UNIVERSIDAD
DE LA REPÚBLICA
URUGUAY

Estrategias de machine learning para el análisis de historias clínicas electrónicas para la detección precoz de enfermedades raras

Tesina Licenciatura en Bioquímica

Matías Rolando

Tutora

Dra. Lucia Spangenberg

Co-tutora

Dra. Leticia Cagnina

Montevideo, Agosto 2023

Unidad de Bioinformática, Institut Pasteur de Montevideo, Uruguay

Facultad de Ciencias, Universidad de la República, Uruguay

Agradecimientos

Le agradezco a mi familia y amigas, por todo el aguante en la carrera, principalmente a mis padres, a mi hermano y mis abuelos. Sí, también a vos madrina. Ile, toda la carrera juntos, llegamos, gracias por el aguante :). Gracias Ale por haber estado siempre. Gracias a les compas de la UBI por bancarme y por siempre decir que la comida que llevaba estaba rica.

Agradezco a mis tutoras, Lucía y Leticia por acompañarme durante este proceso y haberme ayudado a concluirlo de forma satisfactoria, ♡.

Tabla de Contenidos

Resumen

Las enfermedades raras (RD) son un grupo de patologías que afectan individualmente a menos de 1 de cada 2000 personas, pero que en conjunto impactan aproximadamente al 7 % de la población mundial. La mayoría de estas enfermedades afectan a niños, son crónicas y progresivas, y no tienen un tratamiento específico. Los pacientes con RD enfrentan desafíos diagnósticos, con un tiempo promedio de diagnóstico de 5 años, múltiples visitas a especialistas y procedimientos invasivos. Esta demora puede ser perjudicial para su salud.

El aprendizaje automático (ML), a través de diferentes herramientas, tiene el potencial de mejorar la atención médica al proporcionar una gestión más personalizada y precisa de los pacientes, diagnósticos y, en algunos casos, tratamientos. En este trabajo se propuso detectar pacientes con RD en las primeras etapas de su estancia hospitalaria mediante la aplicación de diversas técnicas de ML a la base de datos de cuidados críticos MIMIC-III. Los métodos utilizados incluyeron algoritmos de aprendizaje automático supervisado, como regresiones logísticas, árboles de decisión y máquinas de soporte vectorial (SVM), así como métodos de aprendizaje profundo, como redes neuronales, LSTM y redes neuronales convolucionales (CNN). También se realizaron experimentos considerando modelos de clasificación basados en Transformers. Utilizando LSTM, se logró un AUC del 95,2 % para identificar a

los pacientes con RD, lo que permitió redireccionarlos hacia un camino diagnóstico más preciso. El estudio demuestra el potencial del ML para mejorar la eficiencia y precisión del diagnóstico de las enfermedades raras.

Palabras clave: Enfermedades raras, machine learning, bioquímica, SVM, LR, redes neuronales, transformers

Introducción

Las enfermedades raras son patologías que afectan a un individuo cada 2000 personas (EURODIS, 2009). Aunque cada enfermedad es muy infrecuente, hay alrededor de 7000 enfermedades raras documentadas, y si se agrupan afectan a casi el 7 % de la población mundial (Sireau, 2013). La mayoría de estas tienen base genética, afecta niños, son de carácter crónico y progresivo y no tienen un tratamiento específico. Cuándo se habla del diagnóstico de enfermedades raras se refiere al mismo como una "odisea diagnóstica", debido a que en promedio a un paciente le toma 5 años y por lo menos 7 visitas a especialistas para poder obtener un diagnóstico preciso (Yan, He, y Dong, 2020). Algunos estudios plantean a su vez que en la atención primaria hay un bajo conocimiento de las enfermedades raras y la fuente principal de información es el internet seguido de revistas especializadas (Fernández, Martín, Allés, Delgado, y Martínez, 2012). Es por estas, y varias otras razones, que los pacientes con enfermedades raras son muy vulnerados por el sistema de salud. Para el diagnóstico, se han utilizado aproximaciones genómicas como la secuenciación de exoma completo (WES) o secuenciación del genoma completo (WGS) (Raggio y cols., 2021; Meyer y cols., 2021). El porcentaje de diagnóstico de las enfermedades raras varía entre 30 % al 50 % (Liu y cols., 2019; Clark y cols., 2018), dependiendo del tipo de enferme-

dad. Por estas razones es importante incluir este tipo de análisis en el *pipeline* de diagnóstico de los pacientes, y también hacerlo de una manera anticipada.

Existen varias estrategias de Aprendizaje Automático o *Machine Learning* que son habitualmente utilizadas en diversos campos de la salud, lo que permite el desarrollo de mejores abordajes diagnósticos y tratamientos (Esteva y cols., 2019; Hwang y Lee, 2022). Los algoritmos que se utilizan en este contexto, varían y dependen del tipo de dato de entrada. Por ejemplo existen estudios basados en redes neuronales que utilizan datos de imágenes para detectar enfermedades raras específicas (Schaefer, Lehne, Schepers, Prasser, y Thun, 2020). A su vez, existen trabajos en los que se procesan historias clínicas utilizando herramientas de procesamiento de lenguaje natural (al igual que nuestro abordaje), pero utilizando ontologías y supervisión débil (Dong y cols., 2021).

En este trabajo se parte de historias clínicas/resúmenes de alta obtenidos de una base de datos de la operativa de un hospital y se analizan para identificar signos de presencia de una enfermedad rara. De esta forma, se puede redirigir a un paciente con probabilidad de una enfermedad rara hacia un *pipeline* diagnóstico más adecuado (por ejemplo, que incluya la consulta con un genetista, estudios genómicos, entre otros), para disminuir los tiempos de la odisea diagnóstica

1.1. Procesamiento de texto

El preprocesamiento de texto es necesario al trabajar con modelos de ML debido a que estos están diseñados para trabajar con números y no con texto. Por lo que es necesario utilizar técnicas que permitan realizar dicha conversión sin perder información, para luego ser utilizados al entrenar los algoritmos clasificadores. Éstos, luego, van a ser capaces de reconocer patrones en estos números y realizar predic-

ciones, como puede ser la identificación de paciente con una enfermedad rara o una enfermedad común.

Existen diversas formas para convertir el texto a números, entre ellas la vectorización (transformar las palabras o caracteres a números que formarán parte de un vector que represente el texto) y el uso de *word embeddings* (el vector que representa al texto es "aprendido" o calculado considerando la semántica). El *Vector Space Model* (Salton, Wong, y Yang, 1975) es uno de los modelos más acertados para la codificación de palabras en vectores numéricos, teniendo en cuenta que se preserva la relación semántica de las palabras.

1.1.1. Vectorizador

El vectorizador utilizado fue TF-IDF (Salton y Buckley, 1988). El significado de la sigla es "Frecuencia de Término - Frecuencia del Documento Inversa", donde la Frecuencia de Término ($tf_{t,d}$) se define como la cantidad de veces que una palabra t se encuentra en un texto d. La idea detrás de este algoritmo es obtener el peso de una palabra relativo a una colección de diversos textos para poder categorizar los mismos (Kim y Gil, 2019). Por otro lado, la Frecuencia del Documento inversa (idf_t) se basa en la frecuencia del documento (df_t) definido como el número de textos en una colección que contienen la palabra t, obteniéndose el peso de cada palabra de acuerdo a la Ecuación 1.1.

$$idf_t = log\frac{N}{df_t} \tag{1.1}$$

donde N es el número total de textos en la colección. Por último, para obtener el peso de cada palabra t en el texto se necesita el producto de ambos:

$$tf - idf_{t,d} = tf * idf_t \tag{1.2}$$

Los resultados se podrán dividir en tres categorías (Manning, Raghavan, y Schütze, 2008):

- Valores altos se obtienen cuando una palabra aparece en pocos textos, es decir, que *df* será alto y *idf* también debido a que la palabra aparece pocas veces en relación con todos los textos.
- Valores bajos cuando la palabra aparece pocas veces en un texto o aparee en muchos textos diferentes.
- Valores muy bajos cuando la palabra aparece en todos los textos, por ejemplo los conectores.

En términos de ventajas este algoritmo es simple y eficiente. Dado que el método retorna vectores numéricos para los documentos, se puede calcular fácilmente la similitud entre los mismos. Frente a una tarea particular, como una búsqueda que considere algo relacionado con esos documentos, se podrían encontrar textos similares de forma rápida y efectiva, lo que demuestra la capacidad del algoritmo para condensar características de los textos y como efecto colateral, ayudar en el proceso de las búsquedas.

Este algoritmo tiene limitaciones, el mismo no puede establecer relaciones entre palabras como las similitudes semánticas o por ejemplo los sinónimos (Ramos y cols., 2003); si se buscaran documentos donde la palabra *bruja* es relevante, no incluirá documentos con la palabra *hechicera*.

1.1.2. Embeddings

Word embedding es una técnica en la cual se busca representar las palabras de un texto en un espacio multidimensional, donde cada palabra está mapeada a un vector

numérico. Lo importante es que se mantiene la relación semántica entre las palabras. Una representación de esto se puede ver en la Figura 1.1

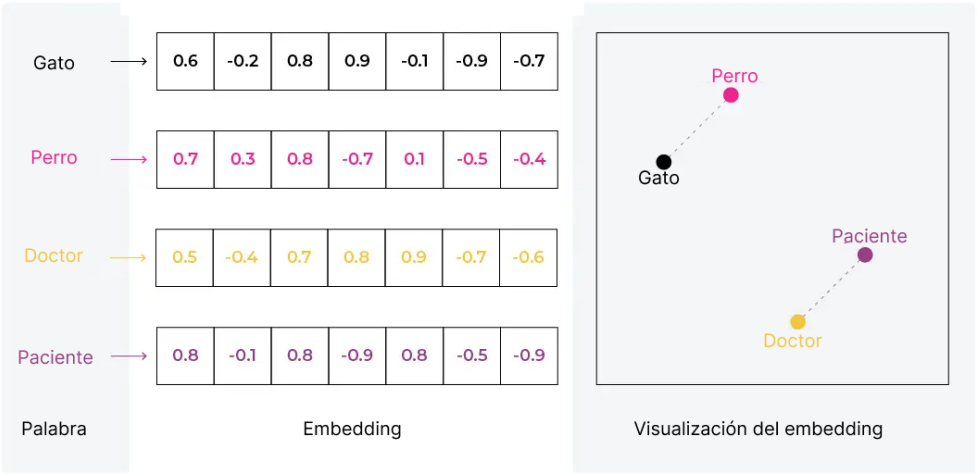


Figura 1.1: Representación de varios *embeddings*. A la izquierda se puede ver la representación en vectores de las palabras y a la derecha se ve como las palabras con relaciones semánticas se encuentran cerca.

El uso de los *embeddings* se popularizó cuando se presentó *word2vec* (Mikolov, Chen, Corrado, y Dean, 2013). Este es un algoritmo basado en redes neuronales artificiales para la obtención de *word embeddings* utilizando la *hipótesis distribucional*. Esta plantea que las palabras que a menudo tienen las mismas palabras vecinas tienden a ser semánticamente similares. Esto ayuda a mapear palabras semánticamente similares a vectores o *embeddings* geométricamente cercanos.

Existen otros métodos para crear *word embeddings* que no son preentrenados, estos métodos utilizan la reducción de dimensionalidad con matrices de co-ocurrencia de palabras, modelos probabilísticos y la representación explícita en términos del

contexto en el cual estas palabras figuran (Levy y Goldberg, 2014; Lebret, 2014).

1.2. Clasificación de texto

Para la clasificación de texto se usaron diferentes algoritmos partiendo desde los modelos *baseline* o básicos, aumentando la complejidad hasta redes neuronales complejas y transformers.

1.2.1. Métodos básicos

Regresión Logística

La regresión logística es un modelo matemático que puede ser utilizado para describir la relación entre diversas variables y una variable dependiente dicotómica, como puede ser tener o no una enfermedad. Este modelo se basa en la función logística 1.3

$$f(z) = \frac{1}{1 + e^{-z}} \tag{1.3}$$

que tiene como codominio 0,1 por lo que sin importar el valor de z, $f(z)$ siempre será un valor entre 0 y 1. Se puede analizar fácilmente, si el valor de z es $-\infty$, $f(-\infty) = \frac{1}{1+e^{-(-\infty)}} = \frac{1}{1+e^{\infty}} = 0$, por otro lado, si el valor de z es $+\infty$, $f(+\infty) = \frac{1}{1+e^{-(+\infty)}} = \frac{1}{1+e^{-\infty}} = 1$. Debido a esto se puede utilizar para describir la probabilidad del suceso Z. Para pasar de la función al modelo logístico se debe describir el valor de Z como la suma lineal de α más β_1 por X_1 y así sucesivamente como se describe:

$$Z = \alpha + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_k X_k \tag{1.4}$$

donde X_k son las variables independientes de interés y $\alpha y \beta_k$ son constantes que representan parámetros desconocidos. Si se sustituye la ecuación de Z 1.4 en $f(Z)$1.3, se obtiene:

$$f(Z) + \frac{1}{1 + e^{-(\alpha + \sum \beta_i X_i)}} \tag{1.5}$$

Para entenderlo como un modelo matemático se debe asignar un contexto, un caso puede ser un modelo epidemiológico en el que un sujeto puede estar enfermo o no, es decir, puede ser 1 o 0. Entonces cada X correspondería a una variable medible y asociada a la probabilidad de estar enfermo $P(D = 1|X_1, X2, ..., X_k)$ y se define el modelo Logístico como:

$$P(D = 1|X_1, X2, ..., X_k) = \frac{1}{1 + e^{\alpha + \sum \beta_i X_i}} \tag{1.6}$$

si se conocen los parámetros α y β se puede inferir a partir del valor de las variables la probabilidad de que un sujeto este o no enfermo. Para obtener el valor de los parámetros se podría utilizar un grupo de sujetos en que el valor de las variables es conocido y su estado también (Manning y cols., 2008).

En el caso de clasificación de texto, el modelo logístico reconoce un vector de variables y luego evalúa los coeficientes para cada variable para predecir la clase del texto (Shah, Patel, Sanghvi, y Shah, 2020). En este modelo se deben hacer ciertas suposiciones:

- La regresión logística binaria requiere que la variable dependiente sea binaria.
- Considerar solo las variables significativas (evitar outliers).
- Las variables independientes deben ser independientes (poca o nula multicolinealidad).
- Las variables independientes deben estar relacionadas linealmente a los log odds (linealidad).

- Habitualmente requiere de muchos datos para lograr un clasificador confiable.

Un modelo de Regresión Logística debe ser entrenado para obtener los valores de α y β para cada observación X. Para esto se necesitan dos métricas, la primera es la distancia entre la predicción y el valor real de la categoría a la que pertenece el conjunto de variables, a lo que se conoce como *loss function* o *cost function*. Por otro lado, se necesita un algoritmo de optimización que va a actualizar los pesos β para poder reducir el valor del resultado de la *loss function* y así acercarse a la categoría real.

Un ejemplo de función de pérdida o *loss function* es la denominada *cross-entropy loss function*, esta es una función que se basa en la estimación por máxima verosimilitud, en la cual se eligen los parámetros α y β que maximizan la probabilidad logarítmica de las categorías y verdaderas en los datos dada las observaciones X (Jurafsky y Martin, 2009).

Los algoritmos de optimización posibles para este modelo son:

- *L-BFGS* es un algoritmo de memoria limitada para resolver optimizaciones no lineares sujeto a restricciones simples. Está pensado para problemas en los que es difícil obtener información sobre la matriz Hessiana (derivada segunda de la función de costo), o para problemas grandes y densos (Zhu, Byrd, Lu, y Nocedal, 1997).

- *LIBLINEAR* es una librería diseñada para manejar datasets grandes e implementa un algoritmo de optimización denominado *Descenso de Coordenadas* para la función de coste. Este, resuelve los problemas de la optimización realizando sucesivas minimizaciones aproximadas a lo largo de hiperplanos de coordenadas (Fan, Chang, Hsieh, Wang, y Lin, 2008).

- El algoritmo *newton-cg* a diferencia de L-BFGS, sí realiza el cálculo de la matriz Hessiana completa (Hanke, 1997) para cada iteración del proceso de optimización de la función de coste.

- *SAGA* es un algoritmo que optimiza la suma de un número finito de funciones convexas suaves, utilizando la regularización L1 (Defazio, Bach, y Lacoste-Julien, 2014).

- *FTRL* es un algoritmo de optimización desarrollado en Google (McMahan y cols., 2013) que utiliza la idea del descenso del gradiente pero considerando métodos de regularización en la elección de los movimientos. Es más adecuado para modelos poco profundos con espacios ralos de características (es decir, muchas son cero).

Support Vector Machine

Support Vector Machines (SVM) (Vapnik, 1982) es otro ejemplo de un algoritmo con la capacidad de aprender basándose en ejemplos, para poder clasificar, luego, eventos desconocidos. Estos se han usado en variadas aplicaciones biomédicas, incluyendo la clasificación de microarrays de expresión génica (Noble, 2006), búsqueda de marcadores genéticos relacionados con el cáncer, descubrimiento de nuevas drogas para el tratamiento del cáncer (Huang y cols., 2018), entre otras.

Se tienen los datos para entrenar $\mathbf{x}_1, ..., \mathbf{x}_n$ que son vectores en un espacio $X \subseteq \mathbb{R}^d$. Estos datos también incluyen la categoría y a la que pertenecen, en caso de una clasificación binaria, $y_i \in 0, 1$. En términos simples, una SVM o una Máquina de Vectores de Soporte corresponden a un hiperplano que separa los datos con base en un margen máximo. Por lo que, todos los vectores que estén de un lado del hiperplano pertenecerán a una categoría y los datos que estén del lado contrario pertenecerán

a la otra categoría. Se denomina *support vectors* o vectores de soporte a todos los datos que queden en las inmediaciones del hiperplano.

De forma general, SVM permite proyectar el plano de datos X a un plano de dimensiones superiores F a través de un operador de kernel K, considerando un conjunto de clasificadores cuya forma se puede observar en la Ecuación 1.7 (Tong y Koller, 2001).

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x}) \tag{1.7}$$

En donde n es la dimensión de X y α_i los valores que obtiene SVM correspondientes al hiperplano en F con máxima separación. Así pues, si el kernel K satisface las condiciones del teorema de Mercer (Burges, 1998), entonces se puede expresar como $K(\mathbf{u}, \mathbf{v}) = \Phi(\mathbf{u}) \cdot \Phi(\mathbf{v})$, con $\Phi : X \to F$. Dicho de otra manera, las condiciones establecen que cuando se emplea un kernel es posible utilizarlo como una métrica de similitud entre pares de datos. Esto permite lograr una separación no lineal de los datos que derivará en el proceso de clasificación.

El uso del kernel es necesario debido a que si se parte de datos complejos, la separación lineal de los mismos no es una opción viable. Las condiciones que debe cumplir la función de kernel entre otras son: ser simétrica, debido a que en el cálculo del producto escalar de dos puntos transformados, el orden de los puntos no debe afectar el resultado, y ser semi-definida positiva, para cualquier conjunto finito de puntos.

Esta última propiedad garantiza que el producto escalar en el espacio de características transformado sea no negativo. Luego, se puede reescribir f como muestra la Ecuación 1.8.

$$f(\mathbf{x}) = \mathbf{w} \cdot \Phi(\mathbf{x}), \ \textit{donde}\ \mathbf{w} = \sum_{i=1}^{n} \alpha_i \Phi(\mathbf{x}_i) \tag{1.8}$$

Al elegir diferentes tipos de kernels se pueden proyectar los datos desde X a F, en donde los hiperplanos van a corresponder a límites de decisión más complejos (Tong y Koller, 2001), dependiendo del kernel usado. Existen varios tipos de kernels, entre los que se encuentran (Schlkopf, Smola, y Bach, 2018):

- *Lineal*: $K(\mathbf{u}, \mathbf{v}) = \mathbf{u}^T\mathbf{v}$
- *Polinómico* (con los parámetros p y c): $K(\mathbf{u}, \mathbf{v}) = (\mathbf{u}^T\mathbf{v} + c)^p$
- *Gaussiano* (con el parámetro σ^2): $K(\mathbf{u}, \mathbf{v}) = exp(-\frac{||\mathbf{u}-\mathbf{v}||^2}{2\sigma^2})$

Árboles de Decisión

Los árboles de decisión son un tipo de clasificador el cual se implementa particionando recursivamente el espacio de instancias hasta asignarles una categoría expresa, esta partición deriva en un árbol. Este árbol va a consistir de nodos que forman un *árbol raíz*, el primer nodo es el denominado *nodo de inicio* y no tiene entradas de información. Por otro lado, el resto de los nodos serán llamados los nodos internos. Estos van a separar el espacio de instancias en dos o más subespacios de acuerdo a una función aplicada a atributos de los datos. En los árboles más simples cada función va a considerar un único atributo de lo que se obtiene subespacios definidos con base en los valores de dicho atributo. Por ejemplo el atributo puede ser la edad, y se generan dos subespacios uno para personas mayores de 30 años y otro para las personas menores de 30 años. A los nodos terminales, aquellos en los que el espacio de datos no se divide más, se los denominan hojas. Las hojas pueden estar asignadas a una clase a la cual un dato puede pertenecer o a la probabilidad de que el dato pertenezca a dicha clase (Rokach y Maimon, 2005).

Una de las ventajas de los árboles de decisión frente a otros algoritmos de clasificación es la fácil interpretación de los resultados, puesto que la separación de los

datos se da usualmente con base en atributos simples (Kingsford y Salzberg, 2008).

El algoritmo debe decidir cuál atributo usar para separar los datos en cada nodo, para esto utiliza una propiedad estadística llamada *information gain* o ganancia de información, esta va a medir cuan acertadamente un atributo separa los datos en un nodo de acuerdo a las clases de dichos datos. Para definir la *ganancia de información* primero se debe definir la *entropía*; esta caracteriza la pureza de una colección arbitraria de instancias. Dada una colección S que contiene ejemplos positivos y negativos de un atributo determinado, la entropía de S relativa a esa clasificación binaria sería como se expresa en la Ecuación 1.9.

$$Entropía(S) = -p_+ log_2 p_+ - p_- log_2 p_- \tag{1.9}$$

Donde p_+ es la proporción de ejemplos positivos en S y p_- es la proporción de los ejemplos negativos en S. Por lo que si la entropía es 0 todos los ejemplos dentro del conjunto S pertenecen a la misma clase, ya sea positiva o negativa. Y será 1 cuando la proporción de ejemplos para cada clase es idéntica. La entropía será, por tanto, un valor que varía entre 0 y 1. De forma general, si el atributo puede tomar c valores diferentes, la entropía se define como:

$$Entropía(S) = \sum_{i=1}^{c} -p_i log_2 p_i \tag{1.10}$$

donde p_i es la proporción de S que pertenece a la clase i. Por lo tanto, la *entropía* se puede definir como la efectividad de un atributo al clasificar el conjunto de datos de acuerdo a su clase. De forma intuitiva, cuánto mayor sea la entropía menos útiles son los atributos para clasificar el conjunto de datos. Por otro lado, cuanto menor sea la entropía más información se puede obtener de los datos.

Entonces, ahora podemos definir la ganancia de información, como la reducción esperada de la entropía con respecto a usar otro atributo al dividir el espacio de

datos. La ganancia de información, $Ganancia(S, A)$ de un atributo A relativa a una colección de ejemplos S, se define como:

$$Ganancia(S, A) = Entropía(S) - \sum_{v \in Valores(A)} \frac{|S_v|}{|S|} Entropía(S_v) \tag{1.11}$$

donde $Valores(A)$ son todos los valores posibles para el atributo A y S_v es el subconjunto de S donde el atributo A tiene valor v. Entonces la ganancia de información es la reducción esperada de la entropía al conocer el valor del atributo A (Mitchell y cols., 2007).

Otro criterio para la selección de atributos es el *índice de Gini*. Este se define como:

$$Gini(S) = 1 - \sum_{i=1}^{n} p_i^2 \tag{1.12}$$

Donde p_i es la frecuencia relativa de la clase i en el conjunto de datos S (Suryakanthi, 2020). El valor de este índice será 0 cuando haya uniformidad en la clase del conjunto. Al realizar una división en un nodo del árbol se calcula el índice de Gini para cada atributo y se selecciona el atributo con menor índice de Gini como el mejor para realizar la división. Esto sucede cuando el atributo logra la mayor reducción de la impureza del conjunto.

1.2.2. Métodos de aprendizaje profundo o *Deep Learning*

En esta sección se expondrán las 3 arquitecturas de aprendizaje profundo o *deep learning* utilizadas en el presente trabajo. En primer lugar, se describirán brevemente los modelos de Redes Neuronales Artificiales (*Artificial Neural Networks*) que constituyen la base de los dos tipos de clasificadores utilizados en este trabajo. Luego, se describirán las Redes Neuronales de Memoria de Largo y Corto Plazo (*Long Short Term Memory (LSTM)*) y Redes Neuronales Convolucionales (*Convolutional Neural*

Networks (CNN)). Se explican sus arquitecturas, las ventajas de estos modelos como también las concesiones.

Redes Neuronales Artificiales

Las redes neuronales artificiales aparecieron como un gran paradigma computacional que fue inspirado en primera instancia por el cerebro humano. Una de las primeras formas de interpretar las neuronas cerebrales como una unidad de cómputo fue en la década del 1950 de la mano de Rosenblatt (Rosenblatt, 1958), quien diseñó un algoritmo llamado *Perceptron*. Este algoritmo permitía asociar entradas de datos con salidas de información de forma similar a como lo hacen los seres humanos.

En términos simples, una red neuronal es un ensamblaje de elementos de procesamiento simples, llamados unidades o nodos, donde su función se asemeja de manera abstracta a una neurona biológica. La habilidad de procesamiento de la neurona se almacena en conexiones entre las unidades llamados pesos o *weights*, los cuales se obtienen en un proceso de aprendizaje a partir de patrones o datos de aprendizaje (Gurney, 1997).

En la Figura 1.2 se muestra un modelo de una neurona. Se evidencian 3 elementos básicos de un modelo neuronal:

- Un conjunto de conexiones o *synapses* donde cada una está caracterizada por un peso propio. Cada entrada de la red (input) x_i es multiplicado por el peso sináptico w_{kj}. La anotación refiere a que es el peso w número j de la neurona k.

- Por otro lado está la suma, en la cual se suman todas las entradas ponderadas por su peso, en este caso es una operación lineal.

- Por último se encuentra la función de activación, en la cual se limita la amplitud de salida de una neurona. Su objetivo es reducir el rango posible de valores de salida a algún valor finito.

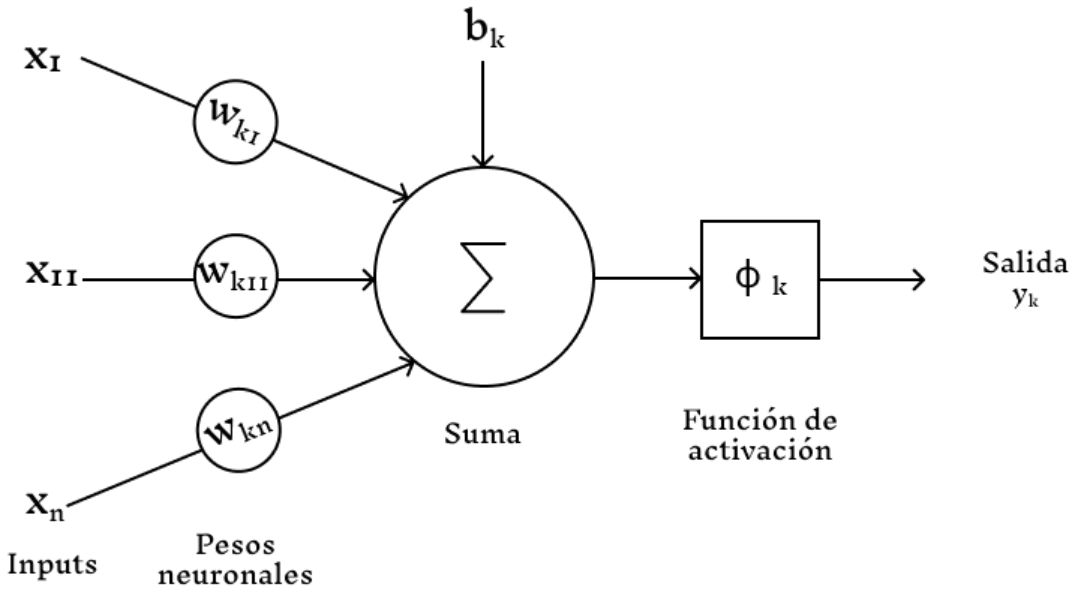


Figura 1.2: Modelo de una neurona no lineal, k. En la misma se observa las entradas o *inputs*, los pesos neuronales, la suma de los anteriores con el *bias* y la función de activación.

En la Figura 1.2 también se incluye un *bias*, denotado como b_k. Este valor tiene como efecto el aumento o la disminución del valor de entrada a la función de activación.

De forma matemática, se puede describir la neurona k como:

$$u_k = \sum_{j=1}^{n} w_{kj} x_j \tag{1.13}$$

donde u_k es el combinador lineal de salida respecto a las entradas o inputs $x_1, x_2, ..., x_n$ y luego, $w_{k1}, w_{k2}, ..., w_{km}$ son los respectivos pesos de cada input para la neurona k. Por otro lado,

$$y_k = \phi(u_k + b_k) \tag{1.14}$$

donde y_k es el output de la neurona k y, ϕ es la función de activación. El bias tiene como efecto la aplicación de una transformación afín a la salida u_k del combinador lineal del modelo de la neurona como se detalla en 1.13 (Haykin, 2008).

La función de activación ϕ, se aplica a $v_k = u_k + b_k$ y define el valor de salida de la neurona. Existen varios tipos de función de activación, las dos más básicas son las siguientes:

- *Función threshold o límite*: para este tipo de función de activación se tiene

$$\phi(v) = \begin{cases} 1, \ si \ v \geq 0 \\ 0, \ si \ v < 0 \end{cases} \tag{1.15}$$

  Esto quiere decir que a valores menores a 0 el valor del output de la neurona será 0 pero a valores mayores o iguales a 0 el valor de output de la neurona será siempre 1, como se muestra en la Figura 1.3.

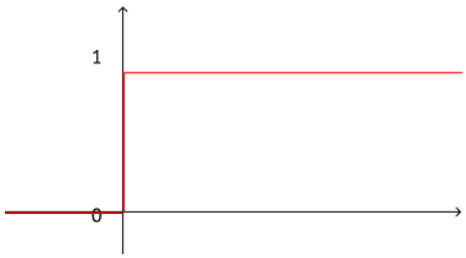


Figura 1.3: Gráfica para una función de activación límite.

A este tipo de neurona se le llama modelo de *McCulloch y Pitts* (McCulloch

y Pitts, 1943). En el modelo, los autores describen la propiedad *todo o nada*, que describe la función de activación (Haykin, 2008).

- *Función sigmoidea:* Se define como una función estrictamente creciente la cual exhibe un balance entre un comportamiento linear y no linear. Un ejemplo de una función sigmoidea es la *función logística*, que ya ha sido mencionada y se puede expresar como:

$$\phi(v) = \frac{1}{1 + \exp -av} \tag{1.16}$$

donde a es la *pendiente* de la función. Al variar el valor de a se obtienen diferentes pendientes como se ve en la Figura 1.4.

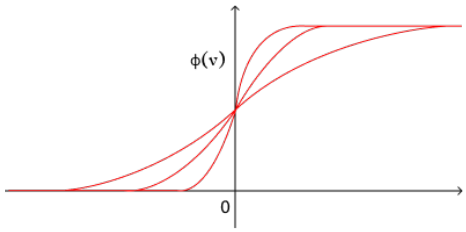


Figura 1.4: Gráfica para una función de activación del tipo sigmoidea.

La similitud que tiene con la función límite es hacia el infinito, en donde la función sigmoidea tiende a los valores finitos 1 o 0. Una gran diferencia entre ambas es que la función sigmoidea asume valores en un rango continuo entre 0 y 1, mientras que la función límite solo toma los valores de 0 o 1 (Haykin, 2008).

- *Función Softmax:* Esta función de activación parte de los vectores numéricos de salida de una red neuronal y retorna un vector con valores reales arbitrarios

en el rango [0,1] (Bridle, 1990). La función está dada por la siguiente ecuación:

$$softmax(z)_i = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{z_j}} \tag{1.17}$$

Donde z es el vector de salida de una red neuronal y N el tamaño del vector de salida de la red neuronal. Un elemento i del vector de salida de la función de softmax se puede pensar como la probabilidad de ese elemento de pertenecer a la clase i, utilizando la teoría de la probabilidad para representar una distribución categórica. Esta es la distribución del vector de entrada sobre N diferentes posibilidades.

Existen varias arquitecturas de redes neuronales artificiales, es decir, cómo las neuronas se interconectan y relacionan entre sí. Los tres tipos más comunes para la clasificación de textos, se detallan a continuación (Haykin, 2008).

Redes Neuronales Multicapa (*Multiperceptron*)

Este tipo de arquitectura es uno de los más básicos y consta de una o más *capas ocultas o hidden layers*, a cuyas neuronas se las denomina neuronas ocultas. El término oculta se refiere a que estas capas no se comunican directamente ni con las entradas (inputs) o las salidas (outputs). La utilidad de las capas ocultas radica en que muchas veces los patrones de entrada no son separables linealmente, por lo que la unidad de salida no puede realizar una clasificación correcta. Si los patrones de entrada son transformados por las neuronas de la capa intermedia, pueden volverse separables linealmente. Por lo tanto, al añadir una o más capas ocultas, se pueden proyectar los patrones de entrada en un espacio cuya dimensión viene dada por el número de unidades de la capa oculta, para que de esta forma se pueda realizar en la unidad de salida una clasificación correcta (Churchland y Sejnowski, 1992).

Este tipo de arquitectura se puede ver reflejada en la Figura 1.5, este ejemplo es una red neuronal *completamente conectada* con una capa oculta, si algunas de las conexiones sinápticas no estuvieran se trataría de una red neuronal *parcialmente conectada*.

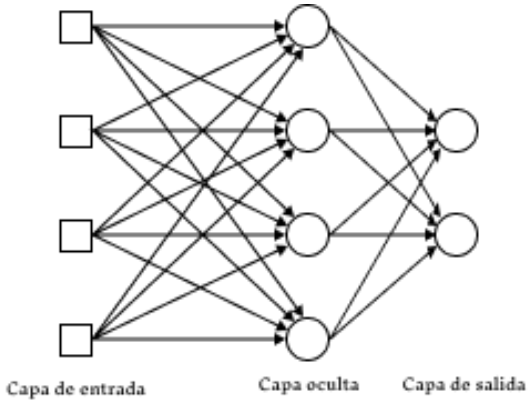


Figura 1.5: Esquema de una red neuronal de múltiples capas.

Redes Neuronales de Largo y Corto Plazo (*LSTM*)

Las redes neuronales recurrentes (RNN) se diferencian de las redes multicapa en que tienen al menos un *circuito de retroalimentación*. Un ejemplo puede ser el que se ve en la Figura 1.6, en la misma se tiene una única capa de neuronas cuyos valores de salida son utilizados como entrada del resto de las neuronas. La arquitectura de RNN fue descrita por Rumelhart, Hinton, y Williams (1986), en donde plantearon las RNN como una extensión de una red multicapa tradicional, permitiendo a la información persistir y ser procesada a través de diversas unidades neuronales secuenciales.

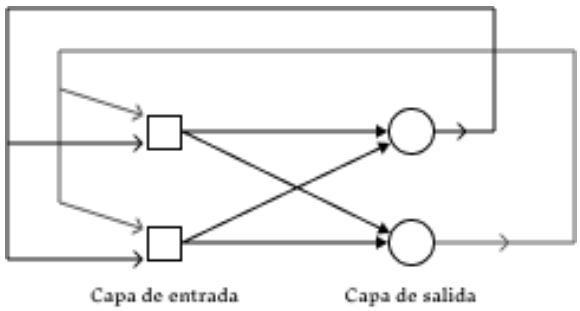


Figura 1.6: Esquema de una red neuronal recurrente sin capas ocultas.

Las redes LSTM son un tipo de red neuronal recurrente y su arquitectura fue por primera vez planteada en el artículo de Hochreiter y Schmidbhuber (Hochreiter y Schmidhuber, 1997).

Este modelo fue diseñado para poder solucionar las limitaciones de las RNN tradicionales, las cuales tenían problemas como capturar dependencias de rango largo. Estas dependencias se refieren (en el contexto del procesamiento del lenguaje natural) a la cantidad de palabras o tokens que serán consideradas para aprender el contexto de cada una de las palabras en las secuencias de entrada de la red.

Otro problema que enfrentan las RNN es el *vanishing gradient*, que surge debido a que los gradientes de las redes neuronales son calculados utilizando *backpropagation*, este calcula las derivadas de los pesos moviéndose capa por capa desde la última a la primera. Debido a esto, las derivadas de cada capa son multiplicadas entre sí y pequeños valores de derivada se multiplican entre sí, obteniendo un valor exponencialmente pequeño de gradiente al aumentar la cantidad de capas, no permitiendo un ajuste correcto de los hiperparámetros del modelo debido a que se diluyen los valores de actualización. En consecuencia, la red neuronal deja de aprender.

Una red LSTM es un modelo que permite aprender y recordar las dependencias largas a partir de datos secuenciales de manera eficaz.

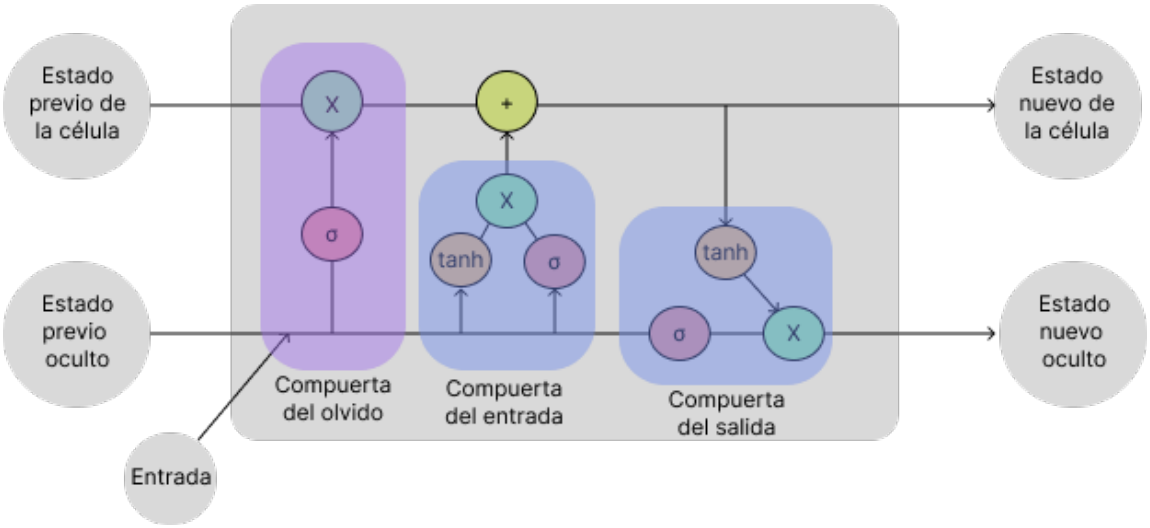


Figura 1.7: Arquitectura de una célula de LSTM. El símbolo + denota una suma, el X una multiplicación y *tanh* una función tangencial.

La idea principal detrás de LSTM es la introducción de la célula de memoria; esta actúa como una unidad de memoria que se autorregula en el marco de cada unidad recurrente de la unidad. Esta célula de memoria va a estar equipada con diferentes compuertas o *gates* que van a controlar el flujo de información, entre las que se encuentran la compuerta de entrada, la compuerta del olvido o *forget gate* y la compuerta de salida. La Figura 1.7 muestra un ejemplo de este tipo de arquitectura.

La compuerta de entrada va a determinar que información se almacena en la célula de memoria a partir de una entrada actual, la compuerta de olvido decide que información descartar de la célula y por último la compuerta de salida regula la salida desde la célula hacia la siguiente *unit* o unidad. Estas compuertas son implementadas usando la función de activación sigmoidea, permitiéndole aprender y controlar el flujo de información de manera apropiada.

Al utilizar estas compuertas y funciones de activación específicas denotadas en

Hochreiter y Schmidhuber (1997), las LSTM van a poder retener o mantener información selectivamente a partir de secuencias largas de entrada. Esto le va a permitir al modelo tener la capacidad de aprender patrones complejos y dependencias en información temporal.

Redes Neuronales Convolucionales (*CNN*)

Una Red Neuronal Convolucional está formada por distintos tipos de capas, entre los que se encuentran las capas convolucionales, capas de agrupación y capas de neuronas totalmente conectadas. Una de las primeras CNN fue descrita en Lecun, Bottou, Bengio, y Haffner (1998); en este artículo se hizo hincapié, por un lado, a la arquitectura de la CNN presentada, que consiste de varias capas en las que se incluyen capas convolucionales, estas son las responsables de aprender patrones locales y *features* o características al aplicar operaciones convolucionales a los datos de entrada. A su vez, incluye varias capas de agrupación que tienen el cometido de disminuir el tamaño de los mapas de características o *feature map*, producto de la capa anterior, sin la pérdida de información. Por último, capas de neuronas totalmente conectadas a las capas siguientes que permiten un aprendizaje de características de alto nivel. Esto se puede ver en la Figura 1.8.

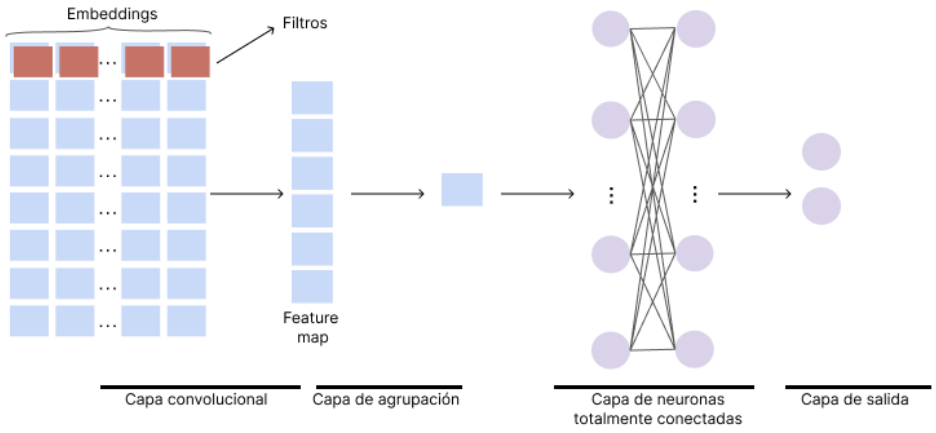


Figura 1.8: Principales componentes de una red CNN

Por otro lado, respecto a las operaciones convolucionales, estas son llevadas a cabo por filtros o *kernels* que extraen patrones locales a partir de la información de entrada al deslizarse sobre los datos. Este movimiento lo realiza por la totalidad de la entrada, en cada posición el filtro se multiplica con elementos con la región de entrada correspondiente (a esta área de interacción también se la denomina campo receptivo). El resultado de estas operaciones es sumado para producir un único valor en esa posición, para luego generar una nueva matriz denominada *feature map*. Este tipo de operación le permite a la red poder obtener información acerca de características espaciales locales como lo pueden ser bordes en una imagen.

A su vez, en el artículo se hace énfasis en el entrenamiento con *backpropagation* o propagación hacia atrás; esta técnica involucra calcular los gradientes de los parámetros de la red respecto a una función de pérdida o *loss function*, y luego actualizar los parámetros en la dirección opuesta de los gradientes para poder disminuir el valor de pérdida. Vale la pena mencionar que este es el algoritmo de actualización de los pesos adoptado por todos los modelos neuronales que se utilizan en la actualidad.

La capa de agrupación o *pooling layer* va a remplazar el *output* de la capa convolucional con un resumen estadístico de las salidas cercanas. Por ejemplo, la operación *Max Pooling* (Zhou y Chellappa, 1988), va a reportar el valor de salida más alto dentro de un rectángulo de un vecindario particular de salidas. En todos los casos esta capa va a ayudar a que la representación no tenga cambios significativos, que permanezca invariante, frente a cambios en el *input*. Esto quiere decir que si hay pequeños cambios en los datos de entrada no tendría que haber cambios en los valores de salida de la capa de agrupación (Goodfellow, Bengio, y Courville, 2016).

1.2.3. Métodos basados en Transformers

Este tipo de arquitectura de aprendizaje profundo fue presentado en Vaswani y cols. (2017) y se plantea como una arquitectura revolucionaria para el procesamiento del lenguaje natural. La idea principal del artículo es reemplazar las RNN con un mecanismo de *self-attention* que captura las dependencias entre palabras en una secuencia de las mismas.

Los autores plantean que las RNN tienen problemas en la paralelización y el tratamiento de dependencias de rango muy largo. Por lo que proponen un modelo de *transformer* que solo depende de mecanismos de atención o *attention mechanisms*. Este último le permite al modelo poder pesar la relevancia de diferentes palabras en una misma frase mientras se procesa la información. Como consecuencia, el modelo puede enfocarse en diferentes partes de la secuencia de entrada a diferentes tiempos y así capturar relaciones de rango largo entre las palabras de la secuencia. Esto genera que el modelo sea capaz de capturar el contexto.

El modelo, en términos simples, consiste de un codificador o *encoder* y un decodificador o *decoder*. El codificador extrae *features* de una secuencia de entrada, y el

decodificador usa las *features* para producir una secuencia de salida.

El codificador en el transformer consiste de múltiples bloques codificadores. Luego, una secuencia de entrada fluye entre los diferentes bloques y la salida de los mismos se convierte en la entrada para el decodificador. Las entradas del codificador son vectores de embeddings. Dentro de cada bloque existen dos sub-capas principales, la primera sub-capa comprende un *multi-head attention* que recibe las entradas. A su vez, una segunda sub-capa comprende una red de retroalimentación completamente conectada.

El decodificador de un transformer también está formado por varios bloques decodificadores, cada bloque recibe *features* desde el codificador. Este decodificador también está formado por diversas capas, entre las que se encuentran una *multi-head attention*, pero a diferencia de la misma en el codificador, esta no presta atención a todas las palabras de la secuencia, si no que solamente a las palabras previas. Por lo que, la predicción de una palabra cualquiera depende solamente de las palabras que la precedían, por ejemplo en el contexto de una traducción. La segunda sub-capa también implementa un *multi-head attention*, pero este recibe información desde la salida del decodificador anterior pero también información de la salida del codificador. Esto se puede ver en términos simples en la Figura 1.9.

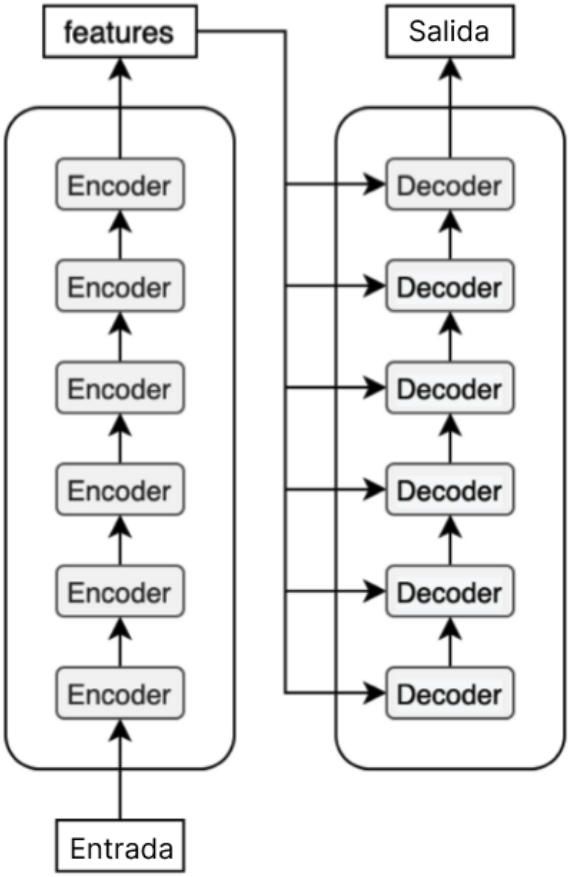


Figura 1.9: Diagrama simple de la arquitectura del transformer presentado en Vaswani y cols. (2017). Figura traducida de *Transformer's Encoder-Decoder: Let's Understand The Model Architecture - KiKaBeN* (2020).

Una de las limitaciones del modelo es la complejidad computacional que posee, debido a la arquitectura del mismo. Los autores plantean diversas técnicas para poder reducir esta limitación.

Un ejemplo de este tipo de modelos es BERT (Devlin, 2019) o *Bidirectional Encoder Representations from Transformers*. Este modelo está basado en un modelo de

lenguaje enmascarado y se entrena previamente mediante transformadores bidireccionales, utilizando únicamente el *encoder* en la arquitectura.

1.3. Historias clínicas

Las historias clínicas son el "conjunto de documentos relativos a los procesos asistenciales de cada paciente, que incluye los datos, valoraciones e informaciones sobre su situación y evolución clínica, así como la identificación de los médicos y de los demás profesionales que han intervenido"(RAE, 2023). Este conjunto de documentos es de difícil acceso y obtención debido a que son datos sensibles de diferentes pacientes y están regidos por diferentes leyes de privacidad.

Debido a estas razones, las fuentes de historias clínicas son varias y de diferente índole. Aunque al usar diferentes fuentes se agrega heterogeneidad a las mismas debido a que el tipo de escritura de la historia clínica depende de cada profesional así como también de su especialización. Las fuentes que se utilizaron en este trabajo fueron, una base de datos de registros médicos a partir de la Oficina Del Libro de la Facultad de Medicina, UdelaR (Bello, Naya, Raggio, y Rosá, 2019). También, una base de datos llamada MIMIC-III (Johnson y cols., 2016), la cual está conformada por documentos y resultados de más de 46.000 pacientes de Unidades de Cuidados Intensivos del Beth Isreael Deaconess Medical Center (Boston, Massachusetts) entre los años 2001 y 2012. Otra fuente es la de historias clínicas de un proyecto previo llamado URUGENOMES (urugenomes.org). A su vez, se realizó *web scraping* a partir de *case reports* de la plataforma de PubMed (*PubMed*, s.f.) utilizando artículos open access. Por último, se utilizó ChatGPT (OpenAI, 2021) para generar algunos ejemplos de historias clínicas y resúmenes de alta tanto para enfermedades raras como para enfermedades comunes; estas fueron curadas por un genetista clínico (ejemplos

de los *prompts* utilizados se pueden ver en la Tabla del Anexo A3).

De la base de datos MIMIC-III solo se usaron los resúmenes de alta o *discharge summary* de las historias clínicas. Estos textos son generados cuando un/una paciente es dado de alta de un servicio de emergencia o de una institución de salud. La ventaja de usar estos textos es que son un resumen de la historia clínica del paciente, ya que en la misma se detallan todos los antecedentes relevantes o no a ese evento de visita a emergencia/institución particular.

Una aclaración importante es el idioma, puesto que la mayoría de las fuentes de historias son en inglés, las procedentes de la Oficina del Libro y el proyecto URUGENOMES son en español y hubo que realizar la traducción de las mismas para poder realizar el trabajo, generando así un potencial foco de ruido. Un ejemplo para cada resumen de alta de MIMIC-III se puede ver en la Figura del Anexo A1.

Objetivos

2.1. Objetivo general

Elaborar estrategias de machine learning para la predicción de enfermedades raras a partir de historias clínicas de pacientes.

2.2. Objetivos específicos

- Buscar nuevos case-reports de enfermedades raras para enriquecer la base de datos.

- Probar diferentes formas de trasnominación de texto a números, en las que se incluyen diversos métodos de representación de textos que abarcan algunos de vectorización y otros de *word embeddings*.

- Entrenar diversos algoritmos de aprendizaje supervisado como los árboles de decisión, SVM, regresión logística y verificar su eficacia, utilizando medidas como ROC, matrices de confusión y F1.

- Entrenar modelos basados en redes neuronales artificiales con creciente complejidad y comprobar si se obtienen resultados adecuados.

Materiales y métodos

3.1. Herramientas

Para el desarrollo práctico del presente trabajo se utilizó como lenguaje de programación Python 3 (Van Rossum y Drake, 2009). Para el espacio de trabajo se utilizó Google Collaboratory (*Google Colab*, 2023), este es un producto de Google Research que permite la ejecución de código Python en el navegador, proveyendo al usuario de poder computacional tanto de CPU como de GPU. Es un producto gratuito basado en Júpiter Notebooks y es de código abierto. El manejo de datos fue llevado a cabo con la librería *Pandas* (pandas development team, 2020).

Para el desarrollo y ejecución de los diferentes modelos se utilizaron diversas librerías. Para los modelos simples se utilizó la librería llamada *Scikit-learn* (Pedregosa y cols., 2011), para los modelos de redes neuronales se utilizaron las librerías *Keras* (Chollet y cols., 2015) y *Tensorflow* (Abadi y cols., 2015). Por último, para los modelos basados en transformers se utilizó *Transformers de HuggingFace* (Wolf y cols., 2020).

3.2. Datos

La base de datos de historias clínicas está compuesta de diversas fuentes, obteniendo así un muestreo heterogéneo. En el caso de MIMIC-III, como se debían obtener solo los resúmenes de alta de pacientes con enfermedades raras y también aquellos que no las tenían, se tomó en cuenta la clasificación llevada a cabo por Dong y cols. (2021) en la cual los autores encontraron historias clínicas de enfermedades raras a partir de un subconjunto de historias de MIMIC-III. Ellos utilizaron un método de dos fases para la clasificación. En primer lugar, los *tokens* o palabras que aparecían en el texto son vinculados con conceptos médicos de la *Unified Medical Lenguage System (UMLS)* utilizando la herramienta SemHER. Los resultados obtenidos fueron refinados usando reglas para la eliminación de abreviaciones y pares de *text-UMLS* con menciones poco frecuentes de dichos textos. En segundo lugar, los conceptos de *UMLS* fueron vinculados con *Orphanet Rare Disease Ontoloy (ORDO)* y así los autores de dicho estudio crearon un dataset con 1073 menciones, 143 de enfermedades raras y 927 de comunes. En nuestro trabajo nos enfocamos en buscar entre los resúmenes de alta de MIMIC-III aquellos que incluían las menciones de (Dong y cols., 2021) y así se obtuvieron 65 historias clínicas vinculadas a enfermedades raras y 274 con enfermedades comunes. A su vez, se incluyeron 100 historias clínicas adicionales, no vinculadas a enfermedades raras de acuerdo a (Dong y cols., 2021) para introducir variabilidad entre los diferentes textos. Por otro lado, se incluyeron historias clínicas de la Oficina del Libro de la Facultad de Medicina. Estas contenían diagnósticos relacionados con cardiología, hematología, neurología y medicina interna. A partir de estos se realizó una revisión por un experto (médico genetista) y así se obtuvieron 98 registros, de los cuales 64 eran de enfermedades

comunes y 34 de enfermedades raras. Estos textos fueron traducidos del español al inglés. Del proyecto URUGENOMES se incluyeron 32 historias clínicas con enfermedades raras y, por último, a partir del *web scrapping* de la plataforma de PubMed se obtuvieron 277 historias de enfermedades raras. Debido a la dificultad de conseguir historias clínicas describiendo el paso de un paciente con una enfermedad rara, se incluyen también 10 historias de enfermedades raras y 13 de enfermedades comunes que fueron generadas por ChatGPT y luego curadas por un experto. La proporción final de las diferentes fuentes se puede ver en la Figura 3.1.

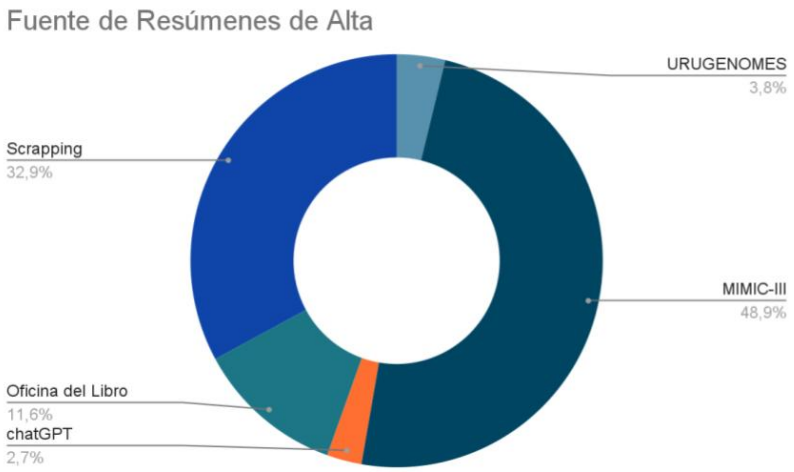


Figura 3.1: Gráfico que resume las diferentes fuentes de las historias clínicas, siendo el total 842.

En suma, el *corpus* cuenta con 842 registros clínicos, 418 correspondientes a enfermedades raras y 423 a comunes. El dataset completo está disponible en github.com/matiasr1608/rare_disease_corpus. A modo de ejemplo, una historia clínica para una enfermedad rara y una común se puede encontrar en la Figura A1 del Anexo 1.

Descripción de los datos

	Historias	Vocabulario	Palabras	Promedio palabras	Oraciones	Promedio oraciones	Min	Max
corpus	842	41765	787277	935	3576	4	1	340
común	424	29671	474648	1119	2753	6	1	340
rara	418	21686	312629	747	823	1	1	39

Tabla 3.1: Estadísticas del corpus.

Para la descripción de los datos se trabaja con los mismos sin procesar. La cantidad de historias clínicas es de 842 (primera fila de la Tabla 3.1). En esta tabla se pueden ver las principales características de los datos, a partir de la primera fila se puede inferir que hay 787277 palabras en 3576 oraciones. Respecto a los diagnósticos se puede ver que estos están escritos en 4 oraciones en promedio, los resúmenes de alta de enfermedades comunes tienen en promedio más oraciones que los resúmenes de las enfermedades raras. Aunque gran parte de los resúmenes tienen una única oración, esto se puede deber a que se escribió como un párrafo entero. Esta distribución se puede ver en la Figura 3.2 panel B.

De las últimas dos filas de la Tabla 3.1, se puede inferir lo que se mencionaba anteriormente respecto a la cantidad de oraciones por tipo de historia. Aunque el vocabulario (palabras únicas) es similar, los resúmenes de alta respecto a enfermedades comunes tienen una mayor cantidad de palabras que las enfermedades raras. Esto mismo ocurre con la cantidad de oraciones, donde las descripciones de enfermedades raras están escritas con 1/3 de la cantidad de oraciones de las enfermedades comunes. De esto se desprende la cantidad de oraciones utilizadas en los resúmenes, donde el

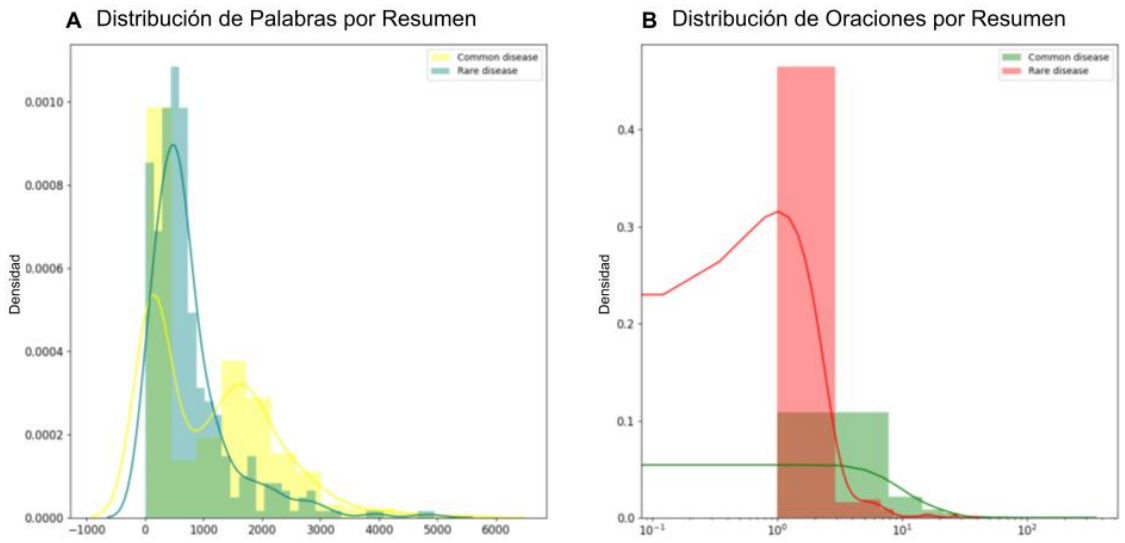


Figura 3.2: Distribuciones de palabras y oraciones a partir de la totalidad de resúmenes de alta sin procesar, paneles A y B respectivamente.

máximo para las enfermedades comunes es de 340 oraciones correspondientes a un único resumen de alta, mientras que el resto se encuentra entre 1 y 70 oraciones. Con respecto a las enfermedades raras, la cantidad de oraciones utilizadas se encuentra entre 1 y 39 oraciones.

Por otra parte, se analizaron los contenidos de los resúmenes de alta a través de nubes de palabras. En la Figura 3.3 Panel A se puede ver la nube de las 50 palabras más frecuentes de todo el conjunto de datos. Del panel B de la Figura 3.3 se pueden observar las palabras más usadas para las enfermedades comunes, que son términos comunes para el ámbito de la salud incluyendo, sangre, dolor y paciente. A su vez, palabras que no están relacionadas con un tratamiento en particular, cápsula, diario y horas. Por otro lado en el panel C de la Figura 3.3 se obtienen palabras más

relacionadas con tratamientos más invasivos y estudios más avanzados como, tumor, tomografía, eliminación y masa.

Figura 3.3: Nube de palabras, para el corpus completo Panel A, para las enfermedades comunes Panel B y para las enfermedades raras Panel C.

Adicionalmente, la Figura 3.2 Panel A muestra la distribución de palabras por historia clínica. La distribución para las notas de enfermedades raras es aproximadamente normal, lo cual es bueno para los modelos de clasificación, aunque hay valores atípicos por encima de las 3500 palabras. La estimación de la densidad se muestra con una curtosis estrecha (leptocúrtica) y tendencia central alrededor de 500. Para el caso de las enfermedades comunes parece ser bimodal aunque la tendencia central está en 100.

3.3. Métodos de evaluación

3.3.1. Exactitud

La exactitud o *accuracy* se define como el porcentaje de predicciones correctas que realiza el modelo. Si las predicciones en el conjunto de prueba son iguales a las categorías originales de los textos, el valor será 1, de lo contrario será 0. El cálculo se puede ver en la Ecuación 3.1

$$accuracy(y, \widehat{y}) = \frac{1}{n_{muestras}} * \sum_{i=0}^{n_{muestras}-1} i(\widehat{y} = y_i) \tag{3.1}$$

donde $\widehat{y}$ es la categoría predicha de la muestra i e y es la categoría real de la muestra. $i(x)$ es una función indicadora. Debido a la fácil interpretación y cálculo, la misma se usa habitualmente.

3.3.2. Área Debajo de la Curva

El Área Debajo de la Curva o AUC, corresponde al área debajo de la curva ROC. La ROC es un gráfico que muestra el desempeño de un modelo de clasificación para todos los umbrales de clasificación. Se utilizan dos parámetros:

- **Tasa de verdaderos positivos (TPR):** $TPR = \frac{TP}{TP+FN}$ siendo TP, verdaderos positivos y FN, falsos negativos.
- **Tasa de falsos positivos (FPR):** $FPR = \frac{FP}{FP+TN}$ siendo FP, falsos positivos y TN, verdaderos negativos.

Entonces la curva ROC gráfica TPR vs. FPR con diferentes umbrales de clasificación. Al bajar el umbral se clasifican más elementos como positivos, así incrementando tanto los falsos positivos como los verdaderos positivos.

3.3.3. F1 score

El valor de F1 está definida como la media armónica de precisión (P) y recuperación (R)(Sasaki, 2007), como se ve en la Ecuación 3.2

$$F = \frac{2PR}{P+R} \tag{3.2}$$

La precisión es el número de los resultados positivos verdaderos divido por el número de resultados positivos totales. La recuperación se define como la división entre el número de verdaderos positivos sobre el número de resultados que tendrían que haber sido clasificados como positivos.

El valor puede oscilar entre 0 y 1, siendo 0 cuando la precisión o la recuperación son 0 y vale 1 cuándo hay perfecta precisión y recuperación.

3.4. Modelos para la clasificación de HC con enfermedades raras

En esta sección se describirán las arquitecturas y características de los modelos de aprendizaje automático utilizados. Comenzando por modelos simples, aquellos que tienen fórmulas matemáticas relativamente sencillas, por lo que son fácilmente entendibles y aplicables comparado a modelos más complejos como los basados en las redes neuronales artificiales. Luego, se describirán modelos de aprendizaje profundo entre los que se encuentran LSTM, CNN y por último los transformers.

3.4.1. Modelos simples

Para estos modelos, la representación del texto se hizo utilizando el vectorizador TF-IDF. Se aplicaron las normalizaciones L1 y L2, estas se encargan de ajustar los

valores obtenidos por el vectorizador para compensar por el bias agregado por la utilización de documentos largos. Esto se debe a que se multiplica la Frecuencia de Término y la Frecuencia de Documento inversa, como se menciona en la Ecuación 1.2. Entonces los documentos más largos tendrían términos con valores más altos debido a que aparecen más veces en el documento solo por ser más lagos. La normalización intenta arreglar el problema anterior. Se utilizaron dos tipos de normalizaciones, por un lado la denominada "L2" que utiliza la norma Euclidiana, una norma que asigna un número estrictamente positivo al largo o el tamaño de todos los vectores en un espacio vectorial, se define de la siguiente manera:

$$||\overrightarrow{u}|| = \sqrt{u_1^2 + u_2^2 + u_3^2 + ... + u_n^2} \tag{3.3}$$

Por otro lado, se tiene la normalización "L1", dada por la siguiente ecuación:

$$||\overrightarrow{u}|| = |u_1| + |u_2| + |u_3| + ... + |u_n| \tag{3.4}$$

que es una simple suma de los componentes del vector, se lo conoce como la distancia *Taxicab*.

También dentro de los parámetros del vectorizador se limitaron el número de características, 5000 o 10000 y también considerando las palabras o tri-gramas (conjunto de tres palabras) como características. Todas las combinaciones utilizadas se encuentran en la Tabla A1 del Anexo 1.

Regresión Logística: Para este modelo se probaron los algoritmos de optimización, *"newton-cg", "lbfgs", "liblinear", "sag", "saga"*.

Árboles de Decisión: En este modelo el parámetro variable fue la profundidad del árbol, variando entre 5 y 25 nodos.

Support Vector Machine: Los kernels utilizados para este modelo fueron *"linear", "poly", "rbf", "sigmoid"*.

Entrenamiento

Estos modelos fueron entrenados con 30 semillas aleatorias y se utilizó una partición del conjunto de datos de 80 % para el entrenamiento y el 20 % para el test. Las diferentes semillas se utilizaron con el objetivo de generar diferentes particiones de los datos y a su vez distintas configuraciones propias de los algoritmos. Para obtener las matrices de confusión y las historias mal clasificadas se utilizó únicamente el modelo con la mejor semilla.

3.4.2. Modelos de aprendizaje profundo

Se utilizaron modelos de redes neuronales profundas como LSTM y CNN, pero también se utilizaron dos arquitecturas compuestas (combinación de las redes previamente mencionadas) para poder aprovechar las ventajas de ambas arquitecturas. La entrada de estos modelos son los embeddings simples, es decir, representaciones estáticas de palabras. Luego de diversos experimentos para elegir los embeddings correctos, entre los que se encontraron el uso de embeddings propios armados con el corpus de entrenamiento (sin resultados favorables debido al tamaño de dicho corpus), se consideraron embeddings preentrenados disponibles para su libre uso (de *word2vec*, *glove* y *fastext*, sin resultados óptimos debido al carácter general de los mismos). Por lo que se decidió por una combinación de nuestros propios vectores preentrenados (se obtuvieron a partir de señales y síntomas de enfermedades raras) y otros específicos para el reconocimiento de texto médico (*GitHub - yao8839836/obesity: obesity challenge*, 2018).

Arquitecturas

Todas las arquitecturas para los 4 modelos, LSTM, CNN, LSTM con CNN, CNN con LSTM, poseen una capa de entrada para el texto a clasificar, esta tiene dos puntos de entrada debido a que posteriormente se utilizan dos matrices de vectores de palabras *(embedding)* diferentes. El resultado de esta operación es una matriz donde cada palabra del texto original tiene asignado un vector de dimensión 200. En caso de que la palabra no existiese en la matriz de *embeddings* preentrenados, el *embedding* resultante tendría la misma dimensión pero es un vector de ceros. Luego, las dos salidas se concatenan para generar un vector con una dimensión de 400 para cada palabra. A partir de este paso la arquitectura varía según el modelo, como se explicará a continuación. En lo que respecta a las combinaciones de hiperparámetros utilizadas en este estudio, como no hay un criterio preestablecido sobre la cantidad de capas, neuronas, funciones de activación y demás hiperparámetros de los modelos, se realizaron diversas pruebas hasta concluir en un conjunto adecuado de los mismos. Luego, se aplicaron esos valores a los 4 modelos y sólo se variaron las semillas con las que se inicializaban los mismos.

LSTM

En la Figura 3.4 se puede ver la arquitectura completa de este modelo. Luego de la capa de concatenación, se encuentra una única capa de LSTM, en este caso con 64 unidades pero se probaron diversas cantidades de unidades. A su vez tiene una única capa densa con los dos posibles *outputs*.

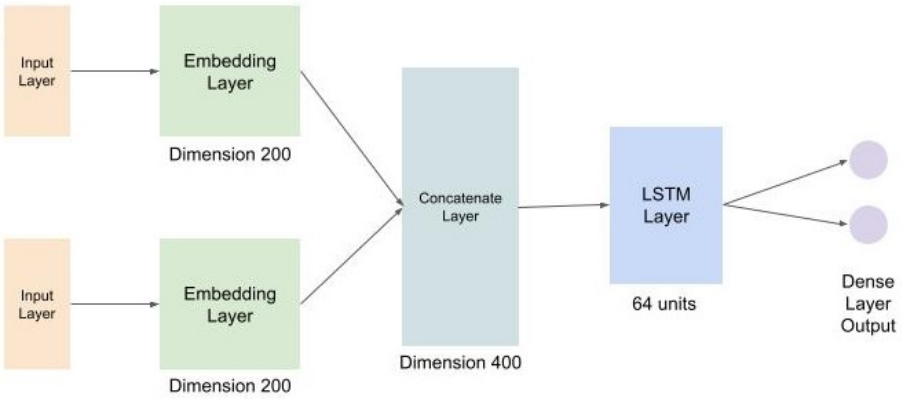


Figura 3.4: Arquitectura del modelo con LSTM.

Los parámetros variables de esta arquitectura son la cantidad de unidades en la capa de LSTM, variando entre 8, 16, 32 y 64, aunque en la figura se muestran 64.

CNN

La arquitectura de este modelo se puede ver en la Figura 3.5. Esta cuenta con 4 capas convolucionales de 1 sola dimensión, pero con tamaños de kernel diferentes, entre 1 y 4. Uno de los hiperparámetros es la cantidad de filtros, que en este caso varía entre los siguientes valores: 10, 25, 50 y 70. Luego se realiza una agrupación de las matrices de características provenientes del paso anterior, para luego concatenar todas las salidas de las 4 capas y se usan como input para una capa totalmente conectada de 64 neuronas. Previo a la capa de salida se encuentra una capa de *dropout*, la cual es utilizada para reducir el *overfitting* o sobre-ajuste con un valor de 0.2 (valor por defecto).

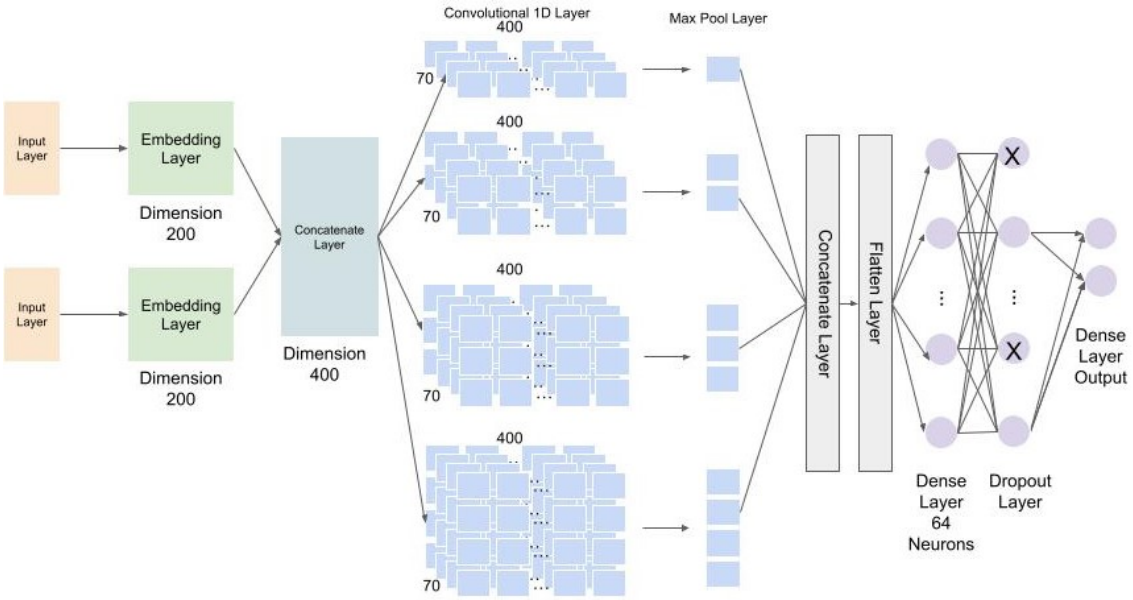


Figura 3.5: Arquitectura del modelo CNN.

CNN + LSTM

La diferencia principal entre la arquitectura de este modelo y la de CNN es la presencia de una capa de LSTM entre la concatenación y la salida. Este modelo posee dos hiperparámetros configurables, por un lado la cantidad de filtros que varía entre 10, 25 y 50 y además la cantidad de unidades de la capa de LSTM, 16, 32 y 64. Por último, se encuentra una capa neuronal totalmente conectada y un *dropout*. Esta arquitectura se puede ver en la Figura 3.6.

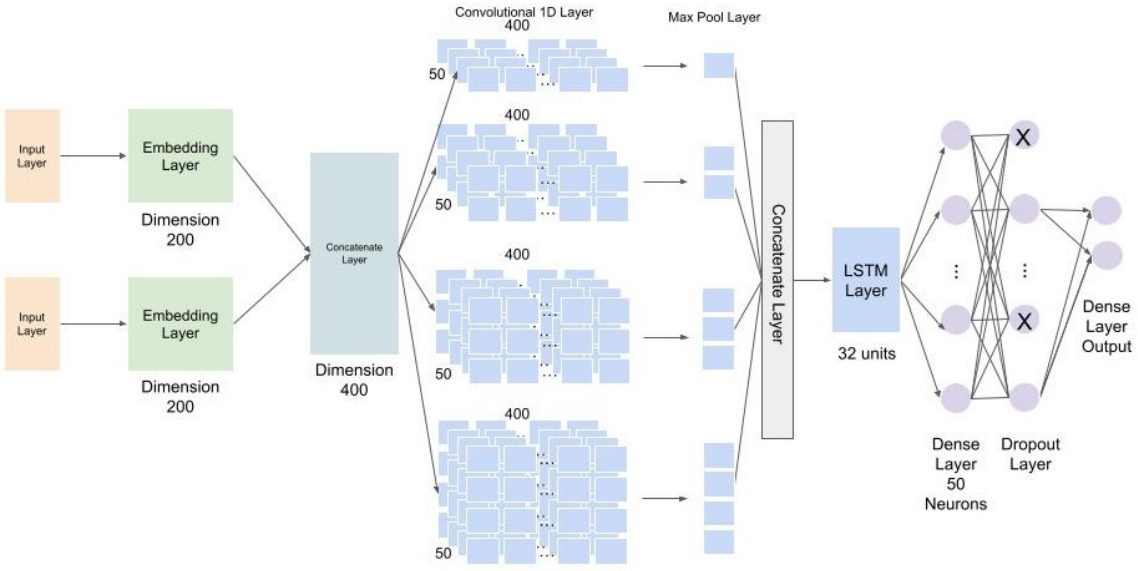


Figura 3.6: Arquitectura del modelo CNN + LSTM.

LSTM + CNN

Este modelo en comparación con el anterior tiene invertido el orden de la capa de LSTM y CNN, además no está presente la última capa totalmente conectada para disminuir la complejidad del modelo. La arquitectura se puede ver en la Figura 3.7

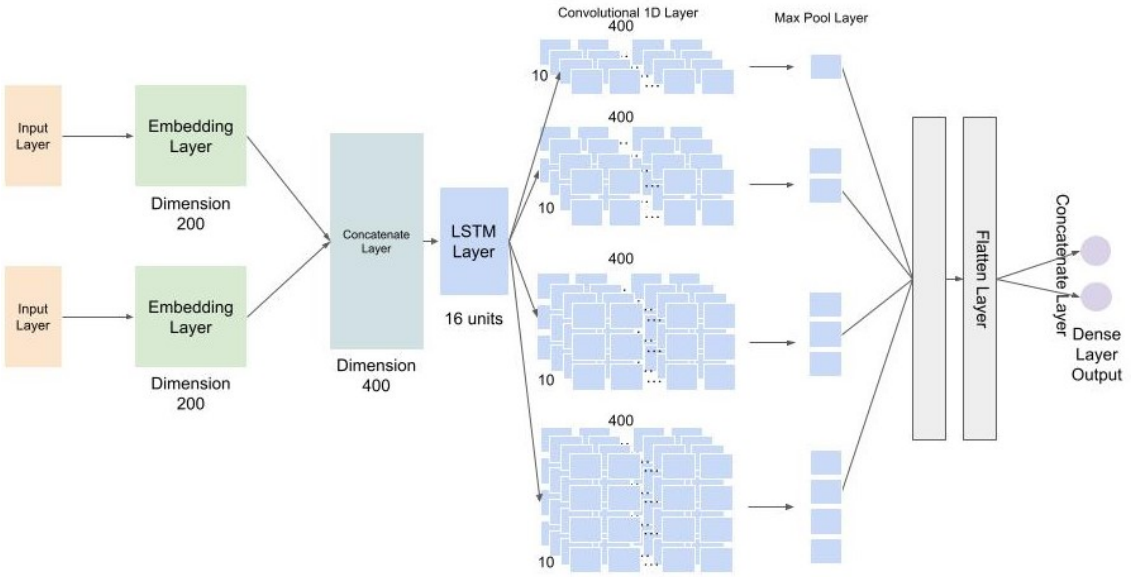


Figura 3.7: Arquitectura del modelo LSTM y CNN.

Los hiperparámetros de este modelo son los mismos que para el modelo anterior.

Otro punto de variación fue el optimizador utilizado por cada modelo, se utilizaron "Adadelta", "Adagrad", "Adam", "Adamax", "Ftrl", "RMSprop", "SGD" y "Nadam".

Si se toman en cuenta todas las opciones posibles de hiperparámetros para cada modelo, resultan para el caso de LSTM y CNN, en 96 combinaciones diferentes utilizadas, mientras que, para LSTM+CNN y CNN+LSTM 216 combinaciones diferentes que se probaron.

Entrenamiento

El entrenamiento de estos modelos involucró el uso de 3, 5 o 10 *epochs*, esto significa la cantidad de veces que la totalidad de los datos pasaron por el modelo para entrenarlo. Los datos fueron divididos en 80 % para entrenar, 20 % para las

pruebas. A partir de estos últimos se separó un 10 % para las pruebas de validación de los modelos.

Entrenar el modelo se refiere a la fase en la cual el modelo "aprende" a partir de los datos para posteriormente hacer predicciones sobre datos desconocidos. Durante el entrenamiento, el modelo es expuesto a los datos catalogados, esto quiere decir que el *input* está asociado a la categoría correspondiente. En este caso sería tener una enfermedad rara o tener una enfermedad común. El modelo va a ajustar los parámetros internos intentando disminuir la diferencia entre las predicciones y las categorías reales de los datos (mediante el algoritmo de backpropagation). Todo este proceso tiene como objetivo el aumento del desempeño del modelo frente a datos desconocidos.

En cuanto a la validación, esta es una parte del entrenamiento en la que luego del aprendizaje de los parámetros de la red, se prueba el modelo con algunos datos que se han separado previamente para poder evaluar el funcionamiento.

Por último, en la prueba de los modelos, estos son enfrentados a datos no mostrados en el entrenamiento y se evalúa la capacidad del modelo al realizar predicciones. En este paso no se realizan ajustes a los parámetros. Se pueden realizar diversas mediciones entre las que se encuentran la exactitud y el área bajo la curva.

Para estos modelos primero se utilizaron las 30 semillas aleatorias utilizadas para los modelos simples en un único conjunto de hiperparámetros por modelo. Luego se eligió la semilla en la que el modelo tuvo un desempeño mayor, para utilizar esta semilla con todas las combinaciones posibles de hiperparámetros para cada modelo.

3.4.3. Modelos basados en *Transformers*

Este tipo de modelos utiliza representaciones contextualizadas de palabras que luego son usadas como entradas de diferentes codificadores apilados. Cada codificador encapsula una capa de *selfattention* y una capa de retroalimentación o *feedfoward* que contribuyen a obtener mejores representaciones, considerando el contexto semántico y sintáctico de cada palabra en la secuencia de entrada. Estos modelos son preentrenados con enormes cantidades de datos de manera no supervisada y luego son *finetuned* o afinados con datos específicos a la tarea a realizar, en este caso el dataset de historias clínicas.

Para esto se seleccionó BERT debido a que, aparte de su buen rendimiento en tareas de procesamiento del lenguaje natural, hay varios sub modelos preentrenados con textos médicos que nos son útiles. Se comparará el desempeño de BERT y Bio_ClinicalBERT, este es una versión afinada con datos de condiciones médicos del primer BERT basado en una gran cantidad de datos biomédicos llamada BioBERT (Lee y cols., 2019).

El modelo BERT fue preentrenado con Wikipedia en Inglés y con *General Book Corpus*. Por otro lado, Bio_ClinicalBERT fue preentrenado con resúmenes de PubMed y artículos completos de PMC y afinado con detalles clínicos de enfermedades.

Resultados

Todos los resultados para todos los modelos se encuentran en el Anexo 1 en la Tabla A1. También se encuentran las matrices de confusión para los mejores 5 resultados por modelo, tanto para los modelos simples y los modelos con redes neuronales, en las figuras A2 y A3 del Anexo 1.

4.1. Modelos simples

En la Tabla 4.2 se muestran los resultados para el mejor conjunto de parámetros para cada modelo. En esta Tabla se puede observar que los modelos LR y SVM tiene un buen y parecido desempeño mientras que DT se desempeña peor. A continuación se analizarán brevemente esos resultados.

	Exactitud	**F1-Score**	**AUC**
LR	0,923	0,922	0,925
SVM	**0,929**	0,927	0,931
DT	0,888	0,893	0,887

Tabla 4.2: Se muestra el mejor resultado por modelo.

4.1.1. Regresión logística

En el caso de este modelo, el mejor desempeño se dio bajo diversas combinaciones de parámetros de los vectorizadores y clasificador. En todos los casos en que *max_features* era igual a 5000 o 10000 se obtuvo una *accuracy* de 0,923, un valor de F1-Score de 0,922 y AUC de 0,25, independientemente del tipo de optimizador del regresor utilizado. Cuando no es utilizado el límite de *features* (vocabulario) sí hay variaciones, donde el modelo que mejor desempeña es aquel que utiliza *stop_words* para eliminar palabras del vocabulario y el optimizador saga para el regresor.

La configuración de vectorizador que peor se desempeña es cuando se usa "L1" como algoritmo de normalización en lugar del uso de "L2".

4.1.2. Support Vector Machine

Para este modelo sí se encuentran diferencias en los resultados, en donde la mejor combinación de normalizador del vectorizador con el kernel son, "L1" y *sigmoid*, respectivamente. Con dichas variables, se obtuvo un accuracy de 0,929, un F1-Score de 0,928 y un AUC de 0,931. Las 5 mejores combinaciones de resultados se muestran

en la Tabla 4.4.

En este modelo hay mucha variación en los resultados respecto a la combinación de parámetros. En el caso de kernels, el tipo *poly* es el que peor se desempeña pero, por otro lado, con el resto de kernels se obtienen resultados dispersos dependiendo de que vectorizador se use.

Vectorizador	**Kernel**	**Accuracy**	**AUC**
Normalizador L1	sigmoid	**0,929**	**0,931**
Normalizador L2	sigmoid	0,923	0,926
Normalizador L2 y trigramas	linear	0,923	0,926
Normalizador L2	linear	0,917	0,919
Normalizador L2 y binarización	rbf	0,917	0,919

Tabla 4.4: Descripción de los parámetros de las 5 mejores combinaciones para SVM.

En el caso de los vectorizadores no hay una combinación de parámetros del mismo que refleje un buen desempeño sin tener en cuenta un kernel asociado.

4.1.3. Árboles de decisión

Para este modelo el mejor resultado de *accuracy* fue de 0,888, acompañado de un F1-Score de 0,893 y AUC de 0,887, utilizando el Normalizador "L2" con binarización y la profundidad del árbol de 23. No se observa una diferencia significativa cuando varía la profundidad del árbol. Las mejores 30 combinaciones (de un total de 120) de vectorizadores y distintos niveles de profundidad tienen una variación de la *accuracy* de 0,888 a 0,864. En la Tabla 4.6 se pueden ver los mejores 5 resultados para este modelo.

Vectorizador	**Profundidad del árbol**	**Accuracy**	**AUC**
Normalizador L2 y binarización	**23**	**0,888**	**0,887**
Normalizador L2 y binarización	9	0,881	0,882
Normalizador L1	6	0,882	0,881
Normalizador L2	9	0,876	0,876
Normalizador L2	23	0,875	0,876

Tabla 4.6: Descripción de los parámetros de las 5 mejores combinaciones para Decision Trees.

4.2. Modelos de Redes Neuronales

En la Tabla 4.8 se muestra el mejor resultado por modelo. La CNN es mejor que LSTM en F1-score pero LSTM es mejor que CNN en accuracy y AUC (0,899 y 0,893 en accuracy, respectivamente). En esta sección se detallan los resultados obtenidos por modelo.

	Accuracy	**F1-Score**	**AUC**
CNN	0,893	0,906	0,944
LSTM	**0,899**	**0,898**	**0,952**
CNN + LSTM	0,893	0,877	0,922
LSTM + CNN	0,888	0,887	0,920
Bio_ClinicalBERT	0,882	0,873	0,881
BERT	0,858	0,852	0,858

Tabla 4.8: Se muestra el mejor resultado por modelo.

4.2.1. LSTM

Para este tipo de red neuronal se obtuvo el valor máximo de AUC (0,952) acompañado de un F1-Score de 0,898 y *Accuracy* de 0,899, cuando es utilizado el optimizador "Adam" y 64 unidades dentro de la capa LSTM. Los errores respecto a la clasificación de historias corresponden únicamente a enfermedades raras, no tiene problemas al clasificar las enfermedades comunes. Esto se puede ver en la Figura 4.1.

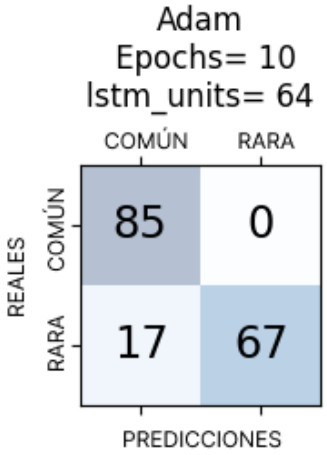


Figura 4.1: Matriz de confusión para el modelo LSTM. En la parte superior se observan los parámetros del modelo.

Los mejores 30 resultados se encuentran por encima de 0,9 aunque no se evidencia un patrón vinculado a la combinación de hiperparámetros utilizada. Aunque el optimizador "Ftrl" es el que peor desempeña en general, sin importar el resto de variables. Si se promedia la *accuracy* por optimizador, solo dos logran un valor superior a 0,860, siendo estos "Nadam" y"Adamax" Si se toman en cuenta únicamente la cantidad de *epochs*, el promedio de la accuracy es de 0,678, 0,695 y 0,719 para 3, 5 y 10 *epochs*, respectivamente.

4.2.2. CNN

En este modelo se obtuvo el desempeño máximo medido por AUC de 0,945, acompañado de *accuracy* de 0,888 y F1-Score de 0,901, cuando el optimizador utilizado es "Adam" y se utilizan 70 filtros. Si tomamos en cuenta las historias mal clasificadas, se clasificaron mal 14 enfermedades raras y 5 enfermedades comunes de un total de 85 y 84, respectivamente. Esto se puede ver en la Figura 4.2.

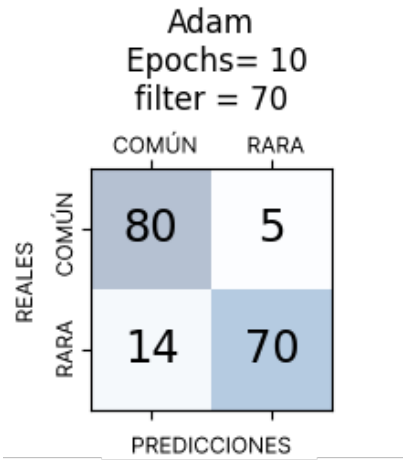


Figura 4.2: Matriz de confusión para el modelo CNN, se incluyen en la parte superior de la figura los parámetros utilizados, siendo la primera línea el optimizador.

En este modelo el optimizador con peor desempeño medido por *accuracy* es "Ftrl" y los mejores dos son "Adam" y "Nadam". En el caso del promedio de *accuracy* en base únicamente a la cantidad de *epochs* no hay grandes diferencias entre 3, 5 y 10 *epochs*.

4.2.3. CNN + LSTM

Esta combinación obtuvo un desempeño máximo de 0,935 en AUC, 0,888 en F1-Score y 0,888 en *accuracy*. El resultado anterior se obtuvo cuándo la cantidad de unidades de la capa de LSTM era de 32 y la cantidad de filtros de CNN eran 50. Teniendo en cuenta las historias mal clasificadas fueron, 16 y 3, siendo estas enfermedades raras y enfermedades comunes, de un total de 84 y 85, como se puede ver en la Figura 4.3.

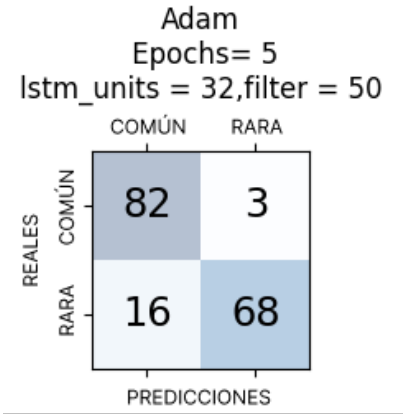


Figura 4.3: Matriz de confusión para el modelo CNN + LSTM. En la parte superior de la figura se pueden ver los parámetros del modelo.

En este caso el optimizador que peor se desempeñó fue “Ftrl”, aunque el resto menos “Adam” y “Nadam”, que dieron los mejores resultados, no están alejados del promedio de desempeño de “Ftrl”. En este caso tampoco hay grandes diferencias entre el uso de diferentes cantidades de *epochs*.

4.2.4. LSTM + CNN

Esta arquitectura siendo inversa a la anterior obtuvo un desempeño máximo medido por AUC de 0,927, acompañado por un F1-Score de 0,833 y *accuracy* de 0,840. Estos valores se obtuvieron cuando el optimizador era “Adam”, la cantidad de unidades de la capa de LSTM era de 16 y 10 filtros en la capa convolucional. Si se toman en cuenta las historias mal clasificadas, clasificó erróneamente 19 enfermedades raras de un total de 84 y 8 enfermedades comunes de un total de 85. Esto se puede ver en la Figura 4.4.

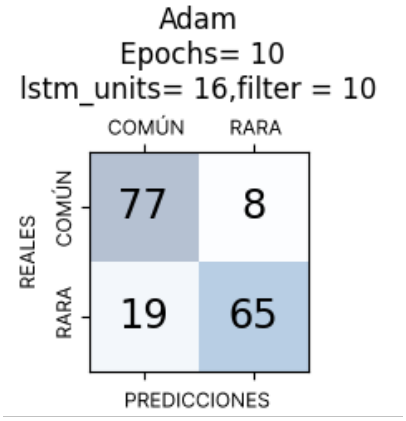


Figura 4.4: Matriz de Confusión para el modelo LSTM + CNN, en la parte superior de la figura se pueden observar los parámetros utilizados.

Por otro lado, los optimizadores que mejor se desempeñaron en general fueron "Adam" y "Nadam". En este caso sí hay una diferencia en los resultados si se tiene en cuenta la cantidad de *epochs*, dado que cuando la cantidad es 10 se obtienen mejores resultados.

4.2.5. Transformers

Las últimas dos filas de la Tabla 4.8 muestran los resultados con los modelos basados en *transformers*, donde Bio_ClinicalBERT obtuvo mejor resultado que BERT. Bio_ClinicalBert obtuvo un valor de *accuracy* de 0,882, F1-Score de 0,873 y 0,881 de AUC. En relación con las historias clínicas mal clasificadas, 5 enfermedades raras fueron mal clasificadas de un total de 74, y 15 enfermedades comunes de un total de 95. Esto se puede ver en la Figura 4.5.

(a) BERT (b) BioBERT

Figura 4.5: Matrices de confusión para los modelos BERT (a) y Bio_ClinicalBERT (b)

Discusión

Para determinar cuál fue el modelo que mejor se desempeñó, se decidió utilizar el valor de AUC. Estudios anteriores han demostrado que el valor de AUC puede proveer información valiosa sobre el rendimiento del clasificador, sobre la capacidad del modelo de distinguir entre las clases y su impacto en un tipo de error determinado, como los falsos positivos y los falsos negativos en el marco del procesamiento del lenguaje natural (Provost y Fawcett, 2001). Por lo que, en nuestro caso, un nivel mayor de AUC significaría que el modelo es mejor para discernir si una historia clínica pertenece a una enfermedad rara o a una enfermedad común.

Dicho esto, los modelos LSTM y CNN fueron los que obtuvieron los mejores valores de AUC (0,95 y 0,94, respectivamente), lo que indica que pueden clasificar de forma correcta, aunque sean modelos relativamente pequeños al compararlos con los *transformers*. Es importante resaltar que el modelo LSTM tiene un mejor desempeño que las combinaciones de LSTM y CNN, esto nos indica que LSTM puede comprender y analizar relaciones de dependencia a lo largo de secuencias de palabras extensas sin necesitar la ayuda de operaciones convolucionales. La Figura 5.1 muestra la matriz de confusión en el Panel A para el modelo LSTM y su curva ROC en el panel B de la que proviene el valor de AUC.

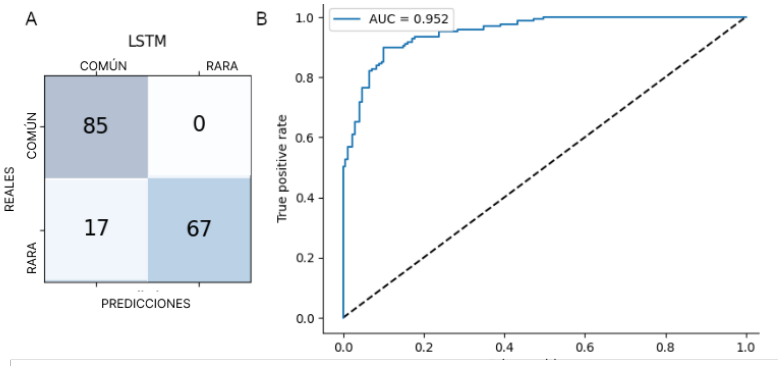


Figura 5.1: *Matriz de Confusión para el modelo LSTM (A) y la curva ROC (B) para dicho modelo.*

En relación con los *transformers* el modelo basado en Bio_ClincalBERT se desempeña mejor que BERT y esto se puede deber a que Bio_ClincalBERT tiene un lenguaje más relacionado con la medicina. Teniendo en cuenta la totalidad de los resultados, se puede ver como BERT y Bio_ClinicalBERT se desempeñan de forma parecida al resto. Lo anterior indicaría que para algunas tareas no es necesario que los modelos sean ultra-complejos, es decir, que aumentar la complejidad de los modelos no es sinónimo de mejores resultados.

5.1. Historias Clínicas sistemáticamente mal clasificadas

El error más común en las historias son las enfermedades raras clasificadas como comunes. En la mayoría de los modelos, sean básicos o no, una mediana de $87,25\,\%$ de todas las historias clínicas mal clasificadas corresponden a enfermedades raras. El *outlier* fueron los árboles de decisión que solo clasificó erróneamente $47\,\%$ de las enfermedades raras. Las historias mal clasificadas se pueden ver en

github.com/matiasr1608/rare_disease_corpus. Una mediana del 80, 95 % de las enfermedades raras mal clasificadas corresponden a las previamente clasificadas por el estudio de (Dong y cols., 2021). Nuestros modelos se basan en las etiquetas generadas por Dong y cols. (2021) y otras fuentes (sección 3.2), por lo que una mala clasificación inicial puede impactar en las performances de los modelos. La segunda fuente con más historias raras mal clasificadas son las que fueron traducidas a partir de los encares médicos obtenidos de la Oficina del Libro de la Facultad de Medicina. Como ya fue mencionado, la traducción puede haber introducido errores en la creación de los modelos y afectado la posterior clasificación de las mismas.

Con respecto únicamente a las historias mal clasificadas por el modelo LSTM, todos los errores corresponden a historias de enfermedades raras clasificadas como comunes. Estas 17 historias fueron revisadas por un médico genetista y se encontró que la gran mayoría son enfermedades comunes pero que fueron mal clasificadas previamente debido a que lo que describen esos resúmenes de alta son complicaciones infrecuentes de procedimientos/enfermedades comunes, probablemente de pacientes de edad avanzada. Por lo tanto, el texto clínico se hace muy largo, complicado, con varias interacciones con procedimientos, personal de especialidades médicas, fármacos, intervenciones, etc. Por lo que la clasificación inicial de estos resúmenes era un desafío desde el comienzo y puede ser una explicación de los errores de clasificación. Estas historias y su correspondiente análisis se puede encontrar en la Tabla A2 del Anexo 1. A pesar de que el corpus puede tener algo de ruido en las etiquetas (lo cual es un escenario realista en el contexto de varias aplicaciones), los clasificadores son capaces de funcionar bastante bien en la práctica.

5.2. Posibles aplicaciones del trabajo

Las enfermedades raras son difíciles de detectar, diagnosticar y de tratar. Pacientes con enfermedades raras se mueven dentro de un sistema de salud de forma extensa, infectiva y con costos económicos que derivan de esta inefectividad. Un diagnóstico precoz de una enfermedad rara, conlleva estrategias tempranas para tratar la enfermedad o para poder convivir con la misma pero con un diagnóstico claro. Esto permite una disminución de los costos y una disminución del impacto que la "Odisea diagnóstica" pueda tener en pacientes de este tipo.

Este método se plantea para la detección precoz de enfermedades raras a partir de historias clínicas y registros médicos, como lo son resúmenes de alta de una unidad de emergencia y notas médicas. Cuando un resumen de alta se clasifica como un posible paciente con una enfermedad rara, se podrían iniciar medidas específicas para estos pacientes. Dentro de las mismas, se encuentra la consulta con un genetista y a partir de ahí pueden surgir diversos estudios moleculares que puedan apresurar un diagnóstico preciso.

Al obtener modelos con *accuracys* mayores al 90 % implica que en ese porcentaje de casos la clasificación fue correcta y, por otro lado, un 10 % mal clasificado. En el caso de que se clasifique una persona con una enfermedad común erróneamente con una enfermedad rara, el costo de este error sería meramente económico para el sistema de salud, puesto que se incluirían en el algoritmo diagnóstico una consulta con un genetista que podría ser innecesaria. Este costo no parece muy alto, teniendo en cuenta lo que se ahorraría el sistema gracias a los pacientes bien clasificados. En este caso, el ahorro generado por pacientes clasificados correctamente con una posible enfermedad rara, posiblemente sea mucho mayor al costo producido por los errores

de la clasificación. Por otro lado, un paciente que sea clasificado con una enfermedad común pero que tiene una enfermedad rara, tendría un impacto mucho mayor en la calidad de vida de los pacientes, debido a que estos irían por los caminos clásicos de diagnóstico y podrían entrar en la "Odisea diagnóstica", generando así costos mayores.

Conclusiones

Se generó un corpus para la clasificación de enfermedades raras. Este se formó a partir de diversas fuentes de forma balanceada y detallada. Para comprobar la posible validez de este dataset se utilizaron diversos modelos de aprendizaje automático, desde los modelos más simples hasta los *transformers*. El modelo con el mejor desempeño fue el LSTM, con un valor de AUC de 0,952. Por lo que se concluye que el modelo basado LSTM es capaz de predecir de forma acertada la condición de la enfermedad de un paciente, ya sea una enfermedad rara o común. Esto permite la posibilidad de ser utilizado en un proceso de un sistema de salud y así llevar a un mejor camino de diagnóstico para el paciente.

Trabajo futuro

El trabajo futuro se basa en la mejora de la optimización de los modelos que ya están relacionados con el tema, como por ejemplo Bio_ClinicalBERT, usando nuestro propio dataset. Esto va a mejorar los resultados debido a que se podrán entender mejor las palabras técnicas en su contexto. Por otro lado, se entrenaron y probaron modelos cuyo fuerte no es la explicación de sus decisiones. En algunos modelos no somos capaces de entender por qué un modelo en particular clasifica a una historia clínica como enfermedad rara. Al entender estos resultados se podría aumentar la comprensión de las enfermedades raras en general y también aprender a escribir resúmenes de alta de una forma que los modelos los puedan comprender mejor. También se espera poder aumentar el tamaño del dataset, se intentará incluir historias clínicas nuevas y curar manualmente las ya existentes, incrementando la eficacia de los modelos.

Referencias

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems.* Descargado de https://www.tensorflow.org/ (Software available from tensorflow.org)

Bello, F. L., Naya, H., Raggio, V., y Rosá, A. (2019). From medical records to research papers: A literature analysis pipeline for supporting medical genomic diagnosis processes. *Informatics in Medicine Unlocked*, *15*, 100181. Descargado de https://doi.org/10.1016%2Fj.imu.2019.100181 doi: 10.1016/j.imu.2019.100181

Bridle, J. S. (1990). Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. En *Neurocomputing* (pp. 227–236). Springer Berlin Heidelberg. Descargado de https://doi.org/10.1007%2F978-3-642-76153-9_28 doi: 10.1007/978-3-642-76153-9_28

Burges, C. J. (1998). *Data Mining and Knowledge Discovery*, *2*(2), 121–167. Descargado de https://doi.org/10.1023%2Fa%3A1009715923555 doi: 10.1023/a:1009715923555

Chollet, F., y cols. (2015). *Keras.* https://keras.io.

Churchland, P. S., y Sejnowski, T. J. (1992). *The computational brain.* MIT Press.

Clark, M. M., Stark, Z., Farnaes, L., Tan, T. Y., White, S. M., Dimmock, D., y Kingsmore, S. F. (2018, jul). Meta-analysis of the diagnostic and clinical utility of genome and exome sequencing and chromosomal microarray in children with suspected genetic diseases. *npj Genomic Medicine*, *3*(1). Descargado de https://doi.org/10.1038%2Fs41525-018-0053-8 doi: 10.1038/s41525-018-0053-8

Defazio, A., Bach, F., y Lacoste-Julien, S. (2014). Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in neural information processing systems*, *27*.

Devlin. (2019, June). BERT: Pre-training of deep bidirectional transformers for language understanding. En *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)* (pp. 4171–4186). Minneapolis, Minnesota: Association for Computational Linguistics. Descargado de https://aclanthology.org/N19-1423 doi: 10.18653/v1/N19-1423

Dong, H., Suarez-Paniagua, V., Zhang, H., Wang, M., Whitfield, E., y Wu, H. (2021, nov). Rare disease identification from clinical notes with ontologies and weak supervision. En *2021 43rd annual international conference of the IEEE engineering in medicine biology society (EMBC).* IEEE. Descargado de https://doi.org/10.1109%2Fembc46164.2021.9630043 doi: 10.1109/embc46164.2021.9630043

Esteva, A., Robicquet, A., Ramsundar, B., Kuleshov, V., DePristo, M., Chou, K., ... Dean, J. (2019, jan). A guide to deep learning in healthcare. *Nature Medicine*, *25*(1), 24–29. Descargado de https://doi.org/10.1038%2Fs41591-018-0316-z doi: 10.1038/s41591-018-0316-z

EURODIS. (2009). *The voice of 12,000 patients. experiences and expectations of*

rare disease patients on diagnosis and care in europe.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., y Lin, C.-J. (2008). Liblinear: A library for large linear classification. *the Journal of machine Learning research*, *9*, 1871–1874.

Fernández, A. A., Martín, A. P., Allés, G. P., Delgado, E. G., y Martínez, M. I. (2012, oct). Percepción de las enfermedades raras por el médico de atención primaria. *SEMERGEN - Medicina de Familia*, *38*(7), 421–431. Descargado de https://doi.org/10.1016%2Fj.semerg.2012.02.011 doi: 10.1016/j.semerg.2012.02.011

Github - yao8839836/obesity: obesity challenge. (2018). Descargado 2023-08-02, de https://github.com/yao8839836/obesity

Goodfellow, I., Bengio, Y., y Courville, A. (2016). *Deep learning.* MIT Press. (http://www.deeplearningbook.org)

Google colab. (2023). https://research.google.com/colaboratory/faq.html. (Accessed: 2023-06-06)

Gurney, K. (1997). *Introduction to neural networks.* Taylor & Francis Group.

Hanke, M. (1997). Regularizing properties of a truncated newton-cg algorithm for nonlinear inverse problems. *Numerical Functional Analysis and Optimization*, *18*(9-10), 971–993.

Haykin, S. S. (2008). *Neural networks and learning machines.* Prentice Hall.

Hochreiter, S., y Schmidhuber, J. (1997, nov). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. Descargado de https://doi.org/10.1162%2Fneco.1997.9.8.1735 doi: 10.1162/neco.1997.9.8.1735

Huang, S., Cai, N., Pacheco, P., Narrandes, S., Wang, Y., y Xu, W. (2018, 01). Applications of support vector machine (svm) learning in cancer genomics. *Cancer genomics I& proteomics*, *15*, 41–51. doi: 10.21873/cgp.20063

Hwang, S., y Lee, B. (2022, feb). Machine learning-based prediction of critical illness in children visiting the emergency department. *PLOS ONE*, *17*(2), e0264184. Descargado de https://doi.org/10.1371%2Fjournal.pone.0264184 doi: 10.1371/journal.pone.0264184

Johnson, A. E., Pollard, T. J., Shen, L., wei H. Lehman, L., Feng, M., Ghassemi, M., ... Mark, R. G. (2016, may). MIMIC-III, a freely accessible critical care database. *Scientific Data*, *3*(1). Descargado de https://doi.org/10.1038%2Fsdata.2016.35 doi: 10.1038/sdata.2016.35

Jurafsky, D., y Martin, J. H. (2009). *Speech and language processing (2nd edition)*. USA: Prentice-Hall, Inc.

Kim, S.-W., y Gil, J.-M. (2019, aug). Research paper classification systems based on TF-IDF and LDA schemes. *Human-centric Computing and Information Sciences*, *9*(1). Descargado de https://doi.org/10.1186%2Fs13673-019-0192-7 doi: 10.1186/s13673-019-0192-7

Kingsford, C., y Salzberg, S. L. (2008). What are decision trees? *Nature biotechnology*, *26*(9), 1011–1013.

Lebret. (2014, April). Word embeddings through hellinger PCA. En *Proceedings of the 14th conference of the European chapter of the association for computational linguistics* (pp. 482–490). Gothenburg, Sweden: Association for Computational Linguistics. Descargado de https://aclanthology.org/E14-1051 doi: 10.3115/v1/E14-1051

Lecun, Y., Bottou, L., Bengio, Y., y Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. Descargado de https://doi.org/10.1109%2F5.726791 doi: 10.1109/5.726791

Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., y Kang, J. (2019, 09). Bio-

BERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, *36*(4), 1234-1240. Descargado de https://doi.org/10.1093/bioinformatics/btz682 doi: 10.1093/bioinformatics/btz682

Levy, O., y Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. En *Proceedings of the 27th international conference on neural information processing systems - volume 2* (p. 2177–2185). Cambridge, MA, USA: MIT Press.

Liu, H.-Y., Zhou, L., Zheng, M.-Y., Huang, J., Wan, S., Zhu, A., ... Lu, Y. (2019, dec). Diagnostic and clinical utility of whole genome sequencing in a cohort of undiagnosed chinese families with rare diseases. *Scientific Reports*, *9*(1). Descargado de https://doi.org/10.1038%2Fs41598-019-55832-1 doi: 10.1038/s41598-019-55832-1

Manning, C. D., Raghavan, P., y Schütze, H. (2008). *Introduction to information retrieval*.

McCulloch, W. S., y Pitts, W. (1943, dec). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, *5*(4), 115–133. Descargado de https://doi.org/10.1007%2Fbf02478259 doi: 10.1007/bf02478259

McMahan, H. B., Holt, G., Sculley, D., Young, M., Ebner, D., Grady, J., ... others (2013). Ad click prediction: a view from the trenches. En *Proceedings of the 19th acm sigkdd international conference on knowledge discovery and data mining* (pp. 1222–1230).

Meyer, E. J., Spangenberg, L., Ramírez, M. J., Sousa, S. M. C. D., Raggio, V., y Torpy, D. J. (2021, jun). CBG montevideo: A clinically novel serpina6 mutation leading to haploinsufficiency of corticosteroid-binding globulin. *Journal of the Endocrine Society*, *5*(9). Descargado de https://doi.org/

10.1210%2Fjendso%2Fbvab115 doi: 10.1210/jendso/bvab115

Mikolov, T., Chen, K., Corrado, G., y Dean, J. (2013). *Efficient estimation of word representations in vector space.*

Mitchell, T. M., y cols. (2007). *Machine learning* (Vol. 1). McGraw-hill New York.

Noble, W. S. (2006). What is a support vector machine? *Nature biotechnology*, *24*(12), 1565–1567.

OpenAI. (2021). *Gpt-3.5: Language models.* https://openai.com. (Accessed: 26 de junio de 2023)

pandas development team, T. (2020, February). *pandas-dev/pandas: Pandas.* Zenodo. Descargado de https://doi.org/10.5281/zenodo.3509134 doi: 10.5281/zenodo.3509134

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, Grisel, O., Blondel, M., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Provost, F., y Fawcett, T. (2001).

Machine Learning, *42*(3), 203–231. Descargado de https://doi.org/10.1023%2Fa%3A1007601015854 doi: 10.1023/a:1007601015854

Pubmed. (s.f.). Descargado 2023-06-26, de https://pubmed.ncbi.nlm.nih.gov/

RAE. (2023). *Definición de historia clínica - diccionario panhispánico del español jurídico - rae.* Descargado 2023-06-26, de https://dpej.rae.es/lema/historia-cl%C3%ADnica

Raggio, V., Dell'Oca, N., Simoes, C., Tapié, A., Medici, C., Costa, G., ... Spangenberg, L. (2021, may). Whole genome sequencing reveals a frameshift mutation and a large deletion in YY1ap1 in a girl with a panvascular artery disease. *Human Genomics*, *15*(1). Descargado de https://doi.org/10.1186%2Fs40246-021-00328-1 doi: 10.1186/s40246-021-00328-1

Ramos, J., y cols. (2003). Using tf-idf to determine word relevance in document queries. En *Proceedings of the first instructional conference on machine learning* (Vol. 242, pp. 29–48).

Rokach, L., y Maimon, O. (2005). Decision trees. En *Data mining and knowledge discovery handbook* (pp. 165–192). Springer-Verlag. Descargado de https://doi.org/10.1007%2F0-387-25465-x_9 doi: 10.1007/0-387-25465-x_9

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

Rumelhart, D. E., Hinton, G. E., y Williams, R. J. (1986, oct). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. Descargado de https://doi.org/10.1038%2F323533a0 doi: 10.1038/323533a0

Salton, G., y Buckley, C. (1988, jan). Term-weighting approaches in automatic text retrieval. *Information Processing Management*, *24*(5), 513–523. Descargado de https://doi.org/10.1016%2F0306-4573%2888%2990021-0 doi: 10.1016/0306-4573(88)90021-0

Salton, G., Wong, A., y Yang, C. S. (1975, nov). A vector space model for automatic indexing. *Communications of the ACM*, *18*(11), 613–620. Descargado de https://doi.org/10.1145%2F361219.361220 doi: 10.1145/361219.361220

Sasaki, Y. (2007, 01). The truth of the f-measure. *Teach Tutor Mater*.

Schaefer, J., Lehne, M., Schepers, J., Prasser, F., y Thun, S. (2020). The use of machine learning in rare diseases: a scoping review. *Orphanet journal of rare diseases*, *15*, 1–10.

Schlkopf, B., Smola, A. J., y Bach, F. (2018). *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. The MIT Press.

Shah, K., Patel, H., Sanghvi, D., y Shah, M. (2020, mar). A comparative analysis of logistic regression, random forest and KNN models for the text classifica-

tion. *Augmented Human Research*, *5*(1). Descargado de https://doi.org/10.1007%2Fs41133-020-00032-0 doi: 10.1007/s41133-020-00032-0

Sireau, N. (2013). *Rare diseases*. Routledge.

Suryakanthi, T. (2020, 01). Evaluating the impact of gini index and information gain on classification using decision tree classifier algorithm*. *International Journal of Advanced Computer Science and Applications*, *11*. doi: 10.14569/IJACSA.2020.0110277

Tong, S., y Koller, D. (2001). Support vector machine active learning with applications to text classification. *Journal of machine learning research*, *2*(Nov), 45–66.

Transformer's encoder-decoder: Let's understand the model architecture - kikaben. (2020). Descargado 2023-08-01, de https://kikaben.com/transformers-encoder-decoder/

Van Rossum, G., y Drake, F. L. (2009). *Python 3 reference manual*. Scotts Valley, CA: CreateSpace.

Vapnik, V. (1982). *Estimation of dependences based on empirical data: Springer series in statistics (springer series in statistics)*. Berlin, Heidelberg: Springer-Verlag.

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

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... Rush, A. M. (2020, October). Transformers: State-of-the-art natural language processing. En *Proceedings of the 2020 conference on empirical methods in natural language processing: System demonstrations* (pp. 38–45). Online: Association for Computational Linguistics. Descargado de https://www.aclweb.org/

anthology/2020.emnlp-demos.6

Yan, X., He, S., y Dong, D. (2020, mar). Determining how far an adult rare disease patient needs to travel for a definitive diagnosis: A cross-sectional examination of the 2018 national rare disease survey in china. *International Journal of Environmental Research and Public Health*, *17*(5), 1757. Descargado de https://doi.org/10.3390%2Fijerph17051757 doi: 10.3390/ijerph17051757

Zhou, y Chellappa. (1988). Computation of optical flow using a neural network. En *Ieee 1988 international conference on neural networks* (p. 71-78 vol.2). doi: 10.1109/ICNN.1988.23914

Zhu, C., Byrd, R. H., Lu, P., y Nocedal, J. (1997, dec). Algorithm 778: L-BFGS-b. *ACM Transactions on Mathematical Software*, *23*(4), 550–560. Descargado de https://doi.org/10.1145%2F279232.279236 doi: 10.1145/279232.279236

Anexo 1

.1. Log Loss

A los efectos de profundizar la *Log loss* utilizada para la Regresión Logística, se introducirá brevemente el mismo. *Log loss* es una forma de medir la efectividad del modelo para clasificar utilizando la probabilidad de que la clase sea real según el modelo. La probabilidad va a ser un valor entre 0 y 1, siendo esto el input para el cálculo de *log loss*. Se mide la disparidad entre la predicción y la categoría real, como se muestra en la ecuación 1

$$L_{log}(y,p) = -logPr(y|p) = -(ylog(p) + (1-y)log(1-p)) \quad (1)$$

donde y es la categoría real y $p = Pr(y = 1)$ es la probabilidad de que la predicción sea la categoría real.

.2. Tablas y Figuras

MÉTODOS BÁSICOS							
		RESULTADO PROMEDIO *					
MODELOS		vectorizadores					
		L2					L1*
		stop_words stop_words	stop_words binary	stop_words features 10k	stop_words, Features 5k	stop_words, trigrams	stop_words
	KERNEL						
SVM	linear	0,8826	0,8848	0,8793	0,8787	0,8732	0,4817
	poly	0,7963	0,8521	0,8049	0,8172	0,8767	0,5751
	rbf	0,8769	0,8821	0,8767	0,8767	0,8742	0,872
	sigmoid	0,8821	0,8836	0,8799	0,8791	0,8682	0,8838

	newton-cg	0,8661	0,8767	0,8686	0,87	0,8487	0,7274
	lbfgs	0,8663	0,8767	0,8696	0,8704	0,8487	0,7284
LR	liblinear	0,8661	0,8767	0,8686	0,87	0,8487	0,7274
	sag	0,8661	0,8767	0,8686	0,87	0,8487	0,7282
	saga	0,8663	0,8765	0,8696	0,8704	0,8483	0,7268
TREE	5	0,7953	0,8152	0,7961	0,7949	0,8174	0,8018
	6	0,8081	0,8164	0,8091	0,8043	0,8154	0,8065
	7	0,8126	0,8152	0,8085	0,8065	0,8095	0,8118
	8	0,8097	0,811	0,8116	0,8122	0,8134	0,8134
	9	0,8085	0,8087	0,8091	0,8075	0,8077	0,8099
	10	0,8108	0,803	0,8112	0,8081	0,8041	0,8081
	11	0,8073	0,8047	0,8116	0,8036	0,8034	0,8014
	12	0,8004	0,8028	0,8071	0,8067	0,7994	0,801
	13	0,8022	0,8041	0,8081	0,8016	0,8	0,8032
	14	0,8014	0,803	0,802	0,8039	0,7968	0,8028
	15	0,803	0,8006	0,7966	0,802	0,801	0,801
	16	0,8014	0,7994	0,8	0,7968	0,7986	0,7959
	17	0,8004	0,8032	0,802	0,802	0,7929	0,7959
	18	0,7984	0,8037	0,799	0,797	0,798	0,7943
	19	0,8	0,7996	0,8028	0,802	0,7949	0,7957
	20	0,8012	0,8008	0,7996	0,7982	0,7994	0,7959
	21	0,8016	0,7945	0,8016	0,8059	0,7911	0,8008
	21	0,8037	0,801	0,8028	0,7978	0,7937	0,7949
	23	0,8002	0,8024	0,8016	0,7907	0,7966	0,7947
	24	0,7996	0,7978	0,8047	0,7963	0,7959	0,7982

REDES NEURONALES**

			LSTM UNITS				
MODELO	**Optim-izador*** **	**Epochs**	8	16	32	64	
		3	0,864	0,893	0,834	0,882	
	Adam	5	0,846	0,852	0,864	0,834	

LSTM

		10	0,876	0,846	0,858	0,899	
		3	0,852	0,888	0,87	0,893	
	RMSprop	5	0,876	0,876	0,858	0,852	
		10	0,87	0,793	0,822	0,775	
			FILTROS CNN				
MODELO	**Optimizer*** **	**Epochs**	10	25	50	70	
		3	0,751	0,87	0,864	0,864	
	Adam	5	0,834	0,882	0,876	0,876	
CNN		10	0,882	0,876	0,858	0,888	
		3	0,734	0,846	0,87	0,876	
	Nadam	5	0,864	0,876	0,864	0,888	
		10	0,864	0,87	0,893	0,876	
				LSTM units			
MODELO	**Optimizador*** **	**Epochs**	**CNN filters**	16	32	64	
			10	0,882	0,834	0,858	
		3	25	0,846	0,852	0,864	
			50	0,87	0,858	0,84	
			10	0,888	0,858	0,846	
	Adam	5	25	0,858	0,876	0,84	
			50	0,864	0,888	0,846	
CNN_			10	0,876	0,84	0,87	
LSTM		10	25	0,858	0,864	0,888	
			50	0,84	0,87	0,811	
			10	0,864	0,828	0,858	
		3	25	0,864	0,864	0,864	
			50	0,87	0,864	0,864	
			10	0,882	0,852	0,846	
	Nadam	5	25	0,87	0,864	0,87	

			50	0,876	0,864	0,87	
			10	0,828	0,822	0,817	
		10	25	0,864	0,858	0,864	
			50	0,645	0,864	0,888	
				LSTM units			
MODELO	**Optim-izador*** **	**Epochs**	**CNN Filters**	16	32	64	
			10	0,84	0,828	0,834	
		3	25	0,846	0,852	0,84	
			50	0,846	0,846	0,864	
			10	0,858	0,876	0,852	
	Adam	5	25	0,858	0,882	0,846	
LSTM			50	0,805	0,858	0,864	
_CNN			10	0,84	0,852	0,864	
		10	25	0,846	0,84	0,817	
			50	0,846	0,852	0,87	
			10	0,817	0,793	0,864	
		3	25	0,846	0,84	0,817	
			50	0,817	0,84	0,822	
			10	0,84	0,84	0,846	
	Nadam	5	25	0,84	0,858	0,852	
			50	0,828	0,876	0,852	
			10	0,852	0,834	0,864	
		10	25	0,858	0,846	0,822	
			50	0,846	0,864	0,858	

ENCODERS			
	Accuracy	AUC	F1
BERT	0,858	0,858	0,852
BIO_BERT	0,882	0,881	0,873

Tabla A1: Resultados de *accuracy* para todos los modelos probados. Para los modelos de base se muestra el promedio para 30 semillas aleatorias, los diferentes kernels están ubicados en la primera columna luego del nombre del modelo y las diferentes configuraciones de los vectorizadores se encuentran en la primera fila. Stops_wors se refiere al uso de stop word para el idioma Inglés. Para los modelos con redes neuronales, la tabla muestra el resultado de la exactitud *accuracy* para cada variación de los parámetros. *Para 30 semillas. **Se muestran los resultados para la mejor semilla luego de probar con 30 semillas aleatorias. ***Se utilizaron 10 optimizadores, de los cuales solo se muestran los mejores 2 con los mejores resultados en general.

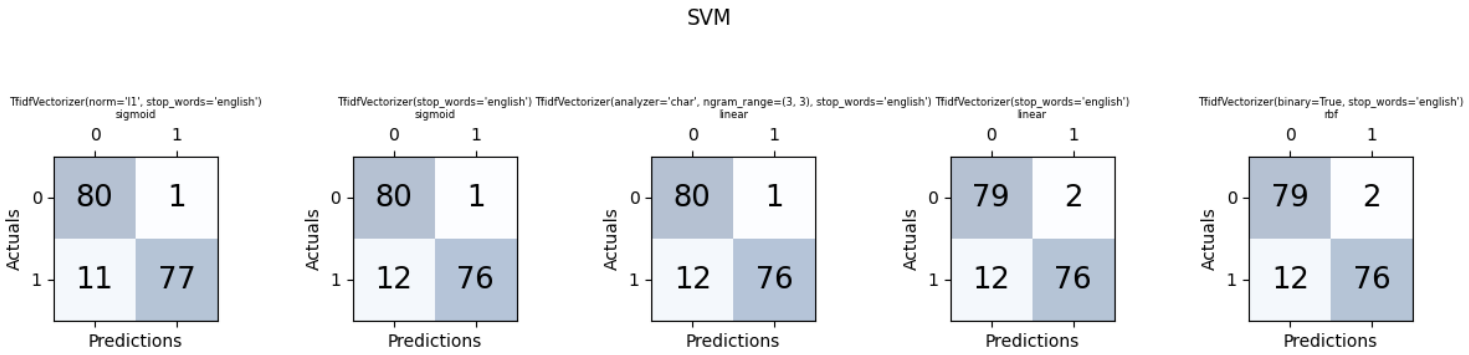
A

Name: [**Known lastname 1469**]; [**Known firstname **] Unit No: [**Numeric Identifier 17474**]Admission Date: [**2173-4-28**] Discharge Date: [**2173-5-4**]Date of Birth: [**2092-4-10**] Sex: F Service: ICU This is a discharge summary addendum from previous dischargesummary dated [**2173-5-3**].HOSPITAL COURSE CONTINUATION OF PROBLEMS:1. C. diff colitis: Patient's stool output continues todecrease. Patient should complete a 14-day course of p.o.Vancomycin and Flagyl p.o. She has remained hemodynamically table.2. Multifocal atrial tachycardia (MAT): Patient's heart rate has improved and her heart rate less in the 80s to low 100scurrently. She remains on a dose of labetalol 100 b.i.d. and has done well with this. If she remains tachycardic; this dose could be increased as an outpatient; however; patient does occasionally become hypotensive with increase of nodal agents. Thus; she was continued on this current dose of p.o.Labetalol.3. End-stage renal disease: Patient is on Tuesdays;Thursdays; and Saturday schedule for hemodialysis and shouldbe continued on this as an outpatient.4. Fluids; electrolytes; and nutrition: Patient appears slightly hypovolemic still and could use additional volumesince she appears intravascularly dry. Her sodium upon the day of discharge has dropped slightly to 131 from her baseline in the mid 130s. This is likely secondary to poor salt intake since she has increased her free water intake.On discharge to rehab; the patient should increase her salt intake for the next day just to improve her sodium; and should recheck her sodium the next day to ensure that her salt intake has been appropriate. One could also consider encouraging chicken broth or more salty foods for the next day or so since patient has increased her p.o. intake; but mostly in the form of free water. She should be continued tobe encouraged to take good p.o. intake.5. Left upper extremity: Patient has new left upper extremity swelling around the site of her A-V fistula on[**5-4**] a.m. She will be having an ultrasound; which results are still pending at the time of this dictation.6. Pulmonary: Patient should continue to use her incentives pirometry since she likely has some atelectasis. Her O2saturations have improved and she is now saturating well on room air.DISCHARGE CONDITION: Stable.DISCHARGE STATUS: To rehab facility at [**Location (un) 2196**] Point.DISCHARGE MEDICATIONS:1. Protonix 40 mg p.o. q.d.2. Heparin subq 5000 units q.12h.3. Metronidazole 500 mg p.o. t.i.d. for nine more days.4. Vancomycin 125 mg p.o. q.6h. for nine more days. Pleasegiven in liquid formulation since patient has difficulty swallowing big pills.5. B complex; vitamin C.6. Folic acid 1 mg capsule p.o. q.d.7. Labetalol 100 mg p.o. b.i.d.8. Tylenol 325 mg p.o. q.4-6h. prn.9. Trazodone 25 mg p.o. q.h.s. prn insomnia.10. Lidocaine hydrochloride 2% gel one application urethralprn.11. Zinc oxide; cod liver oil 40% ointment one application topical prn.DISCHARGE DIAGNOSES:1. Clostridium difficile colitis.2. Multifocal atrial tachycardia.3. End-stage renal disease.FOLLOW-UP PLANS: The patient should likely follow up with her PCP in one week after discharge for followup of the above issues. [**Name6 (MD) 9775**] [**Last Name (NamePattern4) 9776**]; M.D.Dictated By:[**Name8 (MD) 15481**]MEDQUIST36D: [**2173-5-4**] 12:54T: [**2173-5-4**] 13:04JOB#: [**Job Number 17475**]

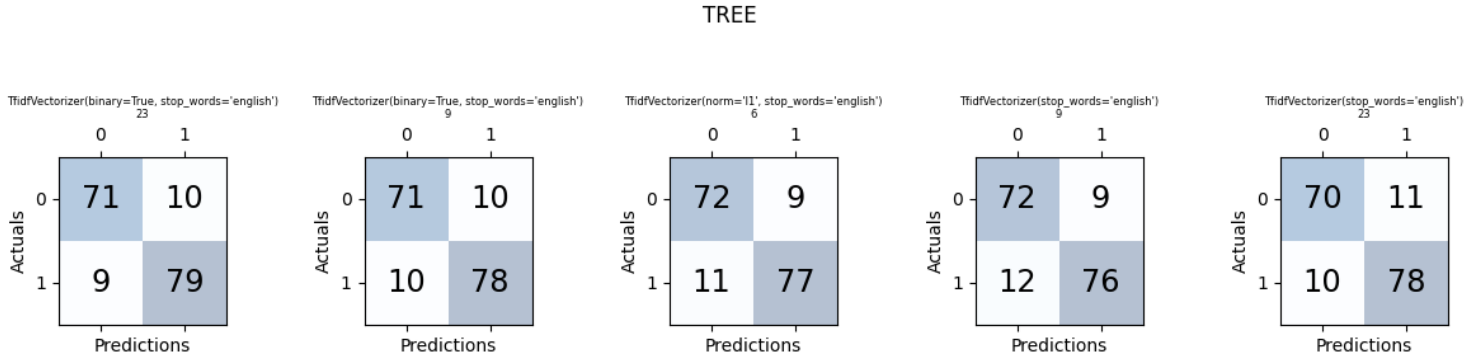
B

Name: [**Known lastname **];[**Known firstname 10852**] Unit No: [**Numeric Identifier 10853**]Admission Date: [**2189-5-11**] Discharge Date: [**2189-5-18**]Date of Birth: [**2133-11-10**] Sex: M Service: MEDICINE Allergies: Patient recorded as having No Known Allergies to Drugs Attending:[**First Name3 (LF) 4226**] Addendum:Pt's stay was extended to [**2189-5-18**] by non-hemodynamically significant oozing at site of new tunneled HD catheter site.This was treated with gelfoam; thrombin spray; and pressure dressings. IR had already stitched the line quite tightly so nosurgical revision was performed. DDAVP was considered but not given due to patient refusing to have peripheral IV replaced..All bleeding had resolved by [**5-18**]; and pt was released after HD.Of note; labetalol was uptitrated to 800mg tid for persistently elevated systolic blood pressures. Discharge Disposition:Home [**Name6 (MD) **] [**Name8 (MD) **] MD [**MD Number(2) 4228**]Completed by:[**2189-5-18**]

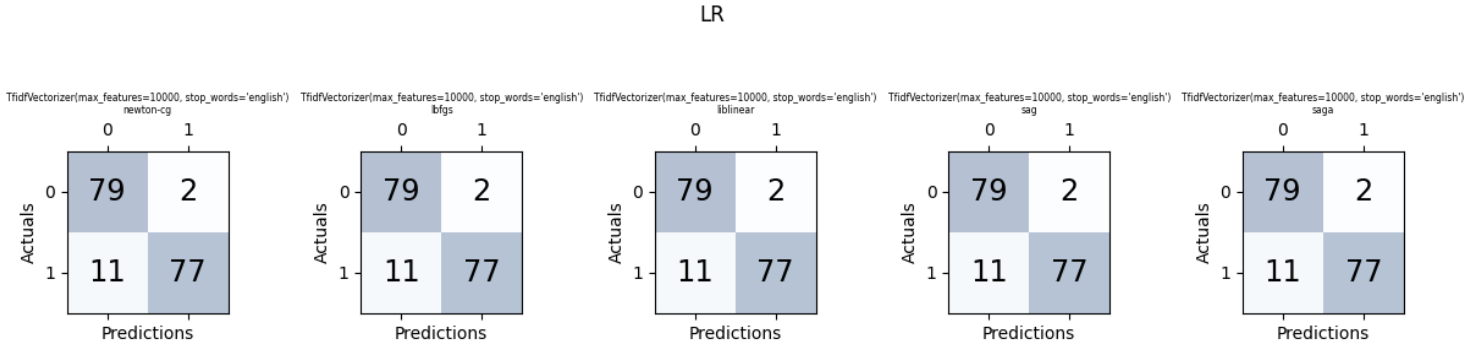
Figura A1: Ejemplo de resúmenes de alta para enfermedad rara en el Panel A y una enfermedad común en el Panel B



(a) SVM

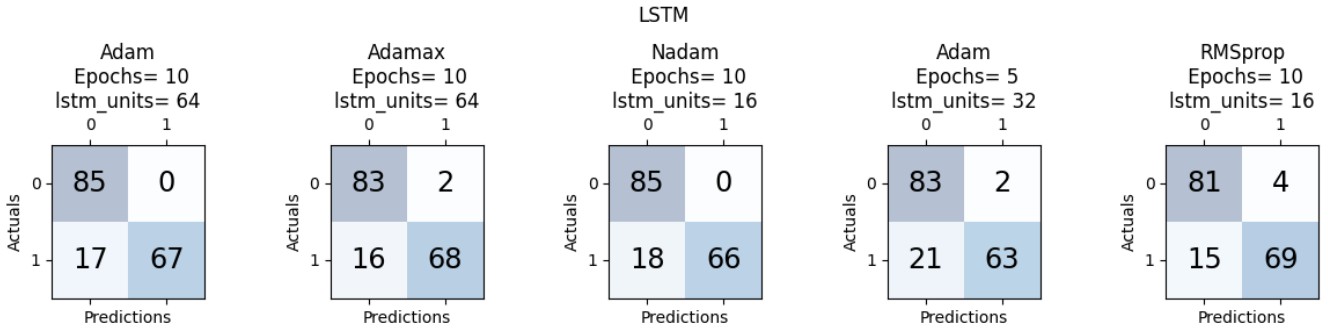


(b) Decision Trees

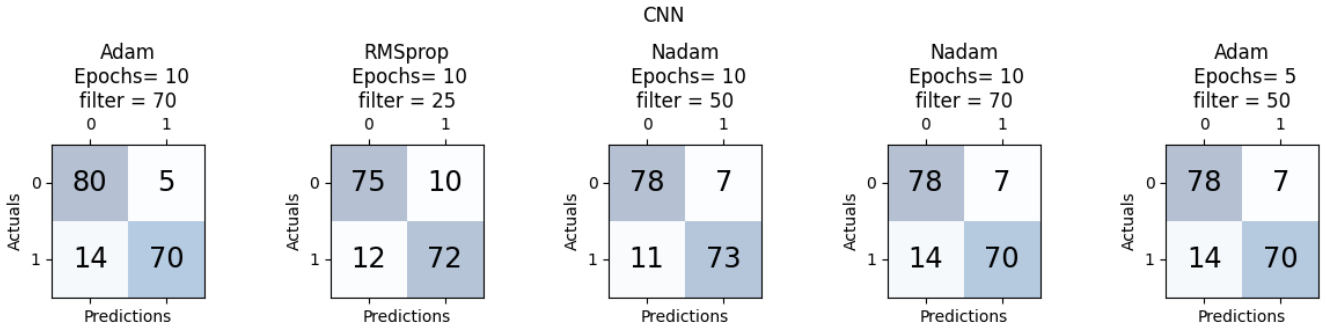


(c) Logistic Regression

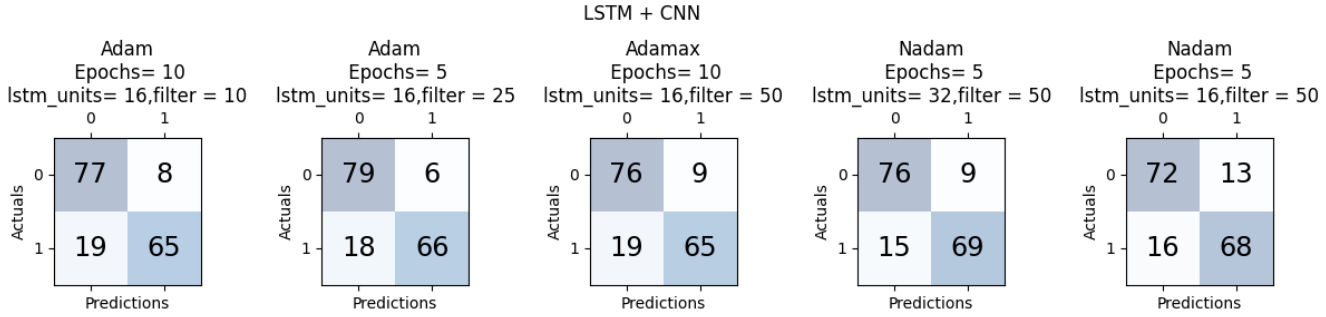
Figura A2: Se muestran las matrices de confusión para los 5 mejores combinaciones de vectorizador y parámetros para la mejor semilla de las 30 semillas aleatorias. En el título de cada recuadro se puede ver el vectorizador utilizado y debajo la variable del modelo. Se muestran los modelos simples, SVM, DT y LR, a, b y c, respectivamente.



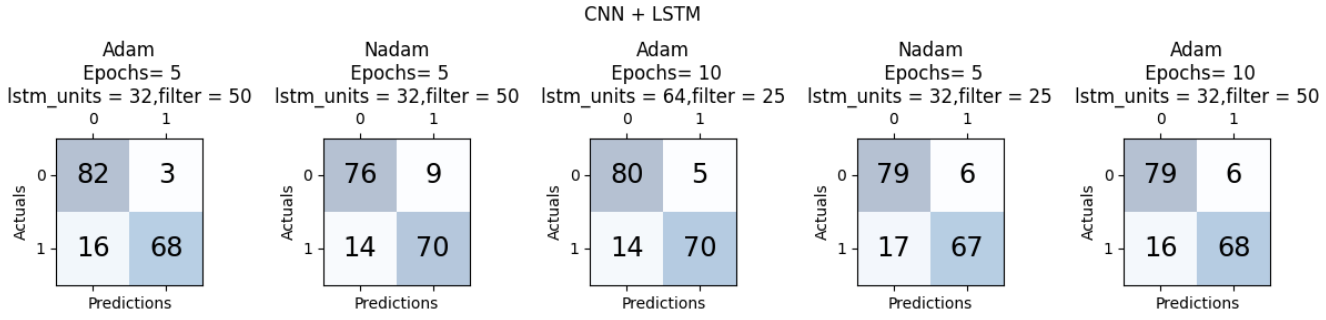
(a) LSTM



(b) CNN



(c) LSTM + CNN



(d) CNN + LSTM

Figura A3: Se muestran las matrices de confusión para los 5 mejores conjuntos de parámetros para cada uno de los modelos basados en redes neuronales, LSTM, CNN, LSTM+CNN y CNN+LSTM, siendo estos a, c y d, respectivamente.

ID paciente	label	real label (expert curated)	comentarios
100161	rare	common	It would be a congenital heart disease (cardiac malformation). Whether it is rare or not depends on the syndromic (rare) or isolated (frequent) etiology. Probably NOT rare
100180	rare	rare	
4806	rare	common	common. It seems that the record is so messy, and has so many things that it get mislabeled. It has a rare complication (aortic dilatation) from a common disease: ̈In the United States, about 13000 patients die because of aortic disease each year, and TAA is the 18 most common causes of death among all individuals. TAA has an incidence of 10 cases per 100000 patient years and a prevalence of 0.16 to 0.34 %.[9] The incidence of TAA is increasing due to better diagnostic imaging and increased life expectancy in the general population. Patients with familial TAA have an average age of presentation at 56.8 years, while patients with TAA due to other causes present around 64.3 years. More men develop TAA, while more women develop worse clinical outcomes and have an increased risk of dissection."
4938	rare	common	it has all possible complications of a common disease
6188	rare	common	Infection in the context of HIV. The infection could be rare, since it is a infrequent germ but the disease is common and the complications are the most frequent.
8979	rare	common	all common diseases combined, an a less frequent complication of one common disease. (not so unfrequent in elderly)

10004	rare	common	all common diseases combined, an a less frequent complication of one common disease. (not so unfrequent in elderly)
11604	rare	common	same as before
24716	rare	rare	chronic pancreatitis and possible pancreatic cancer. See frequencies / Alcoholism / possible hemochromatosis (hemochromatosis is one of the few genetic diseases that are not rare).
26976	rare	common	newborn with extreme prematurity / the disease itself is common, not so extreme prematurity (by frequency), but conceptually it is a common disease, and has all the complications of extreme prematurity.
30504 47026	rare rare	common rare	are all common conditions with summative disease and a rare complication, but not so common in the elderly.
44132	rare	common	
34305	rare	common	patient with multiple complications of common diseases
51061	rare	common	patient with multiple complications of common diseases
49795	rare	common	multiple developmental complications of alcoholism and drug abuse
59366	rare	common	appears to be a clinical progress report, rather than a complete CKD. Relatively frequent infection in patients with underlying chronic kidney disease.

Tabla A2: Comentarios de un experto en enfermedades raras sobre las historias mal clasificadas por el modelo LSTM. Los ID corresponden a las historias respectivas y se pueden ver en el corpus completo.

Etiqueta	**Prompt**

rare	Generate a discharge summary of a rare genetic disease like cystic fibrosis including clinical tests, exams, proposed treatments, medical advice and presented symptoms.
rare	create a discharge summay of a hospital of a patient with a rare genetic disease with symptoms such as sporadic seizures. Include lab tests made, specialists he visited, age of patient, sex, studies ordered.
rare	Write a clinical record of a patient with a genetic disease that doctors are not able to diagnose. Include lab test results, results from studies ordered, results from scans and the age of patient. Include neurological symptoms if present. Include all physicians that he visited in his life. Include please results from all the tests. Include a potential diagnosis according to the results.
rare	Write a clinical record of a patient with a very difficult to diagnose disease that doctors were not able to diagnose in the past. Include lab test results, results from studies ordered, results from scans and the age (baby) and sex of patient. Include all symptoms he presented, especially neurological symptoms . Include all physicians that he visited in his life. Include please results from all the tests. Include a potential diagnosis according to the results.
common	Write a clinical record of a patient with a common disease such as diabetes type 2. Include lab test results, results from studies ordered, results from scans and the age and sex of patient. Include all symptoms he presented, especially cardiological symptoms . Include all physicians that he visited in his life. Include results from all the tests. Include a potential diagnosis according to the results.
common	write a discharge summary of a patient with a rare brain infection that she caught due to a neumonia not totally cured. Note the age and sex of the patient. Number of specialists seen. Write the discharge summary as found in the MIMIc-III database.
rare	please write me a clinical record of a 5 years old boy that came in to the emergency room with a pulmonary infection. This was de fourth one in the last 6 month. Please include all specialists he has seen during his stay at the emergency room, which clinical tests he was submitted to , what are the further steps for him. Could you add a neumologist visit, and a geneticist.

rare	clinical discharge summary similar to those in MIMIC-III data base of a 7 years old girl with neurological symptoms: seizures. Include all the clinical lab test, images, test results, specilaist she had to visit and what is the suspected diagnosis. Include future steps and recommendations.
common	clinical discharge summary similar to those in MIMIC-III data base of a 7 years old girl with an infection in her throat. Include all the clinical lab test and results, specialist she had to visit and what is the suspected diagnosis. Include future steps and recommendations.
common	write a discharge summary similar to those in MIMIC-III data base represeting the life of a hospital. The discharge summary is about a 10 years old boy with acute stomach pain, diarrea and nausea. Include all specialists he has seen in his emergency room stage, all the lab tests and test results. Include possible diagnosis and future recommendations.
rare	can you create clinical record of a patient with probable rare disease such as Primary epilepsy. Include all clinical studies ordered for the patient, results of such studies, images and their results, drugs that she should take. Include all medical and surgical specialists she has visit in her stay at the hospital
rare	please write me a clinical discharge summary of a 3 years old boy that came in to the emergency room with a dislocated shoulder and trouble breathing. This was the second visit in the last 6 month because of bone issues. Please include all specialists he has seen during his stay at the emergency room, which clinical tests he was submitted to , what are the further steps for him. Could you add a neurologist, traumatologist and a geneticist. He has also small stature, which could have a relation with skeletal dysplasia. could you write the clinical record similar to the Mimic-III data base?
rare	please write me a clinical discharge summary of a 7 years old boy that came in to the emergency room with a broken arm and wrist. This was the third visit in the last year because of the same issue of broken bones. Please include all specialists he has seen during his stay at the emergency room, which clinical tests he was submitted to , what are the further steps for him. Could you add consultations with traumatology (x-rays with some finding) and genetics. He has also small stature, and mild cardiac issues, everything together could be relared to osteogenisis imperfeta and bone fragility. could you write the clinical record similar to the Mimic-III data base?

common	write a discharge summary similar to those in MIMIC-III data base represeting the life of a hospital. The discharge summary is about a 10 years old boy with high fever, without clear focus. He presented some caughing but that was it. Include all specialists he has seen in his emergency room stage, all the lab tests and test results. Include possible diagnosis and future recommendations.
common	write a discharge summary of a 8 years old girl that visti the emergency room because of trouble breathing, caugh, agitation. Include all specilaist she has to visit. All clinical tests run and all the results. The results might concurr with bronchospams, which is concurrent with the time of the year (winter). The specialist leading the diagnosis was the neumologist.
common	write a discharge summary of a 12 years old girl that visit the emergency room because of stomach ache, fever, and vomiting. Include all speciliast she has to visit. Include clinical tests such as blood work, to tackle white cells blood counts. Include physical exams, such as stomach palpation. All clinical tests run and all the results. The results might concurr with apendicities. Write the discharge in MIMIc-III format.
common	write a discharge summary of a 28 years old girl that visit the emergency room because of stomach ache, fever, and vomiting. Include all speciliast she has to visit. Include clinical tests such as blood work, to tackle white cells blood counts. Include physical exams, such as stomach palpation to discard apendicities. All clinical tests run and all the results. The results might concurr with kidney stones. Write the discharge in MIMIc-III format.
common	write a discharge summary of a 45 years old man that visit the emergency room because of persistent head ache and disconfort. Include all speciliast he has to visit during his stay in the ER. Include clinical tests such as blood work, to tackle white cells blood counts, which came out normal. Include physical exams, to discard migranie. Include blood presure measurments, which was high. All clinical tests run and all the results. The results might concurr with high blood presure. Write the discharge in MIMIc-III format.

common	write a discharge summary of a 30 years old man that visit the emergency room because of a skin condition. Skin has a rush and is itchy. Topic solutions (creams) were not effective. Include all speciliast he has to visit during his stay in the ER, specially dermatologistis and allergiologist. Include clinical tests such as blood work, to tackle white cells blood counts, which came out normal. All clinical tests ordered and all the results. The results might concurr with an drug-related allergy (ibuprofen). Write the discharge in MIMIc-III format.
common	write a discharge summary of a 30 years old woman that visit the emergency room because of a persisten head ache. She is not able to sleep for days. Include all speciliast he has to visit during his stay in the ER, specially imaging to discard brain tumors (it came out negative). Include clinical tests such as blood work, to tackle white cells blood counts for infections, which came out normal. All clinical tests ordered and all the results. The results might concurr with a strong migranie, that need strong pain medication. Write the discharge in MIMIc-III format.
common	write a discharge summary of a 1 year old baby that visit the emergency room because of problems breathing. Baby is a girl and presents shortness of breath and some fever. Include all speciliast he has to visit during his stay in the ER, specially X-ray to discard pneumonia (it came out negative). Include clinical tests such as blood work, to tackle white cells blood counts for infections, which came out normal. All clinical tests ordered and all the results. The results might concurr with a bronchospams. She will need salbutamol every 4 hours for the first 3 days. Also, oral corticoids. Write the discharge in MIMIc-III format.

Tabla A3: Prompts utilizados para la generación de historias clínicas con ChatGPT de OpenAI.