Universidad Michoacana de San Nicolás de Hidalgo

Facultad de Ciencias Físico Matemáticas

Algoritmos de desidentificación de notas médicas para proteger la identidad de los pacientes.

T E S I S

que para obtener el título de:

licenciado en ciencias físico matemáticas

P R E S E N T A :

jesús mercado ríos

ASESORA

Dra. Karina Mariela Figueroa Mora

COASESOR

Dr. Luis Miguel García Velázquez

Ciudad Universitaria, Morelia, Michoacán. Octubre 2022

Resumen

La digitalización de expedientes médicos ha sido pieza fundamental para el análisis de éstos. Los centros de salud capturan valiosa información de los pacientes en sus expedientes, éstos deberían poder ser estudiados con técnicas automatizadas para descubrir patrones, comportamientos, complicaciones, etc, que puedan apoyar el bienestar de la población. Para que equipos de investigadores puedan tener acceso, y trabajar con la información en esos expedientes, primero se debe salvaguardar la información de los pacientes, por lo que es necesario realizar una desidentificación, es decir eliminar cualquier tipo de información personal. El objetivo de este trabajo es realizar algoritmos que permitan desidentificar de manera automática las descripciones capturadas en expedientes médicos.

Conceptos clave: Desidentificación, Procesamiento de Lenguaje Natural.

Dedicatoria

Este trabajo está dedicado especialmente a mis papás, Jesús Mercado Alviso y Socorro Ríos Medina. Quienes siempre me han apoyado en todo momento, y han hecho todo lo posible porque cumpla mis objetivos.

A mis hermanas Lupita Mercado Ríos y Erika Mercado Ríos, que también han estado conmigo cuando las necesito y me han apoyado incondicionalmente.

A todos los que han confiado en mí, familia, profesores y amigos

Agradecimientos

Quiero agradecer a mis padres por ser un ejemplo de dedicación y ser guías en mi camino, a mi hermana Erika, por apoyarme y ser ejemplo de perseverancia, demostrarme que debo luchar por lo que quiero.

Agradezco a mis compañeros y amigos que me apoyaron y confiaron en mí; a las personas que ahora ya no están conmigo, pero que su apoyo fue fundamental.

Agradezco también a los profesores que supieron apoyarme y guiarme, en especial a la Dra. Karina Figueroa Mora, por creer en mí, y ayudarme a confiar en mí mismo.

Índice general

Capítulo 1

Introducción

Hoy en día es natural que tanto hospitales como clínicas mantengan los registros de los pacientes de manera digital, pues esto permite a los trabajadores del área médica llenar de manera más rápida y acceder de una manera más fácil a dichos expedientes. Mantener de esta manera la información de los pacientes es muy importante pues se permite analizar los datos ahí reflejados, sin embargo, en muchas de estas notas médicas existe información sobre datos personales, i.e. nombre del paciente, dirección, teléfono y ciudad, y de acuerdo con el Instituto Nacional de Transparencia, Acceso a la Información y Protección de Datos Personales (INAI) se debe salvaguardar esta información.

Para eliminar información personal podemos realizar una anonimización o una desidentificación. La anonimización consiste en evitar por completo que se pueda vincular la información hacia una persona utilizando técnicas para alterar los datos. La desidentificación consiste en eliminar los datos que se encuentran de manera explícita. Para facilitar el análisis posterior de las notas médicas se realizará una desidentificación.

En este trabajo, se presentan dos algoritmos que permiten desidentificar notas médicas obtenidas de las bases de datos del Instituto Mexicano del Seguro Social, esto como parte del proyecto aprobado por CONACYT: Estudio longitudinal para el desarrollo de modelos predictivos de complicaciones crónicas de la diabetes mellitus tipo 2, con número de proyecto 10410 y responsable la Dra. Anel Gómez García. Las

notas médicas desidentificadas permitirán proteger los datos personales de los pacientes y que se puedan hacer estudios de la información contenida, por ejemplo, modelos de predicción de pacientes con alguna condición específica, sistemas de recuperación de información, análisis estadísticos, algoritmos de agrupamiento, etc.

La desidentificación se puede llevar a cabo fácilmente cuando se trata de una base de datos relacional, en la que se tienen identificadas las tablas que contienen información sensible y que basta con que sean eliminadas las columnas que contengan estos datos. Otro caso muy diferente es cuando se trata de desidentificar texto elaborado de manera libre con lenguaje natural, por ejemplo cuando un médico captura la descripción del paciente

Para nuestro problema, hablamos del segundo caso, debemos eliminar los datos personales dentro de textos elaborados de manera libre por trabajadores del Instituto Nacional del Seguro Social. Los datos personales que deben ser eliminados dentro de las notas, son: nombres, apellidos, domicilios, números telefónicos y ciudades; estos datos pueden ser del paciente, o incluso de sus familiares directos.

Para resolver este problema de desidentificación, se realizaron dos algoritmos:

- Algoritmo de Desidentificación con Bases de Datos Contextuales
- Algoritmo de Desidentificación con Aprendizaje Profundo

Capítulo 2

Estado del arte

Dada la importancia del estudio de los expedientes clínicos, y de lo dictado por las distintas leyes que protegen la identidad de los pacientes, existen diversos artículos que hablan de la desidentificación de textos clínicos [1] [2] [3]. Cabe mencionar que todos los artículos tratan narrativas en el idioma inglés.

Hablaremos de dos tipos de modelos de desidentificación: modelos basados en comparaciones de una base de datos contextual y modelos de aprendizaje automático.

Los modelos que usan bases de datos contextuales, o diccionarios, intentan identificar si se trata de un dato personal por medio de la comparación con los elementos de dichos diccionarios, así, se puede saber si se trata de un nombre, apellido, parte de un domicilio, ciudad, etc.

Otros modelos son implementados con aprendizaje automático, y pueden hacer uso de *máquinas de vectores de soporte (SVM)* o *redes neuronales*. Incluso puede haber modelos híbridos en los que utilicen ambos tipos de algoritmos.

A continuación se presentan algunos trabajos relacionados con la desidentificación de expedientes médicos, algunos recopilados en [3].

- *De-identification of address, Date and Alphanumeric Identifiers in Narrative Clinical Reports* [1]

  En este modelo emplean, como se menciona en el documento, enfoques simbólicos y no aprendizaje profundo. Toman en cuenta 4 categorías distintas: 1)

nombres, 2) direcciones, 3) fechas y 4) identificadores alfanuméricos. Para el caso de los datos alfanuméricos, se utilizan expresiones regulares para conocer si una cadena de números es precedida por palabras como "número", "protocolo", o incluso el signo "#". Para el caso de las fechas y edades, también son utilizadas las expresiones regulares, en este caso para encontrar patrones; para el caso de las fechas, puede contener sólo números que sigan patrones como: DDMMAAAA, AAAAMMDD, entre otros. Para las edades, igual se reconocen patrones y palabras como: edad, años, meses, etc. En el caso de las direcciones, los patrones que se buscan son: nombres de ciudades y estados, tipos de calles y sus abreviaturas.

Los resultados obtenidos para un conjunto de 3093 reportes clínicos en inglés fueron: **alfanuméricos** *recuperación* $= 100\,\%$, **direcciones** *recuperación* $= 83.6\,\%$, **fechas** *recuperación* $= 98.9\,\%$

- *Development and evaluation of an open source software tool for deidentification of pathology reports* [4]

  En este trabajo se presenta un modelo *open source*, que consiste en 3 pasos. El primer paso preprocesa el texto y lo convierte en un formato XML, que contiene un encabezado, en el cual aparece el nombre, número de seguridad, fecha de nacimiento, identificador del reporte. El segundo paso utiliza la coincidencia de patrones haciendo uso de 50 expresiones regulares para encontrar fechas, números telefónicos o números de seguro social. Además también utiliza palabras disparadoras como "Dr", "Mr", para identificar nombres. En el tercer paso, se usan bases de datos para encontrar nombres y lugares utilizando algoritmos de coincidencia exacta para poder eliminarlos. La evaluación se llevó a cabo con 1800 reportes nuevos en inglés, obteniendo una *recuperación* $= 98\,\%$ y una *precisión* $= 43\,\%$. Los errores de precisión fueron por la cantidad de palabras comunes que aparecen en las bases de datos de nombres y lugares.

- *A software tool for removing patient identifying information from clinical documents* [5]

Este software utiliza expresiones regulares, información del paciente que se encuentra en los encabezados de los reportes y diccionarios de nombres y localidades. Utiliza aproximadamente 50 expresiones regulares para identificar posibles datos personales, como números de seguro social, códigos postales, o parte de los domicilios. También se utilizan para descubrir palabras que pueden indicar nombres de paciente o de trabajadores, como "Dr" o "Mr". Se extraen los nombres de los pacientes y médicos, para luego buscar coincidencias en el texto. Por último, utiliza algoritmos que pueden identificar nombres mal escritos. El software tuvo dos evaluaciones. La primera utiliza 2400 reportes obtenidos de *Indiana Network for Patient Care (INPC)*, que incluye 1400 reportes de laboratorio, 800 reportes narrativos y 200 reportes de diferentes sectores. Obtuvo una $recuperación = 99.06\,\%$, tuvo 4012 errores de borrado excesivo.

La segunda evaluación se realizó con 7193 reportes de patología quirúrgica obtenidos de *INPC*, logrando un $recuperación = 99.74\,\%$.

- *Evaluation of a deidentification (De-ID) software engine to share pathology reports and clinical documents for research* [6]

  La aplicación De-ID es de uso comercial. Utiliza un conjunto de reglas, algoritmos de coincidencia de patrones y diccionarios para encontrar datos personales. Analiza los reportes para obtener los datos explícitos, como el nombre del paciente y así eliminarlo en el reporte completo. No especifica qué tipo de algoritmo se utiliza para la coincidencia de datos como números de teléfonos o códigos postales. Utiliza una lista de nombres proveniente del censo de EUA y busca coincidencias con cada nombre. El algoritmo De-ID permite la creación de diccionarios personalizados para identificar datos locales de cada institución. Las fechas son reemplazadas por etiquetas de tipo fecha que permiten retener información sobre el intervalo de tiempo. El software tuvo tres evaluaciones, las dos primeras usaron aproximadamente 1000 reportes, y fueron utilizados para probar y mejorar el sistema. La tercer evaluación usaba 300 reportes de patología. Los resultados que muestran son sólo de falsos positivos y falsos negativos.

La primer evaluación tuvo 190 $falsos\ negativos$ y 150 $falsos\ positivos$. La segunda evaluación tuvo 102 $falsos\ negativos$. La tercer evaluación se llevó a cabo con 300 reportes, teniendo solamente 8 $falsos\ negativos$. Las últimas evaluaciones mostraban una eliminación muy confiable, por lo que no se consideraron falsos positivos.

- *Identifying Personal Health Information Using Support Vector Machines* [7]
  El equipo de trabajo consideró que el identificar datos personales podría tratarse como una tarea de *reconocimiento de entidades nombradas*; emplearon máquinas de vectores de soporte para su modelo. Se agregaron múltiples funciones para que pudiera identificar variaciones en la escritura de fechas y números telefónicos. Funciones que permiten reconocer nombres de médicos y hospitales.
  Las evaluaciones se realizaron utilizando el corpus de la competencia *i2b2* [8], obteniendo una recuperación, precisión y medida F1 mayores al 86 %. De manera individual se logró una medida F-1 de 83 % para los hospitales, 78 % para los números de teléfono y un 82 % para los números de identificación personal.

- *State-of-the-art Anonymization of Medical Records Using an Iterative Machine Learning Framework* [9]
  Este sistema utiliza aprendizaje de máquina empleando modelos existentes que se usan para el reconocimiento de entidades nombradas. Se lleva a cabo un aprendizaje iterativo basado en árboles de decisión que utilizan la información que se encuentra estructurada en el reporte médico para identificarla en el cuerpo de la nota. Se toman en cuenta características como las mayúsculas, tamaño de palabras, frecuencia, entre otras. Las evaluaciones se realizaron para la competencia *i2b2* [8], obteniendo una precisión, recuperación y medida F-1mayores al 96 % de manera general. De forma individual, el peor resultado fue para las localidades con $F\text{-}1 = 68\,\%$

- *A de-identifier for medical discharge summaries* [10]
  En este artículo se habla sobre el desidentificador *Stat De-id*, el cual está basado en máquinas de vectores de soporte y contexto local de las palabras. Este

sistema identifica características ortográficas, sintácticas y semánticas de las palabras, usando una ventana de ± 2 alrededor de cada una de ellas. Algunas características ortográficas incluyen la palabra misma, uso de letras mayúsculas, puntuación y el largo de la palabra. El sistema Stat De-id se evaluó utilizando 889 resúmenes clínicos obtenidos de distintos departamentos médicos del *Partners Healthcare System* en Boston, MA. Logró una medida *F-1 = 98 %* con una precisión del 99 % y una recuperación promedio, de todos los datos personales, del 97 %.

Capítulo 3

Conceptos básicos

3.1. Anonimización y Desidentificación

Es importante conocer dos conceptos muy relevantes al momento de hablar sobre la eliminación de datos personales y sobre la protección de la identidad; **desidentificación** y **anonimización**. Estos dos conceptos, a menudo, son utilizados sin distinción cuando se trata de la protección de la identidad dentro de los archivos de texto.

La **anonimización** es una tarea que asegura la protección de la identidad, logrando que no haya una relación que se pueda seguir para dar con la persona de la que se habla. La anonimización a su vez, consiste en usar las técnicas de generalización, adición aleatoria, permutación, entre otras.

La **desidentificación**, por su parte, es un proceso de pseudoanonimización, el cual consiste en eliminar o alterar datos personales identificadores que se encuentren de manera explícita. Dicho de otra manera, se eliminan los datos que podrían servir para conocer de manera directa la identidad de una persona. Veamos cómo podría verse la desidentifación y anonimización de los datos de la tabla 3.1

Tabla 3.1: Tabla con datos de alumnos egresados

Nombre estudiante	Año de egreso	Escuela	Promedio general
Natalia Martínez	2018	Contabilidad	9.0
José Campos	2018	Ingeniería mecánica	8.5
Julieta López	2020	Facultad de matemáticas	8.8

Tabla 3.2: Desidentificación de la tabla 3.1

~~Nombre estudiante~~	Año de egreso	Escuela	Promedio general
~~Natalia Martínez~~	2018	Contabilidad	9.0
~~José Campos~~	2018	Ingeniería mecánica	8.5
~~Julieta López~~	2020	Facultad de matemáticas	8.8

Tabla 3.3: Anonimización de la tabla 3.1

Nombre estudiante	Año de egreso	Escuela	Promedio
Fulana 1	2015-2020	UMSNH	>= 9.0
Fulano 1	2015-2020	UMSNH	>= 8.0
Fulana 2	2020-2022	UMSNH	>= 8.0

En la tabla 3.2 se observa que se ha desidentificado, procediendo a la eliminación de la columna de "Nombre estudiante". Aunque aún se pueda obtener la identidad de la persona si se accede a las bases de datos de las facultades y comienza a hacerse una búsqueda con los otros datos.

En la tabla 3.3 se ha realizado una anonimización; esta manera de proteger la identidad es más rigurosa, sin embargo, implica mayor uso de recursos, y para fines de investigación, la generalización podría no dar los resultados que se esperan.

3.2. Algoritmo de Desidentificación con Bases de Datos Contextuales

Para este modelo con bases de datos contextuales, han sido utilizadas técnicas provenientes de la recuperación de información [11] [12], así que es necesario dar a conocer algunos conceptos básicos de esta área. Primero hablaremos sobre el preprocesamiento que debe tener el texto antes de poder utilizarlo. Este preprocesamiento es un proceso que clasifica las palabras y las indexa para después ser usadas.

3.2.1. Tokenización

La tokenización es un proceso que se aplica a un documento dado y consiste en dividirlo en partes llamadas tokens, estos tokens, en este caso, son palabras. Es

importante conocer la cantidad de términos con los que cuentan los documentos. Las palabras válidas formarán parte de un índice o diccionario de tokens. En la fig 3.1 se muestra una separación de una frase en tokens. En el lado derecho, la indexación consistió en enumerar las palabras diferentes.

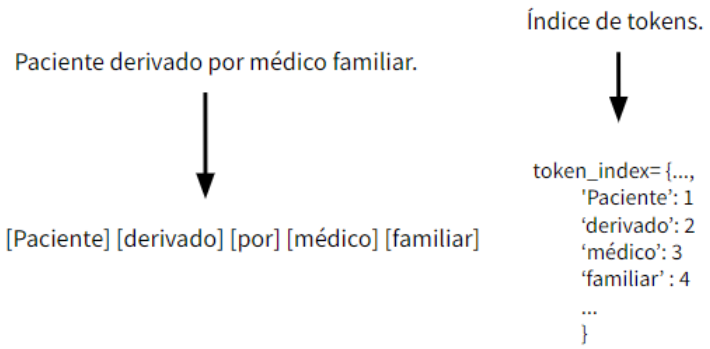


Figura 3.1: Separación de una frase en token y su indexación

3.2.2. Stop Words

Los *Stop words*, o palabras vacías, son aquellas que normalmente no aportan un significado relevante, éstas pueden ser: artículos, preposiciones, pronombres, etc. Eliminar las stop words puede tener beneficios, pero a su vez, puede ser contraproducente hacerlo. La ventaja de eliminarlos es reducir el número de términos que se tomarán en cuenta para llevar a cabo la recuperación, por lo que en las consultas no será relevante la aparición de esas palabras. Por esa misma razón, es que no siempre es bueno eliminar este tipo de palabras, ya que en algunos casos, éstas pueden tener un valor relevante en las oraciones, o al momento de realizar alguna consulta.

3.2.3. Lematización y Stemming

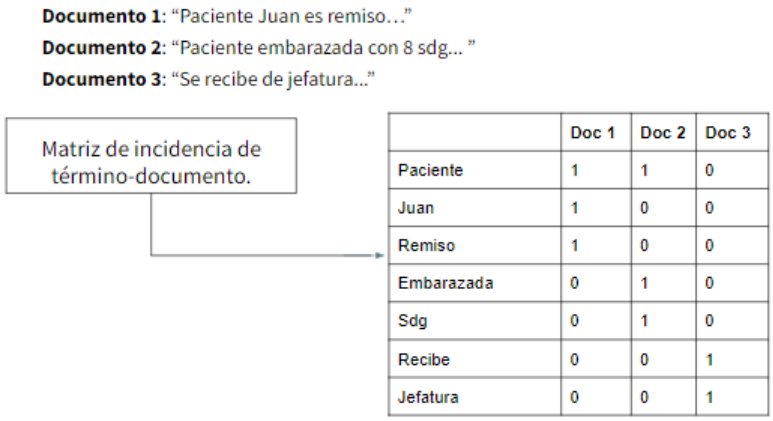
Si se quiere reducir el número de términos aún más, debemos conocer los siguientes dos conceptos: *Stemming* y *lematización* El stemming consiste en reducir las palabras a su lexema o reducirlas a algo que no necesariamente son palabras, por ejemplo: *médico* ->*médic*; *médica* ->*médi*c.

La lematización consiste en transformar una palabra a la forma más general de ella, por ejemplo: *correr* es la forma general de las palabras *corrió*, *corrimos*, etc.

3.2.4. Positional Posting List e Índice Invertido

Antes de definir lo que es la *positional posting list* y el *índice invertido*, es necesario presentar la *matriz de incidencia*.

La matriz de incidencia es una representación que nos permite saber si una palabra específica se encuentra, o no, en algunos textos. En la figura 3.2 se observa el uso de una matriz de incidencia



	Doc 1	Doc 2	Doc 3
Paciente	1	1	0
Juan	1	0	0
Remiso	1	0	0
Embarazada	0	1	0
Sdg	0	1	0
Recibe	0	0	1
Jefatura	0	0	1

Figura 3.2: Matriz de incidencia de 3 notas médicas. Note que se eliminaron los *stop words*

El **índice invertido** es una forma de estructurar la información que almacena una matriz de incidencia, pero sin la necesidad de gastar tanta memoria, además, que permite un acceso más fácil y rápido a la información contenida. Está conformado por un diccionario que contiene todas las palabras distintas de todos los documentos procesados, y una *posting list* o *positional posting list*. La llave de acceso a este índice, es precisamente cada una de las palabras contenidas en el conjunto de tokens.

Positional posting list es una estructura tipo lista que forma parte de un índice invertido, nos indica en qué documento y posición de éste, se encuentra cada palabra obtenida por la Tokenización [13]. La primer componente nos indica en qué número de documento se encuentra la palabra y la segunda componente nos indica en que

posición está.

En la figura 3.3 podemos apreciar cómo es representado un índice invertido posicional. Por ejemplo *paciente* se encuentra en la posición 1 de los documentos 1 y 2. Por su parte, la palabra *embarazada* aparece en el documento 2 en la posición número 2. Y por último, vemos que la palabra *recibe* sólo aparece en el documento 3 en la posición de palabras válidas número 2.

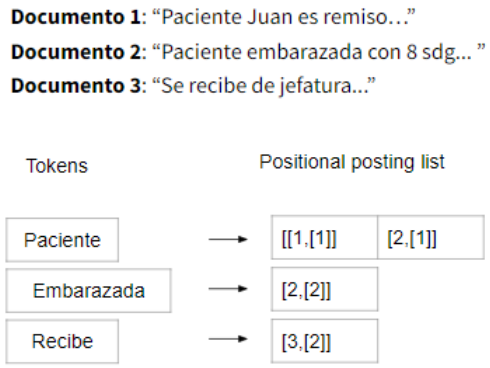


Figura 3.3: Representación de un índice invertido posicional.

3.2.5. Palabras Disparadoras

Existen palabras que pueden indicarnos que enseguida es probable que aparezca un dato específico. Por ejemplo, después de la palabra *paciente*, es posible que aparezca el nombre de dicho paciente. O seguido de la palabra *fecha* podrían aparecer números que indiquen una fecha específica, o palabras que indiquen la fecha de la que se habla. Estas palabras son conocidas como palabras disparadoras o *trigger words*, ya que sirven como detonantes de que puede existir información personal. [1]

3.3. Algoritmo de Desidentificación con Aprendizaje Profundo

Para hablar sobre el modelo de aprendizaje profundo, es necesario definir conceptos importantes que nos permitirán tener una breve introducción en las redes

neuronales artificiales.

3.3.1. Neurona

La neurona es la unidad básica de una red neuronal artificial, en ella se realizan operaciones con los valores de entrada. Los datos que serán utilizados por la red neuronal, deberán tener una representación vectorial: $X = [x_1, x_2, ..., x_d]$, que es un dato de entrada; los pesos asociados a cada elemento de X durante el entrenamiento $W = [w_1, w_2, ..., w_d]$, con d como la dimensión de los datos de entrada [14]. La operación que realiza la neurona es conocida como producto punto entre vectores $W \cdot X = \sum_{i=1}^{d} w_i x_i = W^T X$ [15]. Al resultado de esta operación $z = \sum_{i=1}^{d} w_i x_i$ se le aplica una función ϕ, que es conocida como función de activación. Más adelante se explicará el modelo más sencillo, que consta sólo de una neurona.

3.3.2. Función de Activación

La función de activación es la que se encarga de definir el valor de salida de una neurona luego de realizar el producto punto, con el fin de no mantener la linealidad en los resultados [16] [14].

Algunas de las funciones de activación más usadas son [17] [15]:

- Identidad: $\phi(x) = x$
- Sigmoide: $\phi(x) = \frac{1}{1+e^{-x}}$
- Tanh: $\phi(x) = \frac{e^{2x}-1}{e^{2x}+1}$
- ReLU: $\phi(x) = max(x, 0)$
- Softmax: $\phi(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$

3.3.3. Función de Pérdida

La función de pérdida debe minimizar el error con el fin de obtener los mejores resultados en el entrenamiento. Por lo que la elección de ésta, es de suma importancia. La elección de la función de pérdida se puede hacer dependiendo la tarea que se realiza, por ejemplo una predicción binaria, o una predicción multiclase [16] [15] :

1. **Clasificación binaria**

   - Regresión logística: Para este caso se asume que los valores de y son tomados de $-1, 1$ y la predicción de la red $\hat{y}$ es obtenida por la función de activación de identidad. La función de pérdida para un valor observado y y el valor de la predicción $\hat{y}$ es la siguiente:

   $$L = \log(1 + e^{-y \cdot \hat{y}})$$

   - Alternativamente, se puede utilizar la función de activación sigmoide para obtener valores de $\hat{y} \in 0, 1$. Entonces tendremos que el negativo del logaritmo de $|y/2 - 0.5 + \hat{y}|$ proporciona la pérdida, pues esto indica la probabilidad de que el valor de $\hat{y}$ sea correcto [18].

2. **Clasificación multi clase.** Entropía cruzada. Para esta tarea, supongamos que $\hat{y}_1, ..., \hat{y}_c$ son las probabilidades para las c clases. La función de pérdida para una instancia será [18]:

   $$L = -\log(\hat{y}_r)$$

   También se puede utilizar para problemas binarios, conocida como entropía cruzada binaria.

3.3.4. Perceptrón

El modelo más sencillo de una red neuronal artificial, es el que contiene sólo una neurona, conocido como *perceptrón*. Este modelo con una sola neurona puede resolver algunos problemas de clasificación binaria. Las operaciones son las siguientes:

$$z = b + \sum_i w_i x_i$$

$$\phi(z) = y = \begin{cases} 1 \ si \ z \ \geq \ 0 \\ 0 \ si \ z \ < \ 0 \end{cases}$$

donde x es la entrada, w los pesos y b se conoce como sesgo. Para que el sesgo vaya cambiando en cada iteración del entrenamiento, se puede añadir como un nuevo peso, por lo que se necesita agregar una entrada $x_0 = 1$, y el sesgo será su peso w_0. Nótese que es exactamente lo mismo [14]:

$$z = b + \sum_i w_i x_i = w_0 x_0 (= b) + w_1 x_1 + w_2 x_2 + ... = \sum_{i=0} w_i x_i$$

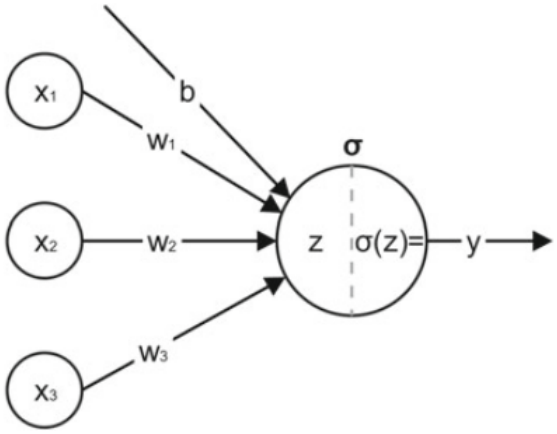


Figura 3.4: Modelo de un perceptrón. Fuente: [14]

3.3.5. Red Neuronal Multicapa

Una red neuronal artificial que tiene una capa de entrada, al menos una capa oculta y una capa de salida, es considerada una red multicapa.

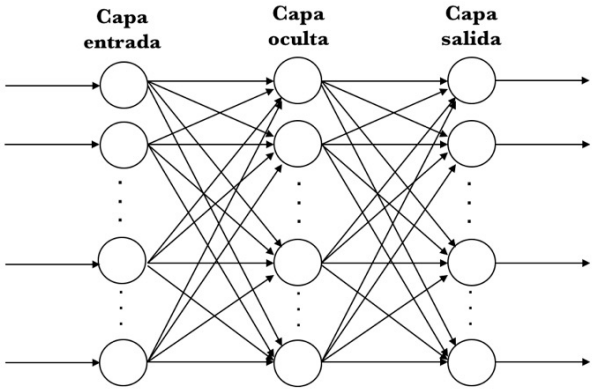


Figura 3.5: Red multicapa. Fuente: https://torres.ai/deep-learning-inteligencia-artificial-keras/

Una manera de entrenar la red es con métodos conocidos como *forward* y *back propagation* que veremos a continuación.

3.3.6. Forward Propagation

Para entrenar la red y conseguir que aprenda, cada neurona de cada capa debe estar conectada con todas las neuronas de la siguiente capa. Es decir: la neurona uno de la capa uno, n_{11}, está conectada con todas las neuronas de la capa dos, n_{2i}, con i que va de 1 hasta el número de neuronas en la segunda capa. Por lo que la salida de la primer capa, es la entrada de la segunda capa [19]. A continuación un ejemplo de propagación del aprendizaje hacia delante:

Sea $x_1 \in X$, el primer elemento con el que se entrenará la red, W_1 los pesos asociados en la primer neurona de la primer capa, así tenemos

$$z_1 = W_1 x_1$$

luego con la función de activación se tiene

$$y_1 = \phi(z_1)$$

en a la segunda capa

$$z_2 = W_2 y_1$$

luego la función de activación

$$y_2 = \phi(z_2)$$

Así, los valores obtenidos en las capas anteriores influyen en las operaciones de las siguientes capas, transmitiendo la información obtenida hacia delante. Ese procedimiento se lleva a cabo en cada neurona de cada capa.

3.3.7. Back Propagation

Es un algoritmo de retropropagación, en el que el error de la última capa, es tomado en cuenta en la penúltima capa, a su vez, éste es tomado en cuenta en la antepenúltima capa y así sucesivamente, para que en la primer capa puedan ajustarse los pesos; después, en la siguiente iteración, gracias a la propagación hacia delante, se espera haber minimizado la función de pérdida.

3.4. Red HugginFace

Hoy en día, y gracias al impacto de las Redes Neuronales Recurrentes, *RNN* para sus siglas en inglés, se usan modelos ya estructurados, algunos especialmente diseñados para el Reconocimiento de Entidades Nombradas, *NER* en inglés, por ejemplo el de la librería de *Hugginface* [20]. Este modelo se basa en el uso de incrustes de palabras o *embeddings* estáticos [21], es decir, la representación vectorial de las palabras no va cambiando con el contexto del texto.

Cabe mencionar que existe un modelo entrenado de esta red neuronal que ya realiza la tarea de reconocer algunas entidades nombradas, tales como: nombres, ciudades

y organizaciones.

Algunos conceptos claves que son necesarios para entender este algoritmo serán descritos a continuación.

3.4.1. Incrustes

Incrustes o *embeddings*, son una manera de representar las palabras de un lenguaje, utilizando vectores de números reales en un espacio de menor dimensión que el número de palabras contenidas en el lenguaje. Estos vectores, a diferencia de otras representaciones de palabras, conservan una relación entre palabras con un significado similar, es decir, que palabras que pueden ser utilizadas dentro de un mismo contexto están a una menor distancia que con otras palabras que no tienen mucha relación.

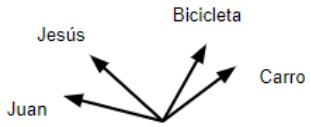


Figura 3.6: Representación de un incruste

Como se puede observar en la figura 3.6, las palabras “Jesús” y “Juan” tienen menor distancia entre sí ya que ambas palabras tienen mayor relación; en cambio, están más alejadas de las palabras “bicicleta” y “carro”, sin embargo, éstas, tienen relación, por lo que también se encuentran cerca entre ellas.

3.4.2. Sentencias

Una sentencia, es un objeto del tipo *Sentence* de la librería *Flair*, que simplemente es una oración formada por tokens. Una oración de tipo Sentence, es capaz de tener etiquetas en sus diferentes tokens, por ejemplo, en la oración:

$$oracion = Sentence('El\ niño\ se\ llama\ Juan',\ use\_tokenizer = True)$$

agregamos una etiqueta de tipo 'NER', llamada 'Nombre' al token Juan (token número 4, ya que se empiezan a enumerar en el número 0)

$$oracion[4].add\_tag(\text{“}ner\text{”}, \text{“}Nombre\text{”})$$

se consulta la oración con la instrucción *print*:

$$print(oracion.to\_tagged\_string())$$

y se muestra

$$El\ niño\ se\ llama\ Juan\ < Nombre >$$

3.4.3. Conjunto de Datos

Los conjuntos de datos necesarios para este modelo son tres, que llamaremos: *train*, *dev* y *test*.

El conjunto **train**, es con el que entrena el modelo, para cambiar los pesos de las neuronas.

El conjunto **dev** sirve para evaluar el entrenamiento, es decir para comprobar los resultados que obtiene la red neuronal; durante esta evaluación se pueden obtener diferentes métricas de los resultados obtenidos, y esta información permite saber si ya se tiene un modelo "suficientemente bueno".

El conjunto **test**, no debe ser procesado por la red neuronal durante el entrenamiento, ya que éste servirá para saber si el modelo realmente tiene buenos resultados. Cada elemento de este conjunto debe ser nuevo para la red, es decir no debe haber sido usado por la red en el entrenamiento.

3.4.4. Corpus

El corpus se puede entender como el conjunto de datos que es utilizado para entrenar el modelo. Para este algoritmo en particular, debe tener una estructura muy específica para que pueda ser utilizado. Todos los datos están formados por una

palabra en cada línea con su respectiva etiqueta. De este modo, la frase: *"El paciente Jesús Mercado"*, en el formato necesario, sería de la siguiente manera:

$$El\ O$$

$$paciente\ O$$

$$Jesús\ B-Nombre$$

$$Mercado\ I-Nombre$$

$$\backslash n$$

La letra *O* indica que es una palabra que no contiene ninguna etiqueta, la palabra *B-Nombre* indica que es la primer palabra que contiene la etiqueta de *Nombre*, la palabra *I-Nombre* indica que es parte de la etiqueta *Nombre*, pero que no es el inicio de ésta; por último \n, el salto de línea indica que la oración terminó.

3.4.5. Red Neuronal Recurrente

El tipo de red neuronal que se utiliza, es una red neuronal recurrente, RNN por sus siglas en inglés (Recurrent Neural Network). Este tipo de redes tienen la característica de *tener memoria*. Esto quiere decir que al recibir oraciones, se toma en cuenta las palabras pasadas para evaluar las palabras siguientes y considerar el contexto en el que se presenta. Una entrada nueva involucra, el dato actual y el dato anterior.

Para esta tarea, en donde se quiere identificar datos personales dentro de un conjunto de textos, la *memoria* es de suma importancia, ya que se necesita que aprenda en qué contexto puede aparecer un tipo de dato personal específico, y así poder detectarlo.

En la figura 3.7 se presentan los esquemas de una red neuronal ordinaria y una red recurrente, se puede observar que en una red recurrente, se hace referencia a que la salida depende de los datos actuales y de los datos que se han procesado anteriormente.

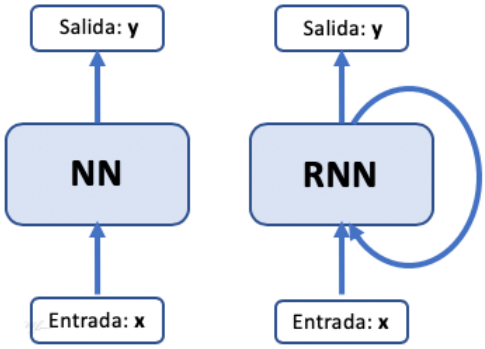


Figura 3.7: Red recurrente. Fuente: [22]

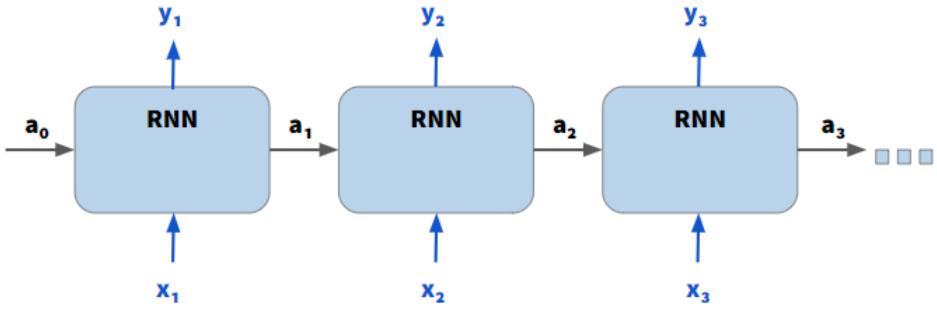


Figura 3.8: Red recurrente vista en diferentes estados de tiempo. Fuente: [23]

En la figura 3.8, se observa como sería una red neuronal a través del tiempo, en donde cada estado tiene dos entradas y dos salidas. Las entradas son x_t que es la entrada en el tiempo t, y el estado oculto, o estado de activación a_{t-1} en el instante anterior. Las salidas son a_t, el estado de activación en el instante previo, y y_t, que será la salida del dato. De esta manera queda más clara la interpretación de la figura 3.7.

Las operaciones que se realizan en este tipo de red son las siguientes:

- En primer lugar, se toman como entradas x_t que es el dato de entrada, y a_{t-1}, que es el estado de activación en el instante anterior.
- Se calcula el nuevo estado oculto, como

$$a_t = \phi_1(Wa_{t-1} + Ux_t)$$

La función ϕ_1 es una función de activación.

- Después de calcular el estado oculto previo, se calcula la salida como

$$y_t = \phi_2(Va_t)$$

donde ϕ_2 es función de activación. Los valores de las matrices de pesos U, V y W serán modificados durante el entrenamiento.

Aquí se puede observar cómo es que las entradas anteriores influyen en la nueva salida, gracias al estado oculto.

3.4.6. Red LSTM

Existe una variante de RNN, llamada LSTM (Long Short-Term Memory), esta red tiene la ventaja de tener *más memoria*, lo que permite almacenar más información de entradas pasadas. También tiene la capacidad de decidir qué cosas sí se conservan en la memoria, y qué cosas no. Esto gracias a la **celda de estado**, o canal de memoria, que es en donde se almacena la información.

Este tipo de red, LSTM, por lo general utiliza la función de activación *tanh*, además de una función sigmoide, que es la que permite el flujo o no de la información en la memoria. [22]

Para entender el problema de memoria a corto plazo en las redes recurrentes es necesario ver cómo afecta el estado oculto inicial al cálculo de la predicción final, y tomando en cuenta la notación empleada en la sección de Red Recurrente, tenemos que para la salida

$$y_3 = \phi_2(Va_3)$$

está involucrado el estado oculto a_3. Y centrando las operaciones sólo en los estados ocultos, vemos que a_3 depende de a_2

$$a_3 = \tanh(W_3 a_2)$$

Así como a_2 depende de a_1

$$a_2 = \tanh(W_2 a_1)$$

Y que a su vez, depende de a_0

$$a_1 = \tanh(W_1 a_0)$$

Así, tenemos que

$$a_3 = \tanh(W_3 \tanh(W_2 \tanh(W_1 a_0)))$$

De modo que para obtener la salida y_3 se debe pasar por 3 funciones tanh anidadas, y ya que la imagen de la función es de $(-1, 1)$, el valor de a_0 irá disminuyendo, así, si la secuencia de salida es mayor, el valor será mínimo.

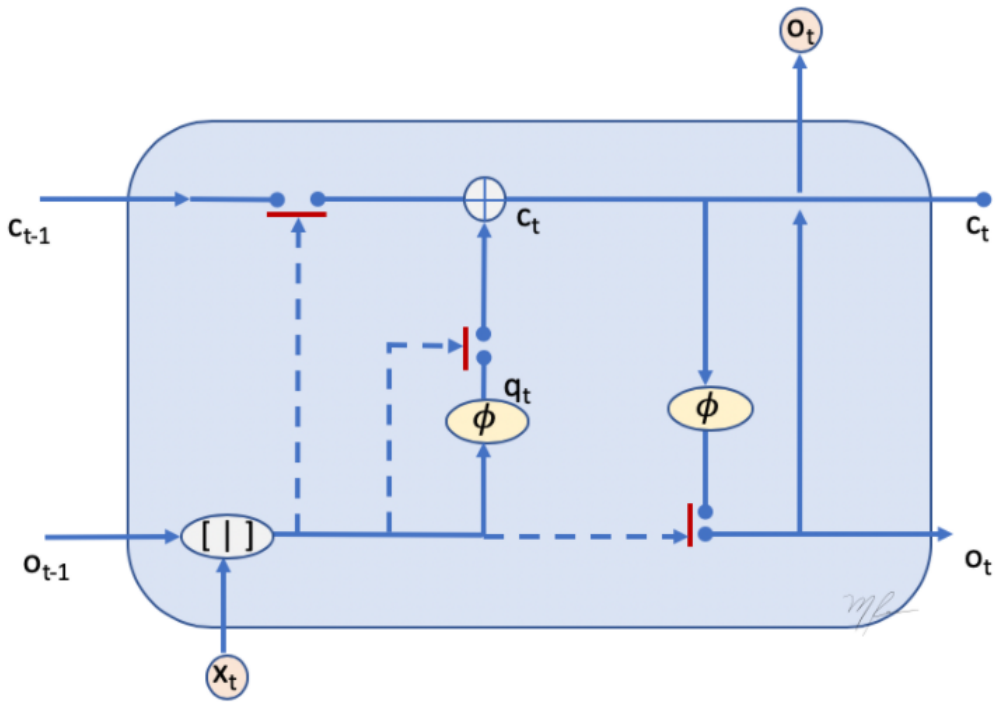


Figura 3.9: Capa LSTM. Fuente: [22]

En la figura 3.9 se observa la estructura que tiene una LSTM. Las partes que la

componen son:

- x_t es la entrada en el tiempo t
- o_{t-1} es la salida en el tiempo $t-1$, la salida anterior, que ahora forma parte de la entrada en el tiempo t
- c_{t-1} es la información contenida en la memoria, obtenida en el tiempo $t-1$
- c_t es la información actualizada en la memoria que será pasada a la siguiente etapa.
- ϕ denota la función de activación (generalmente tangente hiperbólica).
- [|] denota la concatenación.
- $+$ representa la suma.
- $\times$ representa el producto.
- σ representa la función sigmoide, que tiene valores entre $\{0, 1\}$, por lo que funciona como un switch, que permite, o no, el flujo de información.

Así, las operaciones realizadas en el **canal de memoria** son las siguientes. Para la actualización de la memoria del estado c_{t-1} a c_t se necesita de q_t, que es la información nueva.

$$c_t = r_t \times c_{t-1} + s_t \times q_t$$

donde q_t se obtiene con la entrada y la salida anterior

$$q_t = \phi(W_q \left[x_t | o_{t-1}\right] + b_q)$$

r_t y s_t funcionan como switches que controlan qué se elimina y qué se conserva dentro de la memoria. El switch que olvida, o elimina información

$$r_t = \sigma(W_r \left[x_t | o_{t-1}\right] + b_r)$$

donde b_r es el bias correspondiente.

El switch para agregar, o recordar información

$$s_t = \sigma(W_s\,[x_t|o_{t-1}] + b_s)$$

donde b_s representa el bias correspondiente.

Para el **Canal de salida**, se obtendrá una salida o_t después de procesar la información de la memoria.

$$o_t = h_t \times \phi(W_o c_t + b_o)$$

ahora h_t se puede ver como un switch de selección, en donde se eligen qué porciones del canal de memoria se conservarán.

$$h_t = (W_h\,[x_t|o_{t-1}] + b_h)$$

donde b_h es el bias correspondiente.

Todas las matrices de pesos W y los bias b son calculados durante el entrenamiento.

3.5. Conceptos para Medir el Rendimiento

Para entender cómo se mide el rendimiento de una red, es necesario que se conozcan algunos conceptos. Para esto, pensemos que se está creando un programa que sea capaz de decir si un número es par. Sea $A = \{1, 2, 3, 4, 5, 6\}$ el conjunto de números con el que el algoritmo realizará la tarea de identificar cuáles de ellos son pares. Luego del entrenamiento nos dice que los siguientes son números pares. $P = \{2, 3, 4\}$ y en consecuencia, los números impares serían los siguientes $I = \{1, 5, 6\}$.

3.5.1. Positivos y Negativos

Los positivos y negativos se refieren a la predicción del modelo. Para nuestro ejemplo, si el algoritmo decide que un número “x” es par, es un positivo; por otro lado, si no lo etiqueta como par, se considera que “x” es impar, por lo tanto es un

negativo.

3.5.2. Verdaderos y Falsos

Los verdaderos y falsos se refieren a si la predicción es correcta o no. Para el modelo de ejemplo, si predice que "x" es par y efectivamente lo es, entonces es un verdadero; de modo contrario si se tratara de un impar, sería un falso.

Ahora veamos cómo serían los conjuntos de verdaderos positivos VP y falsos positivos FP

$$VP = \{2, 4\}$$

$$FP = \{3\}$$

Entonces los conjuntos de falsos negativos FN y verdaderos negativos VN quedarían así

$$FN = \{6\}$$

$$VN = \{1, 5\}$$

3.5.3. Recuperación

La recuperación es una métrica que nos informa lo que el modelo es capaz de identificar; esto es, sabiendo las etiquetas de los datos con los que se evalua, conocer la fracción de resultados corectos que identificó respecto al total que debía identificar [24].

$$Recuperación = \frac{VP}{VP + FN}$$

3.5.4. Precisión

La precisión evalúa que tan buenos resultados se tuvieron con las etiquetas que la red consideró. Conociendo las etiquetas de los datos, muestra la fracción de verdaderos positivos, respecto de todos los positivos [24].

$$Precisión = \frac{VP}{VP + FP}$$

3.5.5. F1-score

El valor de F1 es una medida que combina la precisión y la recuperación. Se le conoce también como medida F, y es bueno para tener una representación que captura ambas medidas: recuperación y precisión; si la medida F es perfecta, significa que la recuperación y precisión tienen resultados perfectos, es decir, del 100 %. Por otro lado, se verá disminuido si al menos alguna de las dos métricas es inferior al 100 %.

$$F1 = 2 \cdot \frac{precisión \cdot recuperación}{precisión + recuperación}$$

Capítulo 4

Propuesta

Como se ha mencionado, fueron realizados dos algoritmos: Algoritmo de Desidentificación con Bases de Datos Contextuales (ADBDC) y Algoritmo de Desidentificación con Aprendizaje Profundo (ADAP).

4.1. Algoritmo de Desidentificación con Bases de Datos Contextuales

El primer algoritmo, **ADBDC**, utiliza algunas técnicas de recuperación de información, descritas en el capítulo 3. Este algoritmo fue pensado para una tarea específica: eliminar solamente los datos que son necesarios. Para entender eso, es necesario saber que sólo se protege la identidad del paciente, y esto incluye el nombre, número de teléfono y domicilio (que son los datos que pueden aparecer en las notas); sin embargo, el eliminar el nombre del paciente no sirve del todo si se conserva el nombre y número telefónico del hijo, o del padre, o de cualquier familiar, ya que esto permitiría la identificación del paciente. Por esta razón, se decidió eliminar la información de cualquier familiar. Por otro lado, los nombres de los médicos, asistentes médicos, trabajadores sociales, y demás personal hospitalario, no fueron eliminados, ya que no se consideró que hubiera una conexión directa con la identidad del paciente.

Como primer paso, cada documento debe pasar por un parser lingüístico para

identificar cada token. Cada palabra será identificada como token válido o *stop word*. Los tokens válidos serán aquellas palabras que se consideran relevantes, es decir, palabras que no sean artículos, conjunciones, pronombres, entre otros. Las palabras válidas formarán parte de un índice de tokens o diccionario de tokens.

4.1.1. Índice Invertido

El índice invertido posicional se formó para trabajar con 450 notas médicas, el diccionario de tokens, el cual corresponde a las llaves válidas para acceder a éste, consta de 2131 palabras, esto, después de eliminar los stop words de los documentos.

4.1.2. Palabras Disparadoras

Al analizar un conjunto de notas, se identificaron patrones al momento de escribir los datos personales, estos patrones indican que existen palabras que se pueden anteponer a los datos personales.

Algunas palabras disparadoras que se han identificado, son las siguientes:

- Para los nombres, le anteceden palabras como: paciente, hijo, hija, padre, madre, esposo, esposa, cuñado, sra., sr., red de apoyo, entrevista directa.
- Para los domicilios, podemos encontrarlos con las palabras disparadoras: "domicilio particular", "Domicilio calle", o sólo con la palabra "calle". Otros casos están dados por la palabra "dom" seguido del nombre de la calle, el número, la colonia (con palabras disparadoras "col" o "colonia"), y en algunos casos la ciudad y el estado.
- Para los números de teléfono, las palabras disparadoras pueden ser: "tel.", "tel", "teléfono", "cel", "celular", "número", "en caso necesario".
- Para las ciudades, tenemos palabras disparadoras como: "ciudad", "residente", "originario", etc.

4.1.3. Etapa de Búsqueda

Bases de datos contextuales

Para la identificación de nombres y apellidos se utilizaron bases de datos contextuales regionalizadas, es decir, nombres y apellidos comunes en México. Para las ciudades se utilizó una base de datos con los municipios de Michoacán, ya que los derechohabientes, son todos residentes de este estado. De esta manera, son necesarias las siguientes bases contextuales:

- Nombres. Lista con 4400 nombres comunes en México
- Apellidos. Lista con 7456 apellidos en México
- 145 municipios del estado de Michoacán.

Búsqueda

El algoritmo que se utiliza para buscar los datos personales es diferente para cada tipo de dato (nombres, apellidos, domicilio, ciudad y teléfono), sin embargo el procedimiento para las palabras disparadoras es el mismo para todos los casos. Primero, se identifican los documentos en los que aparece cada palabra disparadora, esto por medio del índice invertido posicional. En caso de que fuera más de una palabra disparadora, se verifica que sean contiguas estas palabras en el mismo documento. La continuación del algoritmo para cada tipo de dato se describirá enseguida.

Durante esta fase se hizo uso de todos los elementos, el índice invertido posicional, las bases de datos contextuales y las palabras disparadoras. El proceso fue el siguiente:

Nombres Una vez identificados los documentos y las posiciones en las que se encuentra cada palabra disparadora, se comparan las palabras posteriores una por una contra las bases de datos contextuales de Nombres y Apellidos. Si se encuentra una coincidencia, se sustituye dicha palabra por el identificador $\{NOMBRE\}$

Domicilios

Para el caso de los domicilios se utilizan combinaciones de dos conjuntos de palabras disparadoras, el primer conjunto de palabras disparadoras será: “domicilio” o

"domicilio particular", combinado con alguna de las siguientes palabras: "calle", "colonia", "col".

Específicamente se eliminaron todas las palabras contenidas entre "calle" y "colonia", o "domicilio" y "col", así como las 2 palabras válidas siguientes después de la palabra "colonia", usualmente los nombres de colonias tienen 2 o mas palabras. Una vez que se elimine el domicilio se sustituye por la etiqueta $\{DOM\}$.

Teléfonos

Para eliminar los teléfonos, basta con que después de la palabra disparadora exista un conjunto de números, al menos 7 dígitos; esto se verifica por medio de expresiones regulares, ya que puede haber otros caracteres como: 'y', '-', ',', etc. Si se cumple la condición, se eliminan los números y se coloca el identificador $\{TEL\}$.

Ciudades

De manera similar que con los nombres y apellidos, se verifica si la palabra siguiente a la disparadora, se encuentra en la base contextual de Municipios, si es así, se sustituye por la etiqueta $\{CIUDAD\}$.

4.2. Algoritmo de Desidentificación con Aprendizaje Profundo

Para el segundo algoritmo **ADAP**, se consideró eliminar todos los datos personales que aparecieran en las notas, esto con la finalidad de que hubiera un menor margen de error. Por lo que los nombres del personal médico, de enfermería y demás trabajadores de la unidad médica, no deberían aparecer. Para realizar esta tarea más general, se consideró utilizar una red neuronal recurrente. La cual debe identificar los nombres sin importar a quién hace referencia, identificar los domicilios, ciudades y teléfonos. Este algoritmo no depende estrictamente de cómo es la estructura en la nota, como sucede en ADBDC.

El modelo que tiene preentrenado la librería de *huggin face* para la tarea de reconocimiento de entidades nombradas fue utilizado para la desidentificación de las

notas médicas, sin embargo, no se obtuvieron los resultados esperados, por lo que fue necesario entrenarlo con nuestro propio corpus formado con notas médicas y parte del corpus que utiliza el modelo preentrenado, el cual nombraremos *corpus del modelo*. El corpus estaba formado por:

- Conjunto de entrenamiento (*Train*)
  - 3650 notas médicas
  - 4462 textos del corpus del modelo
- Conjunto de validación (*Dev*)
  - 399 notas médicas
  - 1516 textos del corpus del modelo
- Conjunto de prueba (*Test*)
  - 402 notas médicas
  - 1914 textos del corpus del modelo

4.2.1. Preparación del Corpus

Para poder pasar de las notas, en donde el texto está por oraciones, al formato necesario para el corpus (revisar el capítulo 3), se tuvo que realizar un algoritmo que separara palabra por palabra y además, le asignara la etiqueta correspondiente. Para esto, fue necesario revisar de manera simultánea el conjunto de notas originales, y el conjunto de notas que fueron desidentificadas manualmente.

La primer oración es parte de las notas desidentificadas, la segunda oración es la nota original.

"*Paciente hipertenso* $\{NOMBRE\}$ *es considerado remiso*"

"*Paciente hipertenso Jesús Mendoza Mares es considerado remiso*"

Se utiliza una ventana de palabras para poder identificar las palabras que anteceden y las palabras que le siguen al dato eliminado en la oración original.

"$[Paciente\ hipertenso]\ \{NOMBRE\}\ [es\ considerado]\ remiso$"

"$[Paciente\ hipertenso]\ Jesús\ Mendoza\ Mares\ [es\ considerado]\ remiso$"

Así nos damos cuenta que seguido de las palabras "Paciente hipertenso", aparecerá un dato de tipo Nombre, y que antes de "es considerado" terminará ese dato. El algoritmo coloca una etiqueta que indica el inicio de un dato personal: $\{inicioNombre\}$; y una etiqueta que señala el final de dicho dato: $\{finNombre\}$. Las etiquetas para las diferentes etiquetas son: $\{inicioDom\}$, $\{finDom\}$, $\{inicioTel\}$ y $\{finTel\}$

Realizada esa parte, otro algoritmo utiliza las notas que han sido modificadas para dividir palabra por palabra con su respectiva etiqueta. Funcionando de la siguiente manera: se lee la oración palabra por palabra, si la palabra no es alguna que indique el inicio o final de una etiqueta, la palabra debe ir seguida por una letra "O"

$$Paciente\ O$$

$$hipertenso\ O$$

Cuando sea una etiqueta de inicio de datos personales, a la primer palabra siguiente se le asignará una letra "B" seguido de la etiqueta correspondiente

$$Jesús\ B - Nombre$$

Las siguientes palabras antes de la etiqueta de final de datos personales, llevarán una letra "I" seguido de la etiqueta a la que pertenece

$$Mendoza\ I - Nombre$$

$$Mares\ I - Nombre$$

Al final, la nota quedará de la siguiente manera

$$Paciente\ O$$

$$hipertenso\ O$$

$$Jesús\ B-Nombre$$

$$Mendoza\ I-Nombre$$

$$Mares\ I-Nombre$$

$$es\ O$$

$$considerado\ O$$

$$remiso\ O$$

$$\backslash n$$

Ese procedimiento se lleva a cabo con las notas utilizadas para formar el corpus.

Capítulo 5

Experimentación

5.1. Algoritmo de Desidentificación con Bases de Datos Contextuales

5.1.1. Desidentificación

Para llevar a cabo la desidentificación, se almacena el conjunto de todas las notas en una estructura de datos que permite acceder al elemento deseado por medio de un índice, a este conjunto lo llamaremos N. Así, cada nota puede ser considerada como un documento independiente.

Luego de realizar todo el preprocesamiento al texto y crear el índice invertido posicional, (mencionado en el capítulo 4), se realiza la desidentificación de los datos personales de un solo tipo a la vez en cada nota de N, empezando por los procesos que se basan en diccionarios contextuales y que sólo eliminan las palabras que coincidan:

- Los primeros datos eliminados son los nombres y apellidos.
- Luego se eliminan los municipios.
- Posteriormente se identifican los números telefónicos
- Como paso final, los domicilios son eliminados. Estos datos son los últimos en eliminarse, puesto que implican el borrar un mayor número de palabras, lo que

podría ocasionar problemas en la búsqueda de los demás datos.

5.2. Algoritmo de Desidentificación con Aprendizaje Profundo

5.2.1. Desidentificación

La red se entrenó utilizando una GPU ofrecida por los servicios de Google Colab. Los mejores resultados que se obtuvieron fueron en la época 38, siendo: *loss = 0.0668* y *f1-score = 0.7861*. El entrenamiento tuvo 43 épocas, en un tiempo aproximado de 8 horas y 42 minutos.

Una vez que se tiene la red entrenada, ésta es capaz de identificar el tipo de dato al que corresponden las palabras de las notas. El procedimiento para eliminar los datos personales es el siguiente:

- Cada oración es enviada al modelo.
- Se obtiene una lista de las etiquetas (Name, Loc para los domicilios y ciudades y Cel) que el modelo identificó en la nota, la cual indica las posiciones en las que se encuentra cada una.
- Conociendo las posiciones en las que se encuentran los datos personales, son reemplazados por la etiqueta correspondiente que indica que existía un dato personal ($\{NOMBRE\}$, $\{CEL\}$ o $\{DOM\}$).

5.3. Pruebas

Para poder realizar evaluaciones, es necesario tener un punto de referencia, por lo que se desidentificaron algunas notas que nos permitirán analizar los resultados que se obtengan. De este modo, fueron desidentificadas manualmente un total de 28000 notas. Para esta tarea se contó con la ayuda de los investigadores: Dra. Ricarda Cortés

Vieyra y Dr. Luis Miguel Saavedra Pimentel, como parte del proyecto aprobado por CONACYT.

El conjunto total de notas es de 300,070. Al final de este capítulo se mostrarán los resultados obtenidos por el algoritmo ADAP en todo el conjunto de notas.

Para llevar a cabo las pruebas, se necesitaron conjuntos diferentes de notas, uno de 450 notas, y otro de 2000 notas. Ambos conjuntos fueron etiquetados manualmente para poder hacer la evaluación de los resultados de los algoritmos.

El ADBDC fue evaluado con el conjunto de 450 notas, mientras que el ADAP por su parte, se evaluó con el conjunto de 2000 notas.

Recordando que ADBDC utiliza palabras disparadoras para poder localizar los datos identificadores, este algoritmo logra eliminar los datos de los pacientes, así como de sus familiares. Por otro lado, ADAP elimina, por ejemplo, nombres de otras personas que no necesariamente podrían ayudar a la identificación de los pacientes, por ejemplo: doctores, trabajadores sociales, enfermeras, entre otros. Aclarando esto, la comparación de resultados obtenidos, no está en las mismas condiciones, aún cuando usan el mismo conjunto de datos.

En la tabla 5.1 vemos los resultados del ADBDC obtenidos utilizando el conjunto de 450 notas. Se puede observar que la recuperación de teléfonos es muy buena.

Tabla 5.1: Comparación de resultados entre ADBDC y una identificación manual.

Tipo de datos	ADBDC	Identificados manualmente	Recuperación
Teléfono	296	300	98.66 %
Domicilio	56	68	82.35 %
Nombres	133	164	81.09 %
Ciudad	67	79	84.81 %
General			85.43 %

Para evaluar los resultados del ADAP, se utilizó el conjunto de 2000 notas, mismas que también fueron desidentificadas manualmente, además se hizo una revisión más delicada para obtener las métricas de precisión y recuperación. Los resultados se muestran en la tabla 5.2

Los resultados obtenidos con el conjunto total de notas desidentificadas manualmente se muestra en la tabla 5.3. Recordar que en total es un conjunto de 28000 notas,

Tabla 5.2: Resultados obtenidos por ADAP en el conjunto de 2000 notas

Tipo de datos	ADAP	Identificados manualmente	Recuperación	Precisión	F-1 score
Teléfono	238	241	97.92 %	99.15 %	98.53 %
Nombres	436	501	86.62 %	99.54 %	92.63 %
Domicilio y Ciudad	558	594	89.73 %	95.51 %	92.52 %
General			90.04 %	97.64 %	93.68 %

menos las 4000 notas que se utilizaron para el corpus. Nótese que no se muestra la precisión, esto es porque no se conocen los falsos positivos. Cabe mencionar que los resultados que se muestran sólo es un conteo de etiquetas comparado con el número de etiquetas que se identificaron manualmente.

Tabla 5.3: Resultados obtenidos por ADAP en el conjunto de 24000 notas que han sido desidentificadas

Tipo de datos	ADAP	Identificados manualmente	Recuperación
Teléfono	3623	3918	92.47 %
Nombres	10896	11146	97.75 %
Domicilio y Ciudad	9102	9130	99.69 %
General			97.63 %

Para el conjunto total, que contiene 300,070 notas, los resultados se muestran en la tabla 5.4. La tabla muestra sólo la cantidad de datos eliminados, es decir, el número de etiquetas de cada tipo de dato que colocó, ya que no se puede conocer la recuperación obtenida.

Tabla 5.4: Datos eliminados por ADAP, en el conjunto total de notas

	Nombres	Teléfonos	Domicilio y Ciudad	Totales
ADAP	393,830	24,791	59,245	477,866

Capítulo 6

Conclusiones

Se realizó una investigación sobre trabajos de desidentificación de documentos clínicos, encontrando sólo investigaciones para archivos en el idioma inglés. Se utilizaron algunas técnicas de recuperación de información, además de expresiones regulares, para desarrollar el algoritmo ADBDC. Para el algoritmo ADAP se utilizaron redes neuronales recurrentes, en específico redes LSTM.

Los resultados obtenidos por ADBDC son buenos para el propósito que se realizó, eliminar la información del paciente, y de sus familiares directos, dejando a un lado la información de los médicos, trabajadores sociales, y demás personal del hospital; sin embargo, el colocar de manera manual las reglas que debe seguir para identificar los datos, no es una forma muy óptima de crear un modelo, ya que deberían estar actualizándose constantemente para mejorar los resultados. No obstante, los resultados que se obtuvieron al eliminar los teléfonos fueron bastante buenos.

Las modificaciones de las reglas para que ADBDC identificara los datos personales fueron constantes. Para los domicilios fue un caso particular, ya que puede variar mucho la forma en la que se escriben, si mencionan la colonia, sólo la calle, o el nombre del fraccionamiento. Para el caso de los nombres, pueden o no tener palabra disparadora, en algunos casos sólo se refiere con su nombre. Un problema que también dificulta las cosas, son los errores ortográficos, ya que no coinciden las palabras disparadoras, o los elementos de los diccionarios.

Por su parte, ADAP, que durante su entrenamiento formula sus propias reglas

para identificar los datos personales, obtiene mejores resultados, además que elimina también los nombres del personal médico.

ADAP también presenta sus dificultades, principalmente a la hora de formar el corpus, el algoritmo que prepara cada palabra con su respectiva etiqueta tenía sus complicaciones, y se debía revisar que las etiquetas no tuvieran errores que pudieran afectar el correcto aprendizaje.

Los resultados obtenidos por ADAP en las 24000 notas sólo muestran una recuperación aproximada. Para el caso del conjunto total, que consta de 300,070 notas, sólo se muestra el conteo de etiquetas totales que tuvo, esto debido a que no se puede realizar una comparación sin tener un punto de referencia del cual basarnos.

Bibliografía

[1] Mehmet Kayaalp, Allen Browne, Zeyno Dodd, Pamela Sagan, and Clement Mcdonald. De-identification of address, date, and alphanumeric identifiers in narrative clinical reports. *AMIA Annual Symposium proceedings.*, 2014:767–76, 11 2014.

[2] B. Malin. Guidance regarding methods for de-identification of protected health information in accordance with the health insurance portability and accountability act (hipaa) privacy rule. 2012.

[3] Stephane Meystre, F Friedlin, Brett South, Shuying Shen, and Matthew Samore. Automatic de-identification of textual documents in the electronic health record: A review of recent research. *BMC medical research methodology*, 10:70, 08 2010.

[4] Bruce Beckwith, Rajeshwarri Mahaadevan, Ulysses Balis, and Frank Kuo. Development and evaluation of an open source software tool for deidentification of pathology reports. *BMC medical informatics and decision making*, 6:12, 02 2006.

[5] F Jeff Friedlin and Clement J McDonald. A software tool for removing patient identifying information from clinical documents. *Journal of the American Medical Informatics Association*, 15(5):601–610, 2008.

[6] Dilip Gupta, Melissa Saul, and John Gilbertson. Evaluation of a deidentification (de-id) software engine to share pathology reports and clinical documents for research. *American journal of clinical pathology*, 121(2):176–186, 2004.

[7] Yikun Guo, Robert Gaizauskas, Ian Roberts, George Demetriou, Mark Hepple, et al. Identifying personal health information using support vector machines. In *i2b2 workshop on challenges in natural language processing for clinical data*, pages 10–11. Citeseer, 2006.

[8] Ozlem Uzuner, Yuan Luo, and Peter Szolovits. Evaluating the state-of-the-art in automatic de-identification. *Journal of the American Medical Informatics Association : JAMIA*, 14:550–63, 06 2007.

[9] György Szarvas, Richárd Farkas, and Róbert Busa-Fekete. State-of-the-art anonymization of medical records using an iterative machine learning framework. *Journal of the American Medical Informatics Association*, 14(5):574–580, 2007.

[10] Özlem Uzuner, Tawanda C Sibanda, Yuan Luo, and Peter Szolovits. A de-identifier for medical discharge summaries. *Artificial intelligence in medicine*, 42(1):13–35, 2008.

[11] Claudine Badue, Ricardo Baeza-Yates, Berthier Ribeiro-neto, and Nivio Ziviani. Distributed query processing using partitioned inverted files. pages 10–20, 01 2001.

[12] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. ISBN: 0521865719.

[13] Roi Blanco and Álvaro Barreiro. Boosting static pruning of inverted files. In *SIGIR*, 2007.

[14] Sandro Skansi. *Introduction to deep learning: from logical calculus to artificial intelligence*. Springer, Cham, Switzerland, 2018. ISBN: 978-3-319-73004-2.

[15] Charu Aggarwal. *Neural networks and deep learning : a textbook*. Springer, Cham, Switzerland, 2018. ISBN: 978-3-319-94463-0.

[16] Francois Chollet. *Deep learning with Python.* Manning Publications Co, Shelter Island, NY, 2018. ISBN: 9781617294433.

[17] Bekir Karlik and A Vehbi Olgac. Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, 1(4), 2011.

[18] Alberto Herrera López. Reconocimiento automático de expresiones faciales usando aprendizaje profundo. Tesis de licenciatura, Universidad Michoacana de San Nicolás de Hidalgo. 2019.

[19] Mariano Rivera. Redes neuronales artificiales, 2017. `http://personal.cimat.mx:8181/~mrivera/cursos/aprendizaje_profundo/backprop/backpropagation.html`.

[20] Stefan Schweter and Alan Akbik. Flert: Document-level features for named entity recognition. 2020.

[21] Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649, 2018.

[22] Mariano Rivera. Introducción a redes neuronales recurrentes, 2018. `http://personal.cimat.mx:8181/~mrivera/cursos/aprendizaje_profundo/RNN_LTSM/introduccion_rnn.html#la-capa-lstm-de-keras`.

[23] Codificando Bits. Redes neuronales recurrentes: ExplicaciÓn detallada. `https://youtu.be/hB4XYst_t-I`.

[24] José Martinez Hera. Precision, recall, f1, accuracy en clasificación, 2020. `https://www.iartificial.net/precision-recall-f1-accuracy-en-clasificacion/`.