T2]	819
Elaborar presupuesto [T3]	390
Informe de sostenibilidad [T4]	520
Planificación de la iteración [T5]	455
Reunión de control [T6]	182
Reunión de seguimiento [T7]	45.5
Documentar [T8]	1690
Estudio de estado del arte [T9]	1755
Selección de métricas [T10]	1365
Preparación del sistema [T11]	292.5
Selección de los algoritmos [T12]	390
Preparar la implementación [T13]	390
Implementación de los algoritmos [T14]	351
Pruebas [T15]	487.5
Depuración [T16]	526.5
Entrenamiento de los modelos [T17]	975
Medición de resultados [T18]	585
Comparación de resultados y conclusiones [T19]	1137.5
Total de Costes por Actividad	**14176.5**
Concepto	**Coste (€)**
Depacho de Coworking	900
Portatiles	600
LaTex	0
Python	0
Trello	0
Google Calendar	0
Skype	0
Plataforma de desarollo	0
Datos para el entrenamiento	0
Costes Genéricos Totales	**1500**
Costes Totales(CG + CPA)	**15676.5**
Contingencias (10 %)	**1567.65**
CG + CPA + Contingencias	**17244.15**
Imprevistos	**Coste (€)**
I1(Coste=1690; riesgo=40 %)	676
I2.1(Coste=341.25; riesgo=50 %)	170.625
I2.2(Coste=341.25; riesgo=50 %)	170.625
I2.3(Coste=341.25; riesgo=50 %)	170.625
I2.4(Coste=341.25; riesgo=50 %)	170.625
Coste total de los imprevistos	**1358.5**
Total	**18602.65**

Tabla 8: Presupuesto final

7.3. Control de gestión

Pese a toda la planificación y elaboración de este presupuesto este se encuentra sujeto a fallos, debido a los imprevistos que puedan surgir. Es por ello que debemos elaborar planes para gestionar estas circunstancias.

Como sabemos los principales obstáculos definidos hasta ahora es la falta de tiempo, tanto para el desarrollo de soluciones así como para encontrar los algoritmos óptimos. Para ello ya tenemos planificado las tareas de implementación con amplios márgenes y en iteraciones que empiezan por una tarea de investigación, además de esto hemos considerado dos semanas de reserva para realizar una iteración adicional, en caso de necesidad. Pero esta planificación y horas asignadas a la implementación pueden no ser suficientes. La solución a este problema consistiría en alargar el contrato del ingeniero informático por un mes así como pagar horas extras al director para que adapte la planificación con el nuevo plazo del proyecto.

Para asegurar un desarrollo correcto, con respecto al tiempo, usaremos la medida del desvío de mano de obra en consumo, que calcularemos al final de cada iteración para saber cuál es el coste extra de la etapa, con la siguiente formula: (*consumo_horas_estimado – consumo_horas_real*) **coste_hora_extra*. De la misma manera realizaremos esta medición de la iteración extraordinaria para vigilar una posible desviación así como para el caso extremo de retrasar un mes la entrega, en este último caso asegurando una hora semanal a la comprobación del cumplimiento de la planificación.

Debido al poco coste de los portátiles y a que los datos más críticos no se encontrarán en estos, ya que todos los algoritmos se desarrollaran en la plataforma de ciencias de la computación, en caso de fallo total de un sistema las pérdidas serán pocas y solo económicas, en casos de avería simplemente se enviará a arreglar.

Con respecto el software, es muy improbable que cambien las políticas por no decir imposible para algunos, como Python o Latex que tiene organizaciones sin ánimo de lucro detrás. En el caso de software comerciales deberemos buscar opciones gratuitas si las nuestras dejan de serlo, como son Zoom para la comunicación y Asana y EdoAgenda para la gestión y planificación. Nuestros datos para entrenar modelos y la plataforma para desarrollar algoritmos son proporcionados por nuestra colaboración con el proyecto base, aun así es posible que este sea abortado. En cuyo caso tendremos que mover nuestro sistema a un servicio como AWS, que provee recursos y servicios gratuitos para estudiantes [76], donde pondremos seguir realizando nuestro proyecto. Respecto a los datos, existen muchas formas de acceder gratuitamente a estos, como el Clef Health Lab que pone esa clase de términos etiquetados e informes clínicos a nuestro alcance.

8. Actualización de los costes temporales y económicos respecto al hito inicial

Originalmente este proyecto tenía que llegar a su fin el 14/01/20. Debido a factores ajenos al proyecto, la finalización de este se vio comprometida durante tres meses. El factor principal fue el tiempo de espera de casi tres meses que supuso él envío y reparación de nuestra herramienta de trabajo principal, situación altamente improbable pues se consideraba que a lo sumo una semana sería pérdida. En conjunción con lo anterior, el desarrollador principal se vio infectado de Covid-19 obligándolo a guardar dos semanas de descanso a mitades de la segunda iteración. Debido a esta situación resulta necesario explicar la replanificación y los cálculos de costes adicionales que fueron necesarios realizar.

8.1. Coste temporal

Respecto al número de horas de la planificación original, estas se mantienen así como su distribución en tareas pues el alargamiento del proyecto, como ya comentamos, se produjo debido a circunstancias ajenas al desarrollo de este, en otras palabras no supuso más trabajo en él.

Como vemos en el diagrama actualizado (15), pese a los retrasos, los tiempos de trabajo han sido respetados y, considerando el ritmo propuesto, calculamos que el proyecto finalizará el 2 de abril. Con este nuevo acercamiento mantenemos esas tres semanas extras que ya comentábamos, así como jornadas de 8 horas dejando dos días libres a la semana en caso de algún alargamiento puntual.

Seguimos considerando los riesgos anteriores así cómo los planes y tiempos propuestos para lidiar con ellos. Con la notable excepción de que ahora la posibilidad de que se vuelvan a producir los eventos que retrasaron el proyecto es ínfima, con el añadido de que en caso de avería contamos con un sustituto provisto por la nueva garantía ampliada del ordenador.

8.2. Coste económico

Los retrasos en este proyecto han acarreado una serie de costes añadidos. Concretamente, los relacionados con la gestión del TFG y del envío del portátil no cubierto por la garantía.

Concepto	**Coste (€)**
Gastos relacionados con la reparación y gestión de la garantía	50
Trámites para el segundo turno de lectura	90
Costes totales del retraso	140

Tabla 9: Gastos inesperados

Como podemos observar, tabla 9, estos costes están cubiertos por las contingencias consideradas en el presupuesto inicial (tabla 8), por lo que no se sufren alteraciones en este. Cierto es que en un proyecto empresarial un retraso de tres meses por el fallo de un equipo sería inadmisible.

Semejante situación generaría unos costes relacionados con la contratación del empleado por tres meses adicionales que superarían incluso los supuestos más remotos discutidos en el presupuesto. Debemos considerar, entonces, la hipótesis de un proyecto real, dónde sería preferible adquirir otro equipo.

Bajo la premisa anterior, el coste de adquirir un remplazo ascendería a 600€. Como podemos observar los costes seguirían estando cubiertos por el 10 % de contingencias, así como los dos o tres días de retraso se verían cubiertos por el riesgo de que se alargue una iteración algo más de la cuenta, considerado ya en el presupuesto. Volviendo a nuestro caso particular es un desembolso que no nos podíamos permitir, por lo que el retraso en el proyecto nos era más conveniente que tal suma de dinero.

8.3. Efectos sobre el desarrollo del proyecto

Como ya hemos comentado el proyecto se ha visto retrasado casi tres meses, pese a este retraso los objetivos del TFG no se han visto comprometidos. Ya la inscripción al segundo turno de presentación nos ha dotado del tiempo necesario para realizar todas las iteraciones planificadas. Por eso mismo consideramos que no hay necesidad de cambiar la metodología y rigor; pues seguimos creyendo que el desarrollo en cascada, tal y como lo planteamos entonces, sigue siendo la mejor manera de proceder.

En resumen los retrasos temporales por circunstancias externas supusieron un retraso de tres meses sin más costes adicionales que excedan los riesgos mencionados por el presupuesto o que supongan cambiar la modalidad de trabajo.

9. Identificación de leyes y regulaciones

Las primeras regulaciones a las que estamos sujetos son las licencias y derechos de autor de las tecnologías usadas. Principalmente estamos hablando de Elasticsearch y BERT, dos piezas de *software* que forman parte de nuestro proyecto pese a no haber sido desarrollados por nosotros. Dichos programas caen en la categoría de Free Software, ya que se encuentran bajo la *Apache License 2.0* [77] y *Elastic v2 License* [78], respectivamente. Ambas licencias ofrecen la posibilidad de usar y modificar a placer los productos a los que están sujetos.

Otra pieza de software que usaremos es Snomed-CT, esta Ontología se encuentra ligada a la licencia de Afiliados de IHTSDO [79], permitiéndonos usar sus diversos productos en nuestro sistema sin ningún coste. Obtuvimos dicha licencia al afiliarnos a la organización NLM (*National Library of Medicine*)[80], afiliada de IHTSDO y capacitada para otorgarnos acceso a Snomed-CT, además de a sus propios productos.

Pese a no ser una pieza de *Software* como tal, CIE-10 también está sujeto a una licencia. Más concretamente, hablamos de la licencia de uso de la página web del Gobierno de España, donde podemos encontrar todos los recursos de este estándar [63]. La licencia permite el uso parcial o total de los datos, sea este comercial o no [81]. También usaremos CIM-10 que se encuentra en la web de la Generalitat, y otorga a este la misma normativa comentada[62].

Por último, hemos de hablar de distintos recursos obtenidos a lo largo del desarrollo, la mayoría de ellos obtenidos durante la iteración de la gestión de abreviaciones médicas. Los datos obtenidos caen bajo la licencia de Creative Commons [82], que nos permite usar su trabajo para nuestro proyecto siempre y cuando los mencionemos, cosa que hacemos cuando usamos algún recurso externo. Otros provienen de la web del Gobierno Español y de la Generalitat, por lo que se aplican las mismas licencias que a CIE-10 y CIM-10.

También debemos tener en cuenta las regulaciones relacionadas con los datos que manejamos. Debido al carácter sensible de los documentos, existen un conjunto de medidas y restricciones que se deben respetar cuando estamos trabajando con ellos. Como desarrolladores tenemos un conjunto de responsabilidades que cumplir para con los datos sensibles que ahí se encuentran. No es de extrañar que hayamos tenido que firmar una declaración de buenas intenciones al principio de desarrollo del proyecto. A continuación destacamos los compromisos de confidencialidad que son los más susceptibles de ser incumplidos:

- **Datos anónimos:** En principio todos los documentos clínicos han sido anonimizados, pero cualquier posible identificación de un paciente usando estos debe ser notificado al SIDIAP de forma urgente e inmediata.
- **Uso responsable:** Solo podemos usar los datos en el proyecto acordado.
- **Transferibilidad:** No podemos distribuir a terceros estos datos, eso incluye moverlos del servidor donde se encuentran.

Como vemos, y veremos a lo largo de la documentación, los dos últimos puntos han sido respetados por todo el proyecto. Y, hasta la fecha, no se ha dado la necesidad de tener que reportar al SIDIAP hallazgos de identificación ni información personal. Por lo que todos los puntos han sido respetados.

10. Sostenibilidad y compromiso social

10.1. Autoevaluación

La realización de esta encuesta ha hecho que nos demos cuenta de que nuestro nivel de conocimiento sobre la sostenibilidad no es tan alto como nos pensábamos. Pese a que algunos de los conceptos nos eran conocidos, nos hemos visto limitados a la hora de aplicar algunos de estos o de usar métricas para medirlos, sin ya mencionar los muchos otros conceptos que claramente desconocíamos.

Pese a nuestras limitaciones, algunos de estos conceptos en materia de sostenibilidad social y medioambiental han sido aplicados en la elaboración de este proyecto, aparte de la obvia consideración de la sostenibilidad económica; pues ya entendíamos desde un inicio que para un desarrollo correcto y vida útil del proyecto no solo debíamos considerar la parte económica sino también el alcance y repercusión de este en la sociedad y el medioambiente. Esto anterior lo podemos observar en nuestras consideraciones sobre la pandemia actual y su repercusión sobre la salud mental de nuestro empleado o bien en la de arreglar el hardware estropeado pese a que renovarlo completamente pueda resultar en un ahorro de tiempo y dinero, con una peor repercusión medioambiental.

La realización de esta encuesta nos ha permitido darnos cuenta de las carencias en nuestros métodos, sobre todo en la medición de estos conceptos. Para poder elaborar un proyecto teniendo en cuenta la sostenibilidad en sus tres vértices se emplearan métricas estandarizadas que nos permitirán una percepción más objetiva cuando sea posible, pues hasta ahora la medición de estos se hacía de forma subjetiva. También nos informaremos de estos nuevos conceptos vistos en la encuesta, pues estos conocimientos pueden resultar vitales tanto a nosotros como a la sociedad y al medioambiente a largo plazo.

10.2. Dimensión ambiental

La dimensión ambiental de este proyecto ha sido considerada levemente, pues aunque hayamos tomado ciertas medidas; como la reparación de hardware, para evitar generar residuos electrónicos, o usar un espacio de Coworking, que permita un uso más óptimo de la electricidad; seguimos necesitando una estimación de otros factores importantes. Siendo el principal el impacto que puede tener el desarrollo y entreno de modelos en la plataforma del departamento de Ciencias de la Computación de la FIB. Esto puede ser una información compleja de obtener, pues requiere de saber detalles sobre la infraestructura; sobre otras clases de proyectos que podrían ocupar nuestro espacio, así como ver cuanto ahorraríamos o gastaríamos respecto a ellos, y nuestro consumo total usando esta plataforma. De lo que podemos estar seguro es de que la entidad que la gestiona es consciente del impacto medioambiental de esta, por lo que podemos suponer que se trata de una elección óptima en este aspecto.

Respecto a las medidas tomadas para la reducción del impacto medioambiental, la única variable a la que hemos tenido acceso y control es al tiempo de entreno y los nodos empleados para el desarrollo. Como veremos, la iteración 4 se basa precisamente en encontrar formas de reducir este consumo de tiempo y la posibilidad de entrenar en nodos menos potentes, de esta

manera generaremos un menor impacto ambiental. En caso de volver a realizarse el proyecto se habría requerido los mismos recursos, aunque al usar los resultados de la cuarta iteración habríamos gastado menos recursos a nivel temporal.

En comparación con el resto de soluciones, la nuestra no aporta nada nuevo pues se basa en estudios anteriores. Pero si que podría aportar mejoras aplicadas al proyecto base, ya que los métodos de aprendizaje autónomo (en lugar de otros con mayor coste computacional) pueden suponer un gasto menor en Kw/h, disminuyendo el impacto del proyecto base. Ya que es un estudio de un método para una solución posible, es difícil estimar que recursos usará la implementación real. De hecho la vida útil de este proyecto es la investigación en si, sin otros usos posteriores que la información obtenida, por lo que su impacto es lo ya discutido. Tampoco existen escenarios concretos en los cuáles este proyecto generé mayor huella ecológica, exceptuando que la implementación usada sea entrenada en nodos menos potentes que los empleados en la investigación. Lo cual podría derivar en mayores tiempo de entreno, aunque este factor ya se ha considerado, como ya hemos dicho, en la iteración 4.

10.3. Dimensión económica

La sostenibilidad económica es de lo tres aspectos que más hemos cuidado, pues ya estamos realizando una elaboración de un presupuesto realista, teniendo en cuenta riesgos y contingencias para que este sea sostenible en caso de dificultades, incluyendo planes de acción en caso de imprevistos de gran impacto. Que ha permitido al presupuesto mantenerse pese a los contratiempos discutidos. Cabe destacar que en dicho presupuesto los roles de los tres empleados han sido llevados a cabo por una sola persona lo que repercute en una reducción de dos sueldos reales. Considerando esto hemos tenido un ahorro total de 5269 € destinados a sueldos.

Esta solución aportaría mejoras económicas, como el resto de estudios de este problema, no solo por su aplicación al proyecto base sino porque estos métodos que desarrollaremos puede aplicarse a otras circunstancias, dónde pueden suponer una mejora económica importante como es la automatización de la catalogación de documentos clínicos.

Los coste temporales, que incrementan el coste de desarrollo, son algo ya considerado y discutido en la iteración 4, en caso de hallarse ante esta situación se puede emplear lo discutido allí. La única actualización requerida es el reentreno con mayor cantidad y calidad de datos, cuyos costes a nivel temporal (que es lo que puede incrementar los costes) se podrían reducir de la manera ya mentada. Pese a todo esto debemos recordar que el proyecto base es una investigación realizada en el marco de la UPC, los costes asociados a las máquinas ya se encuentran cubiertos.

El único riesgo real es la existencia de nuevos paradigmas que mejorasen la solución, lo cual implicaría una reimplantación total. Por desgracia no existe ninguna manera de prevenir esto, y es realmente cuestión de tiempo e investigación. La única manera de reducir este riesgo es usar los métodos más nuevos y esperar que sigan siendo relevantes durante cierto tiempo, cosa que ya hemos hecho.

10.4. Dimensión social

La realización de este proyecto me ayudará no solo dándome conocimientos técnicos, pues implica un grado de estudio sobre un sector del Procesamiento del Lenguaje Natural que desconocía, sino de rigor y forma para presentar mejor un proyecto como este. Como futuro ingeniero debo ser capaz no solo de aplicar un conocimiento sino de mejorarlo e incrementarlo de forma metódica. También debo ser capaz de usar habilidades más transversales para la realización de este trabajo, por lo que puliré mis capacidades de análisis, gestión e incluso expresión escrita y oral, los cuales repercutirán no solo en este proyecto sino en todos los posteriores que realice, así como en mi vida en general.

Durante la realización de este proyecto, he tenido que ser conscientes y actuar en consideración de la propiedad de los recursos usados así como la sensibilidad de los datos médicos empleados. He tomado consciencia de que la realización de este proyecto habría sido imposible sin ellos. Un mal uso de ellos por nuestra parte puede significar reducir la existencia y disponibilidad de estos en un futuro, pues fueron dados de buena fe y con la esperanza de que se respetasen sus circunstancias de uso. Por ello es de interés personal y colectivo tratarlos con la responsabilidad debida, para mantener un ecosistema fructífero de desarrollo para esta clase de métodos.

Como ya hemos discutido este proyecto formará parte de los muchos estudios hechos sobre nuestro problema. Por lo que ayudará a investigadores de toda clase, no solamente a los desarrolladores del proyecto base, también a pacientes y hospitales, pues las técnicas usadas aquí podrán ser aplicadas a soluciones diversas (como la clasificación de documentos y enriquecer historiales médicos para facilitar su entendimiento). Los pacientes son también unos de los grupos más gravemente afectados por nuestro proyecto, particularmente los cuáles han dado sus datos al SIDIAP. Si los desarrolladores no tratan estos datos éticamente la privacidad de estos se verá violada en detrimento de sus derechos más básicos. Es por ello que existen procedimientos y mecanismos para vigilar un correcto procedimiento con estos, hecho que minimiza en la medida de lo posible este problema.

Por último debemos considerar el escenario en la que esta investigación ayude a crear un método para automatizar las anotaciones médicas que resulte en la desaparición de los anotadores médicos. Aunque debemos considerar que, debido a la cantidad de documentos generados en la actualidad, las condiciones de trabajo de estos han empeorado considerablemente en los últimos años. Con la llegada de estos métodos la substitución de la mayoría de anotadores también resultará positiva para las condiciones de los que queden y revisen el trabajo generado. Que supondría un punto positivo, para un escenario en el que se eliminarían una gran cantidad de puestos de trabajo.

11. Primera iteración: Desarrollo del sistema inicial

Como ya hemos mencionado anteriormente nuestro método fundacional es el acercamiento por *Information Retrieval* local. Existen diversas implementaciones para este método, pero nosotros nos basaremos en el propuesto por el artículo *BERT-based Ranking for Biomedical Entity Normalization* [29], complementado mediante otras implementaciones [33, 34]. Este, a grandes rasgos, consiste en una búsqueda de entidades en una Base de Conocimiento mediante IR, usando la métrica BM25. Seguido de una reorganización de las lista de entidades devueltas mediante BERT. Como en este caso solo queremos una entidad, asignamos a la mención de una enfermedad la entidad que ocupe el primer puesto, osea que tenga mayor puntuación. El código asociado a esta es, por lo tanto, el dado a la mención. Podemos separar este método en dos sistemas que funcionan en serie, los cuáles explicaremos en mayor profundidad a continuación.

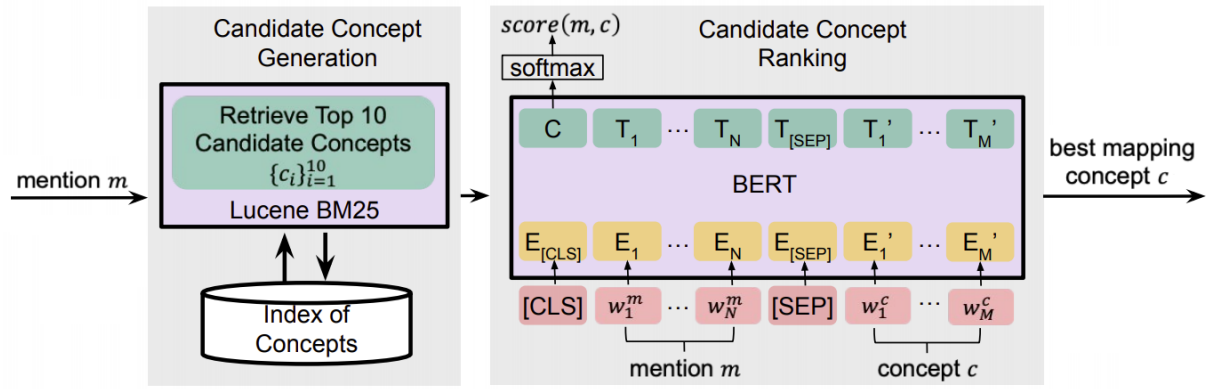


Figura 16: Esquema conceptual del sistema inicial

Fuente: https://arxiv.org/abs/1908.03548

11.1. Sistema de *Information Retrieval*

Como vemos en el esquema 16, para el primer sistema empleamos una base de datos desde la cual obtenemos los 10 conceptos más relevantes en base a la métrica BM25, siendo en nuestro caso los descriptores de códigos. A continuación explicaremos en que consiste más concretamente este sistema de IR, como se implementó y el porqué de las decisiones tomadas.

11.1.1. Funcionamiento de un sistema de *Information Retrieval*

Como sabemos el primer filtrado para elegir entidades potencialmente relevantes consiste en usar un sistema de *Information Retrieval*. Pero, para poder entender las decisiones tomadas sobre la implementación, debemos entender primero qué es y como funciona.

Los sistemas de IR nacen a principios de los años 60 pero no se popularizan hasta la era de Internet, cuando se hace más acuciante la necesidad de encontrar información relevante en un mar de datos cada vez más denso. Estas búsquedas tienen una serie de características que

las diferencian de las de una base de datos relacional, por lo que se requerirán algoritmos y estructuras de datos distintas. Dichas características consisten principalmente en que: no sabemos que buscamos, dónde se encuentra ni siquiera si existe esa información. Por estos motivos las búsquedas se realizan introduciendo un conjunto de términos, o palabras, con el objetivo de obtener resultados relevantes a estos. Se entiende, por lo tanto, que las respuestas van a contener la totalidad, o parte al menos, de la búsqueda presentada.

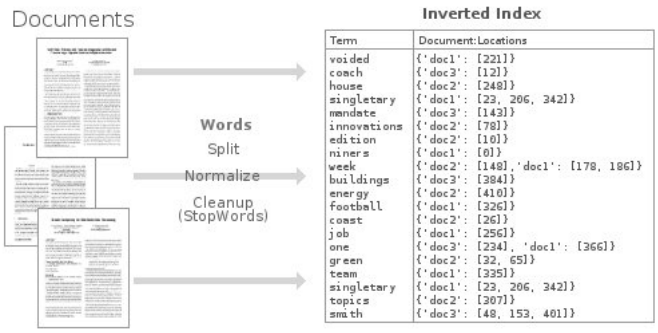


Figura 17: Ejemplo de *Inverted Index*

Fuente: https://nlp.stanford.edu/IR-book/html/htmledition/a-first-take-at-building-an-inverted-index-1.html

La base de todo sistema de IR es la estructura de datos llamada *Inverted Index*, observable en la figura 17. Como indica su nombre se trata de un índice de términos, en la que cada uno tiene asociado una lista de los documentos que lo contienen, además de otras informaciones como cuántas veces se repite en estos. Esta estructura es clave, pues permite un acceso rápido a una información que de otra manera sería muy costosa de calcular al momento de la búsqueda. Los términos encontrados en este índice son todos los existentes en los documentos disponibles, por lo que tratamos con un ente masivo. El proceso de generar un índice invertido se puede dividir en cuatro pasos [83, 84]:

- **Extraer estructuras de un documento:** Si el documento tiene estructuras como HTML o XML estas tienen que ser quitadas.

- ***Tokenization*:** Este paso consiste en procesar el texto y dividirlo en palabras que constituirán las entradas del índice. Para ello se emplean espacios en blanco y signos de puntuación para delimitar palabras. Posteriormente estas se pasan a minúsculas, facilitando así la búsqueda pues no la hace sensible a variaciones en estas. Y, por último, se quitan todas las palabras que se consideran poco significativas; preposiciones, determinantes o de uso muy común como ”ser” o ”estar”; con el objetivo de reducir el tamaño del índice facilitando su procesamiento y eliminando resultados muy triviales.

- **Enriquecer entradas:** Enriquecemos cada término para obtener mejores resultados: como añadirle sinónimos, palabras relacionadas o etiquetas gramaticales.

- ***Lemmatizing* o *Stemming*:** El objetivo de este paso es reducir los términos a sus raíces, lo cuál disminuye el riesgo de que algunas conjugaciones de palabras tengan más peso que otras y que afecte negativamente a los resultados. Existen dos formas de hacerlo: mediante *Lemmatizing*, lo que las convierte a su raíz real (lo que puede ser trivial para algunas pero

más complejo para otras), o bien mediante *Stemming*, una aproximación que quita sufijos y prefijos mediante un conjunto de reglas (menos costosa de ejecutar pero que rinde prácticamente igual [83]).

Una vez generadas las entradas del índice, ya podemos procesar cualquier *query* y devolver los documentos que se podrían considerar relevantes. Aquí entra la segunda parte de un sistema de IR: estimar la relevancia de los documentos. Cabe mencionar que antes de buscar sobre el índice invertido se procesa la *query* como el resto de documentos. Más concretamente se aplica el segundo y cuarto paso.

Podemos entender como relevantes todos los documentos que contengan los términos de la búsqueda, o un subconjunto de ellos. Lo que denominamos una métrica booleana. El problema con dicho tipo de similitud es que devuelve un conjunto de documentos que cumplen las condiciones. Dándonos una puntuación binaria que resulta poco útil si necesitamos algo más que documentos con ciertos términos. Cuando se requiere de calculo de puntuaciones más complejas es dónde entran TF-IDF o BM25.

Las métricas como TF-IDF, o su variante BM25, tienen como objetivo modular una similitud entre documentos y *queries* en base a las palabras que comparten y la importancia de estos términos dentro del documento en si y entre todos los demás. La idea es la siguiente: que una palabra que se repite mucho en un documento tendrá más importancia en este. Pero si esta tiene una frecuencia alta en el resto de ellos es menos discriminadora y, por lo tanto, debería tener menos peso. Bajo estas dos ideas podemos computar TF-IDF, para ello debemos representar cada documento como un vector de N dimensiones, dónde N es el tamaño del índice invertido y la i-essima posición indica el peso del i-essimo término del índice en ese documento. Cada peso se puede computar como la multiplicación de dos expresiones $TF_{d,i}$ e IDF_i que modelan respectivamente estas dos ideas que hemos comentado, podemos observar dichas formulas a continuación:

$$TF_{d,i} = \frac{frecuencia_{d,i}}{Max_j(frecuencia_{d,j})} \tag{1}$$

$$IDF_i = \log \frac{CD}{CDt_i} \tag{2}$$

$$Peso_{d,i} = IDF_i TF_{d,i} \tag{3}$$

$frecuencia_{d,i} =$ Frecuencia del término$_i$ en el documento$_d$
$Peso_{d,i} =$ Peso del término$_i$ en el documento$_d$
$CD =$ Cantidad documentos totales
$CD_i =$ Cantidad de documentos que contienen el término$_i$

Una vez expresados todos los documentos en un espacio vectorial de N-dimensiones, así como la *query* o bien con valores binarios o de la misma manera que los documentos. Podemos computar la similitud como una de coseno entre el vector del documento, D, y el vector de la *query*, Q, de la siguiente manera:

$$SIM(Q,D) = \frac{QD}{|Q||D|} = \frac{\sum_{i=1}^{N} Q_i D_i}{\sqrt{\sum_{i=1}^{N} Q_i^2}\sqrt{\sum_{i=1}^{N} D_i^2}} \tag{4}$$

Obteniendo una puntuación entre 0 y 1 que es mucho más manejable y fácil de entender como medida de relevancia.

La variación de TF-IDF, llamada BM25, opera bajo sus mismos principios, pero con dos añadidos nuevos[85]. El primero es que se considera la longitud del documento: si un término aparece una vez en documento muy corto es que ese término debe ser criticó para entenderlo, pero no pasa lo mismo si es uno muy extenso. El segundo punto que considera es como crece el peso de una palabra en función de la frecuencia. Si esta aparece 100 veces en un texto y 200 en otro, ¿es en este último el doble de importante? Es obvio que es más importante ¿pero cuánto más? En BM25 este factor crece más lentamente a mayor frecuencia, de esta manera limitamos que la saturación de un elemento contamine la métrica. Para conseguir esto se modifican las fórmulas originales de $TF_{d,i}$ e IDF_i, las cuales podemos observar a continuación:

$$TF_{d,i} = \frac{frecuencia_d, i}{Max_j(frecuencia_d, j)} \tag{5}$$

$$TF_{BM25_{d,i}} = \frac{TF_d, i}{TF_d, i + K(1 - B + B\frac{LD}{DML})} \tag{6}$$

$$IDF_{BM25_i} = \log \frac{CD - CDt_i + 0{,}5}{CDt_i + 0{,}5} \tag{7}$$

$K \in \mathbb{N}$
$B \in [0, 1]$
$frecuencia_{d,i}$ = Frecuencia del término$_i$ en el documento$_d$
LD = Longitud del documento
LMD = Longitud media del documento
CD = Cantidad documentos totales
CD_i = Cantidad de documentos que contienen el término$_i$

La razón por la que se cambia el IDF_i es para que palabras que se repitan en exceso no solo no cuenten nada, sino que resten a la similitud. Con el añadido del factor K[12] en $TF_{d,i}$ podemos controlar como crece el peso del término a mayor frecuencia. Y con la expresión $(1 - B + B\frac{LD}{DML})$, la cuál contiene B[13], controlamos como afecta la longitud del documento al peso del término. Una vez entendemos como funcionan estas métricas y el índice invertido podemos hacernos una idea básica de como funciona un sistema de *Information Retrieval*, y discutir nuestro acercamiento.

[12] A mayor valor de K, más rápido se estanca la frecuencia del término como factor en el peso.
[13] Que cuando vale 0 desactiva la importancia de la longitud.

11.1.2. Implementación mediante Elasticsearch

Como hemos observado en el apartado anterior un sistema de IR es bastante complejo de elaborar, pues aun con una idea básica de como funciona hay muchos problemas de implementación que no hemos abordado. Como la computación de las métricas de forma eficiente o la compresión del índice para que ocupe el menor espacio posible en RAM. Es importante este entendimiento de los conceptos pero creemos que es más útil no reinventar la rueda y centrarnos en la clasificación mediante BERT, por lo que usaremos una implementación de un sistema IR: Elasticsearch.

Elasticsearch es un servicio de IR basado en la implementación de Lucene[86], y la base de nuestro primer sistema. El cual se encuentra hospedado en un servidor privado solo accesible desde el mismo que contiene nuestros datos. Esto es debido a que usaremos la mención como búsqueda dentro de este sistema, y al tratarse de datos sensibles no pueden salir de este. De las muchas funcionalidades provistas por esta implementación, nosotros utilizaremos sólo las relacionadas con IR: la gestión del índice invertido y la computación de métricas de similitud, para la obtención de descriptores relevantes.

Inicialmente empleamos los descriptores oficiales de los códigos de CIE-10 y CIM-10 como los conceptos de la base de datos para la búsqueda, provistos por sus creadores respectivos[62, 63]. Más concretamente, usaremos todos los descriptores de códigos completos, subcódigos e infracódigos. La razón de esto es que si solo incluimos los asociados a códigos completos (que son los empleados para nuestra asignación)) estaríamos perdiendo información, pues algunos describen enfermedades que nos pueden resultar útiles. Por ejemplo, ”A48” tiene asociado ”Otras enfermedades bacterianas, no clasificadas en otra parte” que aporta poca información respecto a subcódigos como ”A48.1” cuyo descriptor es ”Enfermedad de los legionarios”. Además de ello, debido a que no existen ciertos tipos de identificadores en nuestros datos (como vimos en el estudio) el uso de sus descriptores dificultaría la obtención de entidades relevantes y, a la larga, una correcta asignación. Es por ello que decidimos descartar los descriptores de las familias de esos identificadores no presentes.

Como sabemos, para poder construir un índice invertido debemos procesar todos los documentos, así como la *query* para la búsqueda. Elasticsearch enfocada esto con unas abstracciones llamadas *Analyzers*, las cuáles se configuran con los procesos que queremos realizar. Al crear un índice asociamos a este nuestro *Analyzer*, de esta manera al insertar nuevos documentos, o hacer una búsqueda, estos procesos se aplican solos. Lo cual facilita el desarrollo y la prueba de nuevos procesos.

Inicialmente nuestra configuración consiste en dos índices, uno con los descriptores en catalán y otro con los castellanos, ambos sujetos a la misma configuración de *Analyzers*. El cuál tokeniza por espacios en blanco, elimina signos de puntuación, pasa todos los caracteres a minúscula, convierte las letras de UTF-8 a ASCII (llamado ASCII folding)[14] y aplica un Stemming dependiente del idioma (con una implementación basada en las especificaciones de *Snowball*). La razón por la que tenemos índices separados por idiomas es para poder usar *Stemming* si ninguna clase de repercusión negativa. Pues es dependiente del idioma y tenemos dicha infor-

[14]Nos permite eliminar así las letras problemáticas para el *Stemming*, ya que ASCII Folding transforma de Ç a C o Ñ a N.

mación sobre las anotaciones. Como estamos realizando pruebas sobre unos datos, y este no es un método final, consideramos positivo poder centrarnos en como rinde el segundo sistema (que es la base de la solución). Aunque si este fuera un método para el proyecto final y nos viésemos obligados a diferenciar los descriptores, lo haríamos mediante el uso de "googletrans 3.0", una API de gestión de idiomas dada por Google. La cuál permitiría detectar en que idioma se encuentra el texto y decidir sobre que índice realizar la búsqueda.

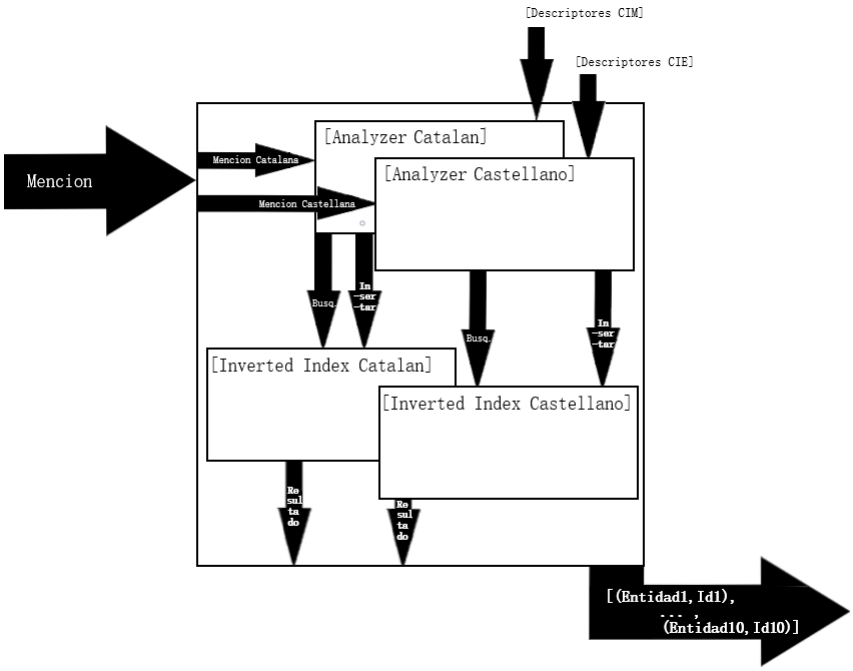


Figura 18: Esquema conceptual del sistema de Elasticsearch

Fuente: Propia

Usamos la métrica BM25, implementada por Lucene y provista por Elasticsearch, para obtener los 10 descriptores más relevantes para cada mención. La razón por la que usamos 10 es algo arbitraria: es la que todos los artículos con sistemas similares usan, por lo que nos parece un buen número para empezar. Si vemos que a lo largo del desarrollo es necesario ampliarlo para mejorar los resultados del sistema lo haremos. La razón por la que usamos BM25 como métrica es que la booleana no expresa tanto una similitud como que contiene ciertos términos, a diferencia de las otras métricas discutidas. También hemos considerado que, aunque en la mayoría de nuestros casos las ventajas de BM25 no sean significativas, no existe ninguna razón para no emplearlo. Pues no supone una desventaja, salvo un pequeño aumento temporal completamente despreciable.

Al acabar la búsqueda, devolvemos la lista de los descriptores y su código junto con la mención empleada para la búsqueda. Cabe destacar que devolvemos la versión original de todos estos textos, ya que esta clase de pre-procesos, aplicados por los *Analyzers*, son útiles para IR pero no para BERT. Podemos observar un esquema de este sistema primario en la figura 18.

11.2. Sistema de puntuación de BERT

Una vez discutido sobre el primer sistema de filtrado profundizaremos en el sistema de puntuación, este recibirá las listas de entidades junto con la mención y las puntuará respecto a como de probable es que esta entidad corresponda a la mención. La asignación de una entidad a una mención consiste en obtener la que tenga mayor puntuación. A continuación explicaremos en detalle que es BERT y como funciona en nuestro sistema.

BERT (o *Bidirectional Encoder Representations from Transformers*) es un nuevo modelo de representación de lenguajes propuesto por los laboratorios de Google. Este modelo se basa en varios principios de los que queremos destacar la transferencia de conocimiento [87]. Dicho concepto se basa en entrenar el modelo sobre un conjunto de documentos genéricos que permita capturar las estructuras intrínsecas del idioma en el que están escritos. Lo cual facilita realizar aprendizajes posteriores sobre tareas más especializadas. Estos dos pasos se conocen como pre-entreno y *fine-tuning*. Pero antes de discutirlos creemos necesario comentar la arquitectura de BERT, para poder entender como funciona concretamente así con las sutilezas de nuestra implementación.

11.2.1. Arquitectura de BERT

La arquitectura interna de BERT se compone por capas denominadas *Transformers*, o Transformadores en castellano. Intentar explicar como funciona exactamente un Transformador es un ejercicio que sobrepasa el abarque de nuestro proyecto. Ya que se trata de una tecnología que no necesitamos modificar de forma interna y bastante compleja de entender a la perfección. Es por ello que la explicaremos a grandes rasgos.

Los transformadores son una nueva tecnología que propone el uso de un mecanismo denominado atención. El cual permite procesar un texto de forma paralela, agilizando mucho su entreno respecto a sus contra-partes que procesan secuencialmente el texto. Además de suponer una mejora respecto a modelizar las dependencias de significado en textos largos, factor que resultaba crítico para estas otras estructuras[88].

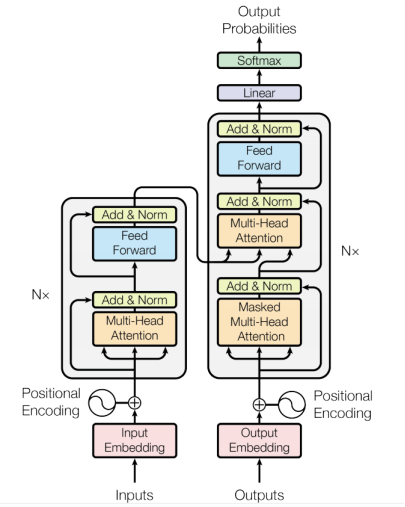


Figura 19: Esquema de la arquitectura de una transformador

Fuente: https://arxiv.org/pdf/1706.03762.pdf

Como hemos mencionado los transformadores conforman BERT. Pero con diferencias en la arquitectura propuesta por el artículo original [88], la cual podemos observar en la figura 19. Esta arquitectura está formada por dos bloques diferenciados, denominados *Encoder* y *Decoder*. Mientras que BERT solo está compuesto por capas de bloques de *Encoder* apilados, cuyo número de capas varía según el modelo, pues van desde 12 en la versión *Base* hasta 24 en la *Large*[87].

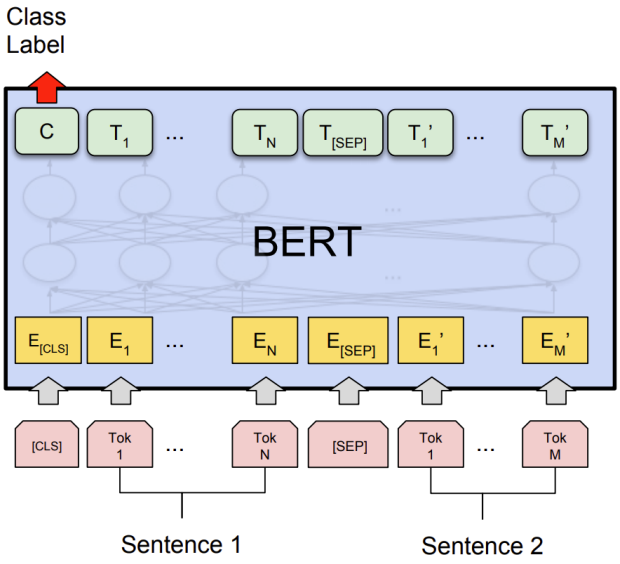


Figura 20: Esquema conceptual de la arquitectura de BERT

Fuente:https://arxiv.org/pdf/1706.03762.pdf

Las redes neuronales son unas construcciones que entienden números, vectores y matrices como entrada. Por eso a la hora de procesar estructuras de datos más complejas; como imágenes, sonidos o textos; se requieren de procesos para transformarlos sin perdidas de información, o

minimizándolas por lo menos. Los *word embeddings* cumplen dicha función en el procesamiento de texto, nos permiten codificar cada palabra a un vector de N dimensiones mucho más procesable. Este vector representa un punto en un espacio N dimensional dónde palabras con mismo significado, o pertenecientes a mismas categorías semánticas, son más cercanas. Podemos observar una representación de este espacio dimensional en la figura 21, perteneciente a un *word embedding* denominado *Word2Vec*.

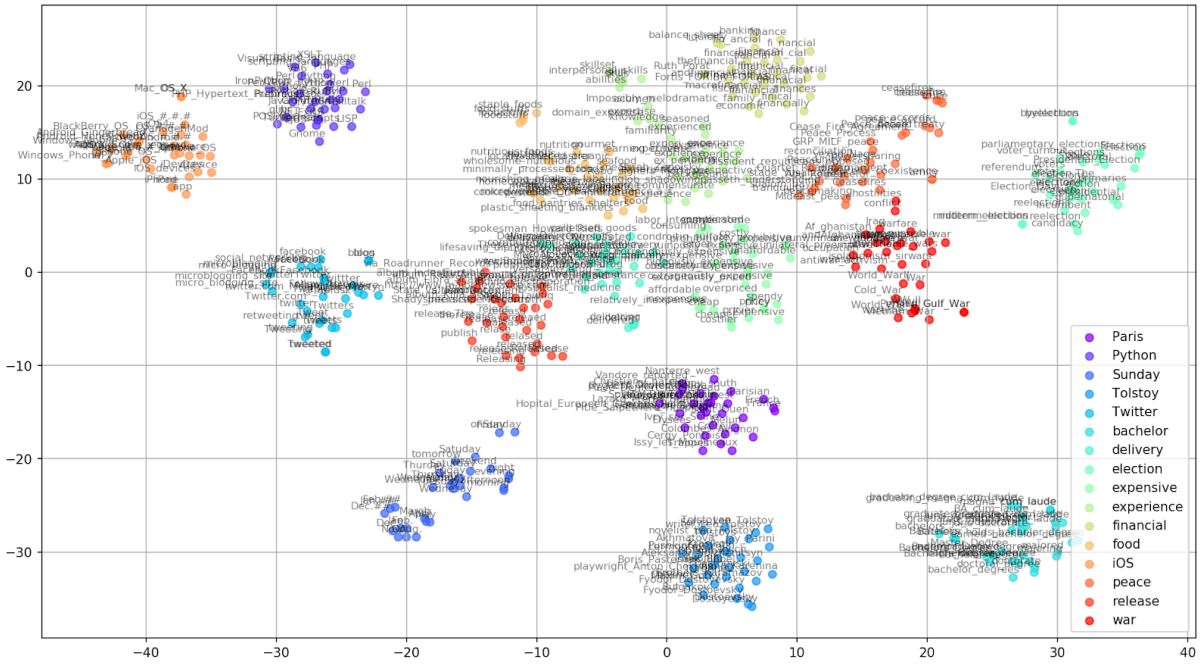


Figura 21: Visualización de Word2Vec

Fuente: https://towardsdatascience.com/google-news-and-leo-tolstoy-visualizing-word2vec-word-embeddings-with-t-sne-11558d8bd4d

Existe una variante, cuyo uso se ha popularizado durante los últimos años debido a las mejoras que supone, llamada *context embeddings*. Cuya idea es dar un *embedding* dependiente del contexto, ya que una palabra puede tener varios significados según este. Un ejemplo de ello es ”vino” en ”Un buen vaso de vino acompañaría bien a este queso.” o ” Vino y me dijo que no aceptaría tales condiciones.”[89]. Debido al uso extendido de estos en NLP existen diversos acercamientos a la hora de generarlos. Pero el resultado siempre es el mismo: una red neuronal que dada una palabra (junto con su contexto si necesario) devuelve su representación en forma de vector.

Mencionamos esto ya que la arquitectura de BERT, como cualquier modelo que procese texto, requiere de estos como entrada, pero con un añadido particular. Como vemos en la figura 22, los *embeddings* empleados por BERT están formados por la suma de tres. El primero, llamado *token embedding*, es simplemente un *context embeddings* al uso. En este caso se trata de Wordpiece[90], el cual da *embeddings* para palabras de forma particular: segmenta estas en raíces y derivativos a los que asigna posteriormente un *embedding*. Un ejemplo de ellos es la separación de ”playing” en ”play” e ”ing”, observable en la figura 22. Por ello decíamos anteriormente que BERT no requiere de pre-procesos, pues este *embedding* no funcionaría tan bien en textos procesados.

Los otros dos a los que se refieren como *positional embedding* y *segment embedding* le sirven a BERT para entender la estructura de los textos que le pasamos. El *positional embedding* sirve para indicarle a BERT la posición de cada *token*, necesaria a indicar ya que los transformadores no tienen dicha noción en su topología. Por último, el *segment embedding* simplemente sirve para indicar si el *token* pertenece a la primera frase o la segunda. Esto último es debido a que BERT puede procesar una o dos sentencias de entrada, para permitir diversas aplicaciones posteriores. Podemos observar también dos *tokens* en la figura anterior (22): "[CLS]"y "[SEP]" que sirven para indicar, respectivamente, el inicio de la entrada de texto y una separación entre la primera y la segunda sentencia.

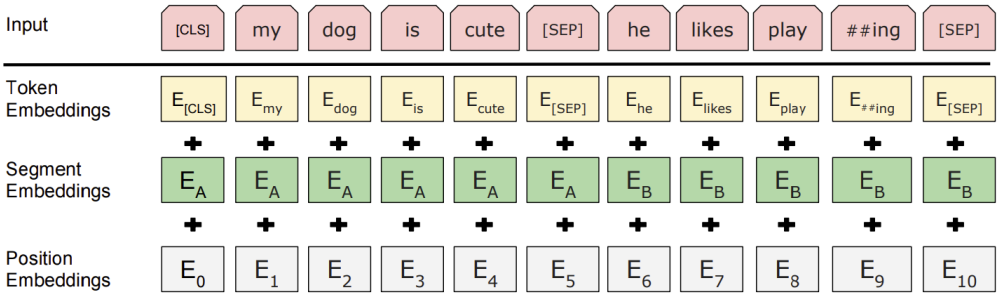


Figura 22: Embedding empleado por BERT

Fuente: https://arxiv.org/pdf/1810.04805.pdf

Respecto a las salidas de BERT podemos ver que, en la figura 20 sobre su arquitectura, se mantiene una para cada *token* de entrada, incluyendo los especiales "CLS" y 'SEP'. Dichas salidas tendrán las misma forma que los *embeddings* de entrada (un vector de N dimensiones) y son usadas en ambos pasos de pre-entreno y *fine-tuning*. De hecho son lo que permite a BERT ser tan polifacético, como explicaremos en las siguientes secciones.

11.2.2. Pre-entrenamiento

Una vez discutida la arquitectura de BERT debemos abordar la transferencia de conocimiento. Este concepto no es nuevo dentro del mundo del aprendizaje autónomo, pues ya existen acercamientos basados en este tanto en NLP [91, 92] como en visión por computador(VC) [93, 94]. Consiste, como ya mencionábamos, en realizar un primer entreno costoso en el que se aprende un conocimiento general empleado posteriormente en una tarea más concreta. En NLP, esto consiste en un entrenamiento inicial sobre un conjunto de documentos genéricos redactados en el idioma que nos interesa modelar. Para, al acabar dicho entreno, pasar el conocimiento, normalmente los pesos resultantes, a una red neuronal de arquitectura similar. La cual se entrena en una tarea más concreta, y que suele requerir de un menor número de texto y coste temporal. Esto funciona ya que lo que se requiere para resolver problemas de NLP consta de un "entendimiento" general del idioma y de una especificación propia del problema. Al eliminar la necesidad de obtener lo general por nuestra cuenta, se ahorra tiempo de desarrollo y permite invertir mayor esfuerzo en la parte especializada.

La manera en la que se realiza la obtención del conocimiento general con BERT es con un

entreno mediante dos tareas denominadas MLM (*Masked Language Model*) y NSP (*Next Sentence Prediction*), como podemos observar en la figura 23. Dichos entrenos son peculiares pues se tratan de un tipo denominados no supervisados. Estos se basan en procesar datos, documentos en nuestro caso, sin ninguna clase de información adicional en la que se intentan modelar ciertas características inherentes a estos. A diferencia de un entreno con datos etiquetados en los que se intenta modelar un conocimiento concreto, el de las etiquetas. Intentar explicarlos en profundidad esta fuera del abarque de este proyecto, por lo que nos limitaremos una visión superficial de estos.

Dichas tareas se realizan simultáneamente durante el pre-entreno e intentan modelar dos comportamientos distintos. La primera, MLM, consiste en resolver la palabra oculta de una frase, mediante un *token* especial denominado "[MASK]", consiguiendo que BERT modele la estructura del lenguaje. La segunda tarea, NSP, tiene como objetivo poder modelar las relaciones entre diversas frases. Para ello se generan pares de frases y BERT debe determinar si la primera realmente precede a la segunda[87]. Cabe destacar que para poder realizar dichos pre-entrenos deben ser procesados un conjunto concreto de salidas de *tokens* mediante redes neuronales[15] que permiten interpretar las respuestas de BERT. Para MLM, se procesan las salidas del *token* "[MASK]" para obtener que palabra genera BERT en cada frase. Y para la tarea NSP, se procesa la salida del token "[CLS]", la cual nos servirá como respuesta a la relación entre estas dos frases. Podemos observar en el esquema 23 como se realizan conjuntamente dichas tareas de entreno.

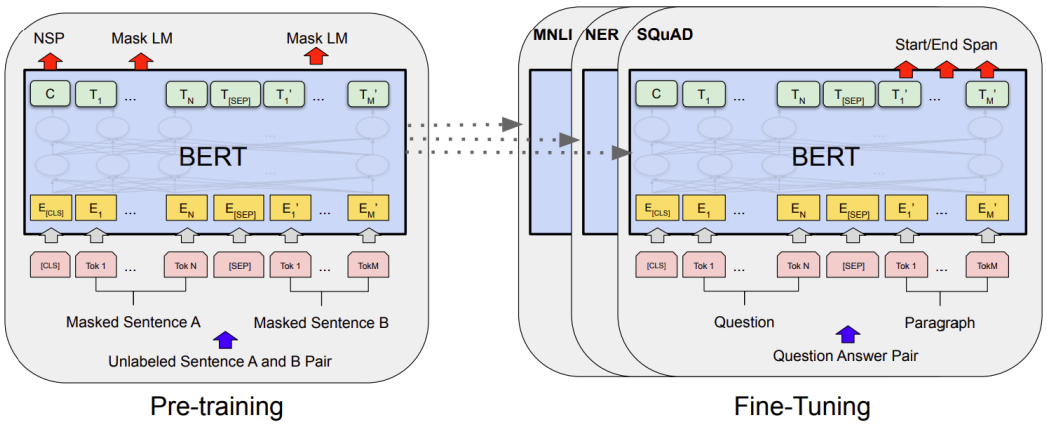


Figura 23: Pre-entreno y *fine-tuning* de BERT

Fuente: https://arxiv.org/pdf/1810.04805.pdf

Según los datos que utilicemos en este primer pre-entreno de BERT estará condicionado a un idioma u otro. Es por ello que los laboratorios de Google liberaron tres versiones[87]:

- **BERT original:** Entrenado sobre documentos en inglés, obtenidos de la Wikipedia y un *corpus* generalista.

[15] En este caso estas segundas estructuras son prescindibles, pues durante este pre-entreno lo que aprenden estas estará relacionado con la tarea concreta. Más que de modelar un entendimiento generalizado.

- **BERT chinese:** Entrenado exclusivamente sobre documentos chinos.
- **BERT Plurilingüe o MULTI-BERT:** Entrenado sobre los 100 idiomas más usados del mundo, entre ellos catalán y castellano.

Como los métodos y el código de BERT son completamente públicos existen también alternativas a las dadas por Google, pre-entrenadas sobre otros conjuntos de datos en terrenos más especializados y/o lenguas. El centrarse en un solo idioma y/o tipología de documentos permite un mejor rendimiento en las tareas objetivo que los empleen. Podemos hallar ejemplos de esto en bioBERT [95] o clinicalBERT[96], que se encuentran pre-entrenados en textos ingleses centrados en biología y medicina. O bien en BETO [97] y ALBERT [98] entrenados sobre documentos castellanos y catalanes, respectivamente.

Lo ideal para nuestra aplicación sería poder desarrollarla sobre un modelo pre-entrenado sobre documentos médicos similares a informes clínicos en ambos idiomas, pero por desgracia semejante modelo no existe. De hecho, no existe siquiera un modelo centrado exclusivamente en estas lenguas. Es por ello que emplearemos MULTI-BERT quién si las modela. Ya que elaborar un método que pueda procesar menciones en ambos idiomas es uno de los objetivos de este proyecto.

11.2.3. *Fine-tuning*

Como ya hemos mencionado BERT consta de dos fases de entreno, en este caso al usar MULTI-BERT tenemos un pre-entreno realizado sobre documentos de gran variedad de temas en 102 idiomas. Por lo que ahora nos centraremos en el *fine-tuning*.

Como ya sabremos, la tarea que debe realizar MULTI-BERT consiste en dado una mención y una entidad devolver una puntuación entre 0 y 1, la cual nos indica como de probable es que esta mención trate de esta entidad. Como vemos esta tarea guarda similitud con NSP. Con la diferencia que la métrica a devolver es semánticamente diferente. Por lo que la arquitectura usada para resolver nuestro problema es bastante similar a esta tarea: cogeremos la salida del *token* "[CLS]"[16] y la conectamos a una *feed-forward neural network* que procesará la información dada por este *embedding* de salida sobre la relación de dos sentencias. Y lo empleará para clasificar en dos clases, las cuales representan no o si a la pregunta planteada. La respuesta de este clasificador binario será procesado por una última capa Softmax que nos dará los resultados entre 0 y 1, más fácilmente entendible como una puntuación. Podemos observar este comportamiento en la figura dada inicialmente sobre el esquema conceptual del sistema (16).

Al tratarse de una clasificación binaria esta arquitectura será entrenada como tal, en este caso la función de pérdida empleada es la estándar "LOGLOSS" para clasificación. A continuación podemos ver como se definen el *score* y la función de pérdida:

[16]Cuyo uso ya estaba pensada para establecer métricas.

$$score(m, e) = softmax(CW^T) \tag{8}$$
$$LossFunction = \log\left(softmax(CW^T)\right) \tag{9}$$
$$\tag{10}$$

$W \in \mathbb{R}^{KXN}$, y representa los pesos del clasificador binario.
$C \in \mathbb{R}^N$, y representa el *embedding* sacado por la salida del *token* "[CLS]".
$K = 2$ En este caso K corresponde al número de clases del clasificador que procesa la salida.

Empleamos ADAM [99] como algoritmo para entrenar nuestra red neuronal, esta es una variante del descenso estocástico del gradiente de la función de pérdida o *Stochastic Gradient Descent* (SGD). Como sabemos una red neuronal se compone de neuronas que se conectan entre ellas mediante una intensidad regulada por pesos. Entrenarla consiste en hallar el valor de dichos parámetros de tal manera que la salida minimice la función de pérdida. Para hallar dicha configuración de pesos ideal empezamos con un conjunto de pesos aleatorios, en nuestro caso los pesos iniciales de BERT son los resultados del pre-entreno de MULTI-BERT, y exploramos estos mediante el gradiente de la función de pérdida avanzando siempre hacia los que la reducen.

Los hiper-parámetros usados, tanto para el entreno como la ejecución, son los propuestos originalmente por BERT para realizar este *fine-tuning*. Estos no sólo están recomendados por los diseñadores sino que lo realizan también la mayoría de investigaciones que seguimos [29, 33, 34]. Con la excepción de que se proponen diversos valores para *learning rate* (de 5e-5, 3e-5 o 2e-5), *batch size* (de 16 o 32) y *trainning epoch*(2, 3 o 4), aunque en esta primera iteración nosotros mantuvimos iguales las configuraciones originales. Más concretamente usamos un *learning rate* de 5e-5 , *batch size* de 32 y *trainning epoch* de 3.

Para la elaboración del método requerimos de un conjunto de datos para el entreno y ejecución. Para ello obtuvimos una partición de 85 % y 15 % de los datos. Dicha proporción es por cada tipo de código completo, de esta manera conseguimos una división total de 85 % y 15 % sin caer en la sobrerrepresentación (o subrepresentación) de algunos de ellos en una de las dos particiones. Cabe destacar que hicimos esta división separada por idiomas, de esta manera al juntar ambas particiones se mantiene el porcentaje ya mentado. Permitiendo además medir el rendimiento separado por idiomas y conjuntamente. Aunque el entrenamiento se hará sobre el conjunto bilingüe de datos, mientras que el *testing* se hará tanto bilingüe como por separado. La razón de esto último es porque no queremos entrenar un modelo por separado, solo queremos observar si nuestro método como tal tiene ciertas inclinaciones hacía uno u otro idioma.

Como sabemos, las menciones tienen asociadas sus identificadores, pero estos datos no nos sirven para entrenar esta solución. Para ello usamos el sistema IR y obtuvimos una lista de entidades relevantes y no-relevantes para esta mención, emulando el comportamiento de una ejecución. Más concretamente, obtendremos para cada mención 5 entidades que tengan el mismo código que esta y 5 que no lo compartan. Estas entidades serán escogidas mediante la métrica BM25, obteniendo así las 5 mejor puntuadas para ejemplos negativos y positivos.

Como ya hemos mencionado anteriormente, todo este sistema se encuentra en el servidor de HPC de rdlab, por lo que entrenaremos esta red en el *cluster* de nodos de este. Debido a que este servidor está compuesto por diversas tipologías de nodos debemos destacar cuáles hemos

usado y en qué cantidad. Debido al ingente tamaño de BERT, hablamos de 110M parámetros, muchos de los entrenos pueden tardar más de un día en realizarse. Es por ello que todos los *trainning* y *testing* han sido realizados sobre 8 CPUs medium con 16 GB de memoria. Estos números fueron elegidos tras un proceso heurístico de entreno, en el cual la red excedía las capacidades de RAM o el tiempo de ejecución. O bien el proceso moría por no poder reservar suficientes nodos para la tarea.

Una vez entrenado BERT, podemos ejecutar nuestro sistema. Una ejecución corriente consiste en tres pasos:

- **Sistema de *Information Retrieval*:** Primero se realiza una búsqueda de las entidades relevantes mediante ElasticSearch.
- **Sistema de *Scoring*:** Se pasa cada par (*mención*, *entidad*) y es puntuada por el sistema de BERT. Dónde el *score* es la probabilidad de que la entidad sea relevante a la mención.
- **Obtención de la entidad relevante:** Para cada mención obtenemos la entidad relevante, mediante el cómputo de la entidad con mayor puntuación dada por BERT. Asignando el identificador de esta a la mención.

‘

11.3. Métricas adicionales usadas durante el desarrollo

Durante el desarrollo de los sistemas mencionados, hallamos interesante medir el rendimiento de los dos sistemas independientemente. Por ello medimos de ElasticSearch el porcentaje de entidades relevantes devueltas por mención y el porcentaje de menciones que no consiguen al menos una entidad relevante. La idea es ver como funciona este primer sistema de búsqueda, pues si para una mención no devuelve ninguna entidad relevante poco puede hacer BERT. Y a cuantas más entidades relevantes se obtengan para una mención, mayor tolerancia a los fallos de BERT podremos tener.

También nos interesa medir como de bien puntúa BERT, por ello mediremos la *accuracy* de este a la hora de determinar si un par (*mención*, *entidad*) está relacionado o no.

11.4. Resultados de la primera iteración

Una vez hablado sobre como funciona nuestra implementación podemos discutir los primeros resultados de esta. Como observamos en la tabla 10, estos son mejorables.

Idioma	***Accuracy* total**	***Accuracy* BERT**	**Porcentaje entidades relevantes devueltas**	**Porcentaje menciones sin entidades relevantes**
Castellano	0,37,99	0,3167	27,34 %	51,62 %
Catalán	0,3513	0,3064	34,79 %	46,61 %
Total	0,3249	0,2038	31,03 %	40,11 %

Tabla 10: Primeros resultados del sistema

En general el rendimiento del sistema total es bajo, ya que solo conseguimos asignar de forma correcta algo menos de un 40 % de las menciones. Si vemos cómo rinden los sistemas separados, observamos que existen problemas tanto en el de ”filtrado” como en el de ”puntuación”. Aunque no es difícil pensar que los errores en este último pueden estar causados, principalmente, por los fallos en el inicial. Ya que casi la mitad de menciones no obtienen una entidad relevante y el porcentaje de entidades relevantes por mención es de algo más de un cuarto, unos números nada halagüeños. Creemos que es importante centrarnos primero en reducir al mínimo los errores que puedan propagarse del sistema de ElasticSearch. Ya que sino podríamos estar cayendo en la sobre-ingeniería de un problema de este. Es por esto último que al ver estos resultados lo primero en lo que pensamos, para la siguiente iteración, es como mejorar el sistema de IR.

12. Segunda iteración: Resolución de abreviaciones y ampliación del índice

Al acabar el desarrollo de la primera iteración decidimos que era necesario elegir una serie de mejoras sobre el sistema IR, antes de centrarnos en perfeccionar BERT. Ya que cualquier problema que se arrastre de ahí contamina el rendimiento de este. Considerando lo anterior, pensamos que para poder entender los fallos del sistema debemos comprender porque se producen en primer lugar. Con ese objetivo condujimos una pequeña revisión de las menciones afectadas por los fallos, más concretamente de las que no obtenían ninguna entidad relevante. De dicho análisis descubrimos dos grandes factores que explican el porqué de estos resultados. El primero es la poca precisión y falta de consideración de las variedades lingüísticas de los descriptores de CIE. Y el segundo, y mayor factor, es el uso de acrónimos y abreviaciones del ámbito médico en muchas de estas menciones. Lo cual dificulta mucho el poder obtener resultados relevantes. A continuación explicaremos, en mayor profundidad, cada uno de estos problemas así como la forma de lidiar con estos.

12.1. Ampliación del índice

El primer problema que afecta a las menciones es el hecho de que los descriptores dados oficialmente, tanto en la versión española como catalana, son muy genéricos. Nos referimos concretamente a dos situaciones. La primera es que existan varias formas de referencia a estos códigos completos; ya sea por uso de sinónimos, otras acepciones ortográficas o bien otra forma de describir esta entidad. Un ejemplo de esto anterior es lo que pasa con el código ”L25” con el descriptor ”Dermatitis de contacto de forma no especificada”, a la que nos podemos referir como ”Eczema de contacto de forma no especificada”, ”Eccema de contacto de forma no especificada” o ”Irritación de la piel debido a contacto no especificado”.

La segunda situación es la generalización de enfermedades. Existen categorías de dolencias que se agrupan bajo un código, por lo que su descriptor engloba semánticamente estas categorías, obviando los nombres concretos. Podemos ver un ejemplo de esto en los subcódigos ”A48.9” y ”A49.9”. Dónde sus descriptores son ”Otras enfermedades bacterianas especificadas” e ”Infección bacteriana, no especificada”, respectivamente. Como vemos estos subcódigos hacen referencia a otras enfermedades no especificadas en la categoría en la que se encuentran, en este ejemplo ”A48” y ”A49”.

Para solucionar estos problemas hemos decidido que lo mejor es ampliar la cantidad de descriptores que tenemos. Para ello realizaremos una ampliación de los descriptores originales de CIE y emplearemos Snomed-CT. A continuación desarrollaremos estas soluciones con mayor profundidad.

12.1.1. Descriptores más diversos

Esta primera parte de la solución es relativamente sencilla. Debemos conseguir mayores formas de expresar los descriptores originales de CIE. Los obtenidos de los documentos oficiales tienen carencias de sinónimos y divergencias comentadas anteriormente, es por ello que es vital

obtener formas alternativas. La mayoría de divergencias ortográficas así como algunos sinónimos y nombres concretos de enfermedades, de hecho, se consideran en los manuales originales[17] [100, 101]. Nuestro primer acercamiento, por lo tanto, fue emplear estas publicaciones para obtener dicha ampliación de descriptores.

El problema surge debido a que estos manuales están estructurados de forma muy compleja. Más pensados para facilitar las tareas de consulta a la hora de elegir un código concreto, que en ser un catálogo de definiciones. Con el añadido de encontrarse en formatos difíciles de procesar, como es el PDF. Fue por ello que modificamos nuestro acercamiento inicial, pues ya existían dichos catálogos procesados en estructuras más accesibles. Más concretamente, acabamos recurriendo a los sitios *web* oficiales [63, 62, 102]. Ya que ofrecen un servicio de búsqueda de códigos en las cuáles muchas de estas formas alternas se presentan, como vemos en la figura 24. Lo que resulta en que los buscadores son mucho más completos que los documentos estructurados dados por las *webs*, y empleados originalmente.

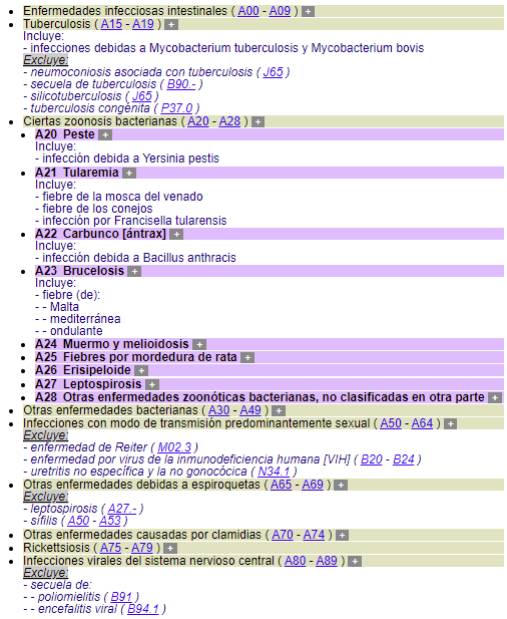


Figura 24: Estructura del buscador

Fuente: https://eciemaps.mscbs.gob.es/ecieMaps/browser/index_10_2008.html

Dichas estructuras son considerablemente grandes, hasta el punto de no poder ser tratadas manualmente sin generar errores y caer en un coste temporal considerable. También son relativamente fáciles de procesar, pues siguen una estructura parecida. Es por ello que nuestro acercamiento para recoger estos datos fue mediante una serie de *Crawlers*. Por *Crawler* entendemos un pequeño *script* que se dedica a acceder a esta información, y transfórmala a un formato de archivos procesables para indexarlos en nuestro sistema de IR. Dichos *scripts* fueron desarrollados mediante Python y la librería *beautiful soup*. Antes de usar los *Crawlers* decidimos que lo mejor sería verificar si se podía a acceder a dicha información mediante *bots*. Lo cual, según

[17] Debemos recordar que CIE es un estándar de codificación usado por profesionales médicos, por lo que en un principio se definió en manuales físicos.

los protocolos de exclusión de robots de los servidores, estaba permitido. También decidimos que lo más respetuoso con estos era realizar dichas recuperaciones de información durante la noche y festivos, interfiriendo en la menor medida posible con el uso humano.

Existen también estructuras de datos alternas a las que hemos usado[103]. Pero hallamos que al final todas bebían de los documentos oficiales. Por lo que fueron descartadas, ya que no aportaban nada nuevo.

12.1.2. Snomed-CT

Como comentamos en el contexto del problema, Snomed-CT es también una ontología médica que modela toda clase de entidades relacionadas con este mundo, no sólo enfermedades[6]. Una de las características que más nos interesan deriva precisamente de esto. Y es que modela muchas clases de relaciones entre conceptos, siendo de nuestro interés el de las diversas acepciones de un término (sinónimos). Esto nos permite solucionar el problema de las categorías genéricas comentado anteriormente. Ya que podemos tener referencias a enfermedades más concretas junto con diversas formas de referirse a ellas. Además de poder cubrir sinónimos no mencionados en los descriptores originales de CIE.

Para conseguir esto debemos hablar de las correspondencias entre Snomed-CT y CIE. Como sabemos estos cubren las mismas áreas relacionadas con enfermedades, con la diferencia de que Snomed-CT está enfocado a mapear más allá de estas. El problema surge con el uso que se les da a estos estándares. Ya que sus enfoques dispares producen distintas granularidades y distribuciones de códigos. Dónde Snomed-CT define enfermedades y circunstancias asociables por separado, CIE define un sólo código (modela enfermedad y contexto a la vez). Por estos motivos, aunque definan las mismas bases, no es trivial la transferencia de uno a otro. Pues no existe una correspondencia 1:1 entre los conceptos.

Pese a lo comentado, transferir los conceptos de Snomed-CT a CIE no es una tarea imposible. De hecho, existen diversos intentos, por parte de la comunidad médica de NLP, de conseguir un mapeo que sirva como estándar en esta tarea [104, 105, 106, 107]. De los existentes, hasta el día de hoy, solo hay uno disponible al público: el desarrollado por NLM (*National Library of Medicine*)[104]. Este mapeo nos ofrece para las entidades de enfermedades de Snomed-CT su correspondiente código en ICD. Como este mapeo depende de las circunstancias también vienen incluidas[18]. Esta transferencia de código es modelada mediante una tabla conformada por muchos datos, de los cuáles nos interesan sólo tres. Hablamos del identificador Snomed-CT original, el ICD resultante y la circunstancia (si es necesaria).

Como hemos mencionado este mapeo es uno inglés, pues está ofrecido por una entidad americana y es sobre ICD-10, pero esto no supone ningún problema. Para empezar los códigos de ICD-10 son trivialmente transferibles a CIE, pues cada identificador representa lo mismo. El único contratiempo es la necesidad de descartar algunos códigos que pertenecen a versiones más recientes. Pues no existe una forma trivial de encontrar una correspondencia entre ellos (como ya hemos comentado en el estudio de los datos). La otra dificultad a la que tenemos que

[18]Para un mismo identificador de Snomed-CT pueden corresponder diversos de CIE, según el contexto de este. Por lo que es necesario incluir bajo que circunstancias se aplica uno u otro código.

enfrentarnos es que el contexto está escrito en inglés y no podemos descartarlo. Si lo descartásemos tendríamos para múltiples identificadores un mismo descriptor. Por ello nos hallamos ante la necesidad de traducir estos contextos. Este no es un acercamiento extraño para algunas aplicaciones de NLP, dónde hay pocas estructuras de conocimiento en el idioma deseado. Con el añadido de que son contextos bastante simples, como localización (pulmón inferior derecho) o posible transmisión (contacto directo). Por lo que son fácilmente traducibles, con un riesgo mínimo de perder significado.

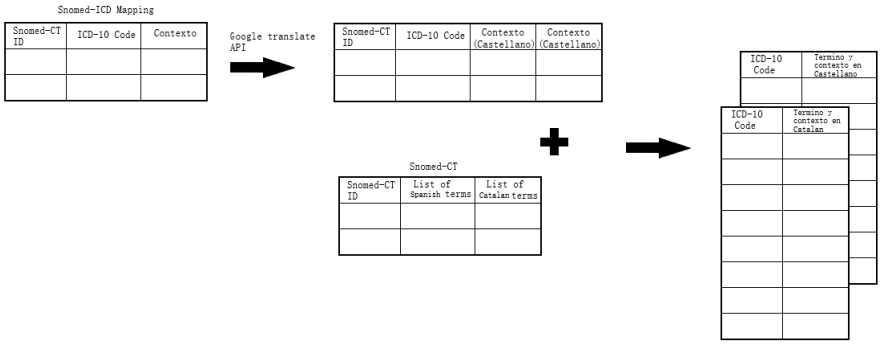


Figura 25: Procesamiento del mapeado de Snomed-CT

Fuente: Propia

Como podemos ver en la figura 25, esta estructura de mapeo es procesada mediante un *script* de Python. El cual obtiene las traducciones del contexto mediante la API de traducción de Google (”googletrans 3.0”). Y que genera una tabla en la que cada código Snomed-CT tiene asociado un identificador CIE junto a su contexto. Como recordaremos cada concepto de Snomed-CT tiene distintas acepciones, también conocidos como sinónimos. Por lo que, para cada concepto dentro de CIE, generamos las diversas acepciones juntando sinónimos con su contexto. El proceso de juntar los términos de Snomed-CT junto con el contexto es bastante sencillo, pues ya son dados para poder hacerlo. Podemos ver esto en el término ”Carcinoma pulmonar” con el contexto ”en el pulmón inferior izquierdo”. Cabe destacar que para obtener los términos, en ambos idiomas, usamos la base de datos Snomed-CT con su correspondiente versión en castellano y catalán, hechas accesibles por NLM [80] y la Generalitat [108].

12.2. Resolución de abreviaciones

Una vez ampliado la base descriptores, debemos pasar al siguiente paso: la resolución de abreviaciones. Éste es el segundo problema que detectamos sobre el primer sistema, así como la principal razón de que falle tanto. Pues las abreviaciones y acrónimos no suelen usarse en las descripciones formales, lo cual hace que calcular BM25 sobre ellos resulte inútil. Podemos ver esto último en la mención ”ciertos elementos indican presencia E. coli” que tendrá más en común con ”presencia de elementos ajenos al tracto digestivo” que ”infección por Escherichia coli”. Ya que comparte el término ”presencia” y ”elementos” con la primera, mientras que solo ”coli” con el segundo. A continuación explicaremos cuál es nuestro acercamiento para resolver este problema.

Para entender como tenemos que solucionar esta dificultad, debemos entender primero que con que clase de siglas tratamos en nuestros documentos. Como sabemos estos son de aspecto más informal, por lo que las abreviaciones no siguen una presentación como tal (no se definen la primera vez que se usan). Y se espera que el lector infiera su significado, en base al contexto y a la formación de esta.

Definir lo anterior es importante, ya que muchos de los estudios vistos tienen acercamientos diferentes según la naturaleza de las abreviaciones. Recordemos que en un texto formal las abreviaciones usadas deben ser definidas con su primer uso. Hecho que aprovechan dichos algoritmos para detectarlas y reconocerlas en usos posteriores [109, 110]. El acercamiento anterior solo funciona si se define en algún momento la abreviación, por lo que se requiere de otra solución debido al carácter informal de los informes clínicos. La manera de proceder en estos casos suele consistir en obtener las definiciones de alguna fuente externa, y aplicarlas mediante el uso de diccionarios y procesamiento de texto [29, 33]. Existen acercamientos como procesar textos formales con los algoritmos discutidos para obtener definiciones [32]. O entrenar una red neuronal para detectar y aplicar abreviaciones [111].

Obviamente entrenar toda una red neuronal se sale de nuestro alcance para este proyecto, pues requiere de un coste de desarrollo y entreno considerable. Por lo que nos pareció que la opción más sensata era conseguir elaborar un diccionario de definiciones de abreviaciones. Para obtener este diccionario exploramos diversas estructuras de datos mediante *Crawlers*, similares a los usados para obtener los descriptores adicionales. Hemos creído que lo mejor era obtener las definiciones de estructuras "públicas" que de textos formales, por tres razones. La primera es el tiempo de búsqueda de dichos textos, los cuales deberían cubrir la extensa variedad de temáticas existentes. La segunda es que no se garantiza la definición de ciertas abreviaciones. Pues muchas de las presentes en nuestros documentos tienen un carácter completamente informal difícilmente presentes en documentos de cierto rigor[19]. Y el tercer motivo es que las estructuras que exploramos son los diccionarios médicos usados por los anotadores. Lo cual nos da ciertas garantías, ya que estamos usando los mismos recursos que se emplearon para etiquetar nuestros datos de entreno.

Respecto a los datos usados para obtener dichas definiciones, hemos de mencionar la diversidad en la naturaleza de estos recursos de apoyo a la anotación. Que cuentan con libros digitalizados[112, 113], bases de datos públicas[114, 115] y buscadores[116]. También hemos obtenido datos de una *Shared Task* llamada BARR2. Cuyo objetivo era el desarrollo de un método para resolver abreviaciones en informes clínicos. Estos datos nos aportan las definiciones de los acrónimos presentes en documentos parecidos a los nuestros, por lo que son extremadamente beneficiosos. Cabe destacar que la cantidad de datos obtenidos para el catalán es más pequeño que en castellano, esto es debido a que las estructuras disponible en este primero son más escasas. Como suplemento a esta escasez algunas definiciones fueron traducidas. Estamos aprovechando el hecho de que el catalán y el castellano son lenguas que comparten las mismas raíces. Por lo que muchas palabras son parecidas, y muchas siglas de entidades como "Infarto de Miocardio"/"Infart de Miocardi" se abrevian a lo mismo: "IM". Con la ayuda de un *script* realizamos una traducción de las siglas que se mantienen en los dos idiomas, seguido de una revisión manual para asegurar que son correctas.

[19] Como varías formas de abreviar un nombre o empleo de abreviaciones no estándares. Las cuales permiten ser entendidas sin escribir toda la palabra. Un ejemplo de esto último es la abreviación "C" o "Ca" para cáncer.

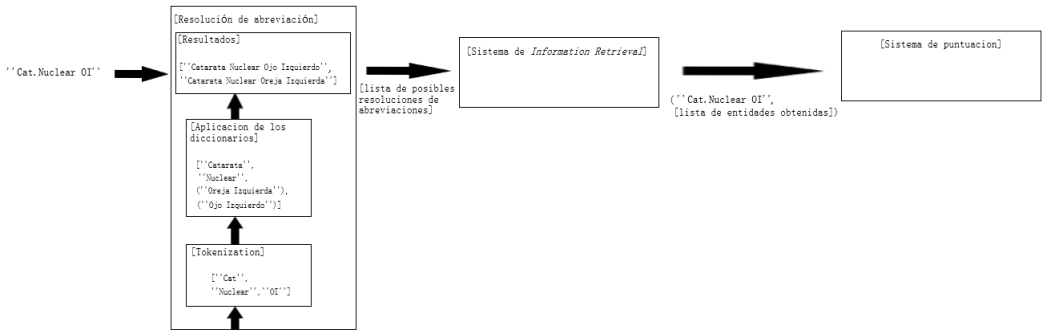


Figura 26: Sistema de resolución de abreviaciones

Fuente: Propia

Una vez obtenida las definiciones, las guardamos en un diccionario. El cual usamos para resolver los acrónimos en las menciones, antes de pasárselas al sistema IR. Usamos dos diccionarios de hecho, en el primero tendremos las siglas originales sin signos de puntuación como claves. Y en el segundo las claves son procesadas para que estén en minúsculas. En ambos la resolución de una sigla es un vector de las distintas definiciones (pues para algunas siglas pueden existir varías).

Para encontrar a abreviaciones, dividimos la mención en *tokens* mediante espacios en blanco y eliminación de puntuación. Esto es porque vimos que, muchas veces, las abreviaciones tendían a no respetar las puntuaciones. Por ejemplo: el uso de ”sdm.” o ”e. coli” es raro y suele emplearse la versión sin puntuación. Después de esto, miramos si para cada *token* existe alguna clave que se corresponda en el diccionario. Primero con el formato normal y después, sino obtiene nada, con el *token* en minúscula. La razón de esto es por la poca formalidad mencionada; es normal ver ”’sdm”, ”SDM” o ”Sdm” que representa ”síndrome”.

Idealmente tendríamos una sola definición por abreviación pero esto no es siempre así. Un ejemplo es ”OI” que puede significar ”Oreja Izquierda” u ”Ojo Izquierdo”. Aunque es cierto que ocurre en pocas ocasiones, es necesario desambiguar estas abreviaciones. Para ello generamos todas las posibles combinaciones de sustituciones de abreviaciones, y para cada una de ellas realizamos una búsqueda dentro de IR. De esta manera, le pasamos a BERT 10 entidades por posible definición[20]. Cabe destacar que pasaremos la mención original a BERT, sin la resolución de abreviaciones. Podemos observar este comportamiento descrito en la figura 26. También cabe destacar que la resolución se hará por idiomas, bajo las mismas premisas que la búsqueda separada por estos.

La idea detrás de pasar abreviaciones sin resolver a BERT es que el mecanismo elaborado solo ayude al sistema IR. Esperamos que, de esta manera, BERT desarrollé un entendimiento de las abreviaciones. Evitándonos tener que elaborar un sistema que las desambigue, y que pueda generar fallos que se propaguen a este. La desambiguación de abreviaciones se hará de forma indirecta al elegir la entidad relevante. Pues, con el entendimiento de dichas abreviaciones, BERT podrá determinar cuáles de las entidades devueltas por cada resolución tienen más sentido.

[20]Si tenemos N combinaciones posibles entonces N*10 entidades serán pasadas a BERT.

12.3. Resultados de la segunda iteración

Una vez aplicadas las mejoras mencionadas, decidimos primero ver como rendía el sistema de ampliación de descriptores por su cuenta. Nos resultaba trivial saber que la resolución de acrónimos, como mínimo, mejoraría el sistema IR en cierta medida. Pero con el aumento de descriptores existía la posibilidad de que se entorpeciese la ejecución.

Idioma	***Accuracy* total**	***Accuracy* BERT**	**Porcentaje entidades relevantes devueltas**	**Porcentaje menciones sin entidades relevantes**
Castellano	0,4745	0,4854	37,11 %	29,72 %
Catalán	0,4056	0,4506	36,27 %	32,03 %
Total	0,4404	0,4649	37,79 %	24,28 %

Tabla 11: Resultados del sistema con la base de datos extendida

Como indica la tabla 11, los resultados muestran una mejoría observable solo con el aumento de los descriptores. Ya que ahora la *accuracy* total y de BERT superan el 40 %. También mejoran las dos métricas que nos interesaban. Hemos conseguido reducir el porcentaje de menciones sin entidades relevantes, así como aumentar las proporciones de resultados relevantes para estas. Creemos que, por lo tanto, tan sólo con el añadido de descriptores existe ya un aumento en el rendimiento de Elasticsearch. El cual repercute positivamente en BERT y en el sistema total.

Idioma	***Accuracy* total**	***Accuracy* BERT**	**Porcentaje entidades relevantes devueltas**	**Porcentaje menciones sin entidades relevantes**
Castellano	0,5279	0,5049	40,10 %	11,31 %
Catalán	0,4188	0,5869	40,38 %	11,47 %
Total	0,4985	0,5266	40,95 %	11,42 %

Tabla 12: Resultados del sistema con la base de datos extendida y resolución de abreviaciones

En la tabla 12 vemos los resultados de nuestro sistemas con la ampliación de descriptores y la resolución de abreviaciones. Respecto a la mejora anterior, podemos observar como la resolución de abreviaciones consigue reducir aun más las menciones sin entidades relevantes y llegar a un 40 % de entidades relevantes por mención. La razón por la que esto supone un incremento mínimo en el sistema total es que el rendimiento de BERT pude ser critico para este. Por mucho que aportemos datos relevantes, si BERT no hace bien su trabajo el sistema general se halla limitado. De hecho, si nos fijamos en como de bien clasifica BERT vemos que lo hace sobre un 50 % de los pares (*mención, entidad*). Esta es una *accuracy* que parece indicar que BERT esta siendo aleatorio. Por lo que consideramos vital mejorar este sistema en la siguiente iteración. Por último, nos gustaría destacar como pese a la menor cantidad de datos disponibles en catalán, el método rinde de forma parecida al castellano. Creemos que esto es debido al menor número de variaciones en el uso de acrónimos y normas lingüísticas.

13. Tercera iteración: Mejora del entreno y rendimiento por profundidad

Una vez acabada la segunda iteración, y observando los resultados de dicha mejora, deducimos que era necesario perfeccionar el sistema BERT. Ya que la métrica del rendimiento de este parecía indicar una puntuación aleatoria del 50 %. A continuación, discutiremos en mayor profundidad dichas soluciones.

13.1. Mejora del entreno

Nuestro acercamiento al perfeccionamiento del entreno fue mediante tres mejoras. La primera fue el uso de los hiper-parámetros de entreno usados en la investigación de "*BERT-based Ranking for Biomedical Entity Normalization*" [29], pues empleaba conjuntos de datos parecidos a los nuestros con excelentes resultados. Además, estos números no se alejan tanto de los propuestos por BERT y se encuentran dentro de los valores recomendados por la literatura actual. Más concretamente usamos: un *learning rate* de 2e-5, *batch size* de 32, y *trainning epochs* de 5.

La segunda mejora al entreno fue el uso de la profundidad en los ejemplos negativos. Como veremos en la sección de resultados, el rendimiento de nuestra solución tiende a mejorar si consideramos menores profundidades. Demostrando de esta manera que los códigos asignados erróneamente tienden a mantenerse dentro de un mismo capítulo o subcapítulo. Por ello creímos interesante en los ejemplos negativos discernir bien entre entidades que estén dentro de estos. Por lo que empleamos dichas profundidades como balance, si hallamos entidades negativas con misma relevancia de BM25. Dando preferencia a entidades dentro del mismo capítulo que la mención, y mayor aun si se encuentra dentro del mismo subcapítulo. La razón por la que empleamos esto como carácter discernidor es que seguimos considerando la métrica BM25 importante. Pero existía un comportamiento en la generación de los ejemplos de entreno, por el cual ciertos ejemplos negativos se quedaban fuera del conjunto. Pues estaban limitados por tener la misma puntuación de relevancia que otros, pese a tener este potencial ya discutido.

La tercera, y última mejora, fue el paso de las abreviaciones al sistema BERT. Debido a que esto no sólo afecta al entreno sino que supone un cambio respecto al la arquitectura del sistema total, lo desarrollaremos en la siguiente sección.

13.2. Paso de abreviaciones a BERT

Una vez mejorado la generación de datos para el entreno, así como los hiper-parámetros usados para este, decidimos que el siguiente curso de acción era la gestión de abreviaciones por parte de BERT. Recordemos que, cuando instauramos la resolución de abreviaciones, era parte de un sistema que ayudaba al Elasticsearch a encontrar entidades relevantes que pasarle a BERT. Originalmente, estas abreviaciones se resolvían de diversas maneras para IR, pero una vez obtenidas las entidades para cada resolución los pares eran pasados con la mención sin resolver. La razón de esto era poder hacer que BERT "aprendiese" a diferenciar las resoluciones por su cuenta.

Como hemos podido observar, los resultados de esto dejan que desear. La tarea dada a BERT es suficientemente compleja de por si, como para que pueda ”aprender” las diversas resoluciones y usarlas correctamente. Por lo que hallamos necesario pasar dichas resoluciones para facilitar la tarea de selección de entidades. Para ello modificamos el sistema original (26) con uno nuevo, observable en la figura 27. Como vemos ahora, los pares (*mención, entidad*) son pasados junto con la mención resuelta usada en el sistema de IR. Por lo que ahora el objetivo de BERT es encontrar la entidad correcta ayudado por las resoluciones correspondientes. Por lo que de forma indirecta también se confirma dicha resolución. Ya que elegir una entidad implica una mención, con su correspondiente resolución.

La idea aquí es que, por el contexto de toda la mención, BERT establezca menores puntuaciones en general a las entidades obtenidas de menciones mal resueltas. Por ejemplo ”catarata de oreja izquierda” nunca debería ser bien puntuada con ninguna de sus entidades, pues es una resolución incorrecta. Este no es un aprendizaje demasiado complejo, pues son construcciones extrañas que BERT no estará acostumbrado a ver. A diferencia de las resoluciones correctas como ”catarata de ojo izquierdo”, a la que puntuará mejor.

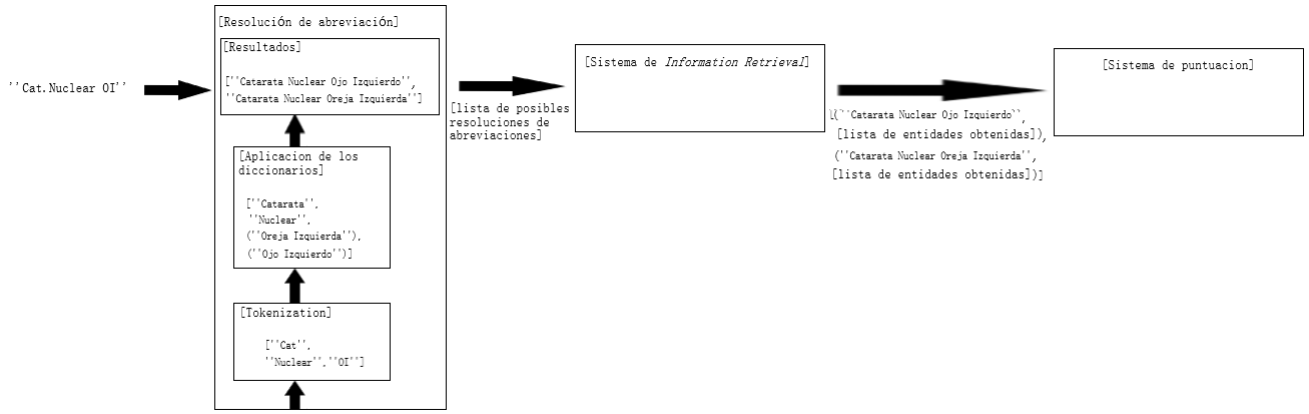


Figura 27: Sistema de resolución de abreviaciones actualizado

Fuente: Propia

Una vez elaborado este sistema debemos discutir como entrenaremos BERT al respecto, pues ya no podemos pasarle las menciones sin resolver. Para ello debimos resolver las presentes en las menciones de entreno; usando guías de anotación, los manuales de CIE y nuestro sistema de resolución de siglas. Más concretamente, empleamos el sistema de diccionarios sobre cada mención del conjunto de entreno. Si este encontraba diversas acepciones para una única abreviación entonces el sistema requería elegir una manualmente. Esta resolución manual era realizada por nosotros mediante el uso de la literatura médica ya comentada. Otro efecto positivo de esto es que permite mayor obtención de entidades relevantes en el entreno, ya que BM25 rinde peor con las abreviaciones. Por lo que ahora las entidades positivas serán realmente más relevantes y las negativas más parecidas a estas. Lo cual implica un mejor entreno.

13.3. Resultados de la tercera iteración

Una vez elaborados, podemos observar los resultados de la mejoras del entreno y del paso de la resolución de abreviaciones a BERT. Lo primero que cabe destacar es que no hemos incluido los resultados del primer sistema, ya que estos se mantienen al no haberse alterado nada de él. Como vemos en la tabla 13, el rendimiento del método general ha mejorado muy positivamente junto al del segundo sistema.

Idioma	***Accuracy* total**	***Accuracy* BERT**
Castellano	0,6477	0,8235
Catalán	0,6180	0,8305
Total	0,6374	0,8201

Tabla 13: Resultados del sistema con las mejoras del sistema de abreviación y entreno

Estos resultados son esperanzadores pero, aunque positivos, un 60 % de *accuracy* es un rendimiento bajo. Por lo tanto decidimos que sería de interés obtener el rendimiento con otros niveles de especificidad, esto es reduciendo los dígitos de profundidad. Realizando la asignación a nivel de subcapítulo y capítulo, en lugar de emplear los códigos completos.

Para poder experimentar con profundidades menores tuvimos que realizar una serie de tareas. La primera fue modificar el índice de Elasticsearch para poder realizar búsquedas en las otras profundidades. Esto resultó en que ahora cada entidad tiene asociado no sólo su códigos de 3 dígitos sino también los capítulos y subcapítulos. Una vez realizada estas, obtuvimos las divisiones de los datos en porciones de 85 % y 15 % para cada profundidad. Estos son generados de la misma manera que hicimos para los códigos completos en esta iteración. Cabe destacar que la mejora del entreno relacionado con las entidades negativas solo es aplicada en la profundidad de subcapítulo, y solo referentes al capítulo. Ya que no existe un concepto más general que capítulo. A continuación, en las tablas 14 y 15, observamos los resultados de dichos experimentos.

Idioma	***Accuracy* total**	***Accuracy* BERT**	**Porcentaje entidades relevantes devueltas**	**Porcentaje menciones sin entidades relevantes**
Castellano	0,7060	0,7908	48 %	11,31 %
Catalán	0,6631	0,7883	47,30 %	11,47 %
Total	0,6880	0,7874	47,92 %	11,42 %

Tabla 14: Resultados del sistema de la tercera iteración a nivel de subcapítulos

En el rendimiento a nivel de subcapítulo (tabla 14) podemos observar una leve mejoría en todos los aspectos, aunque se mantiene el porcentaje de menciones sin entidades relevantes, comportamiento que también podremos observar en la tabla 15, sobre los rendimientos del método a nivel de capítulo. Creemos que es un comportamiento interesante, pues nos muestra que nuestro problema con dichas menciones es debido al carácter de estas junto a las concesiones generadas durante el desarrollo.

Estas mejoras del sistema de subcapítulos son mínimas comparadas con las del método por capítulos. Dónde podemos observar una mejoría enorme, de casi un 90 % de *accuracy* total y del sistema BERT. Esta mejora sustancial nos permitió pensar en una posible idea para mejorar el método entero y obtener un rendimiento similar sobre la profundidad de tres dígitos. Que discutiremos en la cuarta y última iteración.

Idioma	***Accuracy* total**	***Accuracy* BERT**	**Porcentaje entidades relevantes devueltas**	**Porcentaje menciones sin entidades relevantes**
Castellano	0,8852	0,8909	62,26 %	11,31 %
Catalán	0,8523	0,8814	59,42 %	11,47 %
Total	0,8710	0,8775	61,49 %	11,42 %

Tabla 15: Resultados del sistema de la tercera iteración a nivel de capítulos

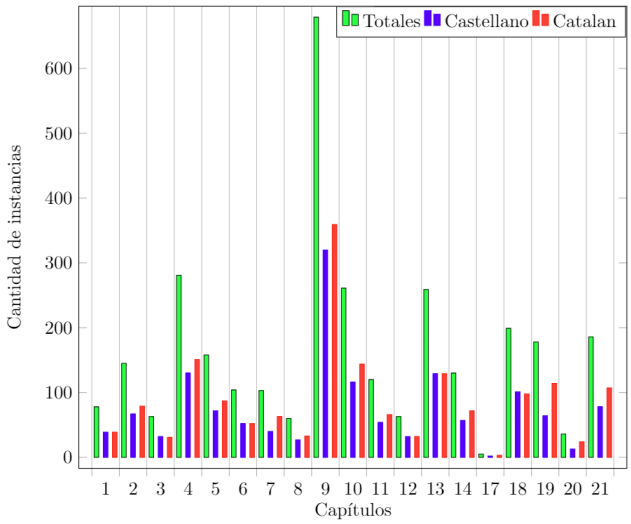
14. Cuarta iteración: Experimentación adicional

Una vez realizada la tercera iteración observamos dos aspectos importantes en el desarrollo de nuestro sistema. La primera es, como vimos en los resultados anteriores, la considerable mejora de los sistemas al emplear subcapítulos y capítulos. Aunque esperable, los resultados obtenidos por capítulos nos hacen ver un método desde el cual poder trabajar. Ya que los resultados de los otros niveles son relativamente bajos, y un rendimiento tan alto nos permite esbozar una posible arquitectura. La cual discutiremos en mayor detalle en la sección de futuros trabajos(14.4). El segundo factor que vimos, fue el incremento de recursos a usar a cada mejora de las iteraciones. Lo cual resulta en un aumento de los tiempos de entreno de nuestra solución. Debido a esto, para esta cuarta iteración hallamos interesante experimentar con reducir las capas de Transformadores, con el objetivo de un menor coste de recursos. Esta experimentación se realizará sobre la asignación a nivel de capítulo, ya que este es el modelo con mayor rendimiento. Y existe la posibilidad de que se reduzca la *accuracy* a mayor número de capas eliminadas. Por lo que creemos que nos permitirá conseguir un balance entre un rendimiento aceptable y un menor coste temporal.

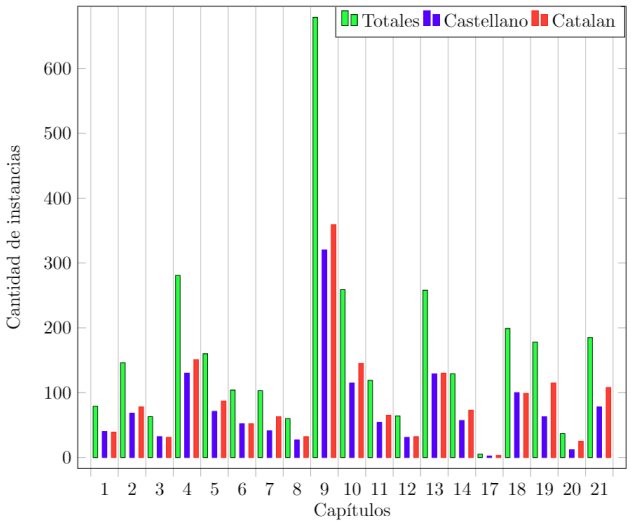
14.1. Particiones de entreno y validación

Antes de realizar los cambios en la topología de BERT, creímos conveniente el uso de *K-fold cross-validation* para la realización de esta experimentación. La razón de ello es que inicialmente realizamos una partición del 85 %/ 15 % para entrenar y validar el método. Este acercamiento nos resultaba válido en un principio, ya que permitía un desarrollo ágil. Pero estas concesiones pueden darnos una concepción errónea de como esta solución puede rendir en datos fuera de los manejados. Podría darse el caso de que nuestra red haya hecho un *overfitting* sobre los datos de entreno. O bien que al generar la partición hayamos escogido, de forma inadvertida, los mejores ejemplos. Esto se hace aun más acuciante con los capítulos, pues como vimos en el estudio de los datos (10), ya existen grandes desequilibrios a este nivel.

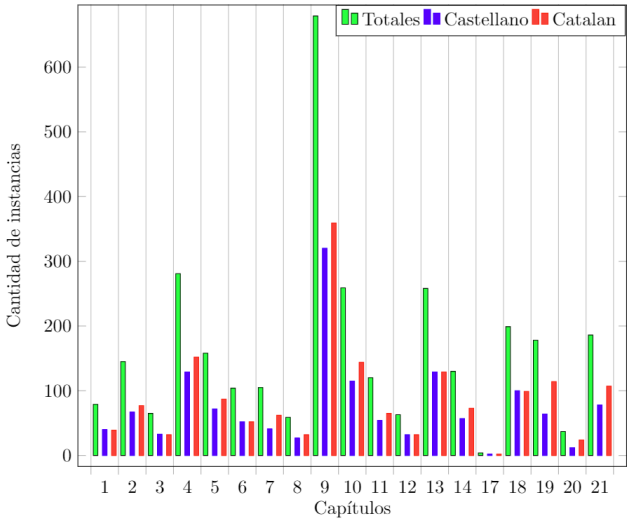
En nuestro caso, la implementación de *K-fold cross-validation* emplea una K de valor 5. Por lo que generamos 5 particiones iguales, por capítulo, de los datos. Para cada una de las iteraciones entrenamos sobre cuatro particiones de estas y la quinta es usada para la validación. Podemos observar la distribución de estas en la figura 28. Como podemos ver, existen capítulos que han sido descartados, como el 15 y 16, debido a que no tienen suficientes instancias para dar una a cada partición. Debemos recordar que estas divisiones son las de las menciones, por lo que debemos realizar la obtención de entidades negativas y positivas así como la resolución de abreviaciones, tal y como hicimos en la tercera iteración. Esto se hizo para todas los datos y se emplean dichas resoluciones cuando se requiera entrenar con dichas particiones. La razón para usar un K valor de 5 es por dos motivos. El primero es que es, junto con 10, uno de los valores recomendados por la literatura. Y el segundo es por cuestiones temporales. Ya que careceríamos del tiempo para realizar todas las pruebas si K fuera 10.



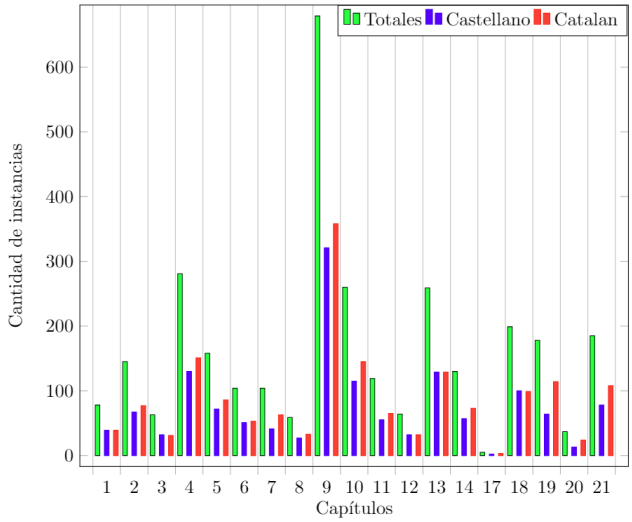
(a) Distribución *fold* 1



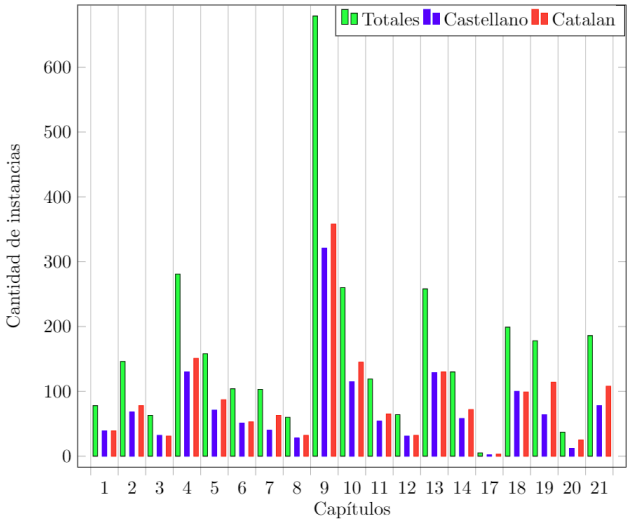
(b) Distribución *fold* 2



(c) Distribución *fold* 3



(d) Distribución *fold* 4



(e) Distribución *fold* 5

Figura 28: Distribución por capítulos de las particiones

Fuente: Propia

14.2. *BERT Pruning*

A lo largo del desarrollo, y a cada mejora propuesta por la iteración, el tiempo de entreno del sistema fue incrementándose hasta llegar a un entreno de 3 días y 20 horas. Es por ello que hallamos interesante experimentar con un concepto denominado poda, o *pruning* en inglés. El cual consiste en podar la red neuronal pre-entrenada intentando mantener el rendimiento en la tarea específica. La razón por la que esta idea funciona es que se basa en el principio de que BERT se encuentra sobreparametrizado. Lo que resulta en información redundante, permitiendo reducir el tamaño de este sin perjudicar gravemente el rendimiento. Y reduciendo en gran medida los costes temporales y de *hardware*.

Más concretamente este consiste en reducir las capas de transformadores de BERT antes del *fine-tuning*. Por lo tanto, la información redundante que intentamos eliminar son las modeladas por estas capas. Diversas investigaciones sobre la topología de BERT han relevado que los transformadores de este modelan a cada nivel un conjunto de conocimientos distintos[117, 118, 119]. Se sabe que las primeras capas modelan tareas de bajo nivel, como identificar la morfología de las palabras. Y tareas de mayor nivel se hallan a mayor profundidad[118]. También se ha demostrado que las capas más externas están más inclinadas a la función objetivo del pre-entreno que a un entendimiento general del lenguaje[119].

Nuestro acercamiento para esta tarea se basa en el artículo "Poor Man's BERT: Smaller and Faster Transformer Models "[120], el cual realiza diversas podas en tamaño y forma. Obteniendo los mejores resultados en la poda de las capas más exteriores, manteniendo un 98 % del rendimiento original. Nosotros realizaremos 3 podas de las capas externas, para ilustrar la evolución del rendimiento a menor capas usadas. Más concretamente realizaremos podas obteniendo 8, 6 y 4 capas. La razón de estos números es que 8 y 6 son las mayores realizadas por ese estudio. Además de que las 6 capas son reminiscentes de la arquitectura original de estos transformadores[88]. Respecto a reducir a 4 capas, creemos que puede ser interesante ver como rinde el sistema con esta poda llevada al extremo.

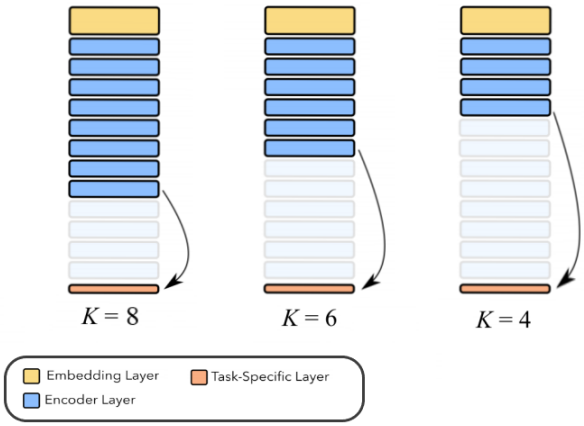


Figura 29: Esquema conceptual de la poda de BERT

Fuente: https://hsajjad.github.io/publication/sajjad-poormanbert-arxiv/sajjad-poormanbert-arxiv.pdf

14.3. Resultados de la cuarta iteración

Una vez realizada las modificaciones necesarias obtuvimos los resultados de cada partición. A continuación podemos observar en las tablas 16 y 17, la media de las *Accuracies* de cada poda así como sus respectivas desviaciones estándar.

Número de capas	**Media *accuracy* total**	**Desviación estándar**
12 capas	0,8730	0,005
8 capas	0,8716	0,005
6 capas	0,8711	0,010
4 capas	0,8641	0,005

Tabla 16: Resultados de la asignación por capítulos para las diversas podas en los datos totales

Podemos ver en las figuras 30a y 30b un decrecimiento lineal sobre los costes temporales de entreno y ejecución a mayor número de capas retiradas. Sorprendentemente durante la poda, la *accuracy* experimenta un leve descenso apenas perceptible. Este fenómeno también se da cuando observamos los resultados de datos partidos por idiomas. Los cuales mantienen, aproximadamente, los mismos niveles de *accuracy*.

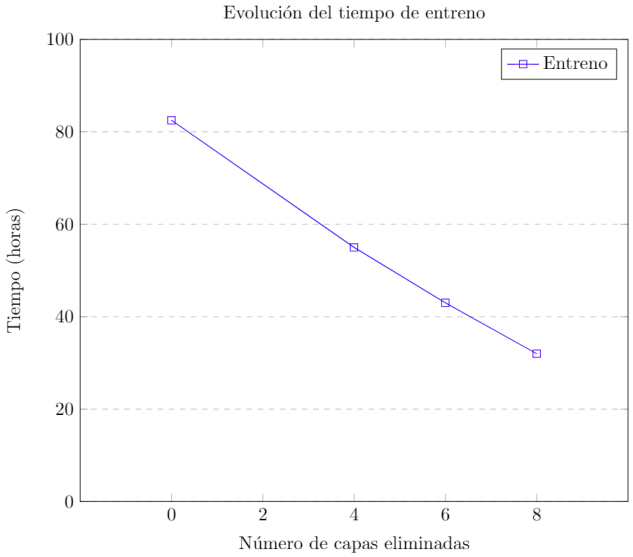
Número de capas	**Media *accuracy* castellano**	**Desviación estándar *accuracy* castellano**	**Media *accuracy* catalán**	**Desviación estándar *accuracy* catalán**
12 capas	0,9084	0,010	0,8756	0,008
8 capas	0,9073	0,010	0,8742	0,005
6 capas	0,9069	0,008	0,8720	0,010
4 capas	0,9020	0,006	0,8661	0,008

Tabla 17: Resultados de la asignación por capítulos para las diversas podas divido por idiomas

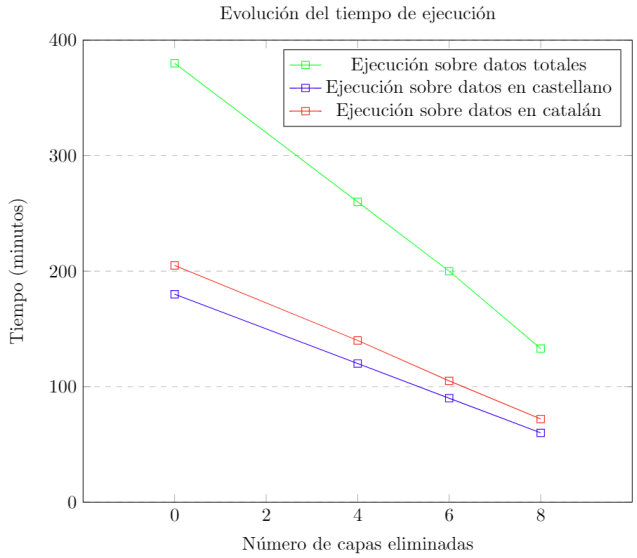
Debido a estos resultados claramente optimistas que nos permiten reducir los costes del entreno sin perjudicar gravemente al rendimiento. Decidimos observar las matrices de confusión (o MC) de cada una de estas podas, con el objetivo de estudiar la evolución de las asignaciones.

Como podemos observar en las matrices de la figura 31, la mayoría de capítulos tienden a clasificarse bien. Con la excepción del séptimo, el cual tiende a fallar más. Tanto en catalán como considerando los datos totales, podemos observar que el octavo y décimo también ofrecen ciertos fallos. Y a lo largo de la poda, estos que fallaban inicialmente van empeorando en su asignación, lo cual era un comportamiento inesperado. Esperábamos que a menor proporción de capas, la asignación se fuera empeorando de forma generalizada.

Esto nos ofrece cierta perspectiva sobre esos tres capítulos, como los más complejos de diferenciar en general. Pues como vemos no existe una preferencia particular hacía ningún otro capítulo, lo que nos muestra que puede ser un problema de falta de contexto. Pues en las instancias de estos capítulos se tiende a hablar de infecciones y objetos extraños sin una definición como tal. Lo cual dificulta considerablemente una asignación correcta.



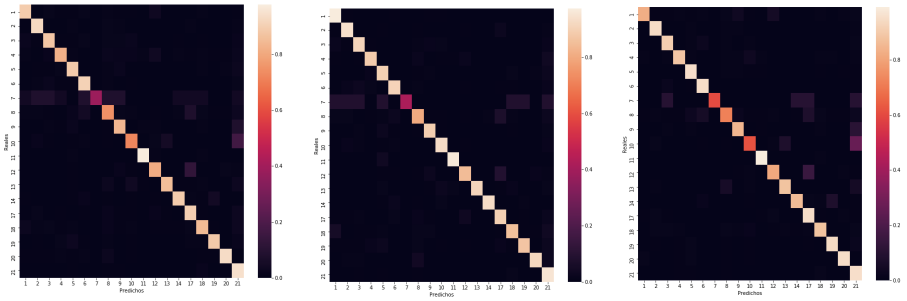
(a) Evolución de los tiempos de entreno por número de capas podadas



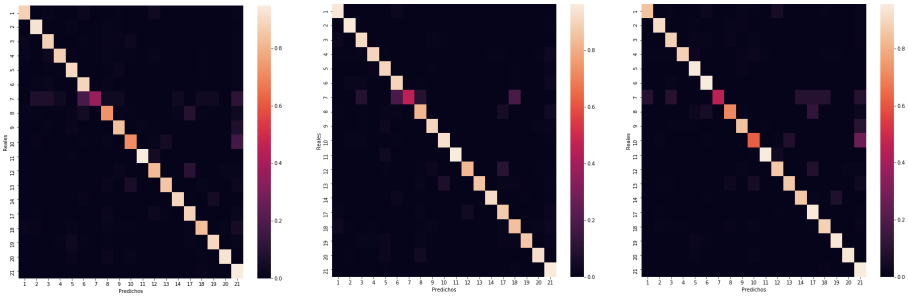
(b) Evolución de los tiempos de ejecución por número de capas podadas

Figura 30: Evolución del coste temporal de los modelos sometidos a podas

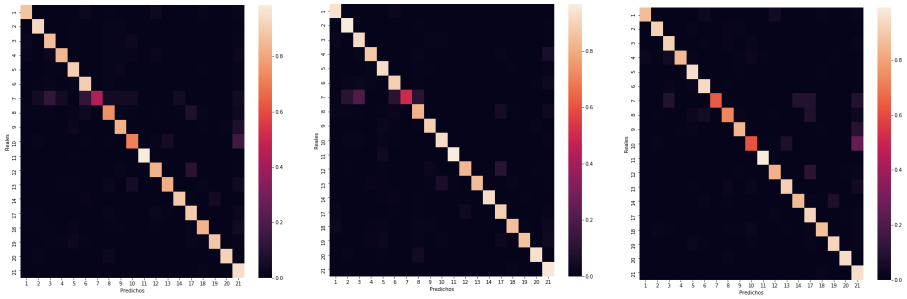
Fuente: Propia



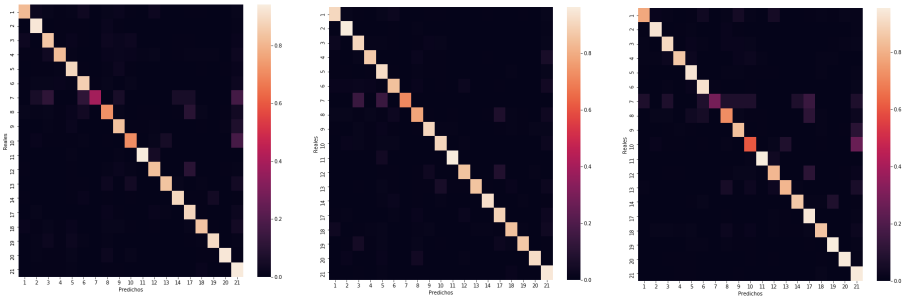
(a) MC de 12 capas de todos los datos (b) MC de 12 capas datos castellanos (c) MC 12 capas datos catalanes



(d) MC 8 capas de todos los datos (e) MC 8 capas datos castellanos (f) MC 8 capas datos catalanes



(g) MC 6 capas de todos los datos (h) MC 6 capas datos castellanos (i) MC 6 capas datos catalanes



(j) MC 4 capas de todos los datos (k) MC 4 capas datos castellanos (l) MC 4 capas datos catalanes

Figura 31: Matrices de confusión en función de las capas y los idiomas

Fuente: Propia

14.4. Trabajos a futuro: Asignación en cascada

Como comentamos en la sección inicial de esta iteración, al ver el rendimiento en la asignación por capítulos esbozamos una posible arquitectura que asigna por cascada. Más concretamente, consistiría en una asignación por etapas. Empleando la misma arquitectura desarrollada, con la diferencia que cada modelo sería entrenado para cada uno de los niveles. Tendríamos un modelo para la clasificación por capítulos. Uno para cada uno de los capítulos, el cual asigna un subcapítulo a las entidades asignadas a este capítulo. Y un último conjunto de modelos para cada subcapítulo que asigna códigos. Una asignación de esta arquitecturas consistiría en: primero una de capítulo, seguida de una por subcapítulo y acabando en un código. Un ejemplo de esta nueva arquitectura se halla en la figura 32.

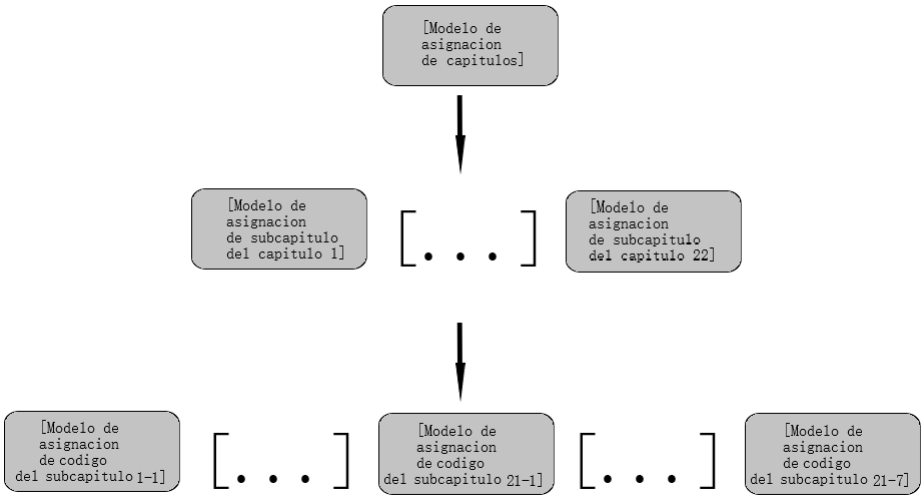


Figura 32: Esquema conceptual del la asignación por cascada

Fuente: Propia

La idea detrás de este modelo es que el conocimiento necesario para encontrar el código relevante de una entidad es una combinación entre conocimientos generales, para discernir entre capítulos, e información muy especifica, para diferenciar entre varios códigos dentro de un mismo subcapítulo. Con esta arquitectura dejamos este conocimiento más general a las primeras fases de la asignación. Permitiendo a las finales gestionar las sutilezas de la diferenciación de códigos. Dotándonos, además, de la posibilidad de hacer podas de ciertos modelos, para aumentar el rendimiento general de la arquitectura.

Como trabajo a futuro, hallamos extremadamente interesante usar dicha arquitectura a como alternativa a un solo modelo que asigne códigos directamente. Sirviendo como introducción a la sustitución del método usado, por un acercamiento basado en la multiclasificación. El cual podría ser la alternativa adecuada, pues esta solución en cascada solventa una de las mayores desventajas de este acercamiento. Hablamos de reducir el número de clases, ya que en esta arquitectura cada modelo tendría que clasificar sobre 20, como máximo.

14.4.1. Estudio adicional de los datos

Debido a la falta de tiempo, esta arquitectura propuesta no se ha podido realizar. Pero, antes de finalizar nuestro proyecto, queríamos esbozar si sería posible realizar dicho acercamiento con los datos disponibles. Para ello profundizamos sobre el estudio hecho inicialmente, en la sección 4.1.

En la figura 10 de dicha sección, pudimos observar una distribución muy desbalanceada. Lo cual repercute en el rendimiento, e incluso la posibilidad de entreno, de algunos modelos. De hecho, vimos que algunos capítulos, con "suficientes" instancias, quedaban con subcapítulos totalmente vacíos. Esto es debido a la distribución desigual ya comentada en el estudio inicial. Este comportamiento se puede observar en la figura 33, donde se ve la distribución de instancias por subcapítulos de los tres capítulos más poblados.

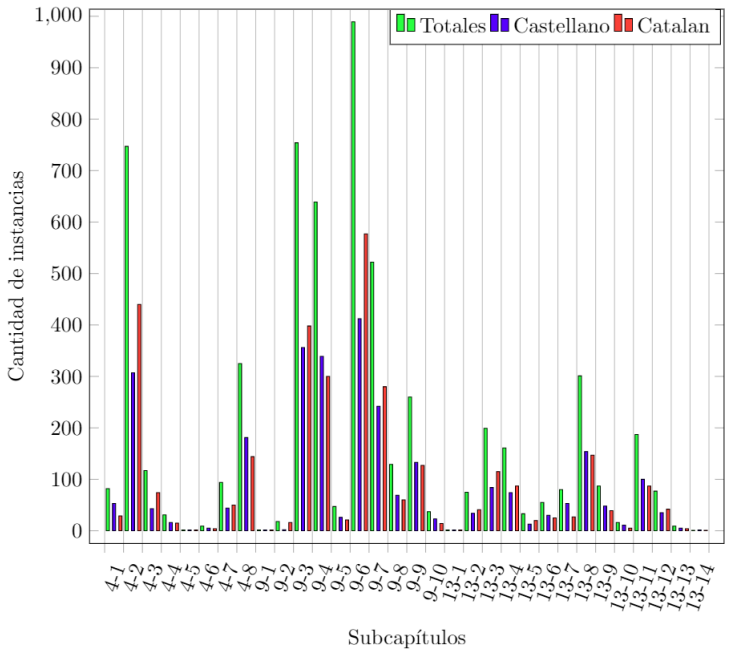


Figura 33: Distribución de las instancias de los capítulos 4,9,13 por subcapítulos

Fuente: Propia

En base a este estudio, deducimos que este desarrollo no podría realizarse sobre todos los capítulos y que solo podríamos emplearlo sobre algún subconjuntos de ellos. Sería ideal probarlo sobre el cuarto, noveno y décimo tercer capítulo. Pues son los tres capítulos con mayor número de instancias, e ilustran diversos comportamientos de distribución. Desde la ideal que se acerca más a una uniforme, hasta la concentrada en algunos subcapítulos y códigos.

15. Conclusiones finales

Como ya podemos intuir la codificación de entidades médicas, de forma automatizada, es un problema con gran variedad de aplicaciones e impactos. Su resolución de forma satisfactoria sobre informes clínicos de atención primaria en castellano y catalán supondrán un gran avance en estos tipos de datos, aún no estudiados.

El método inicialmente propuesto se halla dentro de los más modernos de la literatura actual. Pese a ello no viene sin su conjunto de desventajas, las cuáles nos planteamos resolver a lo largo de las iteraciones. Intentando cumplir, además, con los objetivos propuesto para este desarrollo. Los cuáles consisten en elaborar una resolución exacta e independiente al idioma con un nivel suficiente de especificidad.

A lo largo de la primera iteración se desveló que una de las mayores problemáticas del método estaba actuando en nuestra contra: la dependencia al primer filtrado de entidades. Este factor se vio debido a dos circunstancias: La falta de descriptores y el uso de abreviaciones. Estas se vieron abordadas en la segunda iteración mediante la ampliación de los descriptores y un sistema de resolución de abreviaciones. Durante esta surgió otra problemática considerada: el uso de estructuras externas. Cuyo uso podría resultar en un favoritismo hacía el castellano. Esta problemática fue abordada mediante la generación de algunas estructuras catalanas basadas en la castellanas, para complementar las posibles faltas de información.

La tercera iteración se centró en un refinamiento del entreno. Con el objetivo de obtener un método con mayor exactitud del mostrado hasta el momento. También decidimos experimentar con la reducción de la especificidad del método, con tal de mejorar la exactitud de este. Obteniendo una mejora considerable en su rendimiento.

Por último, en la cuarta iteración se realizó una experimentación más extensa sobre el nuevo nivel de especificidad establecido, asegurando que los resultados obtenidos fueran trasladables a otros datos. Junto a esto, se experimentó con la posibilidad de reducir el tamaño de BERT. Con el objetivo de mantener dichos rendimientos con un menor coste temporal, lo cual se consiguió de forma exitosa.

Como podemos observar en los resultados de la iteración final(14.3), los objetivos de un método con exactitud e independencia del lenguaje fueron obtenidos. Aunque al coste de una menor especificidad, hecho que nos ha permitido reducir los costes de entreno mediante la eliminación de capas de BERT.

Consideramos que este estudio aún debe realizar diversas clases de experimentos que puedan conservar los resultados obtenidos con mayor especificidad. La primera consistiría en la arquitectura planteada en la sección 14.4. Y siendo la segunda el uso del contexto de la mención, factor ignorado por este trabajo y posible clave para una correcta asignación de ciertas menciones complejas. Pese a no haber alcanzado el objetivo de especificidad de forma plena, consideramos este trabajo como la primera piedra fundacional hacía la investigación de este problema en esta clase de documentos.

16. Bibliografía

Referencias

[1] A. Atutxa, A. Casillas, N. Ezeiza, V. Fresno-Fernández, I. Goenaga, K. Gojenola, R. Martínez, M. O. Anchordoqui, and O. P. de Viñaspre, "Ixamed at clef ehealth 2018 task 1: Icd10 coding with a sequence-to-sequence approach," in *CLEF*, 2018.

[2] I. Soriano and J. C. Peña, "Automatic medical concept extraction from free text clinical reports, a new named entity recognition approach," 2017.

[3] Gobierno de España, "Real decreto 69/2015, de 6 de febrero, por el que se regula el registro de actividad de atención sanitaria especializada," 2015. https://www.boe.es/eli/es/rd/2015/02/06/69/con.

[4] M. Almagro, R. Martínez Unanue, V. Fresno Fernández, and S. Montalvo Herranz, "Estudio preliminar de la anotación automática de códigos cie-10 en informes de alta hospitalarios," 2018-03.

[5] World Health Organization, "About icd." https://www.who.int/classifications/icd/en/. Online, Accessed: 2020-09-26.

[6] Snomed International, "Snomed-ct, five step briefing." http://www.snomed.org/snomed-ct/five-step-briefing. Online, Accessed: 2020-09-25.

[7] Unbound Medicine, Inc, "Icd-10 coding." https://www.unboundmedicine.com/icd/view/ICD-10-CM/860000/all/About_ICD_10_CM_Coding_Guide. Online, Accessed: 2020-09-26.

[8] H. Suominen, L. Kelly, L. Goeuriot, and M. Krallinger, "Clef ehealth evaluation lab 2020," *SpringerLink*, Apr 2020.

[9] T. Goodwin and S. Harabagiu, "Automatic generation of a qualified medical knowledge graph and its usage for retrieving patient cohorts from electronic medical records," pp. 363–370, 09 2013.

[10] Y. Wang, L. Wang, M. Rastegar-Mojarad, S. Moon, F. Shen, N. Afzal, S. Liu, Y. Zeng, S. Mehrabi, S. Sohn, and H. Liu, "Clinical information extraction applications: A literature review," *Journal of Biomedical Informatics*, vol. 77, pp. 34 – 49, 2018.

[11] S. J. Nelson, K. Zeng, J. Kilbourne, T. Powell, and R. Moore, "Normalized names for clinical drugs: Rxnorm at 6 years," *Journal of the American Medical Informatics Association : JAMIA*, vol. 18, no. 4, pp. 441–448, 2011. amiajnl-2011-000116[PII].

[12] H. Suominen, L. Kelly, L. Goeuriot, A. Névéol, L. Ramadier, A. Robert, E. Kanoulas, R. Spijker, L. Azzopardi, D. Li, Jimmy, J. Palotti, and G. Zuccon, "Overview of the clef ehealth evaluation lab 2018," in *Experimental IR Meets Multilinguality, Multimodality, and Interaction* (P. Bellot, C. Trabelsi, J. Mothe, F. Murtagh, J. Y. Nie, L. Soulier, E. SanJuan, L. Cappellato, and N. Ferro, eds.), (Cham), pp. 286–301, Springer International Publis-

hing, 2018.

[13] A. Névéol, K. B. Cohen, C. Grouin, T. Hamon, T. Lavergne, L. Kelly, L. Goeuriot, G. Rey, A. Robert, X. Tannier, and P. Zweigenbaum, "Clinical information extraction at the clef ehealth evaluation lab 2016," *CEUR workshop proceedings*, vol. 1609, pp. 28–42, Sep 2016. 29308065[pmid].

[14] A. Miranda-Escalada, A. Gonzalez-Agirre, and M. Krallinger, "Codiesp corpus: gold standard spanish clinical cases coded in icd10 (cie10) - ehealth clef2020," May 2020. Funded by the Plan de Impulso de las Tecnologías del Lenguaje (Plan TL).

[15] A. Miranda-Escalada, E. Farré, and M. Krallinger, "Cantemist corpus: gold standard of oncology clinical cases annotated with CIE-O 3 terminology," Apr. 2020. Funded by the Plan de Impulso de las Tecnologías del Lenguaje (Plan TL).

[16] L. Li, R. Zhou, and D. Huang, "Two-phase biomedical named entity recognition using crfs," *Computational biology and chemistry*, vol. 33, pp. 334–8, 09 2009.

[17] S. Kim, J. Yoon, K.-M. Park, and H.-C. Rim, "Two-phase biomedical named entity recognition using a hybrid method," in *Second International Joint Conference on Natural Language Processing: Full Papers*, 2005.

[18] Y. Xiong, Y. Huanga, Q. Chena, X. Wanga, Y. Nic, and B. Tang, "A joint model for medical named entity recognition and normalization," 2019.

[19] "Vicomtech at cantemist 2020," pp. 489–498, 09 2020.

[20] F. Hassan, D. Sanchez, and J. Domingo-Ferrer, "Tumor entity recognition and coding for spanish electronic health records," 2020.

[21] A. Blanco, A. Casillas, A. Pérez, and A. Ilarraza, "Multi-label clinical document classification: Impact of label-density," *Expert Systems with Applications*, vol. 138, p. 112835, 07 2019.

[22] Y. Lou, T. Qian, F. Li, J. Zhou, D. Ji, and M. Cheng, "Investigating of disease name normalization using neural network and pre-training," *IEEE Access*, vol. 8, pp. 85729–85739, 2020.

[23] Y. Luo, W. Sun, and A. Rumshisky, "A hybrid method for normalization of medical concepts in clinical narrative," in *2018 IEEE International Conference on Healthcare Informatics (ICHI)*, pp. 392–393, 2018.

[24] V. Cotik, J. Vivaldi, and H. Rodríguez, "Semantic tagging of french medical entities using distant learning," in *CLEF*, 2015.

[25] J. Vivaldi and H. Rodriguez, "Medical entities tagging using distant learning," in *Computational Linguistics and Intelligent Text Processing* (A. Gelbukh, ed.), (Cham), pp. 631–642, Springer International Publishing, 2015.

[26] Z. Miftahutdinov and E. Tutubalina, "Deep neural models for medical concept normalization in user-generated texts," *CoRR*, vol. abs/1907.07972, 2019.

[27] A. Pérez, A. Atutxa, A. Casillas, K. Gojenola, and Álvaro Sellart, "Inferred joint multigram models for medical term normalization according to icd," *International Journal of Medical Informatics*, vol. 110, pp. 111 – 117, 2018.

[28] A. Atutxa, A. D. de Ilarraza, K. Gojenola, M. Oronoz, and O. P. de Viñaspre, "Interpretable deep learning to map diagnostic texts to icd-10 codes," *International Journal of Medical Informatics*, vol. 129, pp. 49 – 59, 2019.

[29] Z. Ji, Q. Wei, and H. Xu, "Bert-based ranking for biomedical entity normalization," *CoRR*, vol. abs/1908.03548, 2019.

[30] H. Li, Q. Chen, B. Tang, X. Wang, H. Xu, B. Wang, and D. Huang, "Cnn-based ranking for biomedical entity normalization," *BMC bioinformatics*, vol. 18, pp. 385–385, Oct 2017. PMC5629610[pmcid].

[31] R. Leaman, R. Islamaj Doğan, and Z. Lu, "DNorm: disease name normalization with pairwise learning to rank," *Bioinformatics*, vol. 29, pp. 2909–2917, 08 2013.

[32] L.-H. Chang, "Normalization of disease mentions with convolutional neural networks," May 2019.

[33] D. Xu, M. Gopale, J. Zhang, K. Brown, E. Begoli, and S. Bethard, "Unified Medical Language System resources improve sieve-based generation and Bidirectional Encoder Representations from Transformers (BERT)–based ranking for concept normalization," *Journal of the American Medical Informatics Association*, vol. 27, pp. 1510–1519, 07 2020.

[34] X. Yin, Y. Huang, B. Zhou, A. Li, L. Lan, and Y. Jia, "Deep entity linking via eliminating semantic ambiguity with bert," *IEEE Access*, vol. 7, pp. 169434–169445, 2019.

[35] H. Zhou, S. Ning, Z. Liu, C. Lang, Z. Liu, and B. Lei, "Knowledge-enhanced biomedical named entity recognition and normalization: application to proteins and genes," *BMC Bioinformatics*, vol. 21, p. 35, Jan 2020.

[36] O.-E. Ganea and T. Hofmann, "Deep joint entity disambiguation with local neural attention," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, (Copenhagen, Denmark), pp. 2619–2629, Association for Computational Linguistics, Sept. 2017.

[37] Y. Zhang, X. Ma, and G. Song, "Chinese medical concept normalization by using text and comorbidity network embedding," 2018.

[38] E. Schumacher, A. Mulyar, and M. Dredze, "Clinical concept linking with contextualized neural representations," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (Online), pp. 8585–8592, Association for Computational Linguistics, July 2020.

[39] K. S. Kalyan and S. Sangeetha, "Medical concept normalization in user generated texts by learning target concept embeddings," 2020.

[40] P. Ruas, A. Neves, V. Andrade, and F. Couto, "Lasigebiotm at cantemist: Named entity recognition and normalization of tumour morphology entities and clinical coding of spa-

nish health-related documents," 09 2020.

[41] A. Lamurias, P. Ruas, and F. M. Couto, "Ppr-ssm: personalized pagerank and semantic similarity measures for entity linking," *BMC Bioinformatics*, vol. 20, p. 534, Oct 2019.

[42] Y. Pershina, Maria andHe and R. Grishman, "Personalized page rank for named entity disambiguation," in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (Denver, Colorado), pp. 238–243, Association for Computational Linguistics, May–June 2015.

[43] Z. Guo and D. Barbosa, "Robust named entity disambiguation with random walks," *Semantic Web*, vol. 9, pp. 1–21, 03 2017.

[44] D. Pujary, C. Thorne, and W. Aziz, "Disease normalization with graph embeddings," 2020.

[45] W. Shen, J. Han, J. Wang, X. Yuan, and Z. Yang, "Shine+: A general framework for domain-specific entity linking with heterogeneous information networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 2, pp. 353–366, 2018.

[46] Y. Cao, L. Hou, J. Li, and Z. Liu, "Neural collective entity linking," 2018.

[47] A. E. Johnson, T. J. Pollard, L. Shen, L.-w. H. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. Anthony Celi, and R. G. Mark, "Mimic-iii, a freely accessible critical care database," *Scientific Data*, vol. 3, p. 160035, May 2016.

[48] R. Dogan, R. Leaman, and Z. lu, "Ncbi disease corpus: A resource for disease name recognition and concept normalization," *Journal of biomedical informatics*, vol. 47, 01 2014.

[49] J. Li, Y. Sun, R. Johnson, D. Sciaky, C.-H. Wei, R. Leaman, A. P. Davis, C. Mattingly, T. Wiegers, and Z. lu, "Biocreative v cdr task corpus: a resource for chemical disease relation extraction," *Database*, vol. 2016, p. baw068, 05 2016.

[50] T. Ohta, Y. Tateisi, and J.-D. Kim, "The genia corpus: an annotated research abstract corpus in molecular biology domain," pp. 82–86, 01 2002.

[51] K. Roberts, D. Demner-Fushman, and J. M. Tonning, "Overview of the tac 2017 adverse reaction extraction from drug labels track," *Theory and Applications of Categories*, 2017.

[52] R. Bossy, L. Deléger, E. Chaix, M. Ba, and C. Nédellec, "Bacteria biotope at BioNLP open shared tasks 2019," in *Proceedings of The 5th Workshop on BioNLP Open Shared Tasks*, (Hong Kong, China), pp. 121–131, Association for Computational Linguistics, Nov. 2019.

[53] A. Sarker and G. Gonzalez, "Overview of the second social media mining for health (smm4h) shared tasks at amia 2017," 11 2017.

[54] O. Uzuner, B. R. South, S. Shen, and S. L. DuVall, "2010 i2b2/VA challenge on concepts, assertions, and relations in clinical text," *Journal of the American Medical Informatics Association*, vol. 18, pp. 552–556, 06 2011.

[55] C. Arighi, M. Krallinger, and F. Leitner, "Biocreative - track 1: Interactive bio-id assign-

ment (iat-id)," 2017.

[56] L. Goeuriot, "Datasets - share/clef ehealth 2013." https://sites.google.com/site/shareclefehealth/data, May 2013.

[57] A. Névéol, C. Grouin, J. Leixa, S. Rosset, and P. Zweigenbaum, "The QUAERO French medical corpus: A ressource for medical entity recognition and normalization," in *Proc of BioTextMining Work*, pp. 24–30, 2014.

[58] L. Goeuriot, "Clef ehealth evaluation lab 2015 task 1b:clinical named entity recognition," April 2015.

[59] M. Rasmussen and N. Berggrein, "Named entity recognition and disambiguation in danish electronic health records," 2019.

[60] G. Georgiev, V. Zhikov, B. Popov, and P. Nakov, "Building a named entity recognizer in three days: Application to disease name recognition in bulgarian epicrises," 05 2012.

[61] B. Olivier, "Unified medical language system (umls)." https://www.nlm.nih.gov/research/umls/index.html, May 2021.

[62] "Catsalut. servei català de la salut." https://catsalut.gencat.cat/ca/proveidors-professionals/registres-catalegs/catalegs/diagnostics-procediments/cim-10/, Nov 2017.

[63] "Edición electrónica de la cie-10." https://eciemaps.mscbs.gob.es/ecieMaps/browser/index_10_2008.html, May 2010.

[64] FIB-UPC, "Normativa del treball final de grau del grau en enginyeria informàtica de la fib." https://www.fib.upc.edu/sites/fib/files/documents/actes/normativatfg-gei-2012-09-26.pdf, 2012. Online, Accessed: 2020-09-26.

[65] S. Explorer, "Software Engineer Average Salary in Spain 2020." http://www.salaryexplorer.com/salary-survey.php?loc=203&loctype=1&job=836&jobtype=3, 2020. Online, Accessed: 2020-10-01.

[66] S. Explorer, "Project Manager Average Salary in Spain 2020." http://www.salaryexplorer.com/salary-survey.php?loc=203&loctype=1&job=140&jobtype=3, 2020.

[67] S. Explorer, "Recruiter Average Salary in Spain 2020." http://www.salaryexplorer.com/salary-survey.php?loc=203&loctype=1&jobtype=3&job=445, 2020.

[68] Latex Project, "The latex project public license." https://www.latex-project.org/lppl.txt, 2008. Online, Accessed: 2020-10-02.

[69] The Python Software Foundation, "Python: History and license." https://docs.python.org/3/license.html, 2020. Online, Accessed: 2020-10-02.

[70] Trello, "Precio de Trello." https://trello.com/es/pricing, 2020. Online, Accessed: 2020-10-02.

[71] Google, "¿Qué es Google Calendar?." https://sites.google.com/site/recursosdweb20idiomas/google/google-calendar, 2020. Online, Accessed: 2020-10-02.

[72] M. Skype, "Precio de llamadas de Skype son gratis." https://support.skype.com/es/faq/FA34702/las-llamadas-entre-usuarios-de-skype-son-gratuitas-en-cualquier-lugar-del-mun 2020. Online, Accessed: 2020-10-02.

[73] G. de España, "Real Decreto 2001/1983, de 28 de julio, sobre regulación de la jornada de trabajo, jornadas especiales y descansos." https://www.boe.es/buscar/act.php?id=BOE-A-1983-20906, 2020.

[74] A. Coworking, "Precios Coworking." https://www.aureacoworking.com/precios-coworking/, 2020. Online, Accessed: 2020-10-02.

[75] Amazon, "CHUWI HeroBook Pro Ordenador Portátil Ultrabook." https://www.amazon.es/CHUWI-HeroBook-Ordenador-Port%C3%A1til-Ultrabook/dp/B083B8L7Q9/ref=sr_1_13?dchild=1&keywords=notebook&qid=1602331765&s=electronics&sr=1-13, 2020. Online, Accessed: 2020-10-02.

[76] Amazon, "AWS Educate." https://aws.amazon.com/es/education/awseducate/, 2020. Online, Accessed: 2020-10-02.

[77] Google-Research, "bert-license." https://github.com/google-research/bert/blob/master/LICENSE, Oct 2018.

[78] "Faq sur le changement de licence." https://www.elastic.co/fr/pricing/faq/licensing#section-can-you-summarize-the-changes, 2021.

[79] "Get snomed ct." https://www.snomed.org/snomed-ct/get-snomed#licensing, 2021.

[80] "Nlm, snomed-ct licensing." https://www.nlm.nih.gov/healthit/snomedct/snomed_licensing.html, 2021.

[81] "Ministerio de sanidad, consumo y bienestar social. aviso legal." https://www.mscbs.gob.es/avisoLegal/home.htm, 2021.

[82] "Creatice commons licenses." https://creativecommons.org/licenses/?lang=es_ES, 2017.

[83] C. Manning, P. Raghavan, and H. Schütze, "Introduction to information retrieval." https://nlp.stanford.edu/IR-book/html/htmledition/irbook.html, Apr 2009.

[84] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval: The Concepts and Technology Behind Search*. Addison Wesley, 2011.

[85] R. Seitz, "Understanding tf-idf and bm25." http://www.kmwllc.com/index.php/2020/03/20/understanding-tf-idf-and-bm25/, Mar 2020.

[86] "¿qué es elasticsearch?." https://www.elastic.co/es/what-is/elasticsearch, 2021.

[87] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (Minneapolis, Minnesota), pp. 4171–4186, Association for Computational Linguistics, June 2019.

[88] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[89] F. K. Khattak, S. Jeblee, C. Pou-Prom, M. Abdalla, C. Meaney, and F. Rudzicz, "A survey of word embeddings for clinical text," *Journal of Biomedical Informatics: X*, vol. 4, p. 100057, 2019.

[90] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Łukasz Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, "Google's neural machine translation system: Bridging the gap between human and machine translation," 2016.

[91] A. Radford and K. Narasimhan, "Improving language understanding by generative pre-training," 2018.

[92] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, (New Orleans, Louisiana), pp. 2227–2237, Association for Computational Linguistics, June 2018.

[93] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[94] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," *CoRR*, vol. abs/1411.1792, 2014.

[95] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, "BioBERT: a pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, pp. 1234–1240, 09 2019.

[96] E. Alsentzer, J. Murphy, W. Boag, W.-H. Weng, D. Jindi, T. Naumann, and M. McDermott, "Publicly available clinical BERT embeddings," in *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, (Minneapolis, Minnesota, USA), pp. 72–78, Association for Computational Linguistics, June 2019.

[97] J. Cañete, G. Chaperon, R. Fuentes, J.-H. Ho, H. Kang, and J. Pérez, "Spanish pre-trained bert model and evaluation data," in *PML4DC at ICLR 2020*, 2020.

[98] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," *CoRR*, vol. abs/1909.11942,

2019.

[99] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[100] M. Alfaro Latorre, "Manual de codificacion cie-10-es diagnosticos." https://www.mscbs.gob.es/estadEstudios/estadisticas/normalizacion/CIE10/UT_MANUAL_DIAG_2016_prov1.pdf, 2011.

[101] "Classificació estadística internacional de malalties i problemes relacionats amb la salut." https://catsalut.gencat.cat/web/.content/minisite/catsalut/proveidors_professionals/registres_catalegs/catalegs/cim-10/CIM10-volum1.pdf.

[102] "Consulta interactiva de cim-10." https://catsalut.gencat.cat/ca/proveidors-professionals/registres-catalegs/catalegs/diagnostics-procediments/consulta-interactiva/, 2021.

[103] "Enfermedades y síntomas comunes según cie-10 (clasificación internacional de enfermedades)." https://www.tuotromedico.com/CIE10/, 2021.

[104] "Snomed-ct to icd-10-cm map." https://www.nlm.nih.gov/research/umls/mapping_projects/snomedct_to_icd10cm.html, Mar 2021.

[105] J. N. Nikiema, V. Jouhet, and F. Mougin, "Integrating cancer diagnosis terminologies based on logical definitions of snomed ct concepts," *Journal of Biomedical Informatics*, vol. 74, pp. 46–58, 2017.

[106] K. Giannangelo and J. Millar, "Mapping snomed ct to icd-10.," in *MIE*, pp. 83–87, 2012.

[107] K. W. Fung and J. Xu, "Synergism between the mapping projects from snomed ct to icd-10 and icd-10-cm," in *AMIA Annual Symposium Proceedings*, vol. 2012, p. 218, American Medical Informatics Association, 2012.

[108] "Disponible la 10a versión de la extensión catalana de snomed ct." https://ticsalutsocial.cat/es/actualitat/ja-es-pot-descarregar-la-10a-versio-de-lextensio-catalana-de-snomed-ct/, Jul 2020.

[109] A. S. Schwartz and M. A. Hearst, "A simple algorithm for identifying abbreviation definitions in biomedical text," in *Biocomputing 2003*, pp. 451–462, World Scientific, 2002.

[110] S. Sohn, D. C. Comeau, W. Kim, and W. J. Wilbur, "Abbreviation definition identification based on automatic precision estimates," *BMC bioinformatics*, vol. 9, no. 1, pp. 1–10, 2008.

[111] A. Intxaurrondo, "Biomedical abbreviation recognition and resolution 2nd edition (barr2)," Sep 2018.

[112] J. Y. Laguna and V. Alberola, *Diccionario de siglas medicas y otras abreviaturas, eponimos y terminos medicos relacionados con la codificacion de las altas hospitalarias*. Ministerio de Sanidad y Consumo, Centro de Publicaciones, 2003.

[113] G. Catalunya, "Abreviacions sanitàries." https://canalsalut.gencat.cat/

web/.content/_Professionals/Recursos/altres_recursos_interes/serveis_linguistics/assessorament_linguistic/documents/abreviacions.pdf, Aug 1997.

[114] J. Kellen, "Spanish medical abbreviations - a to z." http://medinelingua.info/spanish/es_abbr.php, 2018.

[115] M. Krallinger and A. Intxaurrondo, "Abremes-db." https://zenodo.org/record/2207130, Nov 2018.

[116] J. Yetano Laguna, V. Alberola Cuñat, and A. Giner Menéndez-Valdés, "Diccionario de siglas médicas y otras abreviaturas desarrollado por sedom." https://www.sedom.es/diccionario/, Feb 2021.

[117] M. Giulianelli, J. Harding, F. Mohnert, D. Hupkes, and W. Zuidema, "Under the hood: Using diagnostic classifiers to investigate and improve how language models track agreement information," 2018.

[118] N. F. Liu, M. Gardner, Y. Belinkov, M. E. Peters, and N. A. Smith, "Linguistic knowledge and transferability of contextual representations," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (Minneapolis, Minnesota), pp. 1073–1094, Association for Computational Linguistics, June 2019.

[119] E. Voita, R. Sennrich, and I. Titov, "The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives," 2019.

[120] H. Sajjad, F. Dalvi, N. Durrani, and P. Nakov, "On the effect of dropping layers of pre-trained transformer models," 2021.

Glosario

BERT "Bidirectional Encoder Representations from Transformers", un nuevo modelo de red neuronal que modela lenguajes con enfoque generalista y polifacético.

BM25 Métrica de relevancia empleada por los sistemas de IR.

Código completo El tercer nivel de identificadores de CIE. Se encuentra codificado en el tercer dígito. Se llama también código de tres dígitos, y suele ser el nivel estándar de especificación.

Capítulo El primer nivel de identificadores de CIE, y más general. Se encuentra codificado en los dos primeros dígitos de este.

Descriptor de identificador Descripción de un identificador de CIE, suele tratarse de una descripción formal del grupo de enfermedades que cubre este.

Entidad Concepto con identificador dentro de un estándar de codificación. Para CIE son enfermedades.

Infracódigo El quinto nivel de identificadores de CIE. Es la mayor especificación que existe, y es bastante infrecuente.

IR "Information Retrieval", sistema de recuperación de información en conjuntos de documentos. Basado en búsquedas por fragmentos de texto.

Mención Fragmento de texto, en el cual se hace mención a una entidad de un estándar.

NEN "Named Entity Normalization", es el problema de codificar menciones a entidades de un estándar. El nombre en inglés de nuestro problema.

NLP "Natural language processing", nombre en inglés de PLN.

PLN Procesamiento del lenguaje natural. Área de la inteligencia artificial que estudia las interacciones entre el lenguaje humano y los ordenadores, particularmente interesado en como estos últimos lo procesan.

Subcódigo El cuarto nivel de identificadores de CIE. Se encuentra codificado en el cuarto dígito.

Subcapítulo El segundo nivel de identificadores de CIE. Se suele expresar como "capítulo-subcapítulo". Como el capítulo, se encuentra codificado en los dos primeros dígitos..