.UBAfiuba
FACULTAD DE INGENIERÍA

PREDICCION DE ENFERMEDADES NEURODEGENERATIVAS MEDIANTE NLP

Eichenbaum, Daniel Matías

APROBADA

10 (diez)

Tesis de Grado de Ingeniería Electrónica

UNIV. de Bs. As. - FACULTAD DE INGENIERIA - Departamento de Electrónica

Pedro Arini

Zanutto

Ricardo Veiga

Director:
Dr. Ing. Lew, Sergio Eduardo

Jurados:
Ing. Veiga Ricardo
Dr. Ing Arini, Pedro D.
Dr. Ing. Zanutto, B. Silvano

CIUDAD AUTONOMA DE BUENOS AIRES, 2 *DE* 08 *DEL 2023*

Prólogo

El objetivo del presente documento será el de conocer en qué medida es posible detectar la presencia de **Alzheimer (AD)** o la **variante de comportamiento de Demencia Frontotemporal (FTDbv)** utilizando el discurso generado por el paciente, al preguntarle que cuente un día de su vida. Para ello se hará uso de los avances científicos más recientes en relación al campo ***Natural Language Processing (NLP)***. El estudio se limita a discursos de pacientes con edades superiores a los 50 años de nacionalidad colombiana o chilena. Entre las mayores dificultades a afrontar se incluye el tratamiento con datos escasos, sesgos de adquisición, modelado y evaluación del modelo con pocas muestras.

Entre los artículos relacionados destaca *Automated Text-Level Semantic Markers of Alzheimer's disease* (Sanz et al. 2022) donde no solo los autores introducen un dataset en español similar al aquí utilizado, sinó que presentan evidencia significativa de diferentes marcadores en el discurso (como la granularidad y cercanía semántica entre palabras consecutivas) que permite discriminar pacientes con **Alzheimer (AD)** de pacientes **sanos de control (HC)**. Además el proceso de adquisición de datos es similar al aquí utilizado. En cuanto al modelado, destaca el artículo *Deep learning-based speech analysis for Alzheimer's disease detection: a literature review* (Yang et al. 2022) donde se presenta una vasta revisión de literatura, modelos y tareas con sus respectivos *datasets* en **inglés**, para el diagnóstico de **Alzheimer**. Entre ellas se incluye el modelo **BERT** y **Longformer** con *Accuracy* entre $82\% - 89.58\%$.

Keywords:
NLP, **Transformers**, **Alzheimer (AD)**, **Demencia Frontotemporal variante comportamiento (FTDbv)**, **BETO**, **Logformer**

Referencias del prologo

Sanz, Camila et al. (2022). “Automated text-level semantic markers of Alzheimer’s disease”. In: *Alzheimer’s & Dementia: Diagnosis, Assessment & Disease Monitoring* 14.1, e12276. DOI: https://doi.org/10.1002/dad2.12276. URL: https://alz-journals.onlinelibrary.wiley.com/doi/abs/10.1002/dad2.12276.

Yang, Qin et al. (2022). “Deep learning-based speech analysis for Alzheimer’s disease detection: a literature review”. In: DOI: 10.1186/s13195-022-01131-3. URL: https://doi.org/10.1186/s13195-022-01131-3.

Agradecimientos

En primer medida, se agradece enormemente a **Adolfo Martín García**, a **Bruno Cernuschi Frías**, a **Patricia Pelle**, a **Ricardo Veiga** y por supuesto a **Sergio E. Lew** por su infinita predisposición y constante asesoramiento en la realización de la presente tesis de grado. Sin su conjunta cooperación este documento no podría haber existido.

Se agradece enormemente a la **Universidad de Stanford** por hacer público el curso CS224N el cual sirvió de inspiración para este documento. En igual medida, se agradece a la **Universidad de San Andrés** y la **Facultad de Ingeniería de Buenos Aires** por brindar el espacio académico.

Se agradece también al equipo de **Google Colab** por permitir el entrenamiento de los modelos aquí presentes de forma gratuita.

Y por supuesto quedo enormemente agradecido con toda mi **familia** y **amigos** que me supieron soportar en este largo recorrido

En memoria de **Bruno Cernuschi Frías**

Índice

Estudio del Problema

Capítulo **1**

1.1 Trastornos Neurodegenerativos

Los **trastornos neurodegenerativos** (*Neurodegenerative diseases : unifying principles* 2017) son enfermedades que afectan al sistema nervioso y se caracterizan por la degeneración progresiva de las células nerviosas. Entre los síndromes clínicos se caracteriza la pérdida de memoria, cambios de personalidad, perturbaciones motoras, dificultades cognitivas. Algunos de los trastornos neurodegenerativo son el Alzheimer (AD), la demencia Fronto Temporal (FTD), la enfermedad de Parkinson (PD), la esclerosis lateral amiotrófica (ELA), la enfermedad de Huntington (HD).

1.1.1 Alzheimer (AD)

El Alzheimer se caracteriza por la pérdida de memoria del paciente. La degeneración suele comenzar en el hipocampo (Fig. 1.1 y 1.2) y luego extenderse hacia áreas cercanas. En edades avanzadas los pacientes suelen perder la habilidad de formular razonamientos lógicos y su discurso suele repetir oraciones frecuentes dichas por otros. Además los pacientes suelen cantar, balbusear o decir palabras por fuera de la conversación. En la Fig. 1.3 puede apreciarse una resonancia magnética de un

paciente con Alzheimer avanzado.

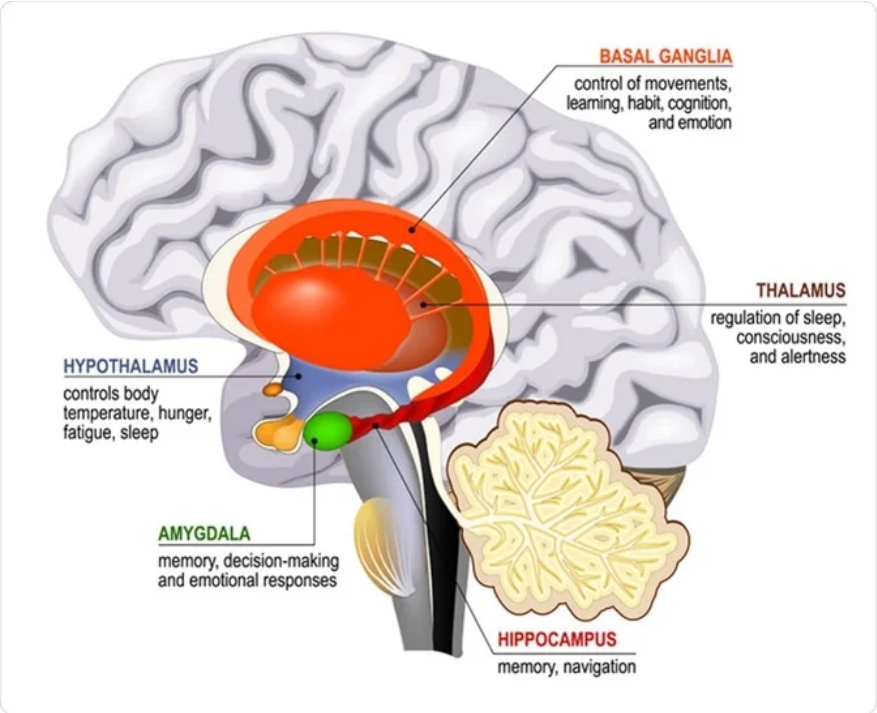


Figure 1.1: Ilustración que sintetiza diferentes tejidos del sistema nervioso central con sus funciones importantes, incluyendo al hipocampo.*

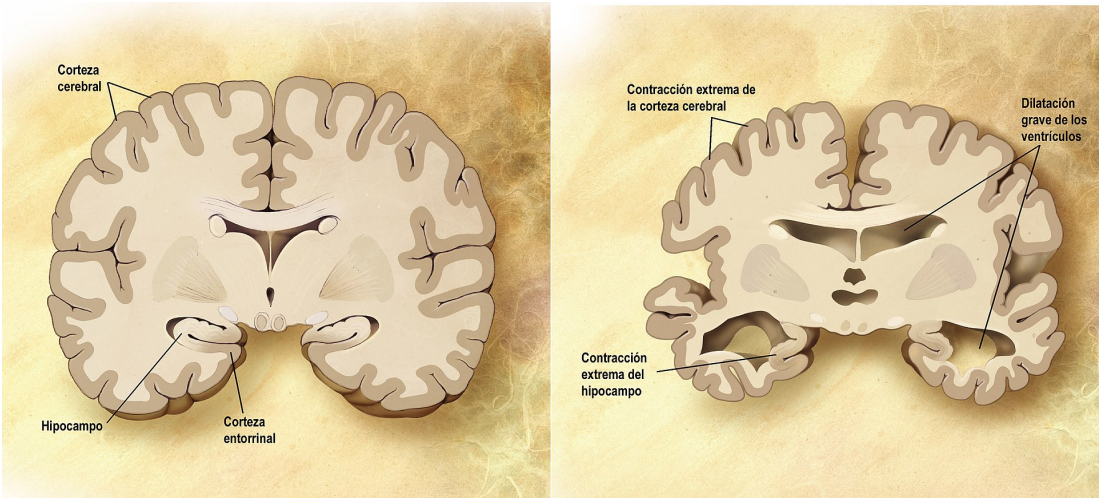


Figure 1.2: Esquema frontal ilustrando la ubicación del hipocampo para un paciente sano (**izq**) y uno con AD (**der**).†

*Fuente: Dutta, Sanchari Sinha. (2019, August 20). Hippocampus Functions. News-Medical. Recuperado 16/02/2023 de `https://www.news-medical.net/health/Hippocampus-Functions.aspx`

†Fuente: Wikimedia Commons. Recuperado el 18/02/2023 de `https://commons.wikimedia.org/wiki/File:Cerebro_corte_frontal_Alzheimer.jpg`

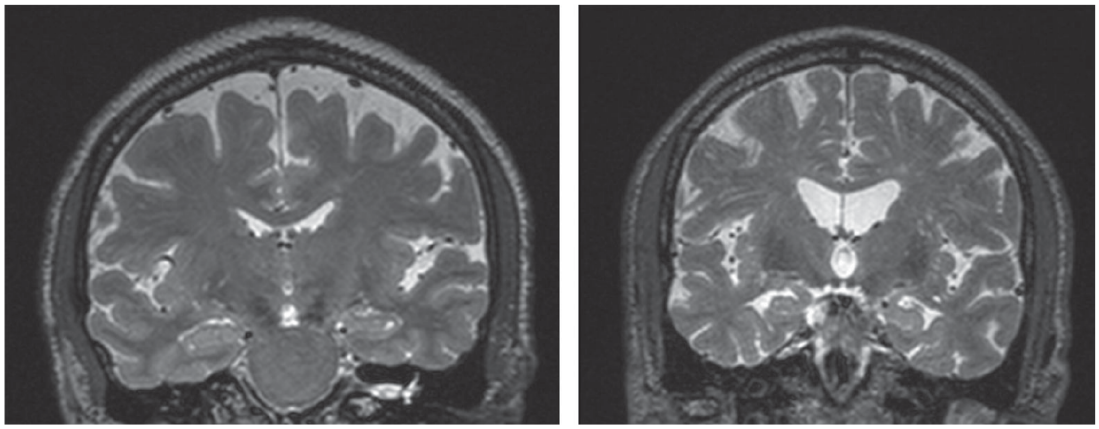


Figure 1.3: Imágen de Resonancia Magnética (MRI) del plano frontal de un paciente normal (**izq**) y un paciente con Alzheimer, mostrando reducido volumen en el hipocampo (**der**).[‡]

1.1.2 Demencia Frontotemporal (FTD)

Las demencias frontotemporales son un grupo de desórdenes que ocurre cuando las células nerviosas del lóbulo temporal y frontal (Fig. 1.4) se deterioran (Fig. 1.5). La demencia frontotemporal puede afectar el comportamiento, la personalidad el lenguaje y el movimiento.

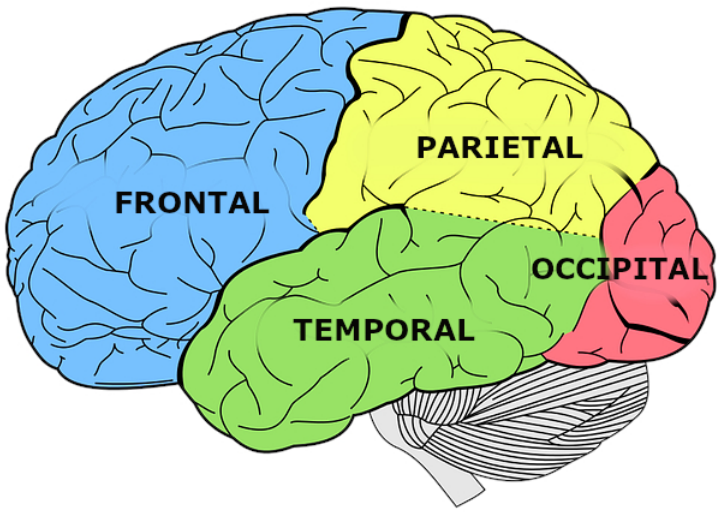


Figure 1.4: Ilustración de la ubicación de los lóbulos que constituyen el encéfalo. [§]

[‡]Fuente: *Neurodegenerative diseases : unifying principles* 2017

[§]Fuente: *Lóbulos cerebrales para principiantes*, Accedido el 19/02/2023 en `https://www.psycolab.com/lobulos-cerebrales-para-principiantes/`

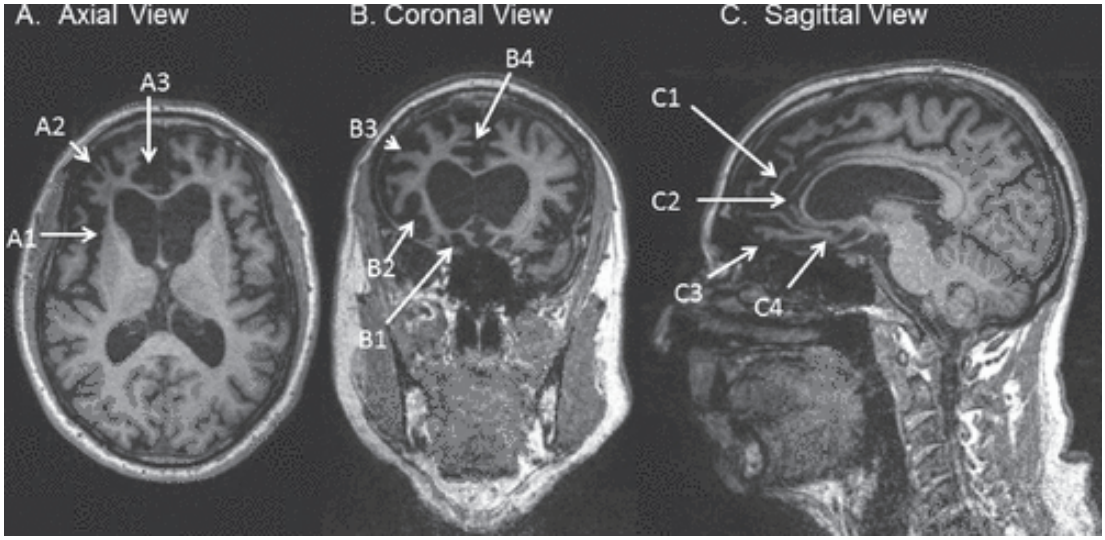


Figure 1.5: Imágen de Resonancia Magnética (MRI) de un paciente con FTDbv. **(A)** Vista axial del cuerpo calloso mostrando atrofia asimetríca del lóbulo frontal. **(B)** Vista frontal del cuerpo calloso y **(C)** Vista Longitudinal del hemisferio derecho, mostrando especialmente atrofia frontal [¶]

La variante conductual **(FTDbv)** es responsable de aproximadamente la mitad de todos los casos de esta enfermedad y suele afectar al comportamiento y la personalidad. A diferencia del Alzheimer, la memoria queda relativamente preservada, sin embargo los pacientes no consiguen reconocer sus propios cambios en comportamiento. Entre los signos más frecuentes se incluyen: Desinhibición o pérdida de moderación basada en normas sociales (comentarios groseros, ignorar espacio personal, arrebatos agresivos), Apatía o indiferencia (pérdida de interés, descuido de la higiene personal), Embotamiento emocional o falta de empatía, Comportamientos compulsivos o ritualistas (Repetición de palabras o frases, releer el mismo libro siempre), déficits en la función ejecutiva o mala toma de decisiones (dificultad para planificar) entre otras. [□]

[¶]Fuente: Peter Ljubenkov y Bruce Miller*A Clinical Guide to Frontotemporal Dementias*, Focus, 13 de oct de 2016, `https://focus.psychiatryonline.org/doi/10.1176/appi.focus.20160018`

[□]Fuente: *Frontotemporal Dementia*, Accedido el 18/02/2023 de `https://www.hopkinsmedicine.org/health/conditions-and-diseases/dementia/frontotemporal-dementia` y
Variante de comportamiento FTD, Accedido el 18/02/2023 de `https://www.theaftd.org/es/what-is-ftd/behavioral-variant-ftd-bvftd/`

1.2 Adquisición de datos

El dataset aquí utilizado es propiedad del **Centro de Neurociencias Cognitivas (CNC)** de la **Universidad de San Andrés** y está sujeto a acuerdos de confidencialidad, por lo que una caracterización más precisa, por el momento, no está disponible. Sin embargo, la metodología empleada para la recopilación es similar al utilizado en las siguientes publicaciones: (Sanz et al. 2022 y Eyigoz et al. 2020). Donde el *dataset* se construye preguntando a diferentes pacientes (Chilenos y Colombianos) que relaten un día de su vida. Estos discursos (en audio) fueron grabados y luego transcriptos por un sistema de reconocimiento del habla, además se han eliminado las partes del discurso que comprometen la integridad y privacidad del paciente. En la **Fig. 1.6** se observan ejemplos del discurso de diferentes voluntarios.

Voluntario con AD (Colombia)

tengo muchos hoy en día felices eran felices cuando nacieron mis hijas eran muy felices uno y cuéntemelo entonces voy a contarte el último si bien feliz el día en que conocí mi actual señora eso fué una cita a ciegas , que llaman una exnovia con la cual estuve una señora con la cual estuve como seis años y cuando cuando yo volvía y me separaba yo la conocí y ella tenía hijos pequeños yo tenía ya grandes mis hijas pues en el transcurso de este tiempo ya habían terminado universidad se casaron y se fueron a vivir a otro paí tonces se fueron mis hijas que yo adoraba la señora con la que andaba claro entonces nos fuimos a organizar esta puerca relación y cuando huy espérame que mis hijas están ya grandes peor que los hijos estaba pequeños dijo que un tiempo adiós luego le dije a ella preséntame a alguien estoy desconectado casado con la vida que no conoce mis hijas que tenían el punto de contacto pues vivían fuera del país tonces esta novia por medio de esta novia conocí a mi actual señora fué a fuí a la masajista y y le dijo mi exnovia mi exnovio Aníbal quiere conocer a alguien porque a ver si me da su nombre que es una buena gente bien? la pacienta que siguió es mi actual señora me dijo el señor me dijo que usted quiere conocer a alguien me dijo ay que jartera! porqué mi mamá trabajó para él la mamá de ella había trabajado para mí y entonces no pues entonces me llamó y me dijo llámate a fulana de tal y la llamé cuando la ví escribí en mi agenda esta mujer será mía para siempre y no le había dicho que hubo y ocurrió así es mi segunda esposa

Voluntario de CTR (Colombia)

mi labor empieza las cuatro y treinta de la mañana me levanto me hago mi aseo personal luego tomo un un desayuno un yogurt con frutas y hago una caminada de treinta minutos caminada trote después me alisto y salgo a las seis y cuarto para la oficina llego mas menos un cuarto para las siete y hacer pues labores leer el correo ver que otras cosas hay atender preguntar como están los pedidos las ventas organizar todo como está la producción que no la hacen en otro sitio pero de luego pues que hace falta de materias primas y demás finalmente tuve unos días ahí reuniones con todos los vendedores para motivarlos y a hacer ya las cosas de rutina ese es más o menos el día almuerzo cuando no tengo compromisos por fuera almuerzo ahí en la empresa y y salgo a las dos dos y dos y media de la tarde ese es más o menos llego a la al apartamento leo el periódico a esa hora luego generalmente cuatro veces a la semana hago meditación hago prácticas de meditación y y veo noticieros nada de telenovelas pero me gusta ver todos los noticieros nacionales e internacionales ese es ese es más o menos digamos el el día

Voluntario con FTDbv (Colombia)

cuando cuando decidimos salirnos de para irnos a a un conegar a Estados Unidos pues resultó como mi marido es médico antes de irnos para Estados Unidos había un congreso aquí de ortopedia entonces el dijo ah asistimos al congreso asisto al congreso de ortopedia y nos vamos y en ese congreso se conoció con el doctor Oscar Escallante un famoso médico de columna de Italia y le dijo no, porque no te vas conmigo para Italia y entonces llegó a la casa todo contento me dijo pues ya no es pa Estados Unidos es para Italia y nos fuimos para Italia dónde fué una belleza duramos como cuatro años porque cada año le aumentaban otro le daban oportunidades y bueno en fin pasamos divino y la gente italiana es adorada mi marido con su guitarra porque él hasta hace muy poquito hace que tocó guitarra eso se arrebató con los los italianos se arrebataron con él hicimos muy buenas amistades nos orinentamos muy bien gracias a dios paramos yo no sé gracias a dios si, y fueron cuatro años muy lindos de la vida hicimos muchas amistades como pudimos después duramos muchísimos años escribiéndonos hasta que al fin se fué como como mermando la frecuencia de la de la comunicación y ya no nos escribimos realmente pero es que ya hace mucho tiempo también que regresamos nosotros tuvimos cincuenta y tres años de casados tonces eso ha pasado mucha cosa y ya

Figure 1.6: Ejemplos de discursos de voluntarios con AD, FTDbv y CTR

1.3 Exploración de los datos

Esta sección describirá diferentes características que son propias de los datos adquiridos.

1.3.1 Muestras disponibles

El dataset está compuesto de 43 muestras de las cuales:

- 21 corresponden a discursos de pacientes con **AZ** (15 Chilenos + 6 Colombianos)
- 21 corresponden a discursos de pacientes con **FTDbv** (21 Colombianos)
- 21 corresponden a discursos de pacientes de **CTR** (12 Chilenos + 9 Colombianos)

En la **Tabla 1.1**, se enumeran los discursos agrupados por país, sexo y diagnóstico.

Diagnóstico	Colombia			Chile	
	M	V	O	M	V
CTR	4	5	0	9	3
AD	1	5	0	9	6
FTDbv	16	4	1	0	0

Tabla 1.1: Cantidad de muestras en el dataset filtrados por país y sexo para cada diagnóstico. (M: Mujer, V: Varon, O:Otros)

1.3.2 Rango de Edades

El rango de edades de los voluntarios se encuentra entre los 50 y 100 años. En la **Fig. 1.7** se observa que cada diagnóstico contiene muestras en un amplio espectro de edades.

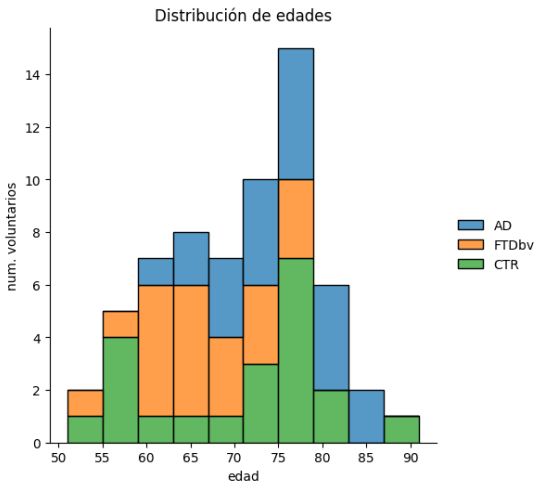


Figure 1.7: Distribución de las edades proveniente de pacientes con **AD**, **FTDbv** ó **CTR**.

1.3.3 Longitud de los discursos

En la **Fig. 1.8** se agrupan los discursos por longitud de texto y diagnóstico. Se observa que el discurso medio contiene un valor medio aproximado de 260 símbolos. Además se observa que la cantidad de discursos cortos (200 símbolos) suele provenir de pacientes diagnosticados con **AD** o **FTDbv**.

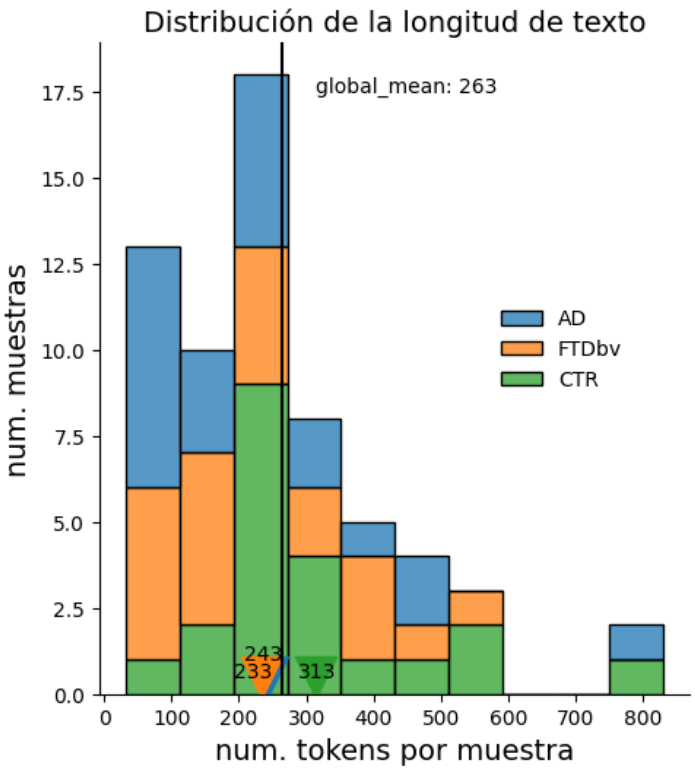


Figure 1.8: Distribución de la longitud de símbolos (tokens) por cada muestra (discurso) proveniente de pacientes con **AD**, **FTDbv** ó **CTR**. Se observa que los pacientes con AD o FTDbv suelen generar discursos cortos en promedio (233 y 243 símbolos respectivamente). Mientras que los voluntarios de CTR generaron discursos con longitud media de 313 símbolos.

1.3.4 Distribución del género

En la **Fig. 1.9** se agrupan los pacientes por género y diagnóstico. Si bien hay una diferencia significativa entre cantidad de mujeres y varones, la distribución entre clases de diagnósticos se mantiene uniforme.

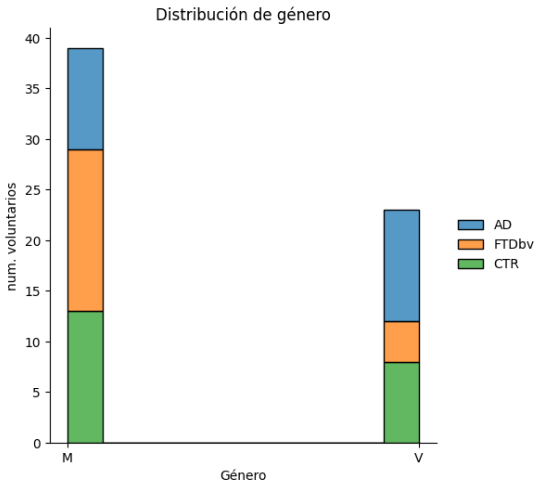


Figure 1.9: Distribución del género en función del diagnóstico del paciente

1.3.5 Sesgos

En la **Tabla 1.1** se observan cantidades asimetrícas de diagnósticos por país. Por ejemplo, todos los pacientes con **FTDbv** son Colombianos. Esto podría sesgar al modelo a creer que los Chilenos no pueden tener tal diagnóstico debido a la insuficiencia de datos. Este incoveniente, será resuelto en la sección 3.1.5 descartando datos hasta obtener la misma población de diagnósticos para todos los países.

Debido a la ausencia de muestras Chilenas con Demencia Frontotemporal (FTDbv) no será posible constituir un único dataset que abarque ambos diagnósticos (o se deduciría erróneamente que los chilenos sean inmunes al FTDbv). Es por ello, que se separán los datos en dos datasets. CTR vs AD y el otro CTR vs FTDbv.

También los datos presentan un sesgo de género, no se dispone de la misma cantidad de muestras mujeres que de varones. Sin embargo este sesgo podría subsanarse en un futuro al recolectar un mayor número de muestras.

Otros sesgos que están presentes, provienen de las variaciones en la adquisición de los datos. Por ejemplo algunos discursos capturaron las muletillas (’eee’ o ’mmm’) o por ejemplo, diferentes profesionales capturaron de forma diferente los discursos de los pacientes ya sea cuando el paciente le realiza una pregunta al entrevistador (¿Ya empiezo?, ¿Un día felíz o uno triste? ¿Usted me entiende no?).

De no reducirse todos estos sesgos a la hora de construir el modelo, existe la posibilidad de que el modelo aprenda a generar diagnosticos por el patrón linguístico equivocado. Está claro que el bajo volumen de datos aquí presentes, sería ingenuo pretender que el sistema esté libre de sesgos. Evaluar el sistema con más datos sería lo recomendado.

1.3.6 Limpieza de los datos

La limpieza de datos tiene la finalidad de reducir los sesgos mencionados en la sección 1.3.5. La limpieza se llevó en 4 pasos, para cada discurso.

- Se eliminaron los caracteres introducidos por el reconocedor de habla tales como $+, -.*, \wedge$, estos símbolos son ajenos al discurso.
- Se eliminaron las muletillas (monosílabos eee, xxx, mmm) puesto que estos símbolos no estaban contemplados homogéneamente en todos los discursos.
- Se eliminaron los pequeños diálogos entre el paciente y el profesional que adquirió la muestra. (Ej. *"Empiezo ya?"*)
- Se reemplazaron los nombres propios de las ciudades para reducir el rastro sobre la procedencia del paciente. Además se reduce el sesgo por detectar una enfermedad basado en lugar de procedencia.

Referencias

Capítulo 1

Eyigoz, Elif et al. (2020). “From discourse to pathology: Automatic identification of Parkinson’s disease patients via morphological measures across three languages”. In: *Cortex* 132, pp. 191–205. DOI: https://doi.org/10.1016/j.cortex.2020.08.020. URL: https://www.sciencedirect.com/science/article/pii/S0010945220303245.

Neurodegenerative diseases : unifying principles (2017). eng. New York, NY: Oxford University Press.

Sanz, Camila et al. (2022). “Automated text-level semantic markers of Alzheimer’s disease”. In: *Alzheimer’s & Dementia: Diagnosis, Assessment & Disease Monitoring* 14.1, e12276. DOI: https://doi.org/10.1002/dad2.12276. URL: https://alz-journals.onlinelibrary.wiley.com/doi/abs/10.1002/dad2.12276.

Método

Capítulo 2

El objetivo del presente capítulo es introducir al lector diferentes modelos que permitan predecir, si un paciente padece o no de determinado trastorno utilizando su discurso como información de entrada. Se busca que el lector, sin abundar en detalles, adquiera cierto conocimiento general del modelo como de su funcionamiento interno.

En principio, se desarrollará el encoder del modelo **Transformers** (Vaswani et al. 2017) elegido por ser paralelizable y por su versatilidad en el desempeño de diferentes campos del procesamiento del lenguaje. A partir de este primer modelo el desempeño irá evolucionando en diferentes variantes, ya sea incorporando *pretraining* **BERT** (Devlin et al. 2019, Cañete et al. 2020), hasta modificando el módulo de *self-attention* para procesar textos largos con mejor rendimiento **Longformer** (Beltagy, Peters, and Cohan 2020 y Romero 2022).

2.1 Modelo Transformer

El modelo **Transformer** (Vaswani et al. 2017) fué pensado originalmente para transformar una secuencia de texto a la entrada del modelo, generar una representación intermedia (vector/matriz) y devolver otra secuencia de texto. Específicamente el

modelo **Transformer** intenta resolver la familia de problemas denominadas **seq2seq** (Sutskever, Vinyals, and Le 2014), entre ellos participa la traducción de textos en diferentes idiomas **Machine Translation** (Bahdanau, Cho, and Bengio 2014a), la generación de resúmenes de texto **Text Summarization** (Allahyari et al. 2017), El responder a preguntas **Question Answering** (Z. Wang 2022) o incluso el conversar con el sistema **Chatbots** (Sojasingarayar 2020) que puede ser visto como la generación de una respuesta acorde al pasado de la conversación y la reciente sentencia de texto.

2.1.1 Codificación de la entrada

La primera dificultar al diseñar sistemas que interactúen con texto es la incompatibilidad del lenguaje simbólico de los humanos con el numérico de las computadoras. Para que el texto pueda ser procesado el primer paso será definir las unidades simbólicas (pueden ser palabras, pueden ser letras, grupos de letras o palabras, etc.) y asignar una representación numérica a cada símbolo.

En el caso más sencillo podemos definir el conjunto ordenado *vocabulario* $\mathbf{V} = \{"era", "la" ... "casa"\}$, para luego **representar una oración** con una secuencia de índices llamados símbolos o **Tokens**. $"La\ casa\ era\ nueva" \mapsto \{2, 4, 1, 3\}$

Otra manera de asignar una representación numérica a cada símbolo puede ser con un vector. Por ejemplo, la representación **One-Hot encoding** asigna un vector que contiene un 1 en la dimensión correspondiente al índice del símbolo en el vocabulario y cero en el resto.

$$ENC_{one-hot}[La] = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix} \qquad ENC_{one-hot}[Casa] = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}$$

En Tomas Mikolov et al. 2013 se proponen otras maneras de representar un símbolo con su vector, en el caso citado significados similares se mapearán a coordenadas similares.

2.1.2 Arquitectura

El modelo **Transformer** ver **Figura 2.2** se compone de un encoder y un decoder. La función del encoder es mapear una secuencia de símbolos de entrada hacia una representación numérica. Esta representación llegará al decoder convirtiendola en otra secuencia de símbolos a la salida.

Se verá más adelante en **BERT** que el encoder puede ser reutilizado para diferentes tareas, mientras que el decoder es intercambiable para la tarea específica a realizar.

En el modelo *Transformer* se propuso un decoder diseñado para la generación de texto. Debido a que la tarea que se busca resolver es **Text Classification** en lugar de generar una secuencia de texto, omitiremos todo análisis del *decoder*. Sin embargo, aquellos que quieran profundizar en la arquitectura del *decoder* podrán remitirse a Vaswani et al. 2017.

> Obs. En la Fig. 2.2, no confundir la entrada del decoder que recibe la representación (salida del encoder) de su segunda entrada (salida del positional Encoding) utilizada para una decodificación recursiva.

Input Embedding

El primer componente a analizar del modelo *Transformer* Fig. 2.2 es el denominado **Input Embedding**. Consiste en definir una matriz $E \in \mathcal{R}^{hidden\_size \times |V|}$ que dado un índice de vocabulario, devuelve una columna de dimensión $hidden\_size$. En el mismo proceso de aprendizaje, la red irá definiendo la representación de cada símbolo, tal como se recuerda en la sección 2.1.1.

Debido a que la entrada está compuesta por una secuencia de símbolos (texto), la salida de este componente será una secuencia de *vectores de significado* de dimensión $hidden\_size$ cada uno.

Positional Encodings

Para codificar la posición de cada símbolo en la secuencia, se propuso a cada símbolo ya vectorizado sumarle una representación de posición. Esta representación es fija (no se aprende) y vale:

$$\begin{aligned} PE_{pos,2i} &= sin(pos/10000^{(2i)/hidden\_size}) \\ PE_{pos,2i+1} &= cos(pos/10000^{(2i)/hidden\_size}) \end{aligned} \tag{2.1}$$

Esta codificación, para cada posición pos altera conjuntamente a todas las coordenadas i. En alusión a la representación binaria, cada posición se codifica en una representación acotada entre $[-1; 1]$ en lugar $\{0, 1\}$. De esta manera es que podemos representar la posición de cada vector en la secuencia sin alterar demasiado la magnitud del vector original.

Self-attention

El bloque de atención, es un mecanismo de correlación entre una secuencia con otra **attention** (Bahdanau, Cho, and Bengio 2014b), que para el caso particular **self-attention** consiste en correlacionar la secuencia consigo misma.

Existen diferentes mecanismos de atención/correlación que explotan diferentes aspectos de rendimiento (por ejemplo, la atención aditiva correlaciona bajo un coste computacional lineal), la variante utilizada en nuestro caso será la denominada **dot attention**, que utiliza el producto interno para cruzar las señales.

A partir de una secuencia de vectores $X = \{x_1 ... x_n\}$, cada uno será proyectado con 3 matrices M_k, M_q, M_v a aprender en el entrenamiento.

$$K = M_k \cdot X$$
$$Q = M_q \cdot X$$
$$V = M_v \cdot X$$

Nomenclatura. En el artículo original se representa a la secuencia de vectores x_i con una matriz, concatenandos a lo largo de las filas.

$$X = \begin{bmatrix} \cdots & x_1 & \cdots \\ \cdots & x_2 & \cdots \\ & \vdots & \\ \cdots & x_n & \cdots \end{bmatrix} \in \mathbf{R}^{seq\_len \times hidden\_size}$$

A continuación, se calcula el **dot atención** para cada vector en la secuencia.

$$Attention(K, Q, V) = softmax(\frac{Q \cdot K^T}{\sqrt{hidden\_size}}) \cdot V$$

El término $\sqrt{hidden\_size}$ se utiliza para contrarrestar que el producto interno sea demasiado grande, llevando a la softmax a una región de gradientes evanescentes (Vaswani et al. 2017).

El producto interno $Q \cdot K^T = M_q\ X \cdot X^T\ M_k^T$ genera una matriz que multiplica todos los vectores x_i con todos los x_j. Al pasar por la softmax, esta tiene el efecto de que **1.** todas las filas suman la unidad y **2.** una suerte de selección del producto interno mas alto.

Finalmente la multiplicación con $\mathbf{V} = [v_1 ... v_n]$ consigue que cada valor v_i esté ponderado por la *softmax*. Si se utilizase la función max en su lugar, conseguiría que cada vector v_i se reordene (o incluso repita) en la secuencia. La atención entonces es un mecanismo que selecciona distintas posiciones v_i en la secuencia.

Notar que los nombres de las matrices provienen de Key, Query, Value en analogía a la estructura de datos ”diccionario”. Donde el diccionario es una tabla de Values direccionadas por el Key, luego el usuario podrá generar consultas Query sobre el redireccionamiento Key y recuperar el Value.

Multi-Head Self-attention

Multi-head Attention **Fig. 2.1** es un resultado empírico que consiste en repetir el mecanismo de atención h veces, pero aplicado en diferentes sectores de la representación. Si cada vector en la secuencia tenía dimensión $hidden\_size$, cada mecanismo de atención *head* recibirá vectores de dimensión $\frac{hidden\_size}{h}$. Aplicada la atención se concatenan los vectores *Value* de cada *Head* y se aplica una capa lineal.

Según Vaswani et al. 2017, la atención Multi-Head permite aplicar diferentes atenciones a diferentes subespacios, con ello conseguir separar la representación en diferentes factores. En **Disentangled Representation Learning** (X. Wang et al. 2022) se estudian otros mecanismos para separar determinada información en factores.

Para la oración ”La niña y su perro miraba al niño”.
Un factor podría atender al género donde se espera que las representaciones vectoriales entre ”perro” y niño” sean similares. Y otro factor podría atender al tipo de mamífero, esperando similitud entre ”niña” y ”niño” pero no con ”perro”.

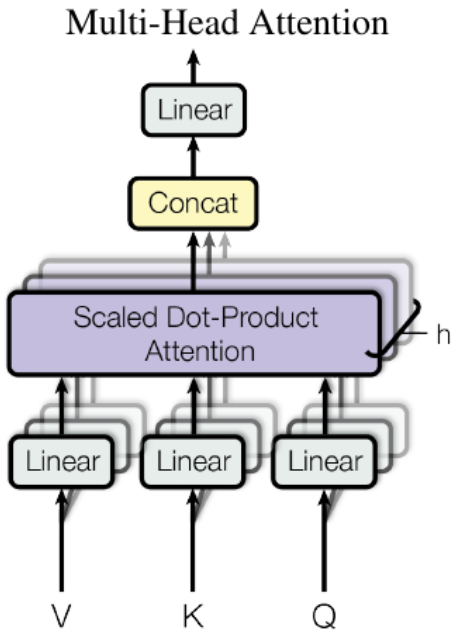


Figure 2.1: Cada vector de entrada será dividido en h *heads* y se aplicará h copias del mecanismo de atención (análogos, pero aprenden diferentes matrices) finalmente se concatenan los vectores Value para reconstruir la dimensión original $hidden\_size$.

Bloque FeedForward

Este bloque corresponde a la típica red densa de dos capas con activación ReLU, descripta por:

$$\begin{aligned} \sigma(x) &= ReLU(x) = max(0, x) \\ FFN_1(x) &= \sigma(xW_1 + b_1) \\ FFN_2(x) &= xW_2 + b_2 \\ FFN &= FFN_2(FFN_1(x)) \end{aligned}$$

Este componente es responsable de introducir las suficientes alinealidades en el modelo para que sea posible transformar la información capa a capa.

Layer Normalization

En el (Cap. 8.7.1 Goodfellow, Bengio, and Courville 2016) se estudia la posibilidad de reducir el tiempo de entrenamiento de las redes profundas, mediante la reparametrización de media y varianza.

Veremos a continuación la variante **Layer Normalization** (Ba, Kiros, and Hinton 2016) que consiste en calcular, para una muestra, la media y varianza entre todos los vectores de la secuencia, luego reparametrizar estos estadísticos con media y varianza globales del proceso de entrenamiento.

Algoritmo: Layer Normalization

1. Sea $\mathbf{X} = \{x_1 \cdots x_n\}$ una secuencia de vectores con $x_i \in \mathbb{R}^{hidden\_size}$
2. Estimar la media y varianza $\overline{\mathbf{X}}$ y $\sigma^2[\mathbf{X}]$
3. Definir parámetros $\begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix}$ a obtener en el entrenamiento por *backpropagation*.
4. Devolver la secuencia $O = \{o_1 \cdots o_n\}$ con

$$o_i = \frac{x_i - \overline{X}}{\sigma(X)} \cdot \theta_2 + \theta_1$$

Este algoritmo es una variante de **Batch Normalization** (Ioffe and Szegedy 2015), donde $batch\_size$ muestras se procesan juntas y los momentos se calculan entre vectores a igual posición en la secuencia (índice). Por el contrario aquí los momentos se calculan a lo largo de cada secuencia de muestra. De esta manera, ya no importará que secuencias en un batch tengan diferentes longitudes.

Shortcut-connections + Stacked Network

Si volvemos a la **Fig. 2.2** observaremos unos bloques que se repiten N veces, tanto para el encoder como para el decoder. Estos bloques junto a la flecha que conecta la entrada de cada submódulo con su salida, son piezas heredadas del modelo **ResNet** (He et al. 2015).

En dicho artículo se estudia la posibilidad de apilar muchas capas *layers* de redes neuronales densas mediante la adición de *shortcut connexions*, que consisten en sumar la entrada a la salida de cada bloque.

Algoritmo: Shortcut-Connections

1. Sea $\mathbf{X} = \{x_1 \cdots x_n\}$ una secuencia de vectores de entrada.
2. Sea $FFN(\cdot)$ la salida de un bloque neuronal.
3. La salida del bloque será $\mathbf{O} = \mathbf{X} + FFN(\mathbf{X})$.

Estas conexiones resuelven el problema del gradiente evanesciente (Pascanu, Tomás Mikolov, and Bengio 2012) permitiendo el entrenamiento de grandes cantidades de capas apiladas.

La intuición detrás de su funcionamiento es que en la medida que se recorre la señal de error desde la salida hacia la entrada, mientras la parte neuronal FFN va desvaneciendo su gradiente, la otra componente genera un atajo que le da paso a la señal de backpropagation.

Sea por ejemplo una red de 3 capas, la señal feedforward será:

$$
\begin{aligned}
O_1 &= X + FFN_1(X) \\
O_2 &= O_1 + FFN_2(O_1) \\
O_3 &= O_2 + FFN_3(O_2)
\end{aligned}
$$

Que si reemplazamos paso a paso, obtendremos

1. $O_3 = O_2 + FFN_3(O_2)$
2. $O_3 = O_1 + FFN_2(O_1) + FFN_3(O_2)$
3. $O_3 = X + FFN_1(X) + FFN_2(O_1) + FFN_3(O_2)$

El gradiente se obtiene según.

$$
\nabla_\theta O_3 = \underbrace{\nabla_\theta X}_{=0} + \nabla_\theta FFN_1(X) + \nabla_\theta FFN_2(O_1) + \nabla_\theta FFN_3(O_2)
$$

Luego por simplicidad de análisis, supondremos que el gradiente de cada FFN respecto a los parámetros que no son de su propia etapa, se desvanecen. Obteniendo.

$$
\begin{aligned}
\nabla_\theta FFN_1(X) &\approx \nabla_{\theta_1} FFN_1(X) \\
\nabla_\theta FFN_2(O_1) &\approx \nabla_{\theta_2} FFN_2(O_1) \\
\nabla_\theta FFN_3(O_2) &\approx \nabla_{\theta_3} FFN_3(O_2)
\end{aligned}
$$

Que es análogo a entrenar cada bloque de forma independiente.

En el artículo original, se advierte que si la dimensión de la entrada no coincidiese con la salida, es posible modificar el modelo de la siguiente manera:

$$O = W \cdot X + FFN(X)$$

con $W \in \mathbb{R}^{dim(O) \times dim(X)}$

Dropout

Para mejorar la capacidad de generalización de una red con muchos parámetros, se propuso el método **Dropout** (Srivastava et al. 2014) como regularizador a la salida de todas las alinealidades. Para determinada capa de activación y en la etapa de entrenamiento, el método muestrea desde una multinomial con probabilidad $p_{dropout}$ una máscara binaria que dejará o no pasar la activación a la siguiente etapa.

Algoritmo: Entrenamiento con Dropout

1. Sea la secuencia de entrada **X** $= \{x_1 \cdots x_n\}$ con $x_i \in \mathbb{R}^{hidden\_size}$
2. Defina $p_{dropout} \in [0, 1]$
3. Muestree **M** $= Multinomial(p_{dropout})$ con $M \in \{0, 1\}^{hidden\_size \times n}$
4. Devuelve **O** $=$ **X** $\odot$ **M** con $\odot$ la multiplicación elemento a elemento.

En el artículo citado, se ha mostrado que este método hace más robusta a la red frente a variaciones aleatorias al encontrar los parámetros internos. El artículo concluye que como regularizador supera a los regularizadores de norma l_1 y l_2 en un amplio abanico de tareas (claisifación de objetos, reconocimiento de dígitos, reconocimiento del habla, clasificación de documentos y análisis de datos biológicos).

Un resultado interesante de este artículo es que la red encuentra parámetros de bajo nivel de activación *sparse* (respecto a sin dropout), esto consigue que cada neurona deba ser útil en una gama de datos evitando la alta especialización.

Una interpretación ampliamente aceptada de este resultado (Cap. 7.12 Goodfellow, Bengio, and Courville 2016) es que por cada batch, se entrenan diferentes modelos (variantes del modelo original con menos neuronas) y en tiempo de inferencia, todos estos modelos se combinan.

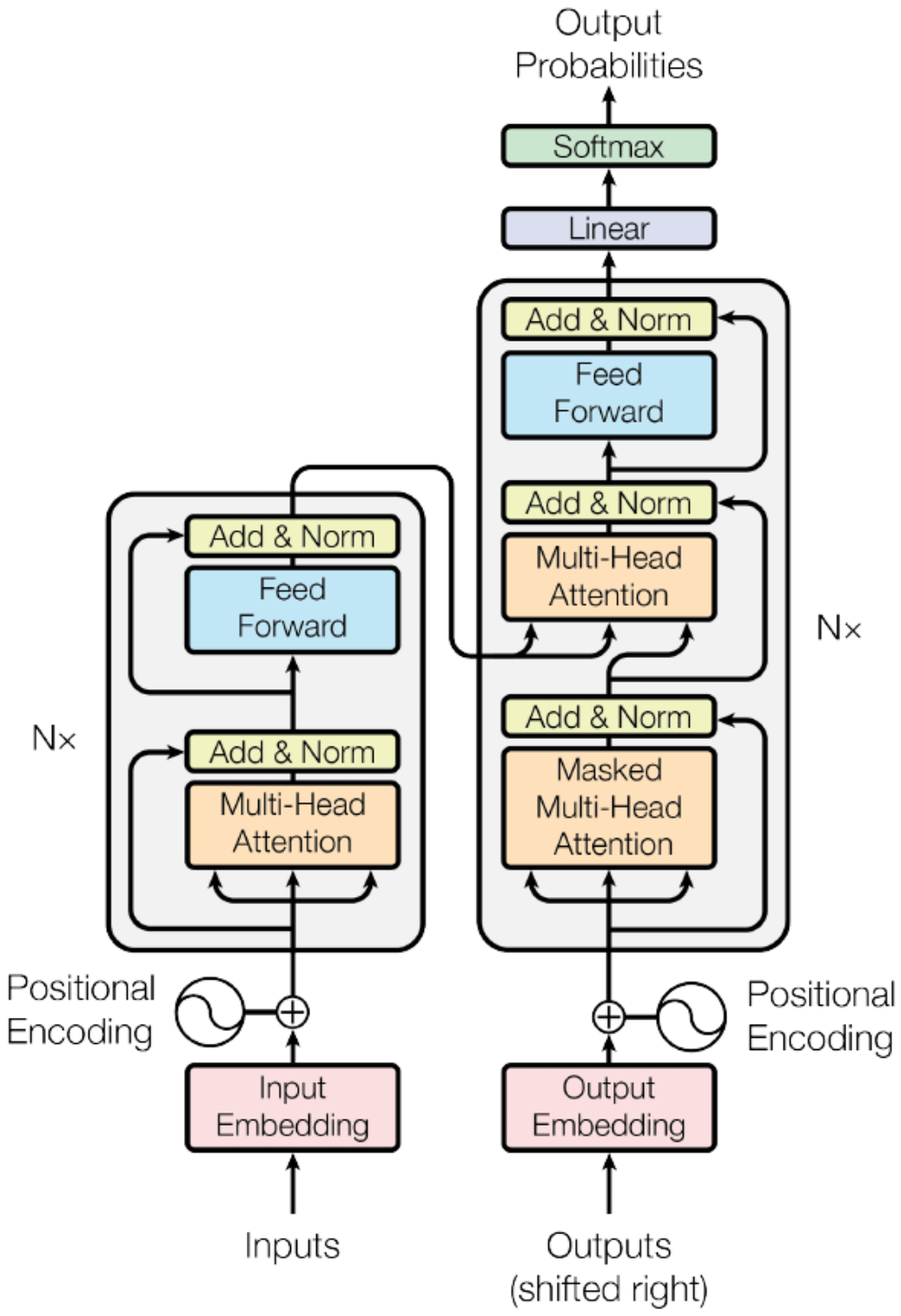


Figure 2.2: Ilustración del Modelo **Transformer** (Vaswani et al. 2017) donde a la izquierda se aprecia la arquitectura del encoder mientras que a la derecha la del decoder. La representación numérica que genera el encoder se visualiza por la flecha que conecta al último bloque del encoder y entra a cada bloque *Multi-Head Attention* del decoder.

2.2 Modelo BERT

El segundo modelo a considerar es el modelo **BERT** (Devlin et al. 2019), este modelo introduce tres mejoras significativas respecto al Transformer estudiado en la **sección 2.1**. Por un lado se modifica el submódulo *input embedding*, se separa la base del encoder de su capa de salida (para resolver múltiples tareas) y finalmente se introduce el preentrenamiento.

2.2.1 Input Embedding

Adelantándonos a la sección siguiente 2.2.2, **BERT** será capaz de clasificar texto *Text classification*, como de responder preguntas *Question Answering*. Cada tarea requiere un procesamiento especial de la entrada y la salida, la arquitectura que encontró Devlin et al. 2019 para salvar las inconsistencias entre tareas, es la de en principio definir el símbolo **[CLS]** ubicado en el principio de las oraciones y definir el símbolo **[SEP]** ubicado en el final.

Se encontró útil definir el símbolo **[CLS]** al comienzo de cada secuencia, con el fin de reservar la primera representación vectorial para lograr realizar clasificaciones globales de texto (no confundir con las clasificaciones símbolo a símbolo).

> El Dataset de *Question Answering* está conformado con tripletes (R, Q, A), Referencia, Pregunta y Respuesta. Mientras que en *Text Classification* el Dataset está compuesto por tuplas (T, C) Texto y Clasificación.

Por otro lado, para resolver las inconsistencias de la entrada, se especifica el símbolo **[SEP]** para separar un tipo de entrada de otro. En el caso de *Question Answering*, la **R**eferencia de texto estará separada de la **P**regunta **"[CLS] R [SEP] Q [SEP]"**; Mientras que el **T**exto en *Text classification* la entrada queda conformada con **"[CLS] T [SEP]"**.

Token embeddings

El modelo **BERT** por defecto utiliza **WordPiece Tokenizer** (Wu et al. 2016) para transformar un texto en una secuencia de símbolos (**tokens**). Los símbolos que genera este algoritmo se denominan **subwords** (por ejemplo la terminación ##ing), esta suerte de compresión consigue reducir el tamaño del vocabulario a aproximadamente 30000 palabras en inglés.

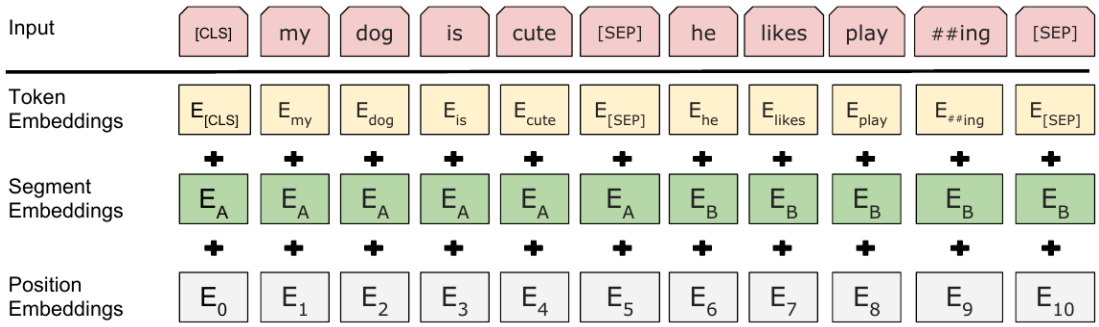


Figure 2.3: A cada muestra de entrada se le añade el token **[CLS]** al inicio de la oración y el token **[SEP]** al final. Los diferentes Embeddings utilizados en BERT son 1. Token Embeddings para mapear cada símbolo con su vector. 2. Segmentation Embedding para diferenciar los tipos de oraciones. 3. Positional Embeddings para codificar el órden de la secuencia

Además de representar el significado del símbolo con un vector, el modelo BERT encuentra útil codificar de forma aditiva la posición del símbolo en la secuencia **Figura 2.3**. El vector que representa la posición **Positional Embeddings** se construye asignandole un vector a cada número natural (en la práctica se define una longitud de secuencia máxima max_length), estos vectores agrupados como matriz, se inicializan aleatoriamente y sus valores se van definiendo en el mismo entrenamiento. A diferencia del modelo **Transformer** Vaswani et al. 2017 la codificación de la posición se realizaba de forma determinística en 2.1. Adicionalmente, para informar a la red sobre el tipo de secuencia introducida (ya sea el documento de referencia o la pregunta sobre el texto en *Question Answering*) se define un nuevo vector **Segment Embeddings** que efectivamente codificará si el símbolo proviene de un tipo de entrada o del otro (en caso de haber más de 1 tipo).

2.2.2 Multi-task learning

Cómo ya se ha adelantado en la introducción, el modelo **BERT** es una generalización del encoder **Transformer** que permite realizar una mayor variedad de tareas definiendo distintas etapas de salida o **Decoders**.

En la sección siguiente 2.2.3, se estudiará la tarea **Masked Language Model (MLM)** que consiste en predecir la palabra faltante en una oración. Para resolver esta tarea, se define como *capa de salida* un clasificador que por cada vector en la secuencia de salida, determine un símbolo de vocabulario. Conocida la palabra correcta y la predicha, es posible definir un criterio de error a minimizar.

A partir de los parámetros obtenidos en **MLM**, el modelo se reentrena bajo la tarea **Next Sentence Prediction (NSP)** que consiste en determinar si dos oraciones son consecutivas en el mismo párrafo o no. Esta tarea utiliza el primer vector reservado por el símbolo **[CLS]** para realizar una clasificación binaria. Gracias a este segundo entrenamiento, se consiguen mejores resultados en tareas que requieran encontrar relaciones entre diferentes oraciones, como es el caso en *Question Answering*.

Finalmente, para resolver tareas más específicas, se conecta a la representación generada por el encoder (secuencia de vectores dispuestos en una matriz) un decoder que se ajuste a las necesidades del problema. Al nunca haber sido entrenado

el *Decoder* el modelo completo se reentrena para la tarea final, este proceso se denomina **finetuning** y se verá en la sección a venir.

> Por ejemplo si en vez de clasificar se requiriese generar una secuencia de texto, luego podría considerarse utilizar el Decoder del modelo Transformer.

En el Cap. 7.7 y 7.9 Goodfellow, Bengio, and Courville 2016 se introduce al lector en *Multi-task learning* y *Parameter Sharing* ofreciendo un poco más de intuición sobre estos procesos tan utilizados en el proceso de entrenamiento.

2.2.3 Pretraining & Fine-Tuning

El mecanismo de preentrenamiento es el pilar que permite al modelo **BERT** alcanzar el *state of art* en al menos 11 tareas de procesamiento del lenguaje (Devlin et al. 2019). El mecanismo consiste en entrenar el modelo en dos etapas. ***Pretraining*** donde se entrena un grán corpus de texto para una tarea generalizada. ***Finetuning*** donde se continúa el entrenamiento pero con un corpus específico al problema a resolver.

> Notar que la capa de salida se suele entrenar desde cero para el *finetuning*.

En el artículo citado, el preentrenamiento intenta resolver primero la tarea de **Masked Language Model** (Mask LM) que consiste en copiar la secuencia de entrada en la salida excepto en los lugares de entrada con el símbolo **[MASK]** donde a la salida, el modelo deberá inferir la palabra más probable.

El éxito de preentrenar con **Mask LM** recae en la sencillez de obtener grandes datasets utilizando **Self-Supervised Training**. La estrategia es recuperar grandes volúmenes de documentos y aleatoriamente reemplazar una palabra por el símbolo **[MASK]**, la palabra reemplazada será el objetivo **tgt** del entrenamiento y el docuemnto con dichas máscaras será la entrada **src**.

> Para ilustrar el concepto de *self-supervised training*, considere:
>
> 1. Sea una muestra de un gran corpus:
>   ”El hombre fué a la tienda a comprar un litro de leche”
> 2. Se muestrea aleatoriamente la palabra ”comprar” y ”de”
> 3. La nueva muestra supervisada será:
>   **SRC**: ”El hombre fué a la tienda a [MASK] un litro [MASK] leche”
>   **TGT**: ”comprar”, ”de”
>
> De esta manera, es muy fácil transformar grándes volúmenes de texto en muestras supervisadas para la tarea **Masked LM**. Por no requerir supervisión humana, el proceso es económico.

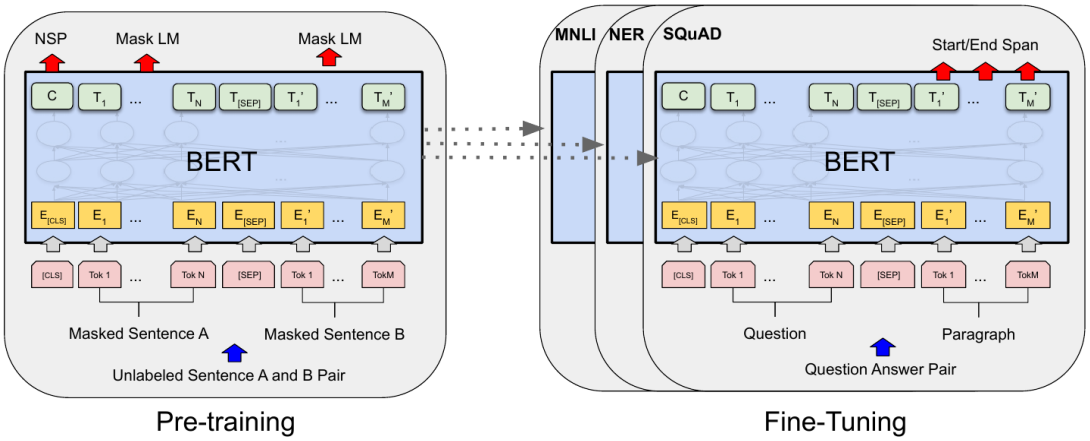


Figure 2.4: En **BERT**, el **Pretraining** se realiza en dos partes, primero bajo la tarea de *Masked Languange Model* y luego bajo la tarea de *Next Sequence Prediction*. Notar que la primera utiliza todos los vectores de salida menos el primero, y la segunda tarea utiliza únicamente el primer vector. Luego, las tareas particulares se pueden resolver mediante **Fine-Tuning** tales como *Question Answering*, *Named Entity Recognition*, *Multi-Genre Natural Language Inference* (Clasificar si una oración contradice, fortalece o es neutra frente a otra) o muchas más

Ya completado el entrenamiento **(MLM)**, el preentrenamiento continúa bajo la tarea **Next Sentence Prediction (NSP)** que como ya se mencionó en la sección anterior, consiste en predecir si dos oraciones se dijeron una luego de la otra. Del mismo modo que sucedía con **MLM** es fácil obtener un dataset automatizado que contenga pares de oraciones con su secuencialidad anotada.

El preentrenamiento está recomendado al trabajar con datos escasos, pues generar datos sintéticos tiene la virtud de cubrir una parte del lenguaje no explorada por los datos insuficientes del problema original. Otra característica del método es la de ahorrar cómputo en el *finetuning* ya que se espera, que luego del *pretraining*, que los parámetros esten cerca de sus coordenadas finales. Comparado con *pretraining*, *finetuning* es relativamente económico (menos datos, menos épocas de entrenamiento).

La intuición detrás de este mecanismo es la de encontrar primero una representación que generalice bien a diferentes tareas, y luego adaptar al modelo para que resuelva una tarea en particular. No todo *pretraining* consigue una mejora en el modelo final. Para que la mejora ocurra, tanto las tareas de *pretraining* como las de *finetuning* deberán guardar alguna relación en común. Es por ello que en lugar de utilizar el modelo **BERT** originalmente entrenado en inglés, se utilizará el modelo **BETO** cuyo preentrenamiento en español coincide con el idioma de nuestro caso de estudio.

2.2.4 Modelo BETO

Habiendo comprendido los modelos anteriores, es momento de introducir el modelo **BETO** (Cañete et al. 2020). **BETO** es una versión revisada del modelo **BERT** entrenada en español. Está disponible en el repositorio **Hugging Face** y viene preentrenada en dos versiones **uncased** y **cased**, es decir para textos de entrada codificados en minúsculas o para textos enriquecidos con mayúsculas. Debido a que los datos del presente proyecto están en minúscula, es de esperar que la versión **uncased** ofrezca mejores resultados.

El modelo se entrenó utilizando todo el corpus de **Wikipedia** y todas las fuentes del **OPUS Project** (Tiedemann 2012) (diarios de las naciones unidas, charlas TED, Subtítulos, Noticias, y más). Una diferencia respecto a **BERT**, no tan estructural, es el uso de **Dynamic Masking (DM)** (introducido en **RoBERTa** Liu et al. 2019) donde las posiciones de las máscaras (MLM) se generan en tiempo de entrenamiento.

2.3 Modelo Longformer

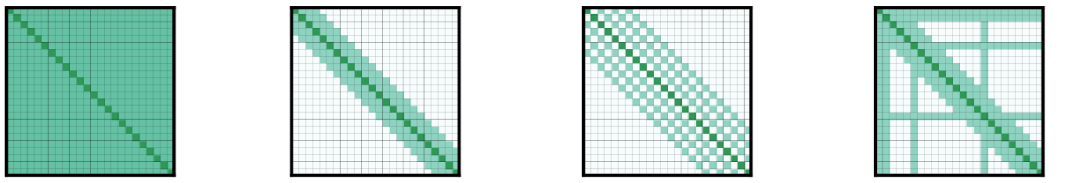
En esta sección presentaremos el modelo **Longformer** (Beltagy, Peters, and Cohan 2020), una variante a **BERT** pero diseñada para procesar textos largos. Gracias a este modelo será posible entrenar y evaluar discursos más largos. Aprovechando que su versión en español **Longformer Narrativa** (Romero 2022) ya está disponible.

En la sección 2.1.2 se definió el mecanismo de atención, este componente escala al menos cuadráticamente $\mathcal{O}(seq\_len^2)$ con la longitud de la secuencia tanto en tiempo como en memoria. En este modelo se propone un mecanismo de atención que escala linearmente con la longitud de sequencia, permitiendo procesar documentos más largos o hacer un uso más eficiente de la memoria ram. El modelo longformer resuelve la limitación de **BERT** de procesar hasta 512 símbolos.

2.3.1 Mejoras en el mecanismo de Atención

El mecanismo de atención combina una ventana de atención local junto a un mecanismo de atención global dependiente de la tarea. La intuición sobre el mecanismo local, es la de construir representaciones que contemplan posiciones vecinas, mientras que el mecanismo global permite construir representaciones de toda la secuencia (texto) de entrada.

Para lograr una complejidad lineal $\mathcal{O}(seq\_len)$, el modelo longformer propone que el mecanismo de atención sea un patrón fijo, entre pares de vectores de entrada.



(a) Full n^2 attention (b) Sliding window attention (c) Dilated sliding window (d) Global+sliding window

Figure 2.5: Comparación de los diferentes patrones de atención utilizados en el modelo **longformer**

Sliding Window

El primer patrón a considerar, son las ventanas deslizantes o **Sliding Window**. Se define una ventana fija de atención de tamaño w centrado en cada posición **Fig. 2.5 (b)**, con el fin de capturar las relaciones locales respecto a cada posición. Utilizando un apilamiento de estos mecanismos, será posible aumentar el **campo receptivo** (Ch.9 Goodfellow, Bengio, and Courville 2016) de cada vector hacia posiciones más y más lejanas. El campo receptivo de la capa l, es $l \times w$. La complejidad algorítmica de este patrón es $\mathcal{O}(seq\_len \times w)$.

Dilated Sliding Window

Para incrementar el campo receptivo aún más (por ejemplo para documentos muy largos), el patrón de atención puede dilatarse **Fig. 2.5 (c)**. Esto es, una ventana con posiciones distanciadas en d unidades. El campo receptivo ahora será $l \times d \times w$. En el mismo artículo, se ha encontrado empíricamente que asignar diferentes d dilataciones a cada cabezal head del *Multi-Head Attention* resulta benefisioso, permitiendo que cada *head* capture representaciones a diferentes lejanías.

Global Attention

Recordando el uso asignado al símbolo **[CLS]** en *BERT*, que reservaba un vector para que pudiese ser correlacionado con toda la secuencia e ir construyendo una representación global que pudiese ser utilizado en la clasificación. Para no perder esta capacidad de construir representaciones globales, se define un conjunto de posiciones fijas que puedan correlacionarse con cualquier otra posición de la secuencia.

Longformer Attention

En la **Fig. 2.5 (d)** se combinan todos los efectos antes descritos para obtener la cascada de representaciones locales sin tener que sacrificar el mecanismo de correlación global.

2.3.2 Longformer Narrativa

Finalmente, se desea que el modelo haya sido preentrenado en español. Para ello, se propone utilizar **Longformer Narrativa** (Romero 2022) entrenado en base a **RoBERTalex** (Gutiérrez-Fandiño et al. 2021a) que a su vez es el modelo **RoBERTa** (Liu et al. 2019) entrenado en español bajo la estrategia de entrenamiento descrita en el artículo original **Longformer** (Beltagy, Peters, and Cohan 2020).

Los datos de preentrenamiento provienen del dataset **Spanish Legal Domain Corpora (SLDC)** (Gutiérrez-Fandiño et al. 2021b) que contiene diferentes textos del dominio legal Español.

En la tabla 2.1 se observan los parámetros de dilatación por capa especificados en el modelo longformer.

Layer	dilatación	window_size
(Layer 0-5)	d=0	512
(Layer 6-7)	d=1 (en dos heads)	512
(Layer 8-9)	d=2 (en dos heads)	512
(Layer 10-11)	d=3 (en dos heads)	512

Tabla 2.1: Parámetros de ventana por capa, para el modelo longformer

2.4 Resumen

En la siguiente tabla 2.2 se comparan los hiperparámetros entre los modelos ya introducidos.

Hiperparámetros	Transformer (Base)	BERT (BETO)	Longformer (Narrativa)
Modelado			
activation_unit	ReLU	GeLU	GeLU
hidden_size	512	768	768
nLayers	6	12	12
nHeads	8	12	12
Regularización			
pDropout	0.1	0.1	0.1
label_smoothing	0.1	0	0
L2	0	0.01	0.01
Entrenamiento			
Optimizador	Adam	Adam	Adam
training_steps	100k	2M	+500K
lr	custom	1E-4	custom
Datasets			
Origen	WMT En-De	Wikipedia OPUS	SLDC
Tokenización			
max_tokens	10k	512	4096
Algorithm	WordPiece	SentencePiece	BPE
vocab_size	37k	31k	52k
Resumen			
num_params	65M	110M	129M

Tabla 2.2: Comparación de los hiperparámetros de los modelos.

Referencias

Capítulo 2

Allahyari, Mehdi et al. (2017). “Text Summarization Techniques: A Brief Survey”. In: *CoRR* abs/1707.02268. arXiv: 1707.02268. URL: http://arxiv.org/abs/1707.02268.

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton (2016). “Layer Normalization”. In: DOI: 10.48550/ARXIV.1607.06450. URL: https://arxiv.org/abs/1607.06450.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2014a). “Neural Machine Translation by Jointly Learning to Align and Translate”. In: DOI: 10.48550/ARXIV.1409.0473. URL: https://arxiv.org/abs/1409.0473.

— (2014b). “Neural Machine Translation by Jointly Learning to Align and Translate”. In: DOI: 10.48550/ARXIV.1409.0473. URL: https://arxiv.org/abs/1409.0473.

Beltagy, Iz, Matthew E. Peters, and Arman Cohan (2020). “Longformer: The Long-Document Transformer”. In: DOI: 10.48550/ARXIV.2004.05150. URL: https://arxiv.org/abs/2004.05150.

Cañete, José et al. (2020). “Spanish Pre-Trained BERT Model and Evaluation Data”. In: *PML4DC at ICLR 2020*. URL: https://users.dcc.uchile.cl/~jperez/papers/pml4dc2020.pdf.

Devlin, Jacob et al. (2019). “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: arXiv: 1810.04805 [cs.CL].

Goodfellow, Ian J., Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. Cambridge, MA, USA: MIT Press.

Gutiérrez-Fandiño, Asier et al. (2021a). *Spanish Legalese Language Model and Corpora*. arXiv: 2110.12201 [cs.CL]. URL: https://huggingface.co/PlanTL-GOB-ES/RoBERTalex?.

— (2021b). *Spanish Legalese Language Model and Corpora*. arXiv: 2110.12201 [cs.CL].

He, Kaiming et al. (2015). "Deep Residual Learning for Image Recognition". In: DOI: 10.48550/ARXIV.1512.03385. URL: https://arxiv.org/abs/1512.03385.

Ioffe, Sergey and Christian Szegedy (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: DOI: 10.48550/ARXIV.1502.03167. URL: https://arxiv.org/abs/1502.03167.

Liu, Yinhan et al. (2019). "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: DOI: 10.48550/ARXIV.1907.11692. URL: https://arxiv.org/abs/1907.11692.

Mikolov, Tomas et al. (2013). "Efficient Estimation of Word Representations in Vector Space". In: arXiv: 1301.3781 [cs.CL].

Pascanu, Razvan, Tomás Mikolov, and Yoshua Bengio (2012). "Understanding the exploding gradient problem". In: *CoRR* abs/1211.5063. arXiv: 1211.5063. URL: http://arxiv.org/abs/1211.5063.

Romero, Manuel (2022). *Legal Spanish LongFormer by Narrativa*. URL: https://huggingface.co/Narrativa/legal-longformer-base-4096-spanish.

Sojasingarayar, Abonia (2020). "Seq2Seq AI Chatbot with Attention Mechanism". In: *CoRR* abs/2006.02767. arXiv: 2006.02767. URL: https://arxiv.org/abs/2006.02767.

Srivastava, Nitish et al. (2014). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *J. Mach. Learn. Res.* 15.1, pp. 1929–1958.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le (2014). "Sequence to Sequence Learning with Neural Networks". In: DOI: 10.48550/ARXIV.1409.3215. URL: https://arxiv.org/abs/1409.3215.

Tiedemann, Jörg (May 2012). "Parallel Data, Tools and Interfaces in OPUS". In: *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*. Istanbul, Turkey: European Language Resources Association (ELRA), pp. 2214–2218. URL: http://www.lrec-conf.org/proceedings/lrec2012/pdf/463_Paper.pdf.

Vaswani, Ashish et al. (2017). "Attention Is All You Need". In: *CoRR* abs/1706.03762. arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

Wang, Xin et al. (2022). "Disentangled Representation Learning". In: DOI: 10.48550/ARXIV.2211.11695. URL: https://arxiv.org/abs/2211.11695.

Wang, Zhen (2022). "Modern Question Answering Datasets and Benchmarks: A Survey". In: DOI: 10.48550/ARXIV.2206.15030. URL: https://arxiv.org/abs/2206.15030.

Wu, Yonghui et al. (2016). "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation". In: *CoRR* abs/1609.08144. arXiv: 1609.08144. URL: http://arxiv.org/abs/1609.08144.

Experimentos

Capítulo 3

En esta sección se realizarán diferentes experimentos apareados, con el propósito de encontrar **modelos** progresivamente más certeros. Se decribirá el proceso de entrenamiento de cada experimento junto a sus criterios de evaluación.

3.1 Criterios de entrenamiento

En la presente sección se describirán los diferentes criterios de entrenamiento, entendiendo como entrenamiento a la búsqueda o ajuste de los parámetros de los modelos descritos en el Capítulo 2 con el fin de minimizar el error de generalización.

3.1.1 Especificaciones de entrenamiento

El mecansimo de entrenamiento utilizado es **Error Backpropagation** (Rumelhart, Hinton, and Williams 1986), específicamente, se utilizará **autograd** de **Pytorch 1.13.1** (Paszke et al. 2017).

En cuanto al Hardware, se utilizará el Notebook de **Google Colab** * que cuenta para uso académico una **GPU NVIDIA Tesla T4** de arquitectura Turing con 16GB RAM y 2560 núcleos †.

3.1.2 Capa de salida & Criterio de Error

La predicción de trastornos neurodegenerativos a través de discursos generados por el paciente, puede ser visto, como la tarea de **clasificación de texto** en dos categorías *Se detecta determinado trastorno* o *No se detecta*.

Hemos visto que el modelo preentrenado (Encoder), devuelve una representación, cuyo primer vector (símbolo [CLS]) servía para realizar clasificaciones globales del texto. Por ello se le añadirá al *Encoder* su **Output Layer** (capa de salida), que consiste en definir una matriz en tiempo de entrenamiento de dimensiones $hidden\_size \times output\_size$ que proyecte linealmente la representación del primer vector hacia la dimensión de salida ($output\_size = 2$).

Por otro lado, se definirá el criterio de error como la minimización de la entropía cruzada entre el diagnóstico predicho y el diagnóstico real.

$$\begin{aligned} \theta^* = \underset{\theta}{argmin} J(\theta) &= \underset{\theta}{argmin} H(y^{True},\ \hat{y}_\theta) \\ &= \underset{cte>0}{H(y^{True})} + \underset{\theta}{argmin} KL(\ y^{True}\ ||\ \hat{y}_\theta\ ) \\ &= \underset{\theta}{argmin} \sum_{i=0}^{1} P(Y^{True}=i)\ ln \frac{P(Y^{True}=i)}{P(\hat{Y}_\theta = i)} \end{aligned} \tag{3.1}$$

Con KL la divergencia de Kullback Leibler. La intuición detrás de este criterio es la de encontrar una distribución para nuestro modelo $\hat{y}(\theta)$ que punto a punto sea similar a la distribución verdadera y^{True}. Cuando ambas distribuciones coinciden ($\hat{y} = y^{True}$), la divergencia se hace cero y se alcanza el mínimo de la solución.

3.1.3 Criterio de finalización del entrenamiento

Habrá dos criterios de finalización del algoritmo de entrenamiento.

1. Cuando el error haya disminuído ϵ veces desde el error inicial. $loss[0]/loss[end] < \epsilon$.
2. Cuando el entrenamiento alcance max_epochs iteraciones

Estos criterios ayudan a que diferentes simulaciones **Bootstrap** (Sec. 3.2.1) alcancen errores de entrenamiento similares. Se recuerda que la etapa **Finetuning** no debe ser muy entrenada (unas pocas épocas) para no destruir la información del preentrenamiento (Zhang et al. 2021, Flender 2022).

*Google Colab: `https://colab.research.google.com/`
†Especificaciones de la GPU: `https://www.nvidia.com/es-la/data-center/tesla-t4/`

3.1.4 Regularización

Debido a la existencia de discursos cortos y discursos largos, se hipotetiza que el modelo deberá poder realizar mejores predicciones cuando reciba más información de discursos largos. Es por ello que se incorpora a la función de coste un regularizador que dependa de la longitud del discurso de entrada.

$$loss^* = loss \cdot len\_seq$$

La intución de este regularizador es la de poder castigar el error de entrenamiento en discursos largos, de este modo, cuanto más largo sea un discurso más se castigan los errores cometidos.

3.1.5 Sesgo por país en pacientes AD

Debido a que la muestra está compuesta por proporciones diferentes de Alzheimer en pacientes Colombianos que de Chilenos, tal como se vió en la sección 1.3.5) **Tabla 3.1**, el modelo podría descubrir que sólamente clasificando por nacionalidad adquiriría una tasa de error (Accuracy) de 0.57 (Si se clasificase a todo chileno con AD y a todo colombianos con CTR).

Para subsanar este problema, se seleccionaron tanto para el dataset de entrenamiento como para el de validación, **igual cantidad de individuos de ambas nacionalidades**. La consecuencia de eliminar este sesgo resultó una caída considerable del rendimiento de todos los modelos (de 42 muestras disponibles quedaron 24). De esta manera, si el modelo supiese de qué país proviene cada individuo, esta información no le ayudaría para predecir un diagnóstico.

Por otro lado, para el ensayo **FTDbv** vs **CTR** el sesgo por país asciende a un Accuracy del 78%. Es por ello que para este ensayo se descartan los voluntarios chilenos. (El ensayo FTDbv vs CTR consistirá en 9 muestras FTDbv colombianas vs 9 muestras CTR también colombianas.

Pero no todo son malas noticias, en la **sección 3.2.1** al muestrear el *dataset*, será posible ir rotando estas muestras sobrantes, para, de cierta manera, aprovecharse en el mecanismo de evaluación.

	Chile	Colombia
AD	15	6
CTR	12	9
FTDbv	0	21

Tabla 3.1: Proporciones de diagnósticos por países

3.1.6 Dataset Augmentation

Entre las técnicas que más destacan en el tratamiento de datos escasos para *Deep Learning* resalta **Dataset Augmentation** (Cap. 7.4 Goodfellow, Bengio, and Courville 2016) como mecanismo de regularización en la fase de entrenamiento. El método consiste en incorporar datos generados sintéticamente al conjunto de entrenamiento.

Dataset Augmentation es una familia de técnicas que vienen utilizandose masivamente en procesamiento de imágenes (Perez and Wang 2017) como añadir versiones desplazadas, rotadas, con ruido de las imágenes de entrenamiento.

Estas técnicas llevadas al procesamiento del lenguaje consisten en proponer transformaciones al texto original, obteniendo variantes del mismo. Entre las técnicas más simples se incluyen, reordenar palabras, insertar/quitar palabras aleatoriamente, reemplazar palabras por sus sinónimos, etc. De todas estas técnicas empíricamente probadas, la que mejores resultados dió fué **Backtranslation** (Sennrich, Haddow, and Birch 2015), la idea de traducir automáticamente el texto a un idioma diferente y retraducirlo al idioma original. Este proceso se llevó a cabo con **Google translate** para diferentes idiomas.

> Obs. Dataset Augmentation será aplicado únicamente sobre el conjunto de entrenamiento. No se desea ni que se filtre información a los datos de validación ni que la evaluación del modelo esté sesgada por muestras no-independientes.

3.1.7 Especificaciones de entrenamiento

El entrenamiento del modelo consistirá en ajustar (*Finetune*) el modelo base.

Apareamiento entre modelos

Debido a que se compararán varios modelos de forma simultánea, se definirá una semilla (**seed**) para generar particiones *Train* y *Test* idénticas para cada modelo en cada simulación **Bootstrap**. Es decir, cada modelo observa los mismos datos de entrenamiento y evalúa en orden las muestras de validación.

3.2 Criterios de evaluación

El objetivo del presente apartado, será la de medir el grado de validez de los modelos propuestos.

3.2.1 Simulaciones Bootstrap

La escases de datos, además de dificultar el proceso de entrenamiento, dificultará la tarea de evaluar los modelos ante situaciones generales. Si bien la mejor opción sería adquirir más datos, el método **Bootstrap** (Efron and Tibshirani 1986) nos aporta una alternativa, basada en simular $sampling\_size$ experimentos (pueden repetirse) donde a partir de $|\mathcal{D}_{dataset}|$ muestras del *dataset*, $|\mathcal{D}_{train}|$ pertenecerán a la partición *"Train"* y las restantes $|\mathcal{D}_{val}|$ a *"Validation"*. No habrá filtraciones entre grupos, pero en cada simulación *bootstrap* el destino de cada muestra a cada grupo será desconocido.

> Cada simulación **Bootstrap** será única, debido a la aleatoriedad del dropout, las inicializaciones de la capa de salida, cómo se agrupan los datos en el *batch*, la selección y secuencialidad aleatoria de los datos.

Las métricas obtenidas con este mecanismo tendrán un mayor grado de credibilidad, ya que al repetir la misma medición durante varias simulaciones con datos muestreados aleatoriamente, será posible estimar la variabilidad de cada métrica.

3.2.2 Test de hipótesis entre modelos

Para poder verificar si efectivamente un sistema modela mejor los datos que otro, se propone realizar un test de hipótesis apareado para la tasa de éxito (Accuracy). Será aplicado el **test de Wilcoxon** (Ch.10 Wild 2010) que es una versión más estricta (al no asumir normalidad) que su equivalente test de student **ttest**. El test se realizará a una cola ($Acc_1 - Acc_2 < 0$) por naturaleza del problema (a mayor Acc_2 mejor). Se define como nivel de singificancia $\alpha_w = 0.05$.

3.2.3 Especificación de error sobre el diagnóstico

Finalmente, se desea determinar si un paciente es diagnosticado correctamente o no. Para ello se proponen las siguientes hipótesis.

H0 El paciente padece de **AD**/**FTDbv**

H1 El paciente es **CTR**.

El **Error de tipo 1** (el más severo) consiste en decirle al paciente que está sano cuando no lo está. Por ello se especifica que la probabilidad de **predecir CTR** erroneamente debe ser menor al nivel de significancia $\alpha = 0.05$.

$$P(H1|H0) \leq \alpha$$

Si bien esta especificación se podría parecer a un test de hipótesis, no lo será, el motivo como más adelante se verá, es que se elige el umbral **ROC** para alcanzar dicho nivel de significancia. Hay que tener cuidado, que para ser un test de hipótesis se deben proporcionar datos nuevos y el p-value se debe descubrir, contrario a obtenerlo por búsqueda de hiperparámetros.

3.2.4 Métricas adicionales

Para continuar evaluando el grado de verdad de nuestros modelos se propone adicionalmente, medir las siguientes variables sobre el dataset de validación.

Matriz de Confusión

La matriz de confusión está compuesta por 4 casos, **True Positive** cuando se detecta correctamente la patología, **True Negative** cuando se detecta correctamente la ausencia de patología, **False Positive** (Error tipo II) cuando se predice erroneamente que la patología está presente, **False Negative** (Error tipo I) cuando no se detecta la patología presente.

Estos parámetros serán acumulados a lo largo de las 400 simulaciones **Bootstrap** para brindar estimaciones más precisas. A partir de esta matriz se podrá deducir las métricas a continuación.

Accuracy

El Accuracy (o tasa de éxito) indicará la cantidad de pacientes bien diagnosticados respecto a los totales. Esta métrica ayuda a conocer la cantidad esperada de pacientes bien diagnosticados

$$Accuracy = \frac{num.\ bien\ diagnosticados}{Diagnosticos\ totales} = 1 - Tasa\ de\ error$$

En caso de una predicción aleatoria, se espera medir $Accuracy = 0.5$ y un clasificador perfecto espera encontrar un $accuracy = 1.0$.

Cota inferior del Accuracy

Para asignarle una mayor confiabilidad al modelo, se desea definir una cota de confianza de significancia $\alpha = 0.05$ para el Accuracy. Esto es:

$$CI\ tal\ que\ P(Accuracy \leq CI) \leq \alpha$$

Es decir, que sólo en una raredad del 5% nuestro modelo tenga un $Accuracy$ tan bajo como CI.

Receiver Operating Characteristic ROC

Recordando que la salida del modelo devuelve una número entre 0 y 1 y por defecto uno creería que si el número supera el umbral de 0.5 se habrá decidido por la clase 1. Pero ¿Qué hubiese pasado si el modelo hubiese predicho 0.0001 ó 0.4999?, pues predecirá por la clase 0, sin embargo no es lo mismo, la predicción fuerte en el primer caso, que en el segundo donde está muy cerca de la incertidumbre.

La curva del **ROC** grafica la potencia del test (ó 1-Error tipo 2) frente al Error tipo 1 para diferentes umbrales de clasificación y es fundamental para comparar el rendimiento entre modelos. En la Fig. 3.1 se visualiza idealmente cómo va variando la curvatura de los modelos cuando estos varían desde clasificadores perfectos hasta clasificadores puramente azarosos.

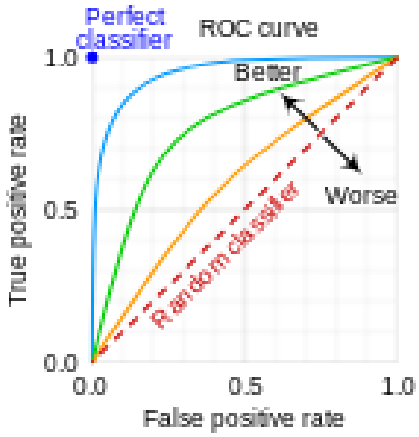


Figure 3.1: En la figura se compara el rendimiento de diferentes modelos. Source: `https://upload.wikimedia.org/wikipedia/commons/1/13/Roc_curve.svg`

Obtenida esta curva, será posible establecer la relación de compromiso entre potencia de un test contra el error de tipo 1.

Area Under the Curve AUC

En la práctica, será posible encontrar modelos que actúen mejor en una región de la curva **ROC** y otros que actúen mejor en otra. Un criterio para determinar qué modelo es mejor que otro consiste en medir el área bajo la curva **AUC** de la curva ROC. Para modelos azarosos, se obtendrá $AUC = 0.5$ y para el clasificador perfecto se obtendrá $AUC = 1.0$.

3.3 Experimentos

3.3.1 Especificación de los experimentos

Se realizarán 4 experimentos para pacientes con **AD vs CTR** para probar bajo las mismas condiciones de entrenamiento (mismo dataset, mismo orden) si efectivamente hubo una mejoría de modelación. Finalmente se realizará un experimento más para **FTDbv vs CTR** utilizando el mejor modelo encontrado para **AD**, de esta manera podremos evaluar el modelo final con datos nunca observados.

1. **Distribución Nula** Consiste en implementar el modelo BERT pero intercambiando aleatoriamente los diagnósticos.
2. **Modelo BERT sin entrenamiento** Consiste en entrenar al modelo **BERT** sin haber sido preentrenado.
3. **Modelo BETO** Consiste en entrenar el modelo **BERT** pero preentrenado en español según **BETO**.
4. **Modelo Longformer Narrativa** Consiste en utilizar el modelo **Longformer** pero preentrenado en español por **Narrativa**.

Los Hiperparámetros de entrenamiento se resumen en la tabla 3.2.

Dataset		
$\|\mathcal{D}_{CTR}\|$	21	
$\|\mathcal{D}_{AD}\|$	21	
$\|\mathcal{D}_{FTDbv}\|$	21	
clean_dataset	True	(Con limpieza de datos)
Simulaciones		
sampling_method	Bootstrap	(Muestreo)
num_simulations	400	
force_equal_country	True	(Num. Países por Batch)
force_equal_tgt	True	(Igual proporción de trastornos)
Batching		
train_val_ratio	75%-25%	
$\|\mathcal{D}_{train}\|$	16	(Balanceado por país)
$\|\mathcal{D}_{val}\|$	8	(Balanceado por país)
Modelo A: Distribución Nula		
pDropout	0.1	
batch_size	8	
lr	1E-5	
shuffling	True	(Mezclar el orden de los datos)
max_epochs	30	
epsilon_convergence	0.05	$(loss[0]/loss[end] < \epsilon)$
truncation	200	(Se toman los primeros tokens)
Modelo B: BERT untrained (sin preentrenamiento)		
pDropout	0.1	
batch_size	8	
lr	1E-5	
shuffling	True	
max_epochs	30	
epsilon_convergence	0.05	$(loss[0]/loss[end] < \epsilon)$
truncation	200	
Modelo C: BETO		
pDropout	0.1	
batch_size	8	
lr	1E-5	
shuffling	True	
max_epochs	30	
epsilon_convergence	0.05	$(loss[0]/loss[end] < \epsilon)$
truncation	200	
Modelo D: Longformer Narrativa		
pDropout	0.1	
batch_size	8	
lr	1E-5	
shuffling	True	
max_epochs	30	
epsilon_convergence	0.05	$(loss[0]/loss[end] < \epsilon)$
truncation	300	
Modelo E: Longformer Narrativa + Dataset Augmentation		
pDropout	0.1	
batch_size	32	
lr	1E-5	
shuffling	True	
max_epochs	30	
epsilon_convergence	0.05	$(loss[0]/loss[end] < \epsilon)$
truncation	300	
dataset_augmentation	En-De-Ch	(retraducido 3 veces)

Tabla 3.2: Hiperparámetros de los experimentos

3.3.2 Resultados de los modelos

Los modelos aquí descritos fueron evaluados bajo el mismo conjunto de entrenamiento y evaluados bajo el mismo conjunto de validación. Se realizaron 400 simulaciones **bootstrap** para tanto los experimentos **AD vs CTR** como para **FTDbv vs CTR**

Comparación de modelos

Los resultados de comparar los modelos A-E para el dataset **AD vs CTR** se ilustran en la figuras 3.2, 3.3 y 3.4, Mientras que en la tabla 3.3 se hace un resumen de las métricas obtenidas.

En la Fig. 3.2 **curva ROC** se ilustra la tasa de rechazar correctamente el trastorno frente a las veces que se rechaza de forma incorrecta para diferentes umbrales de detección.

El área bajo la curva presenta una métrica muy importante a tener en cuenta para medir la calidad del clasificador, se aprecia que a medida el modelo mejora su área bajo la curva también lo hace, siendo 0.5 para la **distribución nula** mientras que el modelo de **longformer con dataset augmentation** alcanza un área bajo la curva de 0.89.

En la curva de **Accuracy vs Error tipo 1**, se observa la relación de compromiso entre Tasa de Éxito (Accuracy) y Error tipo 1, sin violar el requerimiento del error tipo 1 $P(DETECTAR\,CTR|ERA\;AD) < \alpha$. El máximo accuracy lo consigue el modelo Longformer Narrativa que bajo un umbral de detección de 0.9, alcanzando un Accuracy medio de 0.738.

En la Fig. 3.3 se grafican las diferentes matrices de confusión acumuladas en las 400 simulaciones, se observa que en la distribución nula no hay correlación clara entre lo que se intenta detectar y su verdadero valor, sin embargo a medida que mejora el modelo, los errores de detección tanto por falsos negativos/positivos van disminuyendo, consiguiendo correlacionar el diagnóstico real contra el diagnóstico predicho.

Finalmente en la Fig. 3.4 se estiman la distribuciones de las tasas de éxito (Accuracy) tanto con histograma (no paramétrico) como parametrizando con una distribución **Beta** para un umbral de detección ROC de 0.5. Se puede observar que a medida mejoramos el modelo, la masa de probabilidad va acercandose a la unidad, este efecto se puede medir con el tercer momento centrado de la distribución llamado oblicuidad (skew). Por otro lado viendo la distribución acumulada, es posible encontrar la cota inferior de confianza $\alpha = 0.05$ para el *Accuracy*. Es decir, con probabilidad 0.95 podemos decir que el Accuracy superará su cota inferior.

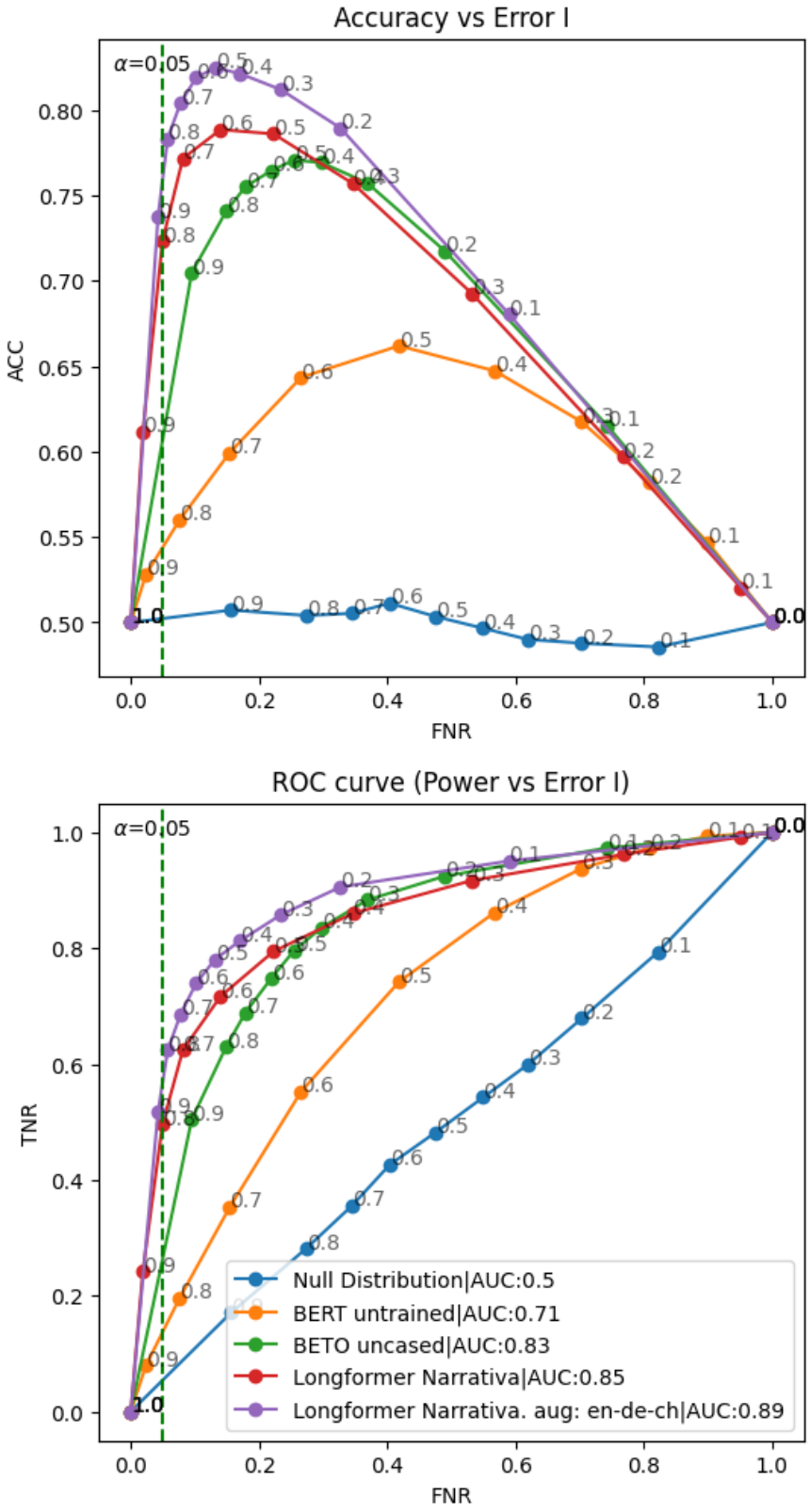


Figure 3.2: Accuracy & ROC obtenidas al realizar 400 simulaciones bootstrap apareadas, aprovechando que el eje horizontal representa el error de tipo 1, se grafica en líneas punteadas el nivel de significancia.

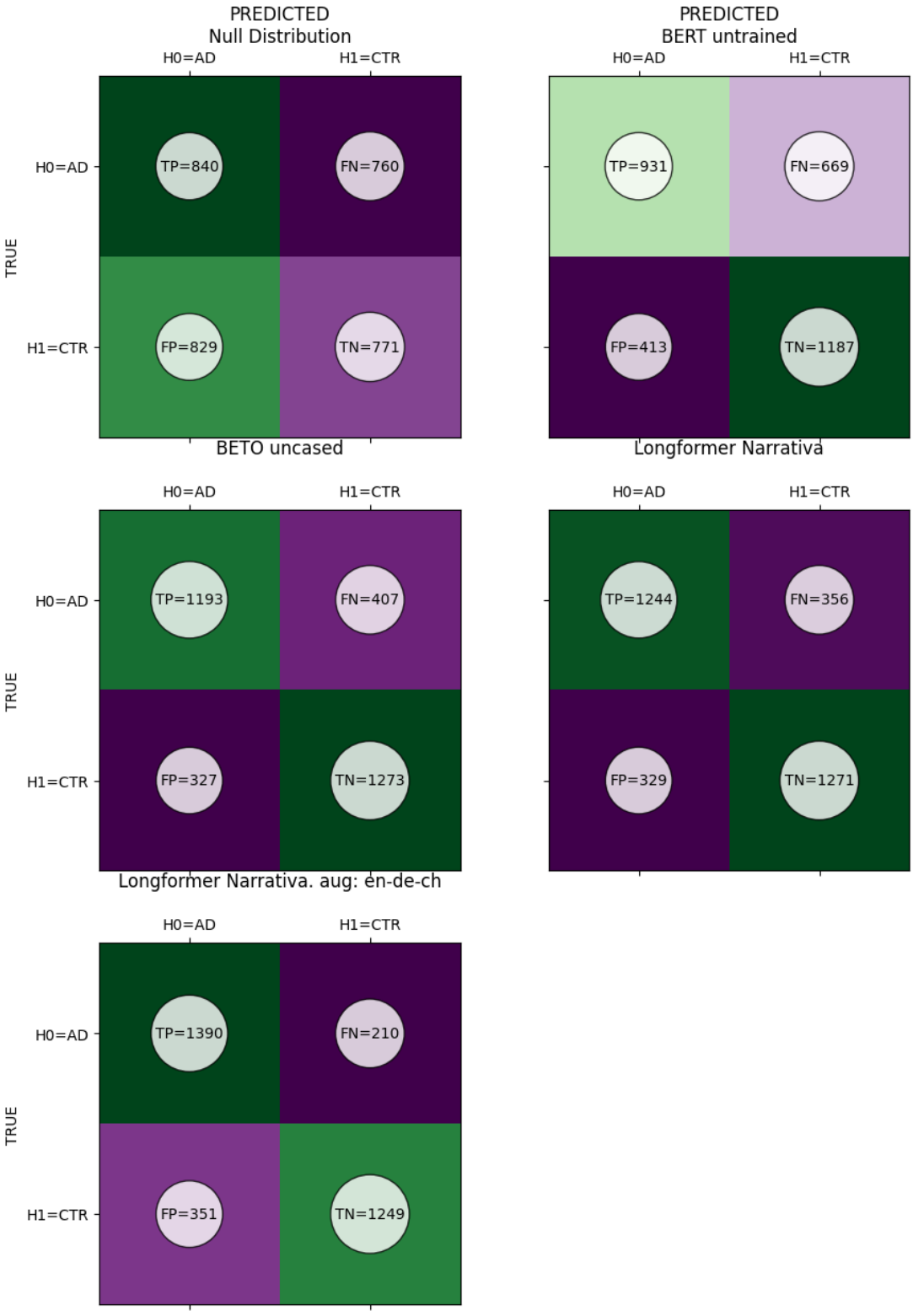
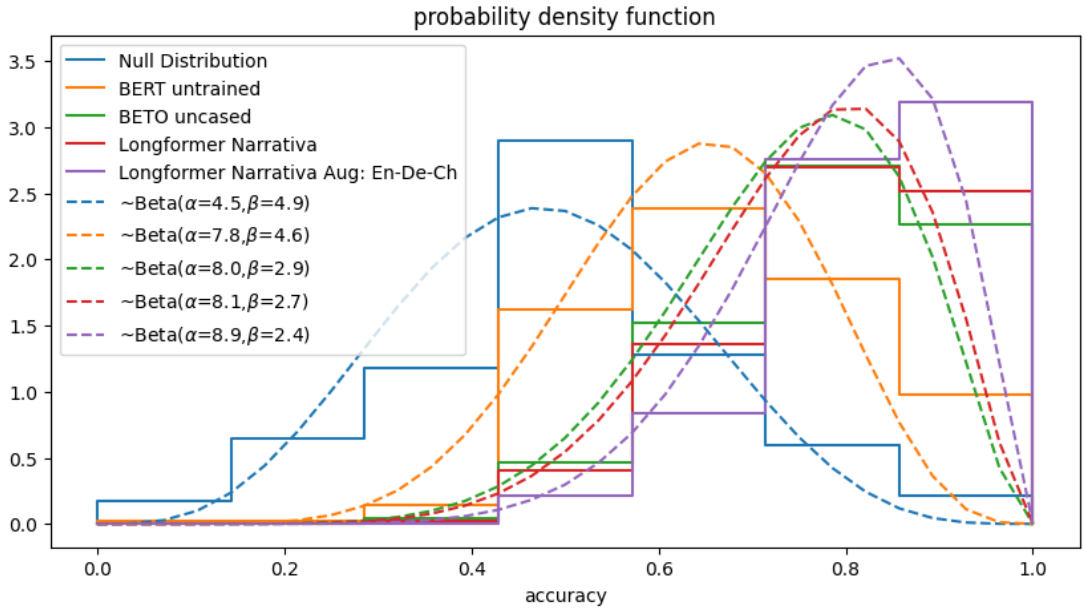


Figure 3.3: Matriz de Confusión obtenida al realizar 400 simulaciones bootstrap apareadas

(A) Función de probabilidad del *Accuracy* para umbral ROC de 0.5



(B) Función de distribución acumulada del *Accuracy*

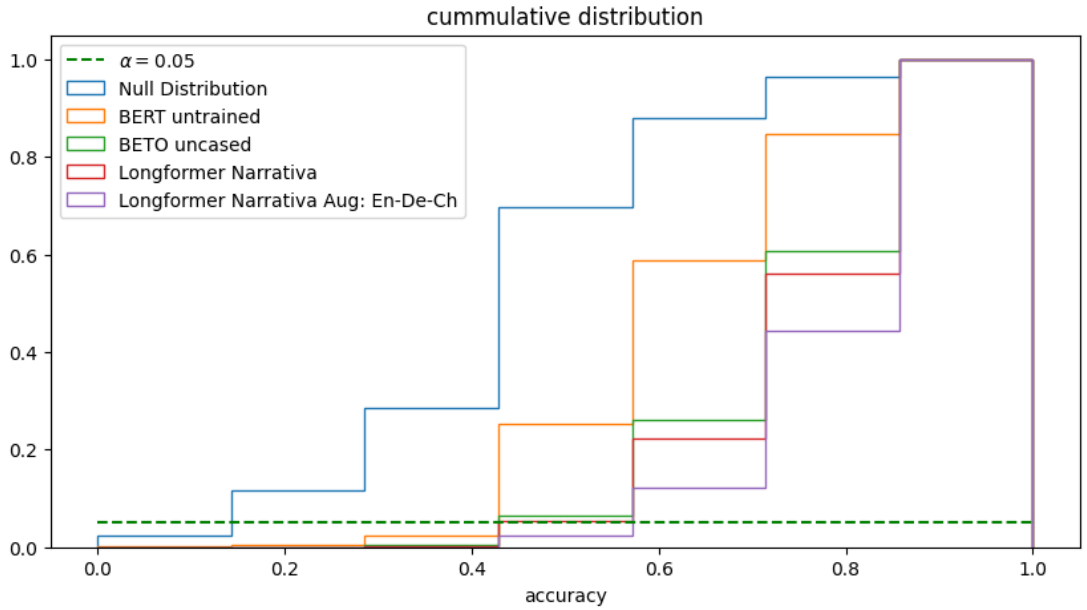


Figure 3.4: Se obtuvo en **(A)** La densidad de probabilidad del *Accuracy* y en **(B)** La distribución acumulada las del *Accuracy*. Al realizar 400 simulaciones apareadas entre la **distribución nula** en azul, el modelo **BERT sin preentrenamiento** en naranja, el modelo **BETO** en verde, el modelo **Longformer Narrativa** en rojo y este mismo último modelo pero entrenado con **Dataset Augmentation** en violeta.

Resumen A Distribución nula		
threshold	0.5	Umbral ROC
alpha	0.05	significancia
Error I	**0.475**	P(CTR|AD)=760/(760+840)
Error II	0.518	P(AD|CTR)=829/(829+771)
AUC	**0.497**	
Cota Inferior(accuracy)	0.143	P(Acc<0.143)=0.023
mean(accuracy)	**0.479**	(Media)
std(accuracy)	0.155	(Desvío)
skew(accuracy)	0.046	(Oblicuidad)
kurt(accuracy)	-0.479	(Curtósis)
Resumen B (BERT untrained)		
threshold	0.5	Umbral ROC
alpha	0.05	significancia
Error I	**0.418**	P(CTR|AD)=669/(669+931)
Error II	0.258	P(AD|CTR)=413/(413+1187)
AUC	**0.715**	
Cota Inferior(accuracy)	0.429	P(Acc<0.429)=0.025
mean(accuracy)	**0.63**	(Media)
std(accuracy)	0.132	(Desvío)
skew(accuracy)	-0.275	(Oblicuidad)
kurt(accuracy)	-0.284	(Curtósis)
Resumen C (BETO)		
threshold	0.5	Umbral ROC
alpha	0.05	significancia
Error I	**0.254**	P(CTR|AD)=407/(407+1193)
Error II	0.204	P(AD|CTR)=327/(327+1273)
AUC	**0.833**	
Cota Inferior(accuracy)	0.429	P(Acc<0.429)=0.005
mean(accuracy)	**0.734**	(Media)
std(accuracy)	0.128	(Desvío)
skew(accuracy)	-0.565	(Oblicuidad)
kurt(accuracy)	0.0148	(Curtósis)
Resumen D (Longformer Narrativa)		
threshold	0.5	Umbral ROC
alpha	0.05	significancia
Error I	**0.223**	P(CTR|AD)=356/(356+1244)
Error II	0.206	P(AD|CTR)=329/(329+1271)
AUC	**0.853**	
Cota Inferior(accuracy)	0.429	P(Acc<0.429)=0.0025
mean(accuracy)	**0.749**	(Media)
std(accuracy)	0.126	(Desvío)
skew(accuracy)	-0.615	(Oblicuidad)
kurt(accuracy)	0.0912	(Curtósis)
Resumen E (Longformer Narrativa + Dataset Augmentation)		
threshold	0.5	Umbral ROC
alpha	0.05	significancia
Error I	**0.131**	P(CTR|AD)=210/(210+1390)
Error II	0.219	P(AD|CTR)=351/(351+1249)
AUC	**0.886**	
Cota Inferior(accuracy)	0.571	P(Acc<0.571)=0.025
mean(accuracy)	**0.785**	(Media)
std(accuracy)	0.117	(Desvío)
skew(accuracy)	-0.731	(Oblicuidad)
kurt(accuracy)	0.329	(Curtósis)

Tabla 3.3: Resultados de los diferentes experimentos para el dataset **AD vs CTR**

A continuación, para garantizar que un modelo es efectivamente mejor que otro, se realizan los siguientes test de signo sobre la diferencia en *Accuracy* apareados. Tabla 3.4. De esta manera se concluye que el Modelo *Longformer Narrativa con Dataset Augmentation* predice con menor error que el modelo *Longformer Narrativa* que a su vez es mejor que el modelo *BETO* mejor que *BERT sin preentrenamiento* y todos ellos son mejores prediciendo que la *Distribución Nula*

Test de Hipótesis (apareado)			
	Test de Wilcoxon	t-test	Result ($\alpha = 0.05$)
Modelo B vs Modelo A	$p\_val = 2.85E-34$	$p\_val = 1.17E-40$	pasa
Modelo C vs Modelo B	$p\_val = 1.98E-26$	$p\_val = 5.66E-31$	pasa
Modelo D vs Modelo C	$p\_val = 0.044$	$p\_val = 0.032$	pasa
Modelo E vs Modelo D	$p\_val = 4.35E-8$	$p\_val = 3.45E-8$	pasa

Tabla 3.4: **Test de hipótesis apareado**, siendo el **Modelo A** la *Distribución Nula*, el **Modelo B** el modelo *BERT sin preentrenamiento*, el **Modelo C** el modelo *BETO*, el **Modelo D** el modelo *Longformer Narrativa* y el **Modelo E** el modelo *Longformer Narrativa con Dataset Augmentation*

Resultados finales

A continuación se repite el **Modelo E: Longformer Narrativa + Dataset Augmentation** tanto para el ensayo **AD vs CTR** como para **FTDbv vs CTR**, si bien se podría argumentar que el primer ensayo pudo sobreajustar los datos por la intensiva búsqueda de hiperparámetros, el segundo experimento observa los datos de entrenamiento y evaluación por vez primera, sirviendo como mecanismo de validación.

En la Fig. 3.5 se grafica la curva ROC, para el **modelo E: Longformer Narrativa + Dataset Augmentation** tanto para el dataset **AD** como para el **FTDbv**. Se puede observar que ambos modelos entrenados con diferentes datos tienen rendimientos muy similares.

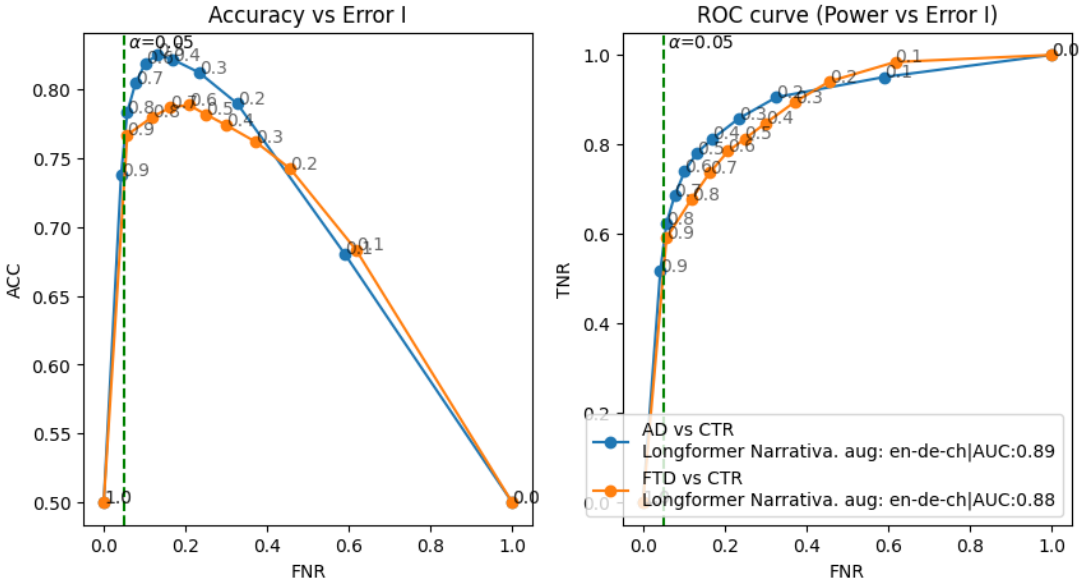


Figure 3.5: Curva ROC final para **AD vs CTR** y **FTDbv vs CTR** utilizando el **MODELO E: Longformer Narrativa + Dataset Augmentation**

Para cumplir la especificación de error tipo 1, se eligieron los umbrales de deteccion de $th_{AD} = 0.84$ y $th_{FTDbv} = 0.92$, para que el modelo decida que un paciente está sano, deberá superar este umbral con probabilidad mayor a 0.84. Las nuevas matrices de confusión para 400 simulaciones Bootstrap, se representan en la Fig. 3.6, mientras que las distribuciones de *Accuracy* en la Fig. 3.7. En la tabla 3.5 se resumen las métricas finales.

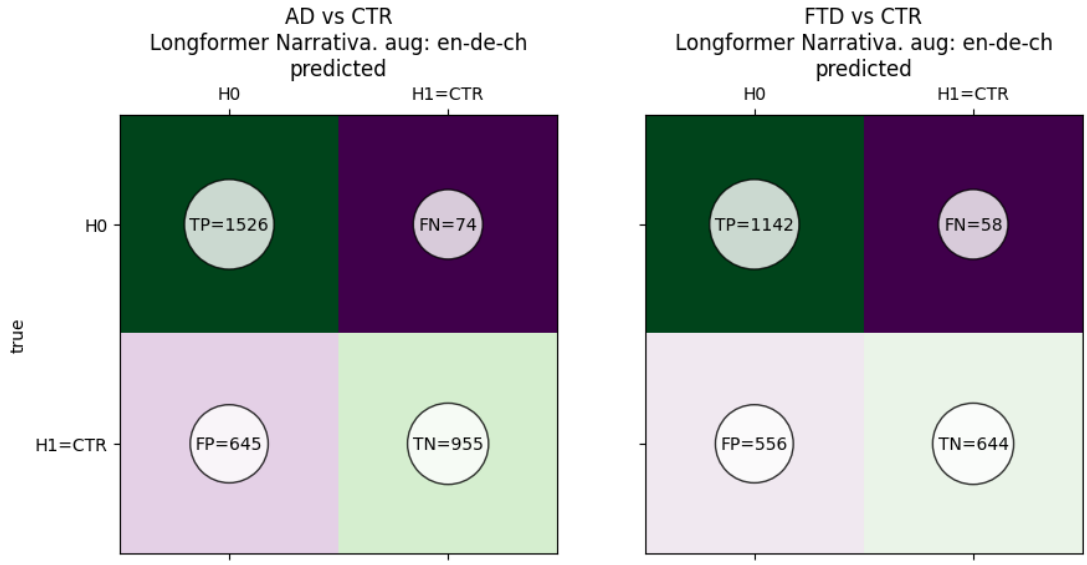


Figure 3.6: Matriz de Confusión final para **AD vs CTR** y **FTDbv vs CTR** utilizando el **MODELO E: Longformer Narrativa + Dataset Augmentation**, utilizando un umbral de detección $th = 0.84$

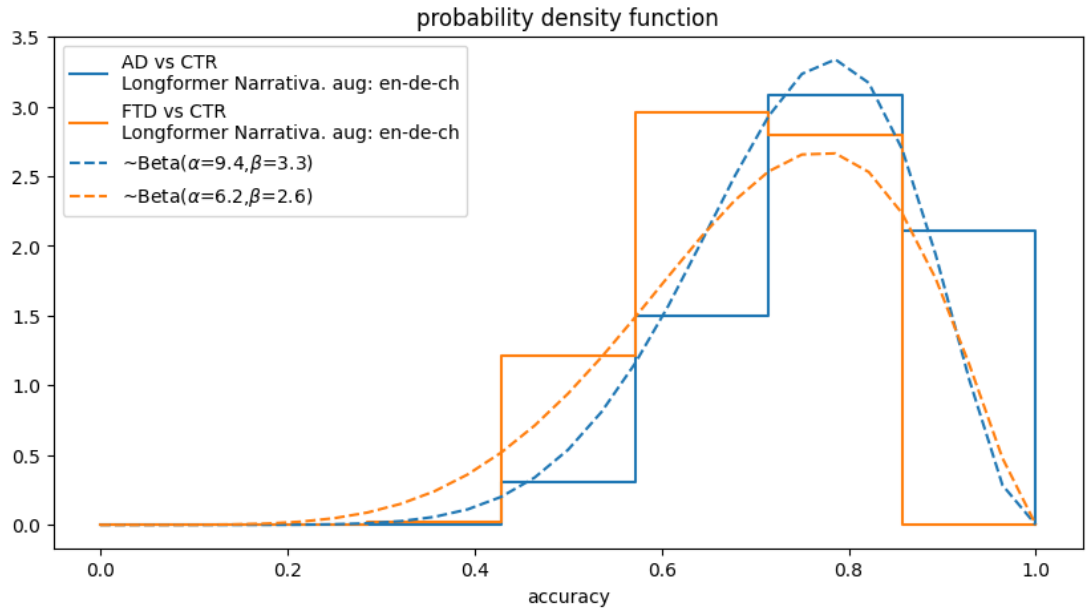


Figure 3.7: Distribución de Accuracy final para **AD vs CTR** y **FTDbv vs CTR** utilizando el **MODELO E: Longformer Narrativa + Dataset Augmentation** utilizando un umbral de detección $th = 0.84$

AD vs CTR (Modelo E: Longformer Narrativa Aug: En-De-Ch)		
threshodld	0.84	Umbral ROC
alpha	0.05	significance
Error I	**0.046**	P(CTR|AD)=74/(74+1526)
Error II	0.403	P(AD|CTR)=645/(645+955)
AUC	**0.89**	
Cota Inferior(accuracy)	0.571	P(Acc<0.571)=0.04
mean(accuracy)	**0.738**	(Media)
std(accuracy)	0.119	(Desvío)
skew(accuracy)	-0.545	(Oblicuidad)
kurt(accuracy)	0.037	(Curtósis)
FTDbv vs CTR (Modelo F: Longformer Narrativa Aug: En-De-Ch)		
threshold	0.92	Umbral ROC
alpha	0.05	significance
Error I	**0.048**	P(CTR|FTDbv)=58/(58+1142)
Error II	0.463	P(FTDbv|CTR)=556/(556+644)
AUC	**0.877**	
Cota Inferior(accuracy)	0.43	P(Acc<0.429)=0.0025
mean(accuracy)	**0.709**	
std(accuracy)	0.145	
skew(accuracy)	-0.533	
kurt(accuracy)	-0.119	

Tabla 3.5: Resumen de resultados finales para **AD vs CTR** y **FTDbv vs CTR**

Finalmente en las **Fig. 3.8 y 3.9** se observan diferentes resultados al predecir **AD** vs **CTR** y **FTDbv** vs **CTR** utilizando el modelo final con umbral $th = 0.5$.

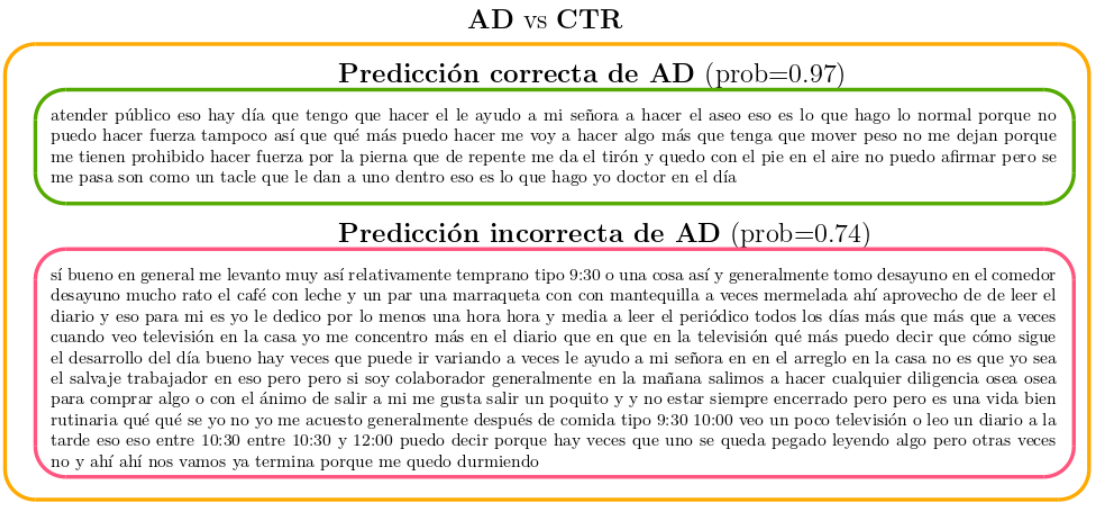


Figure 3.8: Ejemplo positivo y negativo para el ensayo AD vs CTR.

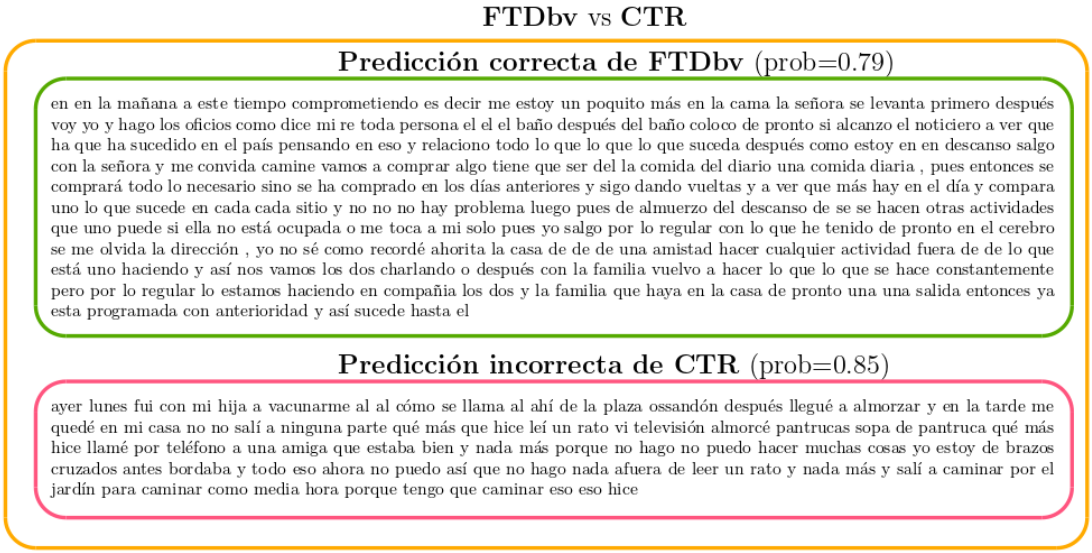


Figure 3.9: Ejemplo positivo y negativo para el ensayo FTDbv vs CTR.

Referencias

Capítulo 3

Efron, B. and R. Tibshirani (1986). “Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy”. In: *Statistical Science* 1.1, pp. 54–75. DOI: 10.1214/ss/1177013815. URL: https://doi.org/10.1214/ss/1177013815.

Flender, Samuel (Feb. 2022). *What exactly happens when we fine-tune BERT?*

Goodfellow, Ian J., Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. Cambridge, MA, USA: MIT Press.

Paszke, A. et al. (2017). “Automatic differentiation in PyTorch”. In: *NIPS 2017 Workshop Autodiff Decision Program Chairs*. URL: https://openreview.net/pdf?id=BJJsrmfCZ.

Perez, Luis and Jason Wang (2017). “The Effectiveness of Data Augmentation in Image Classification using Deep Learning”. In: *CoRR* abs/1712.04621. arXiv: 1712.04621. URL: http://arxiv.org/abs/1712.04621.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams (1986). “Learning Representations by Back-propagating Errors”. In: *Nature* 323.6088, pp. 533–536. DOI: 10.1038/323533a0. URL: http://www.nature.com/articles/323533a0.

Sennrich, Rico, Barry Haddow, and Alexandra Birch (2015). “Improving Neural Machine Translation Models with Monolingual Data”. In: *CoRR* abs/1511.06709. arXiv: 1511.06709. URL: http://arxiv.org/abs/1511.06709.

Wild, Christopher (Dec. 2010). *Chance encounters: A first course in data analysis and inference*. Wiley.

Zhang, Tianyi et al. (2021). *Revisiting Few-sample BERT Fine-tuning*. arXiv: 2006.05987 [cs.CL].

Conclusiones

Capítulo **4**

4.1 Conclusión

A lo largo del presente documento, se han propuesto diferentes modelos para detectar tanto **AD** como **FTDbv** en base al discurso del paciente, se ha probado de forma empírica que el modelo **Longformer Narrativa con Dataset Augmentation** ha sido el mejor candidato para predicir la presencia del trastorno, respecto a los otros candidatos. Sin embargo, debido a la falta de ensayos clínicos este modelo no puede usarse como diagnóstico o reemplazo de un especialista aplicado a este campo. El presente ensayo de *analizar el discurso del paciente*, es económico y puede realizarse de forma automatizada de forma *online*.

4.2 Discusión

Queda en discusión el análisis con mayores volúmenes de datos con el fin de aportar incluso más evidencia a la detección de enfermedades neurodegenerativas. Por otro lado, no hay garantías de que el modelo decida un diagnóstico en base a un sesgo (como la nacionalidad en este caso) en lugar de analizar los patrones distintivos de

cada trastorno (Por ejemplo, algunos pacientes con AD suelen ser poco específicos al contar un día de su vida).

4.3 Futuro del proyecto

Una futura continuación del proyecto incluye la adquisición de nuevas muestras tanto para entrenamiento como validación, con el fin de mejorar tanto el rendimiento del modelo cómo obtener una medida más certera de su rendimiento. Por otro lado, este proyecto podrá ser extendido hacia otros trastornos neurodegenerativos como el Parkinson. Finalmente se espera ampliar el espectro de edades y generalizar hacia una población hispanohablante de mayor tamaño.

Todas las Referencias

Allahyari, Mehdi et al. (2017). “Text Summarization Techniques: A Brief Survey”. In: *CoRR* abs/1707.02268. arXiv: 1707.02268. URL: http://arxiv.org/abs/1707.02268.

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton (2016). “Layer Normalization”. In: DOI: 10.48550/ARXIV.1607.06450. URL: https://arxiv.org/abs/1607.06450.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2014a). “Neural Machine Translation by Jointly Learning to Align and Translate”. In: DOI: 10.48550/ARXIV.1409.0473. URL: https://arxiv.org/abs/1409.0473.

— (2014b). “Neural Machine Translation by Jointly Learning to Align and Translate”. In: DOI: 10.48550/ARXIV.1409.0473. URL: https://arxiv.org/abs/1409.0473.

Beltagy, Iz, Matthew E. Peters, and Arman Cohan (2020). “Longformer: The Long-Document Transformer”. In: DOI: 10.48550/ARXIV.2004.05150. URL: https://arxiv.org/abs/2004.05150.

Cañete, José et al. (2020). “Spanish Pre-Trained BERT Model and Evaluation Data”. In: *PML4DC at ICLR 2020*. URL: https://users.dcc.uchile.cl/~jperez/papers/pml4dc2020.pdf.

Devlin, Jacob et al. (2019). “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: arXiv: 1810.04805 [cs.CL].

Efron, B. and R. Tibshirani (1986). “Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy”. In: *Statistical Science* 1.1, pp. 54–75. DOI: 10.1214/ss/1177013815. URL: https://doi.org/10.1214/ss/1177013815.

Eyigoz, Elif et al. (2020). “From discourse to pathology: Automatic identification of Parkinson’s disease patients via morphological measures across three languages”. In: *Cortex* 132, pp. 191–205. DOI: https://doi.org/10.1016/j.cortex.2020.08.020. URL: https://www.sciencedirect.com/science/article/pii/S0010945220303245.

Flender, Samuel (Feb. 2022). *What exactly happens when we fine-tune BERT?*

Goodfellow, Ian J., Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. Cambridge, MA, USA: MIT Press.

Gutiérrez-Fandiño, Asier et al. (2021a). *Spanish Legalese Language Model and Corpora*. arXiv: 2110.12201 [cs.CL]. URL: https://huggingface.co/PlanTL-GOB-ES/RoBERTalex?.

— (2021b). *Spanish Legalese Language Model and Corpora*. arXiv: 2110.12201 [cs.CL].

He, Kaiming et al. (2015). “Deep Residual Learning for Image Recognition”. In: DOI: 10.48550/ARXIV.1512.03385. URL: https://arxiv.org/abs/1512.03385.

Ioffe, Sergey and Christian Szegedy (2015). “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: DOI: 10.48550/ARXIV.1502.03167. URL: https://arxiv.org/abs/1502.03167.

Liu, Yinhan et al. (2019). “RoBERTa: A Robustly Optimized BERT Pretraining Approach”. In: DOI: 10.48550/ARXIV.1907.11692. URL: https://arxiv.org/abs/1907.11692.

Mikolov, Tomas et al. (2013). “Efficient Estimation of Word Representations in Vector Space”. In: arXiv: 1301.3781 [cs.CL].

Neurodegenerative diseases : unifying principles (2017). eng. New York, NY: Oxford University Press.

Pascanu, Razvan, Tomás Mikolov, and Yoshua Bengio (2012). “Understanding the exploding gradient problem”. In: *CoRR* abs/1211.5063. arXiv: 1211.5063. URL: http://arxiv.org/abs/1211.5063.

Paszke, A. et al. (2017). “Automatic differentiation in PyTorch”. In: *NIPS 2017 Workshop Autodiff Decision Program Chairs*. URL: https://openreview.net/pdf?id=BJJsrmfCZ.

Perez, Luis and Jason Wang (2017). “The Effectiveness of Data Augmentation in Image Classification using Deep Learning”. In: *CoRR* abs/1712.04621. arXiv: 1712.04621. URL: http://arxiv.org/abs/1712.04621.

Romero, Manuel (2022). *Legal Spanish LongFormer by Narrativa*. URL: https://huggingface.co/Narrativa/legal-longformer-base-4096-spanish.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams (1986). “Learning Representations by Back-propagating Errors”. In: *Nature* 323.6088, pp. 533–536. DOI: 10.1038/323533a0. URL: http://www.nature.com/articles/323533a0.

Sanz, Camila et al. (2022). “Automated text-level semantic markers of Alzheimer’s disease”. In: *Alzheimer’s & Dementia: Diagnosis, Assessment & Disease Monitoring* 14.1, e12276. DOI: https://doi.org/10.1002/dad2.12276. URL: https://alz-journals.onlinelibrary.wiley.com/doi/abs/10.1002/dad2.12276.

Sennrich, Rico, Barry Haddow, and Alexandra Birch (2015a). “Improving Neural Machine Translation Models with Monolingual Data”. In: *CoRR* abs/1511.06709. arXiv: 1511.06709. URL: http://arxiv.org/abs/1511.06709.

— (2015b). “Neural Machine Translation of Rare Words with Subword Units”. In: *CoRR* abs/1508.07909. arXiv: 1508.07909. URL: http://arxiv.org/abs/1508.07909.

Sojasingarayar, Abonia (2020). “Seq2Seq AI Chatbot with Attention Mechanism”. In: *CoRR* abs/2006.02767. arXiv: 2006.02767. URL: https://arxiv.org/abs/2006.02767.

Srivastava, Nitish et al. (2014). “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *J. Mach. Learn. Res.* 15.1, pp. 1929–1958.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le (2014). “Sequence to Sequence Learning with Neural Networks”. In: DOI: 10.48550/ARXIV.1409.3215. URL: https://arxiv.org/abs/1409.3215.

Tiedemann, Jörg (May 2012). “Parallel Data, Tools and Interfaces in OPUS”. In: *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC’12)*. Istanbul, Turkey: European Language Resources Association (ELRA), pp. 2214–2218. URL: http://www.lrec-conf.org/proceedings/lrec2012/pdf/463_Paper.pdf.

Vaswani, Ashish et al. (2017). “Attention Is All You Need”. In: *CoRR* abs/1706.03762. arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.