Universitat Oberta de Catalunya (UOC)

Máster Universitario en Ciencia de Datos (*Data Science*)

TRABAJO FINAL DE MÁSTER

Área: Procesamiento del lenguaje

Aplicación de métodos de aprendizaje semi-supervisados para el reconocimiento del habla en personas con afasia

---

Autor: Mónica Romero Ferrón

Tutor: Aitor Álvarez Muniain

Profesor: Iván Gonzalez Torre

Barcelona, 6 de junio de 2021

Créditos/Copyright

FICHA DEL TRABAJO FINAL

Título del trabajo:	Aplicación de métodos de aprendizaje semi-supervisados para el reconocimiento del habla en personas con afasia
Nombre del autor:	Mónica Romero Ferrón
Nombre del colaborador/a docente:	Iván Gonzalez Torre — Aitor Álvarez Muniain
Nombre del PRA:	Jordi Conesa Caralt
Fecha de entrega (mm/aaaa):	06/2021
Titulación o programa:	Máster universitario en Ciencia de Datos
Área del Trabajo Final:	Área 4
Idioma del trabajo:	Español
Palabras clave	Reconocimiento Automático del Habla, RAH, wav2vec2.0, afasia, redes neuronales

Agradecimientos

En primer lugar, quiero agradecer al centro Vicomtech por la oportunidad para desarrollar este proyecto con ellos. A mis tutores, en especial a Iván Gonzalez por su ayuda, motivación y su continua predisposición para resolver mis dudas y preguntas. A Aitor Álvarez por sus inestimable ayuda, correcciones y consejos. Muchas gracias a ambos por estar siempre disponibles y guiarme en este proceso, he aprendido muchísimo con y de vosotros.

También gracias a mi familia por su inestimable apoyo y a todas las personas que me han apoyado.

Resumen

Tradicionalmente, los sistemas de reconocimiento automático del habla (RAH) requieren de algoritmos que utilizan bases de datos etiquetadas para su aprendizaje. Sin embargo, un reciente y novedoso enfoque desarrolla modelos semi-supervisados que tienen la capacidad de realizar una parte de su entrenamiento con datos no etiquetados, facilitando así su uso en entornos donde los datos etiquetados son escasos. Este trabajo de investigación está centrado en la aplicación de estos métodos de aprendizaje en el ámbito de la salud y, más concretamente, en voces patológicas proveniente de hablantes con diferentes tipos de afasia. Se ha trabajado con la base de datos de referencia AphasiaBack, que contiene 78 horas de audios de pacientes con diferentes grados de afasia, y que ya ha sido empleada por otros grupos de investigación. A nivel de modelado, se ha optimizado y afinado la arquitectura de aprendizaje semi-supervisado empleada sobre estos datos de dominio, a través de la aplicación de la técnica Grid Search y de la búsqueda exhaustiva de los hiperparámetros del modelo. En este estudio se comparan los resultados obtenidos con los que se reflejan en el estado del arte. Se demuestra que el modelo de reconocimiento obtenido presenta resultados que mejoran otro tipo de enfoques publicados anteriormente.

Palabras clave: Reconocimiento Automático del Habla, RAH, wav2vec2.0, afasia, redes neuronales

Abstract

Automatic Speech Recognition (ASR) systems traditionally require algorithms that use labelled data for learning. However, a recent novel approach develops semi-supervised models that have the ability to perform part of their training on unlabelled data, thus facilitating their use in environments where labelled data is scarce. This research work is focused on the application of these learning methods in the health domain and, more specifically, on pathological voices of aphasia patients. We have trained our algorithm on the AphasiaBank database which has previously often been used as a benchmark. At the modelling level, the semi-supervised learning architecture used on this domain data has been optimised and refined by applying the Grid Search technique and the exhaustive search of the model's hyperparameters. The algorithms achieves high accuracy which we contrast with reported results from the literature. It is shown that the obtained recognition model presents results that improve on previous state of the art.

Keywords: Automatic Speech Recognition, ASR, wav2vec2.0, aphasia, Neural Network

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

El objetivo principal de este trabajo es la aplicación de métodos de aprendizaje semi-supervisado al reconocimiento automático del habla en situaciones caracterizadas por la escasez de datos etiquetados. En concreto, el estudio se centra en la arquitectura del algoritmo *wav2vec2.0* [6] publicado por Facebook IA en octubre de 2020. Se analiza el uso de este algoritmo al dominio de la salud y, mas concretamente, para el reconocimiento de voces patológicas afectadas por afasia.

El reconocimiento automático del habla (RAH) es un subcampo de la lingüística computacional que mediante la aplicación de diferentes tecnologías y técnicas permite convertir el habla en texto. El RAH está cada vez más presente en nuestro día a día como por ejemplo, integrado en diferentes aplicaciones, como en nuestros teléfonos móviles, relojes inteligentes o asistentes de voz del hogar cada vez más populares. A pesar de que el RAH ha ido evolucionando a lo largo del tiempo mediante la aplicación de diferentes técnicas, en los últimos años se ha conseguido dar un salto cualitativo de gran impacto, gracias entre otros a los numerosos avances relacionados con la Inteligencia Artificial y el Aprendizaje Profundo (*Deep Learning*), la cada vez mayor disponibilidad de datos de entrenamiento y capacidad de procesamiento en los sistemas hardware actuales se ha mejorado su precisión y rendimiento, especialmente desde que se han incorporado las redes neuronales profundas.

Los sistemas RAH más tradicionales requieren de algoritmos que utilizan bases de datos etiquetadas para su aprendizaje. Estas bases datos se componen de cientos o miles de horas de audios anotados. Aun así, disponer de datos de audio anotados a semejante escala es una tarea ardua y costosa, más incluso en los idiomas minoritarios.

Como evolución de los sistemas RAH tradicionales, en 2020, Facebook publicó una nueva arquitectura basada en técnicas de aprendizaje semi-supervisado, que permiten explotar una gran cantidad de datos no anotados en la fase inicial del entrenamiento. Sobre este modelo preentrenado y con una escasa cantidad de datos anotados, da la posibilidad de construir

modelos con resultados similares a los obtenidos por sistemas RAH tradicionales sobre bases de datos de referencia. La necesidad de una menor cantidad de datos etiquetados es una de las principales ventajas de los sistemas RAH. Además, estos ofrecen la posibilidad de construir sistemas para idiomas minoritarios, como los 122 idiomas principales de la india [8]

Para la realización de este análisis nos hemos apoyado en la base de datos de referencia, accesible en AphasiaBank, la cual contiene datos de audio para estudios enfocados en la comunicación con personas con afasia en diferentes idiomas. En este trabajo se ha elegido el idioma inglés para desarrollar el estudio.

La afasia es una alteración del lenguaje que afecta a la producción y/o comprensión del habla. Principalmente, lo padecen personas de la tercera edad y es causada por una lesión cerebral provocada comúnmente por accidentes cerebrovasculares, traumatismos craneoencefálicos, tumores cerebrales o infecciones. La afasia puede agravarse hasta el punto de incapacitar la capacidad de comunicación del paciente [34].

El objetivo principal de este estudio es avanzar en el estado del arte de los sistemas de reconocimiento del habla aplicados a las voces afectadas por afasia. En este sentido, un campo de aplicación importante es el desarrollo de aplicaciones que mejoren la rehabilitación del lenguaje y capacidad de comunicación de pacientes con afasia, mediante la identificación de patrones.. Igualmente, aplicaciones que incluyan reconocedores adaptados a estas voces y que permitan mejorar la transcripción para sistemas de interacción.

Capítulo 2

Estado del arte

2.1. Descripción y evolución de los sistemas reconocimiento automático del habla

En este capítulo se abordan los aspectos que ayudarán a contextualizar este trabajo. En primer lugar se realizó un análisis de los diferentes tipos de metodologías usadas en los sistemas de reconocimiento automático del habla; desde los métodos más tradicionales, pasando por modelos basados en redes neuronales y hasta llegar al algoritmo *wav2vec2.0* y los tipos de redes neuronales que integra.

2.1.1. Breve introducción e historia

A mediados del siglo XX se construyó Audrey, el primer sistema de reconocimiento basado en dígitos. Este podía reconocer dígitos numéricos hablados buscando huellas digitales de audios llamadas formants [19]. En 1960 IBM desarrolló un sistema llamado Shoebox, el cual reconocía dígitos y comandos aritméticos como "más" y "total" [16]. Durante este mismo periodo en Japón se construyó un hardware que podía reconocer partes que constituyen una palabra, como, por ejemplo, las vocales. También en la University College de Inglaterra se desarrolló un sistema que podía reconocer 4 vocales y 9 consonantes analizando fonemas [19].

Durante los años 70, la Agencia de Proyectos de Investigación Avanzada (o sus siglas en ingles ARPA, actualmente DARPA) del Departamento de Defensa de EEUU financió el programa *Speech Understanding Research*. Este programa desarrolló varios sistemas de RAH nuevos. En concreto, el sistema *Harpy* desarrollado por la Universidad Carnegie Mellon fue el mas exitoso de todos, que demostró ser capaz de reconocer el habla utilizando un vocabulario de 1.011 palabras con una precisión razonable. De forma paralela, IBM y Bell Laboratories centraron sus esfuerzos en el desarrollo de los sistemas de reconocimiento del habla para aplicaciones

comerciales [19].

La década de los ochenta se caracterizó por un cambio de metodología enfocándose en los modelos ocultos de Markov (HMM por sus siglas en inglés de Hidden Markov Models). Esto fue un punto de inflexión en las investigaciones sobre el reconocimiento del habla. Los HMM se utilizaba como la base principal para los sistemas automáticos de reconocimiento del habla. Esta técnica se han mantenido durante las últimas dos décadas gracias a las mejoras constantes y desarrollos tecnológicos [19].

En la década de 1990 se logró un gran progreso en el desarrollo de herramientas de software que permitieron el desarrollo de programas de investigación en todo el mundo pero no hubo avances significativos en las técnicas de reconocimiento del habla. Además en esta década también se introdujo el modelo de lenguaje N-gram que permitió mejorar el rendimiento de los sistemas de reconocimiento [19].

A partir del año 2000 fue cuando se comenzó a combinar las redes neuronales artificiales con modelos ocultos de Markov. Esto ya había sido explorado durante los años 80 y 90. Sin embargo, debido a la falta de recursos tecnológicos no se logró extraer grandes avances en este campo. En los últimos años, los esfuerzos en la construcción de arquitecturas de reconocimiento del habla han estado centrados en mejorar el desempeño de estos sistemas. Además, otro de los retos presentes en este ámbito, es optimizar los procesos de formación para que estos sean más simples y eficientes cuando se manejan grandes volúmenes de datos. Google o Facebook son algunas de las grandes corporaciones centradas en liderar este desarrollo de este tipo de arquitecturas y sistemas [19, 10].

2.1.2. De los sistemas de reconocimiento automático del habla tradicionales a los sistemas neuronales

El reconocimiento automático del habla (RAH) es un campo que ha evolucionado mucho a lo largo del tiempo. Anterior al aprendizaje profundo (*Deep Learning*, DL), el reconocimiento del habla ha estado marcado por un modelado estadístico basado en HMM (*Hidden Markov Model*) y GMM (*Gaussian Mixture Model*). Antes de empezar a definir en profundidad estos conceptos estadísticos, es necesario entender como se construía un sistema de reconocimiento del habla de manera clásica [17].

Como explica Navdeep Jaitly en su conferencia "End-to-End Models for Speech Processing" [25] sobre procesamiento de Lenguaje Natural con Deep Learning el objetivo del reconocimiento del habla clásico ha sido siempre encontrar la mejor secuencia de palabras creando modelos de lenguaje mediante N-grams, modelos de pronunciación mediante tablas, modelos acústicos mediante modelos mixtos Gaussianos y procesamiento de la señal de la voz [17].

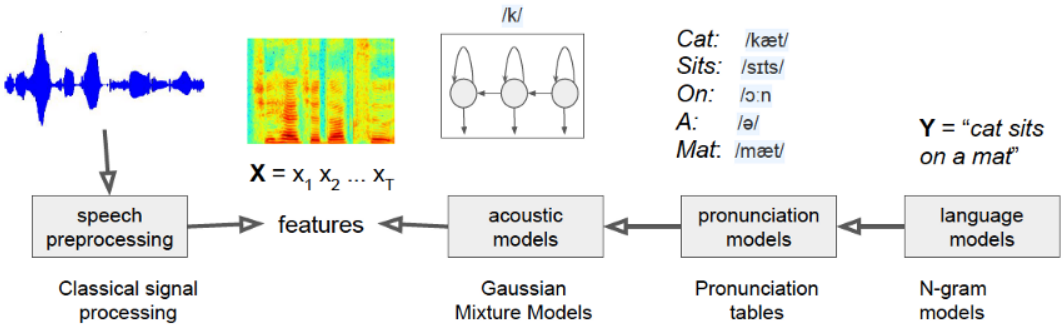

Figura 2.1: Modelo de reconocimiento del habla clásico [17]

De derecha a izquierda de la figura 2.1 se tiene:

- En la figura tenemos el procesamiento del habla que está predefinido por 39 características del audio extraídas mediante los *Mel-frequency cepstral coefficients* (MFCC) [17]. En este tipo de modelos clásicos de lenguaje natural el objetivo es afinar cada modelo nombrado anteriormente. Una vez que se ha construido este modelo se realizan inferencias sobre los datos que recibe y se intenta buscar cual es la secuencia de palabras W con mas probabilidad que se obtenga a patir de las X features que obtenemos, es decir, calculamos [17]:

$$W^* = \arg\max_W P(X|W)P(W)$$

- Los modelos acústicos se generan mediante modelo de mezclas Gaussianas. Un modelo de mezclas Gaussianas es un modelo probabilístico que supone que todos los puntos de datos se generan a partir de una mezcla de un número finito de distribuciones Gaussianas con parámetros desconocidos [31].

- Los modelos de pronunciación son tablas de tamaño muy grande que contiene un diccionario fonético de cada palabra [17].

- Los modelos de lenguaje basados en N-gramas se componen de tablas que describen las probabilidades de la secuencias de palabras. Esta secuencia de probabilidades se construye mediante un modelo Bayesiano de N-grams:

$$\begin{aligned} P(W) &= P(w_1)P(w_2|w_1)P(w_3|w_1w_2...P(w_n|w_1w_2...w_{n-1}) \\ &= \prod_{i=1}^{n} P(w_i|w_1w_2...w_{i-1}) \end{aligned}$$

Esto explica como calcular la probabilidad de la secuencia de palabras W. Primero se calcula la probabilidad de la primera palabra w_1 que aparece en la secuencia de palabras. Tras esto se calcula la probabilidad de que aparezca la segunda palabra w_2 teniendo en cuenta la anterior y así sucesivamente hasta llegar a calcular la última palabra de la secuencia teniendo en cuenta todas las anteriores [17]. Por ejemplo, es más probable la secuencia de palabras "Yo veo un gato" que "Yo miro un gato".

2.2. Modelos basados en redes neuronales

En muchos campos del *Machine Learning* ha habido una revolución gracias al gran avance en el área del *Deep Learning* y en la aplicación de redes neuronales profundas. Con respecto al procesamiento de lenguaje natural se comenzaron a realizar cambios en la forma de abordarlos [17]:

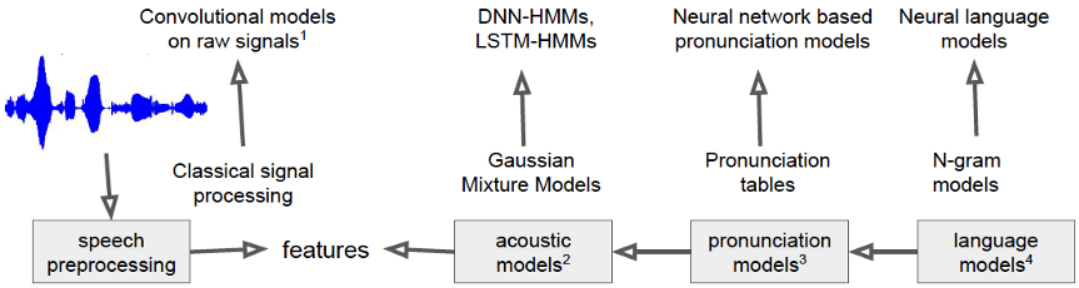

Figura 2.2: Modelo de reconocimiento del habla basados en redes neuronales [17]

Como se observa en el esquema representado en la figura 2.2 se han sustituido cierto procesos por otros:

- En lugar de usar modelos de lenguaje N-gram se construirían modelos de lenguaje neuronal.
- Los modelos de pronunciación se podrían sustituir por redes neuronales profundas basadas en modelos de pronunciación para obtener clasificaciones con mejor precisión.
- Si bien en los modelos tradicionales GMM-HMM la probabilidad de transición entre estados era estimada con modelos probabilísticos basados en mezclas de gaussianas, en los nuevos modelos DNN-HMM esta probabilidad de transcripción es computada mediante una red neuronal.

En esta arquitectura se entrena y optimiza cada componente por separado y luego se combinan en una gráfica de reconocimiento común. Los sistemas End-to-End (E2E) tienen como uno

de los objetivos principales resolver este problema mediante la optimización de todos los componentes de manera conjunta en un único proceso de entrenamiento usando un único criterio de optimización.

Para conseguir este objetivo se crearon los modelos E2E Neural Network. Antes de describir como es un modelo E2E vamos a definir que es una red neuronal y algunos de los tipos de capa más comunes [17].

2.2.1. Redes Neuronales - Deep Learning

Las redes neuronales son un caso especial de los algoritmos de Machine Learning. La idea general y por la que recibe el nombre este tipo de algoritmos es que se comportan de forma similar a neuronas humanas y la forma de obtener resultados.

El *Machine learning*, las Redes Neuronales y el *Deep Learning* se organizan como se detalla en el siguiente esquema:

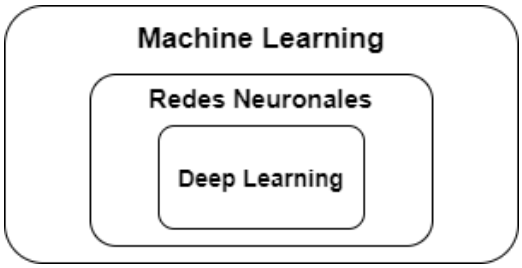

Figura 2.3: Esquema Machine Learning, Redes Neuronales y Deep Learning

En la figura 2.3 se indica que el *Deep learning* es el caso concreto de las redes neuronales. Esto sucede cuando tenemos una red neuronal que esta compuesta por múltiples capas de procesamiento para aprender.

Una red neuronal esta formada por interconexiones entre neuronas. A cada una de ellas se les pueden aplicar transformaciones tanto lineales como no lineales a los valores de sus entradas procedentes de las conexiones con otras neuronas y con ello obtenemos un valor nuevo que se convierta en la salida de la neurona [7].

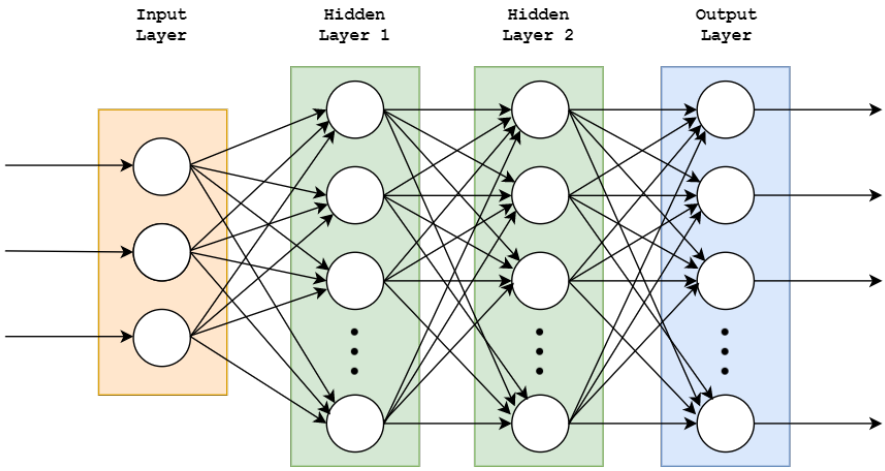

Figura 2.4: Red Neuronal

En la figura 2.4 se muestra como se forma una red neuronal de 4 capas. Comienza con una capa de entrada o *input layer*, esta capa recibe los datos de entrada. Tras esta capa se hayan dos capas ocultas denominadas *hidden layers*, no necesariamente tiene que estar formada por 2 capas ocultas se podría crear con todas las que se desee y cada una podría tener una cantidad de neuronas distinta. Por último, tendremos una capa de salida. Estas combinaciones hacen que podamos obtener patrones complejos [7].

2.3. Métodos semi-supervisados - Wav2Vec2.0

2.3.1. Introducción y arquitectura

Como referencia de este tipo de modelos, Facebook publicó en el estudio [9] la arquitectura *Wav2vec2.0* fue propuesta en el estudio "Wav2vec2.0: A Framework for Self-Supervised Learning of Speech Representations" [6]. Este modelo está enfocado a tareas como el reconocimiento del habla, segmentación de audio o detección de anomalías mediante la explotación de datos sin etiquetar [33]. Poco después de esta publicación *Facebbok AI* publicó *XLSR-Wav2vec2.0* [9]. Las siglas XLSR significan *cross-lingual speech representations* y hacen referencia a la capacidad que tiene el modelo *XLSR-Wav2vec2.0* para aprender las representaciones del habla que son útiles en varios idiomas. Ambos algoritmos aprenden representaciones de voz de cientos de miles de horas de voz en más de 50 idiomas sin etiquetar. *XLSR-Wav2Vec2* se entrenó previamente con los datos de audio de Babel [1] (1.700 horas de datos de voz en 17 idiomas), Multilingual LibriSpeech [29] (más de 50.000 horas de audios en 8 idiomas) y Common Voice [2] (3.600 horas de audios en 36 idiomas) [27].

[1] https://catalog.ldc.upenn.edu/byyear
[2] https://commonvoice.mozilla.org/en/languages

Wav2Vec2.0 codifica el audio de voz a través de una red neuronal convolucional de múltiples capas (CNN), para realiza posteriormente un enmascaramiento en la representaciones de voz latentes resultantes. Las representaciones latentes que hemos generado se alimentar de una red *Transformer* para construir representaciones contextualizadas. Después del entrenamiento previo del habla sin etiquetar, el modelo se ajusta con precisión a los datos etiquetados con una perdida de clasificación temporal conexionista (CTC), detallada en la sección 2.3.2.2, que se utilizará para tareas de reconocimiento del habla posteriores [33].

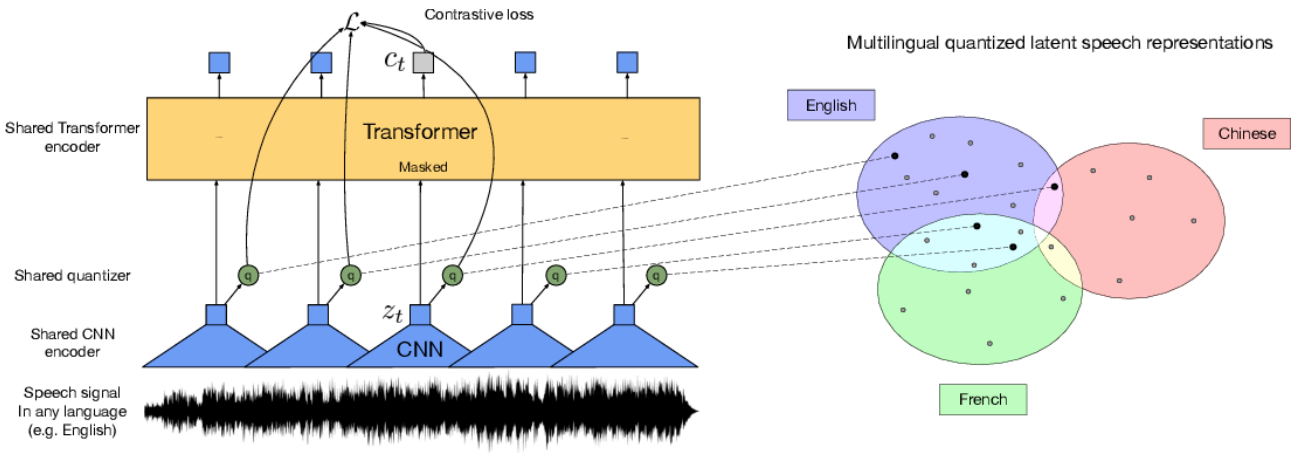

Figura 2.5: Diagrama del modelo XLSR-Wav2vec2.0 [9]

A pesar de ser un algoritmo con menos de un año de publicación se está convirtiendo en un algoritmo de mucho interés por la comunidad, gracias a su gran potencial. Estudios como el referenciado [41] en por en concreto consiguen mejoras de mas del 20 % en seis idiomas de bajo recursos en comparación con trabajos anteriores. Además en otros idiomas como ingles, que no es considerado de bajos recursos, logra una mejora de un 52,4 %.

Otro trabajo publicado recientemente es el realizado en el estudio [42] donde fusiona el modelo *wav2vec2.0* y un codificador lingüístico previamente entrenado (BERT) en un modelo E2E. Utilizando el corpus *CALLHOME* [3] de solo 15 horas logran mejorarlo con respecto a otros modelos E2E usado anteriormente .

Esta arquitectura está formada por las redes neuronales CNN, Transformer y CTC. En los siguiente apartados se explica y describe cada una de estas redes neuronales.

[3]https://catalog.ldc.upenn.edu/LDC97S42

2.3.2. Componentes y tipos de redes

2.3.2.1. Redes Neuronales Convolucionales

Las redes neuronales convolucionales (*Convolutional Neural Network* - CNN) son un tipo en concreto de redes neuronales profundas donde los datos que se usan en la entrada son imágenes. Jordi Torres en su libro "Deep Learning: práctica con Keras" [35] expone como trabajan este tipo de redes neuronales con el siguiente ejemplo:

"Si vemos una cara, la reconocemos porque tiene orejas, ojos, una nariz, cabello, etc. Entonces, para decidir si algo es una cara, lo hacemos como si tuviéramos unas casillas mentales de verificación de las características que vamos marcando. Algunas veces una cara puede no tener una oreja por estar tapada por el pelo, pero igualmente lo clasificamos con una cierta probabilidad como cara porque vemos los ojos, la nariz y la boca. Pero en realidad, antes debemos saber cómo es una oreja o una nariz para saber si están en una imagen; es decir, previamente debemos identificar líneas, bordes, texturas o formas que sean similares a las que contiene las orejas o narices que hemos visto antes. Y esto es lo que las capas de una red neuronal convolucional tienen encomendado hacer" [35].

En la figura 2.6 se explica los diferentes tipos de capas que encontramos en una CNN.

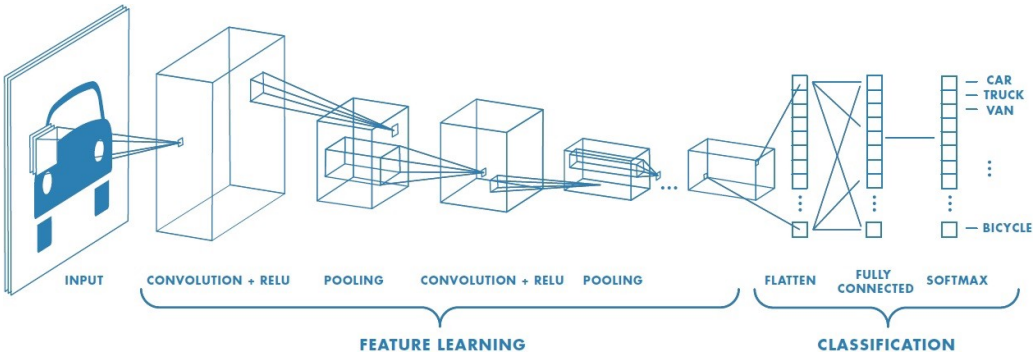

Figura 2.6: Red Neuronal Convolucional con varias capas convolucionales [23]

Se comienza con una capa de convolución. Esta capa realiza una función de transformación de la imagen para poder extraer características de esta, es decir, cada capa puede detectar bordes, colores y su distribución, tamaños, texturas , formas similares, etc. en regiones distintas de la imagen. Lo que se consigue con esto es que la red neuronal tolere una traslación en la imagen [35].

Otra función que poseen estas capas es la de aprender jerarquías espaciales de patrones preservando relaciones espaciales. En la imagen anterior se puede observar como se tienen varias capas convolucionales, por ejemplo, en la primera capa convolucional esta red neuronal ha podido aprender elementos básicos como bordes. En la siguiente capa convolucional patrones

basándose en el elemento básico que aprendió anteriormente y así sucesivamente aprender patrones más complejos y abstractos [35].

La siguiente capa que normalmente se aplican después de una capa convolucional es una capa de pooling, como se puede observar en la imagen 2.5. Estas capas simplifican la información recogida en las capas convolucional y generan una información condensada obtenida anteriormente. La forma mas usada para condensar la información es *max-pooling*. El método consiste en obtener el valor máximo de los obtenidos en la capa convolucional [35].

Por último, se encuentra la sección de clasificación de la red neuronal. Esta comienza con una capa de aplanar, convierte los datos en una matriz unidimensional para así agregarla a otra capa con la red neuronal, por ejemplo, a una capa densamente conectada (*fully connected*). La última capa de salida es la función softmax, con la cual se obtendrá un vector N-dimensional que representa la probabilidad de pertenecer a cada clase [7].

Las redes neuronales convolucionales han sido durante varios años una revolución en el reconocimiento de imágenes. Se han aplicado en numerosos estudios con resultados óptimos como "Convolutional Neural Networks for Speech Recognition" en 2014 donde utilizando combinaciones de modelos ocultos de Markov y redes neuronales convolucionales se obtienen reducciones de la tasa de error de en un 6 % - 10 % [3]. Otros estudios realizados como "An analysis of convolutional neural networks for speech recognition" aportan un análisis más detallado de las CNN, donde encuentran varios dominios en los cuales usando este tipo de redes se obtienen mejores resultados [15].

2.3.2.2. Clasificación temporal conexionista

Clasificación temporal conexionista o conocida en ingles como *Connectionist temporal classification (CTC)*. Este tipo de red neuronal de salida se utiliza en el entrenamiento de las redes neuronales recurrentes (RNN). La función de perdida de las CTC permite entrenar redes neuronales E2E enfocadas a RAH, en concreto estas redes se centran en la necesidad de segmentar el sonido en trozos que representan palabras. Las CTC permiten predecir una transcripción directamente desde una entrada de audio [21].

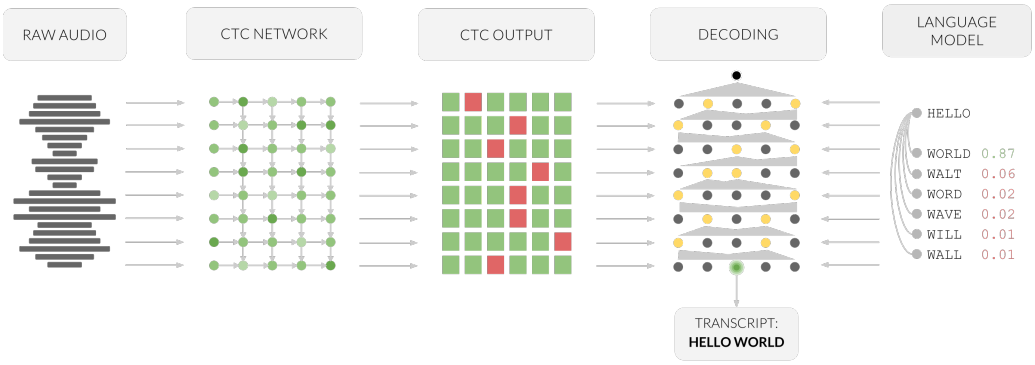

Figura 2.7: Red neuronal CTC [21]

Para obtener esta transcripción las CTC se basan en una matriz donde las columnas son el paso del tiempo y las filas corresponde a una letra del alfabeto, un espacio en blanco (-) donde indica que acaba esa letra y un caracter final (>) para predecir cual es la ultima palabra, básicamente es una predicción de letras. Este corresponde al tercer paso de la imagen anterior donde los cuadrados rojos son la probabilidad máxima en ese tiempo y corresponde a una letra o caracter especial [21].

El paso siguiente seria calcular la probabilidad de todas las rutas posibles en estas matriz, sumar la probabilidad de que las rutas produzcan la misma etiqueta y tras esto seleccionar la etiqueta con la probabilidad más alta [21].

Las redes CTC aportan muy buenos resultados tanto combinandolas con otras redes como por si solas en aplicaciones del reconocimiento automático del habla. Estudios como "Joint CTC-attention based end-to-end speech recognition using multi-task learning" expone las dificultades que tienen las redes CTC en condiciones ruidosas pero solventadolo con un modelo de atención CTC conjunto dentro de un aprendizaje multitarea [20]. Por otro lado, tenemos "Hybrid CTC/Attention Architecture for End-to-End Speech Recognition" donde generan un modelo híbrido generado con el modelo de voz convencional basado en modelos de Markov ocultos con redes CTC mediante programación dinámica [39].

2.3.2.3. Transformer

Transformer es un tipo de red neuronal usado principalmente en el procesamiento del lenguaje natural y de gran popularidad en los últimos años. La característica y diferencia principal de las transformer es que no procesa los datos de la secuencia en orden como sucedía con las RNN.

Esta red es una combinación de redes neuronales convolucionales y modelos de atención con los que conseguimos aumentar la velocidad con la que el modelo puede traducir una secuencia. Mediante una secuencia de entrada la atención decide en cada paso que otras partes de esta

secuencia son importantes. Esto quiere decir que la atención selecciona y memoriza las palabras claves de un texto y las utiliza para proporcionar contexto más adelante [24, 38].

Esta red neuronal está compuesta por 6 codificadores y 6 decodificadores. Una particularidad de las *Transformer* es que trabajan de forma muy eficiente en paralelo durante el entrenamiento, lo que hace reduzca el tiempo requerido en el entrenamiento. Esto es gracias a que en los codificadores se tiene una capa de auto-atención la cual ayuda a buscar otras palabras en la sentencia de entrada mientras codifica una palabra en concreto. Por otro lado, los decodificadores tiene esta capa de auto-atención además de una capa de atención que le permite enfocarse en las partes mas relevantes la codificación realizada anteriormente por los codificadores. Por último, se tiene una capa *softmax* para obtener la probabilidades de salida por cada palabra [24, 38].

2.4. Métodos no supervisados - Wav2vec-U

Los sistemas de reconocimiento del habla han tenido un rápido desarrollo y progreso, aún así, estas técnicas requieren de datos etiquetados que no están disponibles para la gran mayoría de los casi 7.000 idiomas del mundo [32]. En el estudio publicado en mayor del 2021 por Facebook IA, "Unsupervised Speech Recognition" [5], se muestra una revolucionario y nuevo algoritmo no supervisados *wav2vec-U*, abreviatura de wav2vec Unsupervised, un método para entrenar modelos de reconocimiento del habla sin ningún dato etiquetado.

Este modelo utiliza las representaciones auto-supervisadas de *wav2vec2.0* para segmentar el audio en unidades con un método simple de agrupación de k-means. A continuación, se construyen los segmentos obtenidos mediante la combinaciones de representaciones de *wav2vec2.0* y se realiza un análisis de componentes principales. Estas son las entradas que se le proporcionan al generador, el cual crea una secuencia de fonemas que se envían al discriminador para la realización de un aprendizaje adverso [5].

A continuación, en la figura 2.8, se puede ver el esquema del algoritmo *wav2vec-U*:

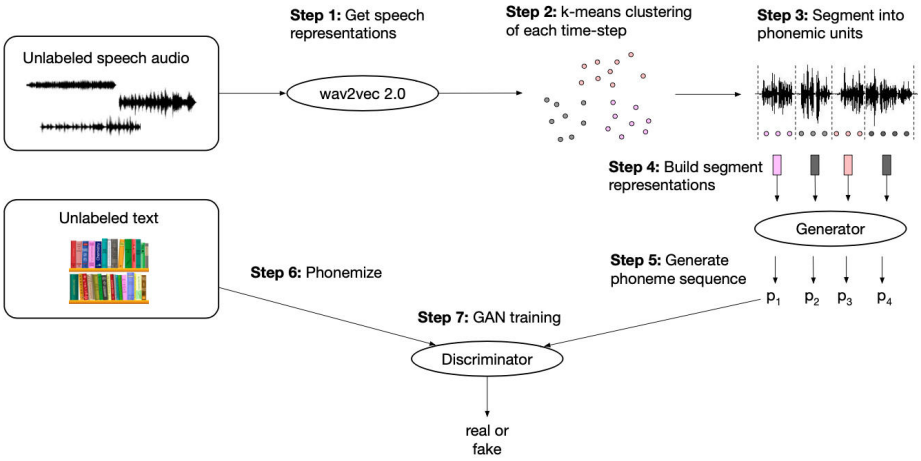

Figura 2.8: Esquema algoritmo wav2vec-U [5]

Este trabajo compara sus resultados con anteriores estudios y demuestra como *wav2vec-U* reduce la tasa de error en fonemas en la base de datos acústica TIMIT [4] de 26.1 a 11.3 [5].

2.5. Aplicación del reconocimiento automático del habla en la salud

2.5.1. Breve introducción

Existen diversas aplicaciones del reconocimiento automático del habla en el ámbito sanitario, por ejemplo en enfermedades o patologías como el Alzheimer, tartamudez o demencia. En concreto se han realizado numerosos estudios sobre datos de voces de pacientes con Alzheimer. Como el realizado por la universidad de Sheffield donde se obtuvieron mejores resultados basándose en segmentos de audio de buena calidad [26].Estudios donde se analizan en aspectos como la cohesión y la coherencia además de una combinación de otras características que poseen las personas con Alzheimer mediante algoritmos de RAH [28]. La publicación [13] se centra en enfermedades caracterizadas por el deterioro de la memoria o lenguaje. Mediante sistemas RAH se extrajeron parámetros como el tiempo del habla, tasa de articulación entre otros que describen en que estado se encuentra el paciente. Estos parámetros ayudan a la detección temprana de esta enfermedad.

La disartria es una alteración en la articulación de las palabras, es decir, el paciente manifiesta dificultades asociadas con la articulación de fonemas [1]. Se han publicado estudios basados

[4]https://catalog.ldc.upenn.edu/LDC93S1

en modelos acústicos de RAH de última generación donde se consigue una mejora relativa del 7.6 % [40]. Otras publicaciones como la realizada por la universidad de Sheffield realizan un estudio del impacto de utilizar modelos de lenguaje combinados con modelos acústicos de ultima generación sobre una base de datos de voces de pacientes con disartria [43].

2.5.2. Reconocimiento automático del habla aplicado a la afasia

La afasia es un trastorno del habla debido a lesiones producidas por un accidente cerebrovascular o una lesión cerebral.

La publicación [18] enfoca su estudio en este tipo de lesión hace una revisión de las tecnologías RAH aplicadas en personas con trastornos del habla y lenguaje como DNN.

El trabajo realizado por la universidad de Michigan donde estudian usando AphasiaBank una base de datos compartida sobre afasia. En el estudio obtienen mejoras en la tasa de reconocimiento en un corpus de habla afásico mas pequeño, además muestra el potencial de esta base de datos para poder realizar trabajos posteriores [22].

El estudio realizado [30] con la base de datos AphasiaBank utilizando el idioma cantones. Desarrollar un sistema de reconocimiento del habla basados en redes neuronales profundas mediante un entrenamiento de múltiples tareas con datos de voz tanto fuera como dentro del dominio. Las características del texto se combinan con características acústicas para cubrir los diferentes aspectos de la afasia.

2.6. Contribuciones principales

Una de las contribuciones de este trabajo es la aplicación del novedoso algoritmo *wav2vec2.0* para el reconocimiento del habla en personas con afasia. Hasta ahora este algoritmo solo había sido aplicado a el aprendizaje de idiomas con bajo recursos por lo que este trabajo aporta una nueva aplicación a otro tipo de bases de datos relacionados con el ámbito sanitario. Otra de las aplicaciones es la realización de un estudio de los diferentes hiperparámetros que se pueden utilizar en el algoritmo. El objetivo es dar la combinación y valores óptimos de los hiperparámetros para poder obtener los mejores resultados.

Capítulo 3

Arquitectura del modelo semi-supervisado XLSR-Wav2vec2.0

Para este estudio se ha utilizado el algoritmo semi-supervisado *XLSR-Wav2vec2.0*[9] y el corpus usado es la base de datos alojada AphasiaBank. A continuación, se detalla cada uno de los pasos que se han llevado a cabo para el entrenamiento del modelo.

3.1. Modelo semi-supervisado XLSR-Wav2vec2.0

Wav2vec2.0 es un modelo preentrenado para el reconocimiento automático del habla y fue lanzado en octubre de 2020 [6]. Poco después de que se demostrase el gran potencial y rendimiento de *Wav2vec2.0* en el conjunto de datos de RAH en inglés LibriSpeech [29], Facebook AI publico el modelo *XLSR-Wav2vec2.0* [9]. XLSR (*cross-lingual speech representations*) significa representaciones del habla cruzada lingual, a lo que se refiere es a la capacidad de *XLSR-Wav2vec2.0* para aprender representaciones del habla que son útiles en varios idiomas [27, 6].

De manera análoga a *Wav2vec2.0*, *XLSR-Wav2vec2.0* es un modelo preentrenado con 50 idiomas de voz sin etiquetar. De manera similar al como se comporta el modelado de lenguaje enmascarado de BERT, *XLSR-Wav2vec2.0* aprende representaciones de voz contextualizadas enmascarando aleatoriamente los vectores de características antes de pasarlo a una red neuronal transformer [27, 6].

En este estudio se ha utilizado el modelo preentrenado *wav2vec2-large-xlsr-53 checkpoint*. Este modelo se entrenó previamente en audios de voz muestreados de 16 kHz. Por ello hay asegurarse de que las entradas de voz se muestrean a 16kHz. *XLSR-Wav2vec2.0* se ajusta con precisión mediante redes neuronales CTC [14, 6].

El entrenamiento del modelo se basa en una de las APIs de *Hugging Face* concretamente la

API *Trainer* [1]. Para hacer uso de esta API hay que seguir los siguientes pasos:

1. Definir un clasificador de datos. A diferencia de muchos modelos de NLP, *XLSR-Wav2vec2.0* tiene una longitud de entrada mucho mayor que la de salida. Con esto se consigue rellenar los lotes de entrenamiento de forma dinámica y por lo tanto, el *fine-tuning* requiere de un *padding* especial.

2. Se define una función llamada *compute metrics* donde evaluará el modelo en función de la tasa de error de palabras.

3. Se carga un modelo ya entrenado previamente y configurado para el entrenamiento. En este caso será con el modelo preentrenado *wav2vec2-large-xlsr-53 checkpoint*.

4. Se configuran los hiperparámetros de entrenamiento.

Una vez realizados estos pasos el modelo quedará ajustado. Se evaluará con los datos de *train* y se verificará que el modelo haya aprendido mediante los datos del test.

Este modelo ha sido evaluado mediante las métricas WER y CER. Además en la comunidad para los sistemas basados en el modelado de fonemas también se utiliza la métrica PER.

A continuación se detallan cada una de ellas:

- La tasa de error de palabras, WER (Word Error Rate), se calcula de la siguiente manera:

$$WER = \frac{S + D + I}{N} = \frac{S + D + I}{S + D + C}$$

  Dónde:

  - S es el número de sustituciones.
  - D es el número de eliminaciones.
  - I es el número de inserciones.
  - N es el número de palabras en la transcripción.
  - C es el número de palabras correctas.

- La tasa de error de caracteres, CER (Character Error Rate), se calcula de forma análoga al WER pero opera con caracteres en lugar de palabras.

- La tasa de error de fonemas, PER (Phone Error Rate), se calcula de forma análoga al WER y PER pero opera con fonemas en lugar de palabras o caracteres.

[1] https://huggingface.co/transformers/main_classes/trainer.html

Cuanto menor sean los valores tanto de WER, CER y PER, mejor será el rendimiento del sistema RAH. Cabe destacar que, en general, las tasas de errores de palabras (WER) es mucho más alta que las tasas de errores de caracteres (CER).

3.2. Hiperparámetros

Antes de iniciar el *Trainer* [2] se crea un *TrainingArguments*, que representa al subconjunto de argumentos usados para acceder a la personalización de los hiperparámetros del entrenamiento. Entre los hiperparámetros posibles que ofrece esta librería, para este trabajo se han considerado los siguientes:

- *group_by_length* (bool) agrupa muestras de aproximadamente la misma longitud en el conjunto de datos de entrenamiento [36].
- *per_device_train_batch_size* (int) es el tamaño de muestras por GPU/TPU/CPU para el entrenamiento [36].
- *gradient_accumulation_steps* (int) es número de pasos de actualización para los que se acumularán los gradientes [36].
- *evaluation_strategy* (str) los posibles valores son[36]:
  - no: no se realizará ninguna evaluación durante el entrenamiento.
  - *steps*: la evaluación se realiza cada eval_steps.
  - *epoch*: la evaluación se realiza al final de cada época.
- *num_train_epochs* (float) número total de épocas de entrenamiento a realizar [36].
- *fp16* (bool) precisión de 16 bits en lugar de 32 bits [36].
- *save_steps* (int) número de pasos de actualización antes de que se guarden [36].
- *eval_steps* (int) número de pasos de actualización entre dos evaluaciones si *evaluation_strategy* es steps [36].
- *logging_steps* (int) número de pasos de actualización entre dos registros si *evaluation_strategy* es steps [36].
- *learning_rate* (float) la tasa de aprendizaje inicial para el optimizador Adam [36].

[2] https://huggingface.co/transformers/main_*classes*/*trainer.html*

- *warmup_ratio* (float) proporción aplicada a los pasos de entrenamiento totales usados para un calentamiento lineal (linear warmup) de 0 a *learning_rate* [36]. Esto ayuda a no sufrir un sobre ajuste temprano del modelo debido a un conjunto de datos que este muy diferenciado.

- *save_total_limit* (int) este valor limitará la cantidad total de puntos de control [36].

- *lr_scheduler_type* (str) el tipo de programador que se utilizará [36]. En este estudio se utilizará el coseno y lineal.

- *preprocessing_num_workers* (int) número de procesos que generan lotes en paralelo con PyTorch [2].

- *mask_time_prob* (float) probabilidad de cada vector de características a lo largo del tiempo para ser elegido como el inicio del intervalo de vectores a enmascarar. Este es un hiperparámetro de las transformers [27].

- *hidden_dropout* (float) probabilidad de abandono de todas las capas completamente conectadas [27].

Para la realización de este estudio se ha hecho uso de los servidores de Vicomtech, con un docker previamente configurado para el entrenamiento del modelo. Dentro del servidor se genera la imagen del docker mediante la siguiente linea de comando:

```
docker build -t {nombre del docker} .
```

Una vez que se ha generado la imagen, se crea el *container* con el siguiente código:

```
NV_GPU={número de la tarjeta gráfica dentro del servidor} nvidia-docker
run -it -d --rm --name {nombre del container}
--runtime=nvidia --shm-size=4g --ulimit memlock=-1 --ulimit stack=67108864
-v {directorio workspace}/ -v /data/:/data/:ro {nombre de la imagen}
```

Las tarjetas gráficas usadas para este trabajo han sido las Titan Pascal [3] y las GeForce RTX 3090 [4].

[3] https://huggingface.co/transformers/main_classes/trainer.html
[4] https://www.nvidia.com/es-es/geforce/graphics-cards/30-series/rtx-3090/

3.3. Hugging Face Fine-Tuning Week

Durante la semana del 22 al 29 de marzo la comunidad de *Hugging Face* realizó una competición para afinar el modelo de reconocimiento del habla *XLSR-Wav2vec2.0* en todos los idiomas del conjunto de datos Common Voice [4].

El evento consistió en que los participantes tenían una semana para ajustar tantos modelos de *XLSR-Wav2vec2.0* en tantos idiomas del conjunto de datos de Common Voice [4] quisieran. Tras esto, se debía evaluar con los datos de prueba de Common Voice [4] del idioma elegido. Para ello el organizador del evento, dejó a disposición de los participantes un script en Google Colab para poder realizar el ajuste del modelo.

Como parte de este trabajo, durante esta semana, se realizaron inferencias y ajustes del modelo sobre el idioma Turco. Esto permitió analizar como funciona el modelo y cuales eran los hiperparámetros o combinaciones más interesantes para realizar una inferencia. Además esto permitió contrastar toda esta información con la comunidad de *Hugging Face* que participaba durante esa semana.

El código realizado se encuentra en el anexo 2 de este trabajo. Donde están los scripts con los que se realizaron las inferencias. En estos códigos se puede encontrar las combinaciones de hiperparámetros utilizados y el resultado obtenido con cada uno de ellos.

[4]https://commonvoice.mozilla.org/en/languages

Capítulo 4

Corpus AphasiaBank

4.1. Descripción del corpus

El corpus usado para el estudio se ha obtenido de la base de datos compartida AphasiaBank, creada por y para investigaciones del estudio de la afasia. Esta base de datos contiene sesiones en diferentes idiomas en las que interactúan pacientes con afasia e investigadores. AphasiaBank es una base de datos multimedia formada por subcolecciones de datos realizadas por diferentes grupos de investigación. La base de datos está compuesta por vídeos en formato mp4 y transcripciones en formato CHAT. Los vídeos fueron descargados desde su repositorio de datos mediante el programa *Multi-File Downloader* [1] como recomiendan desde la propia web AphasiaBank . Las transcripciones se realizan en formato CHAT y se codifican con los programas CLAN (Computerized Language Analysis) [4].

Para este trabajo se ha utilizado la base de datos en inglés, compuesta por 24 sub-datasets con 426 participantes de los cuales 255 son hombres y 171 mujeres.

El rango de edad media de los participantes es de 60 ± 10 años. Se observa como en el género masculino predomina el rango de edad de entre 60 y 69 años. Se puede ver su distribución en la figura 4.1 a continuación:

[1] https://chrome.google.com/webstore/detail/multi-file-downloader/dpecplbkinpdbedgejddhepkgcppgchk?hl=en-GB

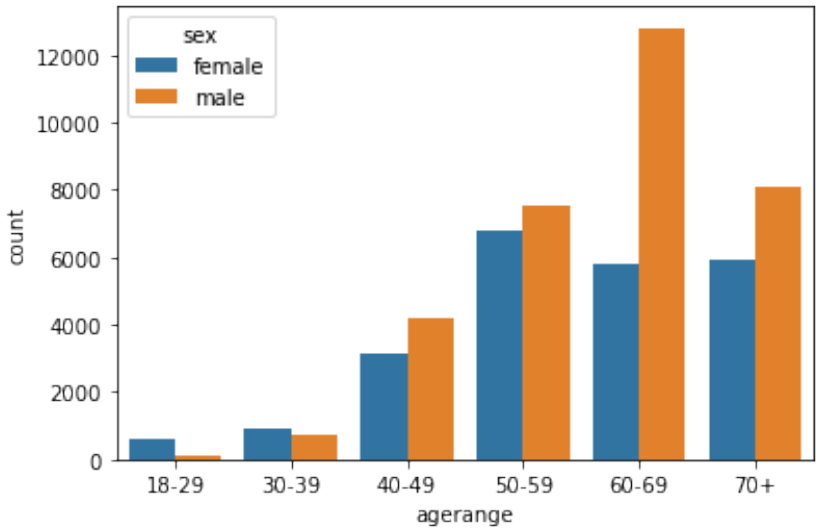

Figura 4.1: Género y edad de los participantes

En el cuadro 4.1 se realiza un desglose por *WAB-R's Aphasia Quotient (AQ)*, indicador de la gravedad general de la discapacidad del lenguaje que posee el paciente. Como puede observarse, el dataset está compuesto por un porcentaje alto de pacientes con una afasia moderada y suave:

Aphasia Quotient	Nº Pacientes	%
Mild	189	43.4 %
Moderate	142	32.6 %
Severe	40	9.3 %
Very Severe	15	3.4 %
Unknown	49	11.3 %

Cuadro 4.1: Aphasia Quotient

Por el tipo de afasia que poseen los pacientes se puede determinar la fluidez en el habla. El dataset está compuesto por un 61 % de pacientes con fluidez en el habla y un 39 % sin fluidez.

Este dataset incluye por 96 horas de audios, 78.031 transcripciones compuestas por un total de 469.207 palabras y 8.505 palabras únicas.

4.2. Preprocesamiento y limpieza de los datos

Las transcripciones recopiladas en formato CHAT contienen registros de una conversación entre el paciente y el investigador. Además, se halla información sobre el paciente (edad, genero,

grado de afasia, entre otros) y sobre el investigador. Para la limpieza de las transcripciones se ha utilizado la librería de Python *PyLangAcq* [2].

Este preprocesamiento y limpieza se comenzó con la lectura de los archivos y extracción de métricas de interés. Para ello fur creado un dataframe en Python donde se almacena la siguiente información:

- mark_start: marca temporal en milisegundos en la que comienza el paciente de hablar.
- mark_end: marca temporal en milisegundos en la que termina el paciente de hablar.
- transcription: las transcripciones correspondientes a la frase del paciente.
- sex: género del paciente.
- age: edad del paciente.
- file: el nombre del archivo de vídeo correspondiente a la entrevista realizada al paciente.
- WAB_AQ: promedio ponderado de todas las puntuaciones de las subpruebas relacionadas con el lenguaje hablado, que mide la capacidad del lenguaje. Esta ponderación se muestra en la tabla 4.2:

AQ score	Tipo de Severidad
0-25	Very severe
26-50	Severe
51-75	Moderate
76+	Mild

Cuadro 4.2: Tabla de Taxonomía del WAB (Western Aphasia Battery) [37]

.

- aphasia_type: diferentes tipos de afasia como afasia global, de Broca o de Wernicke.

A continuación, se obtuvieron nuevas métricas a partir de las anteriores. Estas nuevas métricas fueron la siguientes:

- file_cut: corresponde al nombre del audio, el segundo exacto en el que comienza el paciente a hablar y duración de este. Tendrá la forma *nombre archivo_inicio_duración*. Esta métrica sirve como identificador único de cada fragmento de audio en el que el paciente habla. Posteriormente este audio se ha utilizado en el entrenamiento del algoritmo para la llamada a los archivos de audio correspondientes.

[2] https://pylangacq.org/

- WAB_AQ_category: basado en la tabla 4.2 tipo de severidad que posee cada paciente según su AQ score.

- fluency_speech: categoría sobre la fluidez del habla en afasia, su clasificación se muestra en la tabla 4.3:

Fluidez	Tipo de afasia
Fluidez	Wrnicke's aphasia Trasncortical sensory aphasia Conduction aphasia Anomic aphasia
No Fluidez	Global aphasia Broca's aphasia Transcortical motor aphasia Mixed transcortical aphasia

Cuadro 4.3: Tipos de afasia [34]

.

Los procesos de limpieza y preprocesamiento de los datos que se realizaron fueron los siguientes:

1. Búsqueda de frases compuestas por *"x x x"* , estas son transcripciones de frases o palabras ininteligibles. Se sustituyeron por espacios vacíos.

2. Se observa que las transcripciones contiene caracteres especiales como *,.?!;:* que se han sustituido por espacios vacíos porque no corresponden a ninguna unidad del sonido en particular. No se ha eliminado el caracter *'*, ya que en inglés hay frases donde se utilizan y el sonido es diferente en función de su uso. El uso de apóstrofe en inglés refleja la abreviatura de diferentes verbos, con lo cual el sonido primigenio anterior al apóstrofe se vería afectado por la terminación del verbo siguiente.

3. Además, el texto de las transcripciones se ha normalizado para que todas las palabras estén en minúsculas. Este paso es necesario ya que usar una letra mayúscula al principio de una frase es una regla gramatical, pero no refleja la oralidad del interlocutor. El uso de mayúsculas hace que el modelo tenga menos facilidad para aprender a transcribir oraciones.

4. Se hallaron transcripciones que contenían caracteres fonéticos como *æðə*. Se eliminaron estas transcripciones y sus respectivos audios del estudio por la idiosincrasia y similaridad fonética del sonido.

5. Se eliminaron las transcripciones que solo tuvieran espacios vacíos, es decir, valores NaN del dataset.

6. La transcripción de la grabación *adler07a* contiene transcripciones que no corresponden con el audio. Para su limpieza, se filtraron aquellas que tuvieran su marca de comienzo mayor a la duración del audio. En este caso el audio *adler07a* tiene una duración de 1.036.208 milisegundos, es decir, 17:27 minutos. Por lo tanto, se eliminó toda transcripción de este subdataset en concreto que empiece más tarde del minuto 17:27 minutos.

7. Se separó el dataset en dos diferentes. Uno de ellos está formado por audios mayores o iguales de 25 segundos y otro con el resto. Esta separación se realiza por la falta de memoria de GPU de entrenamiento con pistas de audios de larga duración. Para entrenar el dataset, se ha utilizado audios menores de 25 segundos.

En la tabla 4.4 se puede observar cómo han ido disminuyendo el número de transcripciones y horas de audios por cada paso realizado en la limpieza de la base de datos:

Steps	Nº Transcripciones	Horas de audio
Dataset Original	78.031	116.0
Limpieza de transcripciones con caracteres fonéticos	77.885	115.7
Transcripciones vacías	67.887	91.5
Limpieza de transcripciones no validas de *adler07a*	51.889	67.1
Filtrar audios menores de 25 segundos	51.499	63.6

Cuadro 4.4: Número de transcripciones y horas de audio

A continuación, se generó un script para la extracción del audio en formato PCM WAV desde los videos MP4 con una frecuencia de muestreo de 16000Hz. Para ello, se ha utilizado el programa *FFmpeg* [3], un software gratuito de código abierto diseñado para el procesamiento de archivos de audio y vídeo basado en líneas de comandos [12]. Mediante la siguiente llamada se realizó la extracción de los audios:

```
ffmpeg -i {file} -ar 16000 -ac 1 {file[:-4]}.wav
```

[3] https://www.ffmpeg.org/

Tras la conversión de los vídeos a audio, se creó un script que genera los cortes de audios correspondientes a las transcripciones. Esto fue posible gracias a la herramienta *sox* [4], un software libre de edición de audio multiplataforma, en la que se ha utilizado para ello el parámetro *trim*. Para generar los cortes es necesario indicar donde comienza el fragmento de audio y duración de este en segundos, de tal manera:

```
sox {file[0]} {file[0][:-4]}_{start}_{duration}.wav trim {start} {duration}
```

En este código se puede observar como el nuevo archivo que se ha creado tiene la misma estructura que la métrica *file_cut* la cual se generó con anterioridad para poder alinear el fragmento de audio con su transcripción durante el entrenamiento del modelo.

Una vez se ha preparado y personalizado los hiperparámetros del modelo y el dataset esta preprocesado y limpio, se realizaron entrenamientos del algoritmo.

[4] http://sox.sourceforge.net/sox.html

Capítulo 5

Experimentos y resultados

5.1. Descripción general

A lo largo de esta investigación, y con el objetivo de lograr la mejor precisión posible, ha sido necesario llevar a cabo una serie de experimentos con el algoritmo *XLSR-Wav2vec2.0* para optimizar los resultados. En este apartado del proyecto se exponen y detallan cada uno de los entrenamientos realizados sobre la base de datos AphasiaBank.

Los primeros experimentos que se realizaron, se han denominado "experimentos iniciales", estos han servido como punto de partida para analizar los diferentes parámetros, testar los scripts que se han programado, identificar errores en el código y outliers en los datos procesados. Tras esto, y con unas primeras conclusiones, se ha podido realizar los llamados "experimentos usando Grid Search" con el fin de mejorar los resultados de los llamados "experimentos iniciales". Estos "experimentos usando Grid Search" se han dividido, a su vez, en dos conjuntos de experimentos: un primer grupo basado en un learning rate con un warm-up lineal donde el paso crece hasta un valor y luego tiene un decaimiento lineal hasta otro valor mínimo; y un segundo experimento con un learning rate de coseno donde el paso crece de forma similar al learning rate lineal pero el decaimiento es con forma de coseno.

En cada uno de estos experimentos se ha utilizado la tasa de error de palabras y la tasa de error de caracteres, y que han sido definidas con anterioridad. Estas han servido para concluir una determinada precisión del algoritmo. En el siguiente apartado se detallan tanto los parámetros utilizados como las conclusiones obtenidas de estos experimentos.

5.2. Experimentos iniciales

Los experimentos iniciales se han realizado con el objetivo de analizar el comportamiento del algoritmo sobre el dataset así como, definir los hiperparámetros con mayor representación en

el modelo. Estos experimentos se han entrenado teniendo en cuenta una serie de pasos previos de limpieza y preprocesamiento de la base de datos.

La estructura utilizada para la puesta en marcha de estos experimentos iniciales partió de unos parámetros que se mantienen fijos durante el resto de entrenamientos que se han llevado a cabo a lo largo de este proyecto. A continuación, en la tabla 5.1 se muestran los parámetros:

Hiperparámetros	Valor
Base Model	wav2vec2-large-xlsr-53
Feat Proj dropout	0.2
Layer dropout	0.05
Acumm steps	2
Mask time	0.1
Hidden dropout	0.04
Activation dropout	0.07
Attention dropout	0.07
FP16	True
Learning rate scheduler type	linear
Learning rate	0.0004
Epochs	30

Cuadro 5.1

Esta elección de hiperparámetros iniciales proviene del conocimiento adquirido durante el evento de fine-tuning week de *Hugging Face*. Tras la definición de los hiperparámetros se ha procedido a la selección de aquellos que varían de un experimento a otro. A continuación se de tallan los parámetros que van a ser variables:

- Experimento: versión la base de datos de afasia se ha limpiado y preprocesado en diferentes pasos.
- Detalles: detalles sobre la base de datos.
- Train: el número de registros usados para el entrenamiento.
- Test: el número de registros usados para el testeo.
- Filter: filtros aplicados a la duración de los fragmentos de audios
- Batch size: 2 u 8

El primer experimento inicial, llamado V1, contaba con un database al cual se le había hecho un primer preprocesamiento. Este contenía aun algunos caracteres extraños, fonemas, audios vacíos y espacios al final de cada transcripción. La división de este database se hizo con 30.000· datos para train y 10.000 para test. Se filtraron audios de menos de 20 segundos para evitar una posible falta de memoria en las tarjetas gráficas. El número de épocas en este experimento fue de 30, el batch size de 8, learning rate igual a 0.0004 y el tipo de learning rate lineal.

El segundo experimento inicial, o V2, contaba con un segundo preprocesamiento y limpieza de los datos. Aun así, la base de datos todavía contenía ciertos caracteres , audios vacíos y espacios al final de las transcripciones. La división en este caso fue de 10.000 datos de train y 2.000 para test. El resto de parámetros continuaron de forma análoga.

En el tercer experimento el dataset aun contenía caracteres y espacios en blanco que debían ser eliminados. El resto de parámetros se mantuvieron igual que en el experimento V2.

Por último, en el experimento V4 la base de datos estaba totalmente preprocesada y limpia. Se hizo una separación de un 80 % para los datos en train (esto equivale 45.062 registros) y un 20 % para test (11.266 registros). A diferencia de los anteriores experimentos se aplicó un filtro en el dataset donde se descartaban audios mayores de 25 segundos y un batch size de 8.

5.2.1. Análisis y discusión

Los reusltados de WER y CER obtenidos en los experimentos iniciales y la información de cada uno de ellos, se recoge en las tablas 5.2 y 5.3 como se puede ver a continuación:

Experimentos Iniciales Detalles	
Experimento	Detalles Dataset
V1	Caracteres, Fonemas, audios vacíos, espacios al final
V2	Pocos caracteres, audios vacíos, espacios al final
V3	Pocos caracteres, audios vacíos
V4	Limpio

Cuadro 5.2: Destalles de los Experimentos iniciales

Experimentos Iniciales Resultados						
Experimento	Train	Test	Filter	Batch size	WER	CER
V1	30.000	3.000	$< 20s$	8	43.9	-
V2	30.000	3.000	$< 20s$	2	55.3	34.4
V3	30.000	3.000	$< 20s$	2	54.4	33.9
V4	41.199	10.299	$< 25s$	8	35.8	22.4

Cuadro 5.3: Resultado de los Experimentos iniciales

El experimento V1 ha sido el primero realizado con la base de datos de afasia sin llevar a cabo ningún proceso de limpieza anterior. La finalidad de este era observar si el modelo obtenía un WER inferior a 100 %. Esto ayudó a demostrar que el algoritmo *XLSR-Wav2vec2.0* consigue aprender de esta base de datos.

Una vez realizado el experimento el resultado obtenido en torno a este parámetro WER ha sido de un 43.9 %. El porcentaje obtenido confirma que es posible aplicar el modelo en esta base de datos y que, además, hay un gran potencial de mejora.

En los experimentos V2 y V3 se han utilizado los mismos hiperparámetros y la misma proporción de datos tanto para el entrenamiento como para el test. La única diferencia entre ambos experimentos ha sido la limpieza del dataset. Para el experimento V3 se utilizó el dataset con un preprocesamiento y limpieza mayor que para el experimento V2.

Se observa como el experimento V3 tiene un WER y un CER menor que el V2, 54.4 % de WER frente a 55.3 % del experimento V2. Con lo cual, se pudo inferir que tener un dataset más limpio permite al modelo entrenar mejor y que, a su vez, obtenga mejores resultados.

Por último, el experimento V4 fue el realizado con el dataset totalmente limpio, con una división en los datos de train y test basados en un 80 %/20 % y un batch size de 8. Tanto el WER como CER obtenido son mas bajos que en los experimentos anteriores. En concreto, en este experimento se ha conseguido una mejora del WER de un 18 % respecto al primero (V1).

Gracias a estos primeros experimentos se ha podido concluir que tanto la calidad de las transcripciones como la cantidad de datos que usen el modelo para entrenar tienen relación directa con el WER y CER obtenidos.

5.3. Experimentos usando Grid Search

Los siguientes experimentos realizados son los mencionados anteriormente como 'Experimientos usasando Grid Search'. Grid Search es una técnica de ajuste que intenta calcular los valores óptimos de los hiperparámetros haciendo una búsqueda exhaustiva. Esta búsqueda se

realiza sobre un rango de valores de los parámetros específicos de un modelo [11].

Esta técnica se ha utilizado en este estudio para simplificar y optimizar la selección de los hiperparámetros. Un hiperparámetro es un conjunto de valores que se preconfiguran antes del entrenamiento del modelo. Estos valores no son innatos de los datos analizados.

Se han realizado 5 experimentos a los cuales se le han mantenido una serie de hiperparámetros fijos. A continuación se detalla el valor asociado a cada uno de ellos:

- *Base Model* = wav2vec2-large-xlsr-53
- *FP16* = True
- *Optimizador* = Adam
- *Batch_size* = 2
- *Epochs* = 10
- *Warmup_ratio* = 0.1
- *Train* = 60 % de los datos. Se establece una semilla para que sea el mismo conjunto de datos de entrenamiento en cada experimento.
- *Test* = 25 % de los datos. Se establece una semilla para que sea el mismo conjunto de datos de test en cada experimento.
- *Eval* = 15 % de los datos. Se establece una semilla para que sea el mismo conjunto de datos de test en cada experimento.

La nomenclatura utilizada para nombrar cada experimento realizado es la siguiente:

Nombre del experimento + número del experimento	Nomenclatura
Experimento 1	EXP.1

Cuadro 5.4: Nomenclatura utilizada para los experimentos

5.3.1. Experimentos con learning rate lineal

Elección de hiperparámetros

Los hiperparámetros que han sido optimizados mediante la técnica de Grid Search y sus respectivos intervalos de valores han sido los siguientes:

- Activation_dropout: distribución log-uniforme (-3.9, -1.9)

- Attetion_dropout: distribución log-uniforme (-3.9, -0.7)
- Feat_proj_dropout: distribución log-uniforme (-3.9, -0.7)
- Gradient_accumulation_steps: 2 ó 4
- Hidden_dropout: distribución log-uniforme (-3.9, -1.9)
- Layerdrop: distribución log-uniforme (- 4.6, -1.6)
- Learning_rate: distribución log-uniforme (-9.2, -6.9)
- Mask_time_prob: distribución log-uniforme (-3.9, -1.6)

A continuación en la tabla 5.5, se recoge las combinaciones de hiperparámetros distintas que se han seleccionado en los 5 experimentos realizados:

Hiperparámetros	EXP.1	EXP.2	EXP.3	EXP.4	EXP.5
Activation_dropout	$2.6x10^{-2}$	$3.5x10^{-2}$	$2.2x10^{-2}$	$1.1x10^{-1}$	$3.7x10^{-2}$
Attention_dropout	$3.7x10^{-2}$	$3.2x10^{-2}$	$2.5x10^{-2}$	$3.2x10^{-1}$	$1.5x10^{-1}$
Feat_proj_dropout	$5.6x10^{-2}$	$8.9x10^{-2}$	$3.2x10^{-2}$	$2x10^{-1}$	$5.8x10^{-2}$
Gradient_accumulation_steps	2	4	2	4	2
Hidden_dropout	$3x10^{-2}$	$4x10^{-2}$	$4x10^{-2}$	$7.8x10^{-2}$	$3.3x10^{-2}$
Layerdrop	$1.4x10^{-2}$	$2x10^{-2}$	$5.6x10^{-2}$	$1.6x10^{-1}$	$1.1x10^{-2}$
Learning_rate	$1.4x10^{-4}$	$3.1x10^{-4}$	$3.8x10^{-4}$	$6x10^{-4}$	$7.1x10^{-4}$
Mask_time_prob	$5.7x10^{-2}$	$4.7x10^{-2}$	$9.5x10^{-2}$	$3.3x10^{-2}$	$1.9x10^{-1}$

Cuadro 5.5: Valores de los hiperparámetros

Con la realización de estos experimentos se ha podido generar una clasificación de qué hiperparámetros son más importantes en los entrenamientos y su correlación con respecto al WER y CER obtenidos. La importancia muestra el grado en que cada hiperparámetros fue útil para predecir la métrica elegida y las correlaciones son relaciones lineales entre hiperparámetros individuales y valores métricos. En la siguiente tabla se recoge la importancia y correlación de los hiperparámetros de los 5 experimentos realizados sobre las métricas WER y CER.

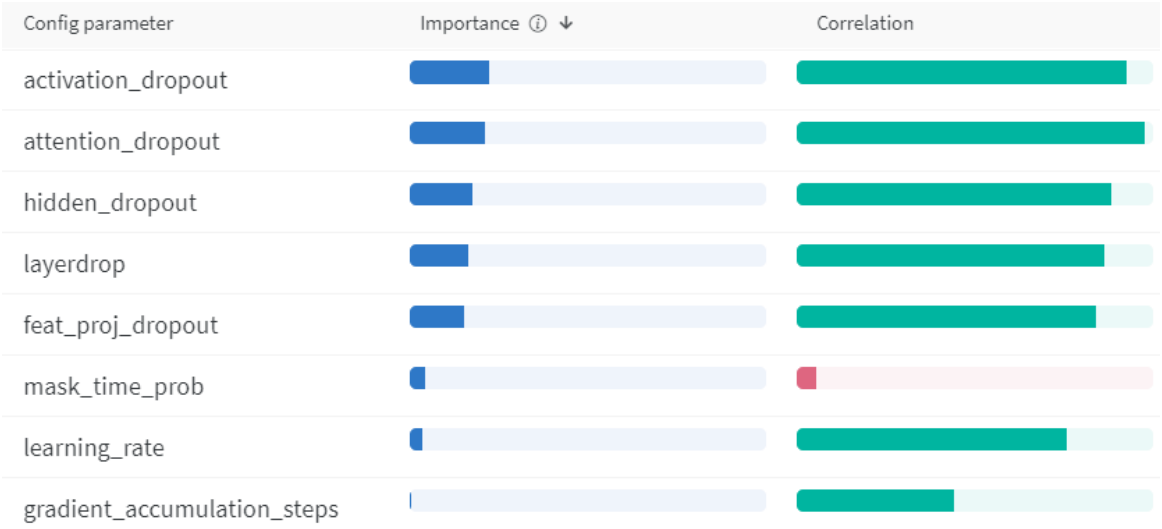

Figura 5.1: Importancia y correlación de los hiperparámetros sobre la métrica WER

En esta primera figura 5.1 se observa que los hiperparámetros con mayor grado de importancia son activation_droput, acttention_dropout, hidden_dropout, layerdrop y feat_proj_dropout. Por otro lado, el hiperparámetro mask_time_prob es el único con una correlación negativa sobre el modelo, el resto tiene una correlación positiva y muy alta con la métrica WER. Esto quiere decir que si variamos estos parámetros nuestro experimento se verá más afectado que por otros del modelo.

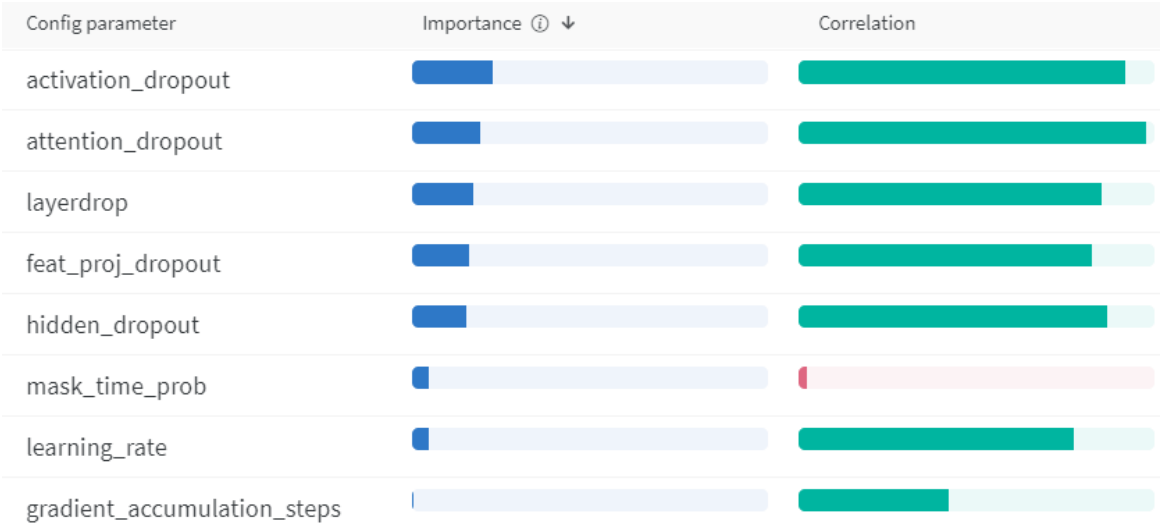

Figura 5.2: Importancia y correlación de los hiperparámetros sobre la métrica CER

En la figura 5.2, se puede observar la importancia y la correlación de los hiperparámetros con la métrica CER. Tanto la métrica CER como la WER tienen resultados muy similares. Los hiperparámetros con una alta correlación positiva, ordenados de mayor a menor importancia,

son: activation_dropout, attention_dropout, layerdrop, feat_proj_dropout y hidde_dropout. Por el contrario, el hiperparámetro mask_time_prob es el único con una correlación negativa.

Comportamiento del hiperparámetro Learning Rate

En esta sección se analiza el learning rate obtenido en cada uno de los experimentos. Learning rate o velocidad de aprendizaje en español, es uno de los hiperparámetros que define los pesos en cada iteración durante el entrenamiento, estos pesos se deben variar para poder dar cada vez una mejor aproximación [7].

Como se observa en al figura 5.3 se ha comenzado con un warmup lineal en todos los experimentos y tras ello comienzan a descender de forma lineal. El experimento 1 es el que mejor learning rate se ha conseguido ajustar a lo largo del experimento .

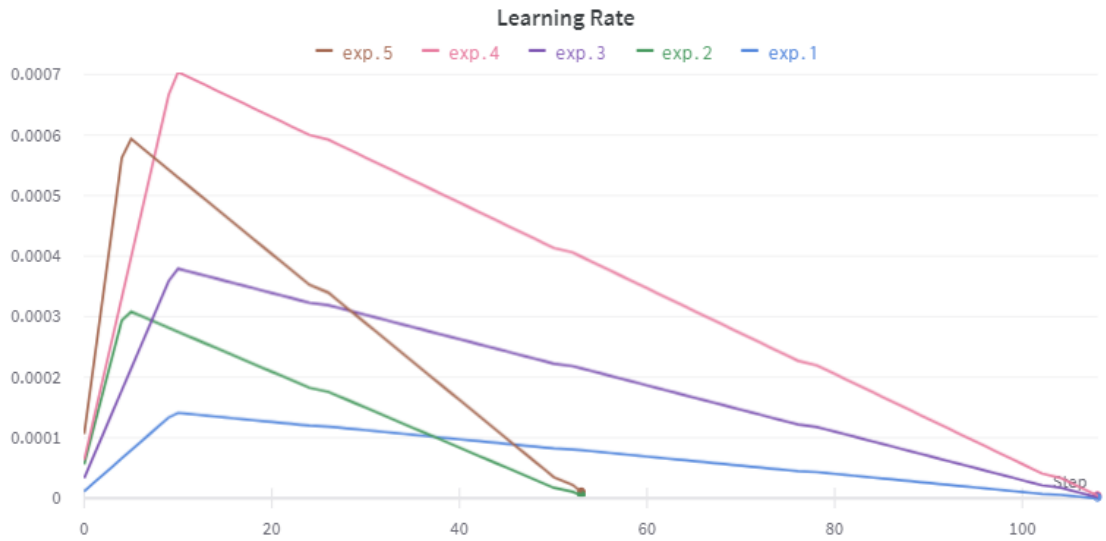

Figura 5.3: Convergencia de learning rate

Como se puede observar en las figuras 5.1 y 5.2, en estos experimentos el hiperparámetro *learning_rate* no es el que mas importancia tiene, aun así, es clave que su decrecimiento sea proporcional a lo largo de las iteraciones realizadas en el experimento.

5.3.1.1. Resultados WER y CER

Una vez entrenados los diferentes experimentos con los hiperparámetros detallados en la tabla 5.4 se han obtenido diferentes resultados.

Como se puede observar en el cuadro 5.5 el mejor WER y CER obtenido fue en el experimento 1 con 30.8 % y 19.27 % respectivamente. El experimento 4 es el que peor WER y CER ha obtenido, esto se debe a los valores de los hiperparámetros elegidos que más importancia y

correlación tienen en el modelo, detallados en las figuras 5.1 y 5.2. Tomando como referencia estos hiperparámetros, en la tabla 5.6, se puede ver como los valores elegidos para el experimento 4 son muy diferentes a los elegidos en el experimento 1.

Experimentos	WER	CER
EXP.1	**0.3082**	**0.1927**
EXP.2	0.3366	0.2058
EXP.3	0.3796	0.2295
EXP.4	0.6776	0.3761
EXP.5	0.4718	0.2785

Cuadro 5.6: WER y CER por experimento

En las siguientes figuras, figura 5.4 y 5.5, se puede observar el comportamiento de la precisión de cada uno de los experimentos durante las iteraciones realizadas en el entrenamiento del modelo:

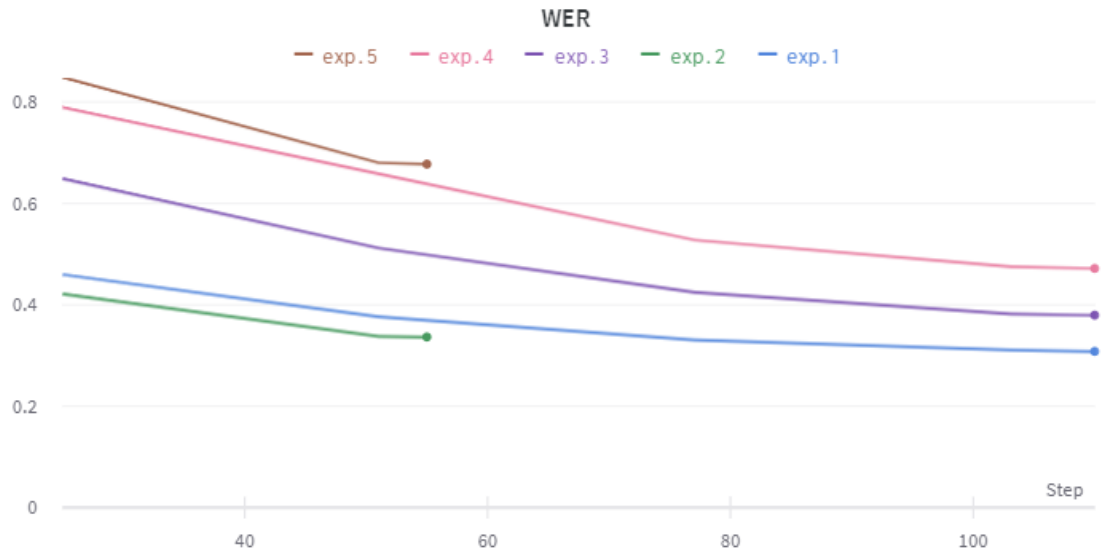

Figura 5.4: Curva WER durante las evaluaciones

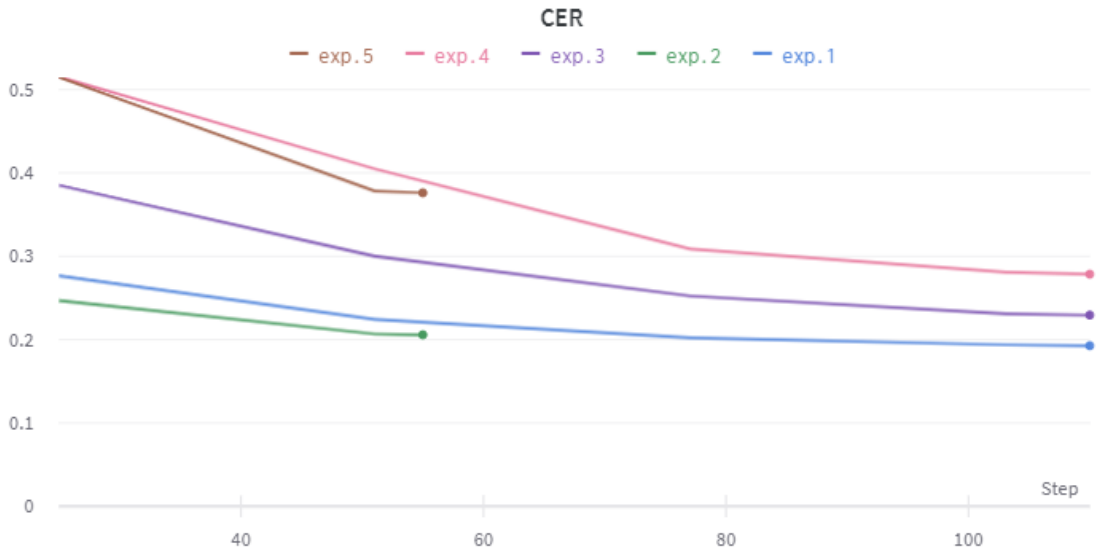

Figura 5.5: Curva CER durante las evaluaciones

En ambas figuras se observa que los modelos decrecen a mayor velocidad hasta llegar a la mitad del entrenamiento que es cuando el decrecimiento se ralentiza hasta casi estabilizarse.

5.3.1.2. Experimento con mejor resultado

En esta sección se analiza con más profundidad el experimento 1 que ha sido el que mejor resultados de WER y CER se ha obtenido.

Entrenar un modelo consiste en determinar valores correctos basándose en datos etiquetados. El objetivo entonces ha sido minimizar la pérdida, es decir, reducir el número de errores en las predicciones. A continuación, se va a mostrar el comportamiento del modelo durante el entrenamiento.

Con la combinación de hiperparámetros del experimento 1 se ha obtenido un 30.8 % de WER y 19.3 % de CER. Como se puede observar en la figura 5.6 tanto el WER como el CER se han ido mejorando a lo largo de todo el entrenamiento.

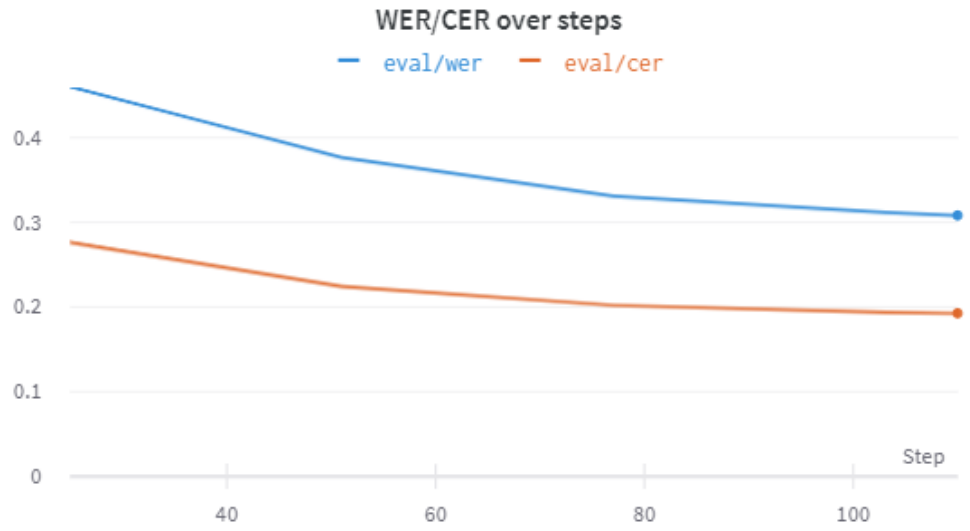

Figura 5.6: Curva WER/CER durante las evaluaciones. Experimento 1.

Se ha analizado el comportamiento de la variable pérdida durante el entrenamiento y evaluación. Con estas curvas de aprendizaje del modelo se podría inferir si se produce un sobre ajuste, un desajuste o e un buen ajuste en el modelo. En la figura 5.7 se observa como la perdida (loss) durante las evaluaciones es mayor que la perdida que se obtiene durante el entrenamiento. Esto significa que hay un sobre ajuste, es decir, el modelo aprende de los datos de entrenamiento y me memoriza. A pesar de haber un sobre ajuste el modelo sigue consiguiendo mejor precisión a lo largo de las iteraciones. Este comportamiento es algo que han reportado otros investigadores, por ejemplo en las discusiones surgidas durante la fine-tuning week.

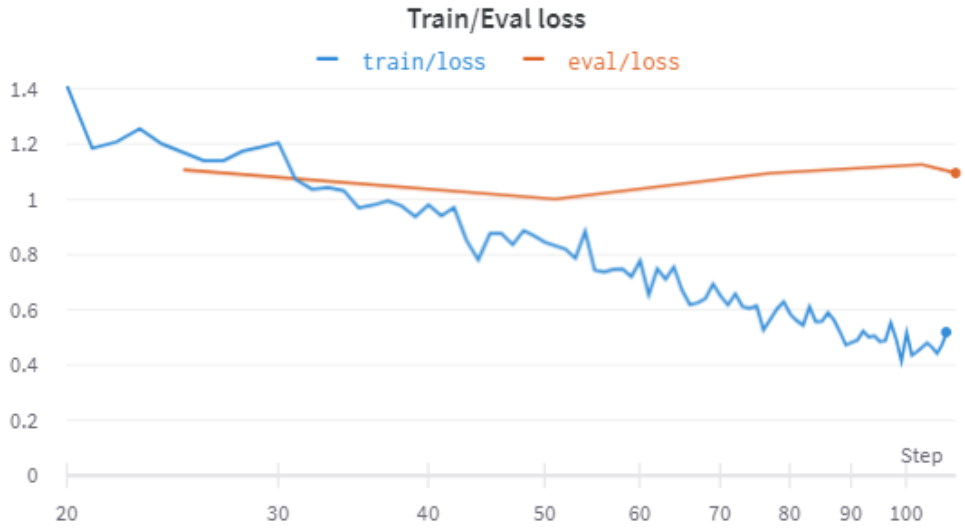

Figura 5.7: Curva de pérdida durante el entrenamiento. Experimento 1.

Ejemplos de Referencia-Evaluación

A continuación se muestran una serie de ejemplos de transcripciones originales y la predicción realizadas por el algoritmo:

Transcripción original	**Predicción**
"my speech better but it's hard the speech is not there"	"my speech better but it's hard the speech is o there"
"well it's a little boy who is kicking the soccer ball"	"well it's little boy who is kicking the soccer ball"
"they danced until twele o'clock"	"they danced till twele o'clock"
"but the sisters hated her because she was good"	"but the sisters hated her because she was good"
"i don't know what his name is right now"	"i don't kow what his name is right now"
"a lot of things that i hae that you talk to me are things that were funny things"	"a lot of things that i hae that you talk to me ar things that were funny things"

Cuadro 5.7: Ejemplos de transcripción y predicción

Como se puede observar en algunos casos la predicción es exactamente igual a pesar de que en algunos casos se encuentran algún error en palabras o caracteres.

5.3.2. Experimentos con learning rate de coseno

Elección de hiperparámetros

A continuación se detallan los hiperparámetros que han sido optimizados mediante la técnica de Grid Search y los intervalos de valores elegidos:

- *Activation_dropout*: distribución uniforme (0.01, 0.04)
- *Attetion_dropout*: distribución uniforme (0.01, 0.1)
- *Feat_proj_dropout*: distribución uniforme (0.01, 0.1)
- *Gradient_accumulation_steps*: 2, 3 ó 4
- *Hidden_dropout*: distribución uniforme (0.01, 0.05)
- *Layerdrop*: distribución uniforme (0.01, 0.1)
- *Learning_rate*: distribución uniforme (5×10^{-5}, 5×10^{-4})
- *Mask_time_prob*: distribución uniforme (0.03, 0.15)

En este conjunto se han realizado 12 experimentos distintos. Las tablas 5.8, 5.9 y 5.10 recogen las combinaciones de hiperparámetros escogidos en cada uno de los experimentos realizados:

Hiperparámetros	EXP.1	EXP.2	EXP.3	EXP.4
Activation_dropout	$1.1\text{x}10^{-2}$	$3.2\text{x}10^{-2}$	$3.1\text{x}10^{-2}$	$2.5\text{x}10^{-2}$
Attetion_dropout	$8.7\text{x}10^{-2}$	$3.5\text{x}10^{-2}$	$4.9\text{x}10^{-2}$	$8.8\text{x}10^{-2}$
Feat_proj_dropout	$7.8\text{x}10^{-2}$	$1.7\text{x}10^{-2}$	$4.5\text{x}10^{-2}$	$8.8\text{x}10^{-2}$
Gradient_accumulation_steps	4	3	3	2
Hidden_dropout	$2.2\text{x}10^{-2}$	$4.5\text{x}10^{-2}$	$3.6\text{x}10^{-2}$	$2.4\text{x}10^{-2}$
Layerdrop	$1.4\text{x}10^{-2}$	$4.3\text{x}10^{-2}$	$2.11\text{x}10^{-2}$	$3.3\text{x}10^{-2}$
Learning_rate	$1.3\text{x}10^{-4}$	$2.7\text{x}10^{-4}$	$4.1\text{x}10^{-4}$	$3.4\text{x}10^{-4}$
Mask_time_prob	$4.9\text{x}10^{-2}$	$1.4\text{x}10^{-2}$	$6.6\text{x}10^{-2}$	$5.7\text{x}10^{-2}$

Cuadro 5.8: Valores de los hiperparámetros. Experimentos 1-4.

Hiperparámetros	EXP.5	EXP.6	EXP.7	EXP.8
Activation_dropout	$3.2x10^{-2}$	$3.3x10^{-2}$	$3.8x10^{-2}$	$2.2x10^{-2}$
Attetion_dropout	$8.1x10^{-2}$	$8.7x10^{-2}$	$6.9x10^{-2}$	$9.7x10^{-2}$
Feat_proj_dropout	$8.6x10^{-2}$	$6.1x10^{-2}$	$5.6x10^{-2}$	$1.7x10^{-2}$
Gradient_accumulation_steps	2	4	3	2
Hidden_dropout	$3.6x10^{-2}$	$4.8x10^{-2}$	$3.2x10^{-2}$	$3.7x10^{-2}$
Layerdrop	$6.1x10^{-2}$	$6.4x10^{-2}$	$1.9x10^{-2}$	$4.2x10^{-2}$
Learning_rate	$4.7x10^{-4}$	$3.2x10^{-4}$	$4.3x10^{-4}$	$4.6x10^{-4}$
Mask_time_prob	$1.2x10^{-2}$	$4x10^{-2}$	$1.1x10^{-2}$	$1.4x10^{-2}$

Cuadro 5.9: Valores de los hiperparámetros. Experimentos 5-8.

Hiperparámetros	EXP.9	EXP.10	EXP.11	EXP.12
Activation_dropout	$1.6x10^{-2}$	$3.1x10^{-2}$	$1.1x10^{-2}$	$2.8x10^{-2}$
Attetion_dropout	$1.5x10^{-2}$	$1.9x10^{-2}$	$7.1x10^{-2}$	$1.4x10^{-2}$
Feat_proj_dropout	$5.5x10^{-2}$	$6.1x10^{-2}$	$8.6x10^{-2}$	$9.7x10^{-2}$
Gradient_accumulation_steps	3	2	3	4
Hidden_dropout	2.610^{-2}	$2.6x10^{-2}$	$1.3x10^{-2}$	$1.1x10^{-2}$
Layerdrop	$3.2x10^{-2}$	$6.4x10^{-2}$	$8.4x10^{-2}$	$2.4x10^{-2}$
Learning_rate	$1.5x10^{-4}$	$3.5x10^{-4}$	$2.3x10^{-4}$	$1x10^{-4}$
Mask_time_prob	$1.3x10^{-2}$	$1.1x10^{-2}$	$5x10^{-2}$	$8.8x10^{-2}$

Cuadro 5.10: Valores de los hiperparámetros. Experimentos 9-12.

En las siguientes figuras 5.8 y 5.9 se recogen la importancia y correlación de los hiperparámetros de los 12 experimentos realizados sobre las métricas WER y CER.

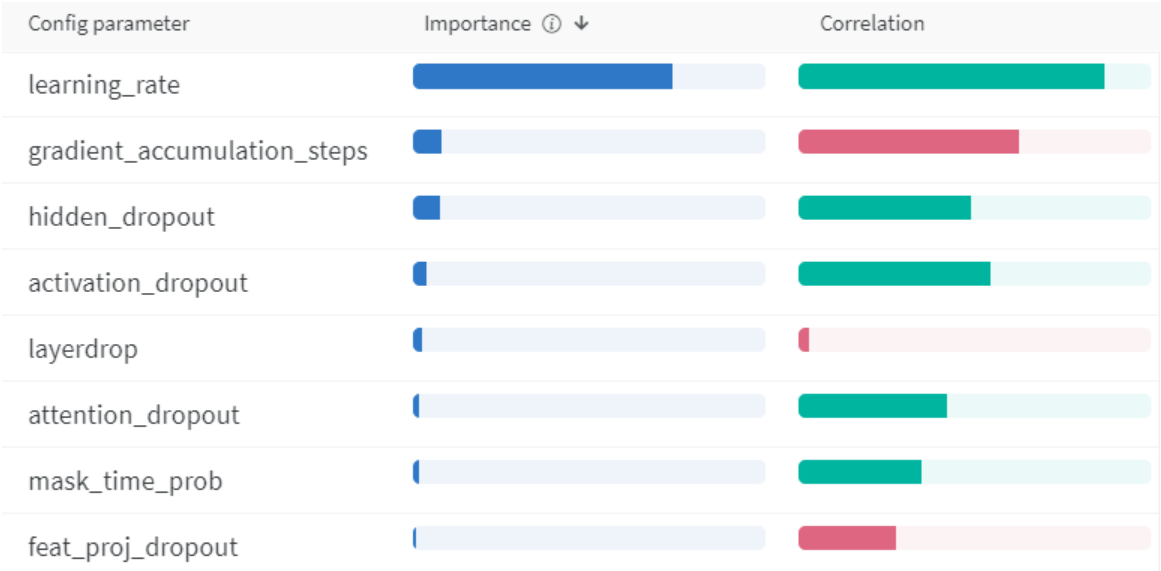

Figura 5.8: Importancia y correlación de los hiperparámetros sobre la métrica WER

En esta primera figura 5.9 se observa que que el hiperparámetro learning_rate es el que más importancia y correlación positiva tiene sobre la métrica WER en estos modelos. Por otro lado, el hiperparámetro gradient_accumulation_steps, feat_proj_dropout y layerdrop tienen una correlación negativa muy alta aunque no tengan una importancia muy alta.

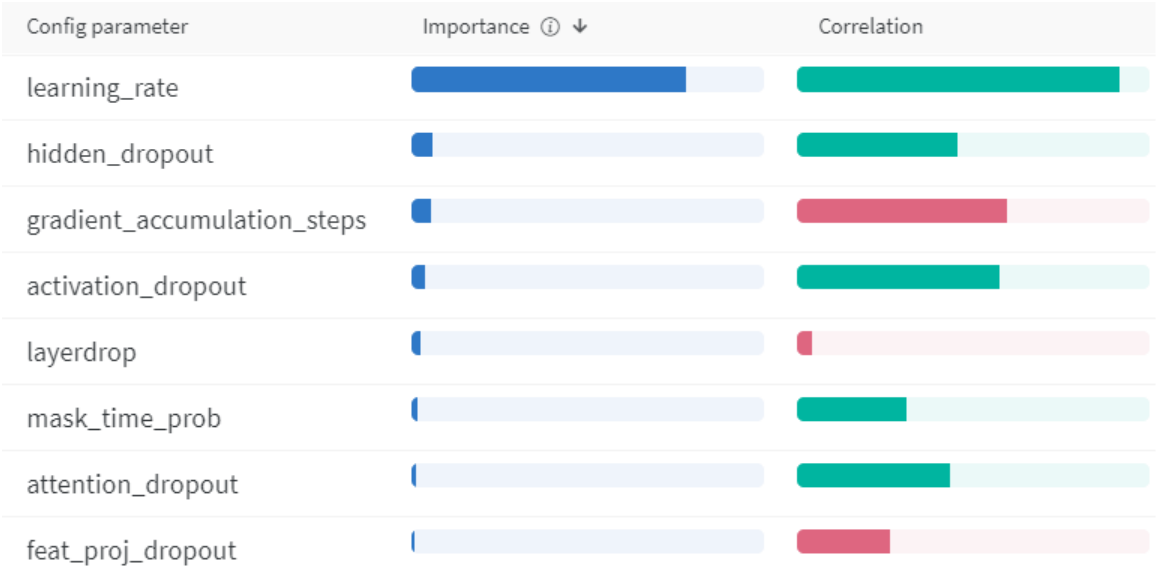

Figura 5.9: Importancia y correlación de los hiperparámetros sobre la métrica CER

La distribución de la importancia y correlación de los hiperparámetros sobre la métrica CER es análoga a la métrica WER.

Comportamiento del hiperparámetro Learning Rate

En esta sección se va a analizar el learning rate obtenido en cada uno de los experimentos. Se puede observar en la figura 5.10 que los experimentos comienzan con un warmup y tras ello comienzan a descender. El experimento 9 es el que mejor learning rate se ha conseguido ajustar a lo largo del experimento.

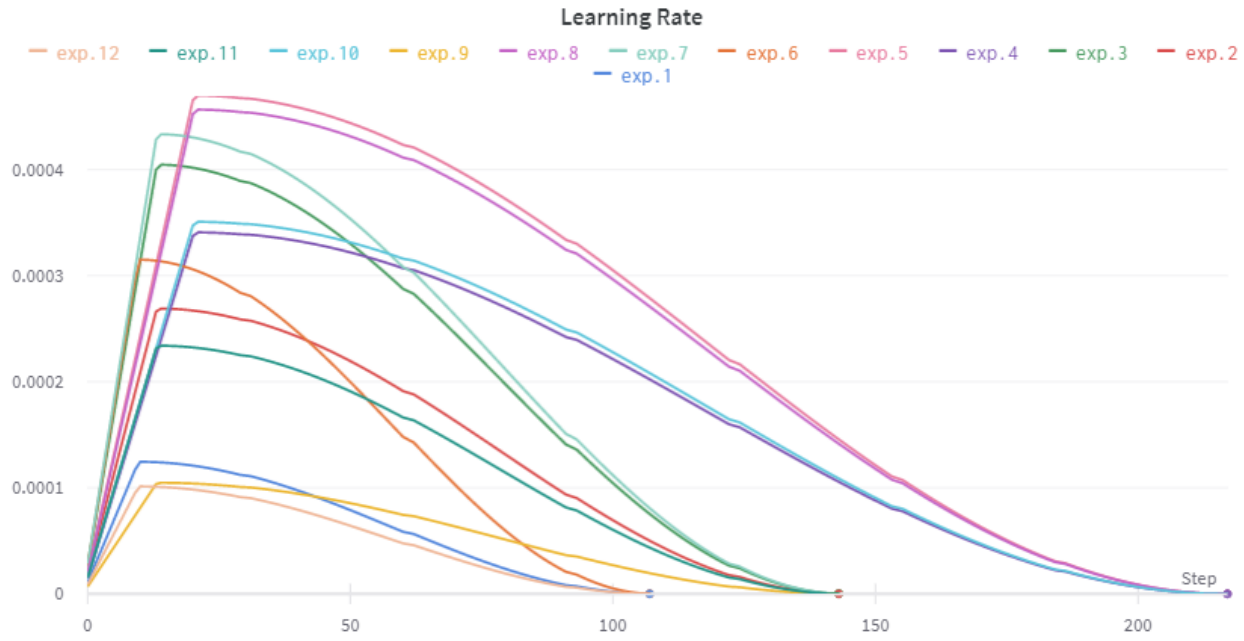

Figura 5.10: Convergencia de learning rate

Como se ha visto en la figura 5.8 y 5.9 el hiperparámetro learning_rate es el que mas importancia tiene en los experimentos, con lo cual, el que mejor ajuste tenga a lo largo de las iteraciones es el que mejor posiblemente mejor WER y CER pueda conseguir.

5.3.2.1. Resultados WER y CER

Una vez entrenados los 12 experimentos con los hiperparámetros detallados en las tablas 5.7, 5.8 y 5.9, se han obtenido diferentes resultados.

Como se puede observa en la tabla 5.11 el mejor WER y CER obtenido fue en el experimento 9 con 31.1 % y 18.9 % respectivamente. El resto de experimentos han obtenidos un WER y CER muy similares.

Experimentos	WER	CER
EXP.1	0.3147	0.1925
EXP.2	0.3396	0.2071
EXP.3	0.3777	0.2298
EXP.4	0.3558	0.2184
EXP.5	0.417	0.2505
EXP.6	0.3459	0.2109
EXP.7	0.3803	0.2314
EXP.8	0.401	0.2419
EXP.9	**0.3105**	**0.189**
EXP.10	0.3642	0.2224
EXP.11	0.3212	0.1981
EXP.12	0.3146	0.1926

Cuadro 5.11: WER y CER por experimento

En las gráficas 5.11 y 5.12 se representa el comportamiento de las métricas WER y CER durante el entrenamiento por cada uno de los experimentos realizados:

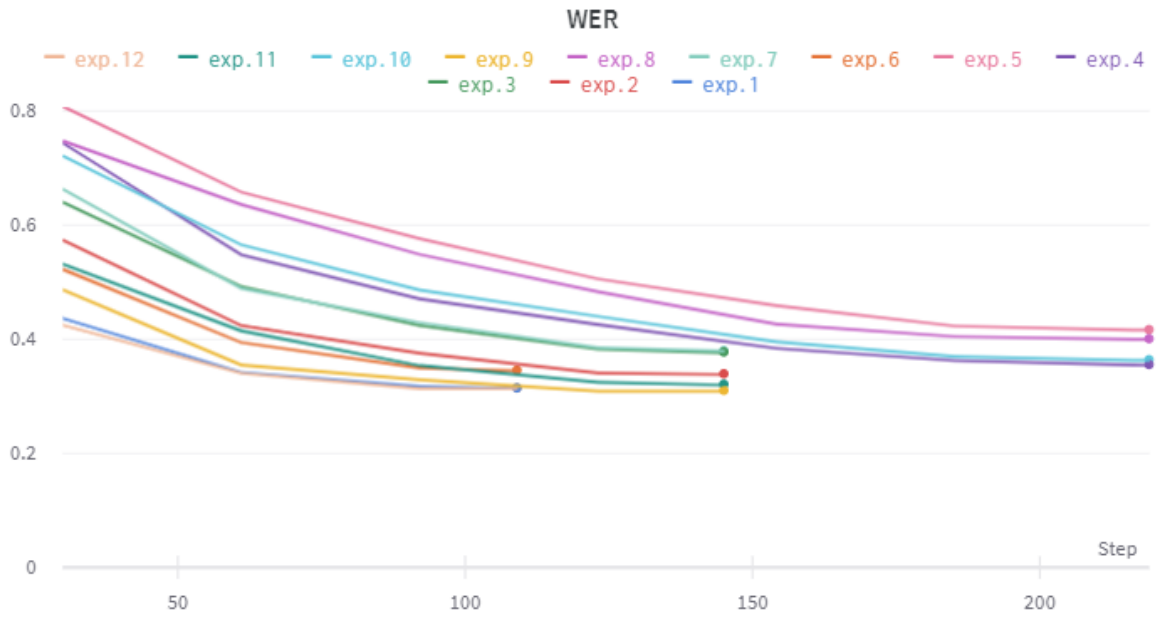

Figura 5.11: Curva WER durante las evaluaciones

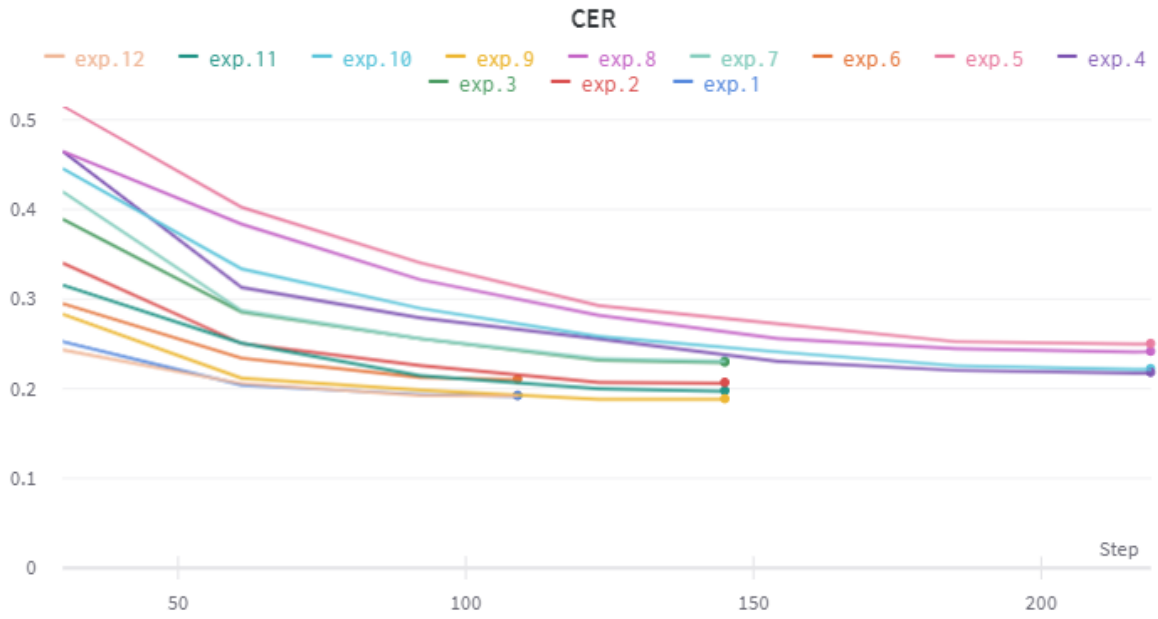

Figura 5.12: Curva CER durante las evaluaciones

Tanto en la gráfica 5.11 como en la gráfica 5.12 se observa que los modelos decrecen hasta llegar a la mitad del entrenamiento que comienzan a decrecer mas lentamente hasta casi estabilizarse.

5.3.2.2. Experimento con mejor resultado

En esta sección se va a analizar con más profundidad el experimento 9 que ha sido el que mejor resultados de WER y CER se ha obtenido.

Con la combinación de hiperparámetros del experimento 9 se ha obtenido un 31.1 % de WER y 18.9 % de CER. A continuación, se va a estudiar el comportamiento del modelo durante el entrenamiento. En la gráfica se observa el WER y CER obtenido en cada iteración.

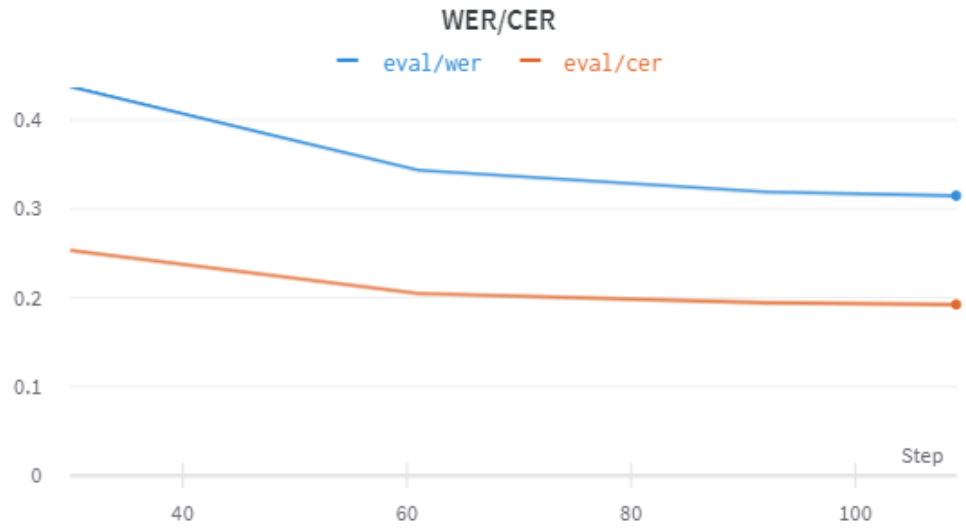

Figura 5.13: Curva WER/CER durante las evaluaciones. Experimento 9.

En la gráfica 5.14 aparece representada la función de perdida en el entrenamiento y evaluación. La perdida (loss) durante las evaluaciones es mayor que la perdida que se obtiene durante el entrenamiento, es decir, el modelo tiene un sobre ajuste de los datos. A pesar de esto el modelo consigue mejorar su precisión a lo largo de las iteraciones.

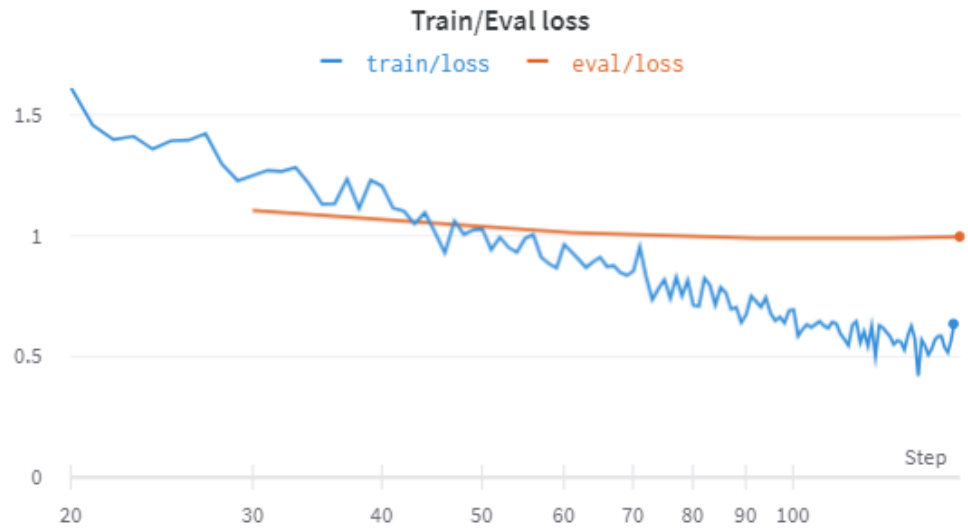

Figura 5.14: Curva de pérdida durante el entrenamiento. Experimento 9.

Capítulo 6

Conclusiones

6.1. Conclusiones

Los algoritmos semi-supervisados son un campo reciente de investigación y desarrollo en sistemas de reconocimiento de habla. Este trabajo presenta una descripción general del algoritmo semi-supervisado *Wav2vec2.0* y detalla una serie de experimentos que han permitido alcanzar los objetivos del estudio. *Wav2vec2.0*, hasta ahora, solo había sido aplicado a bases de datos de idiomas de bajos recursos, es decir, idiomas con poca cantidad de datos etiquetados. Sin embargo, este estudio logra aplicar este algoritmo en el reconocimiento del habla de personas con patologías que les afecta el habla, en concreto, a la afasia.

El primer objetivo de este trabajo ha sido realizar un estudio de la arquitectura del algoritmo *Wav2vec2.0* y analizar los hiperparámetros que lo conforman con el fin de encontrar la mejor combinación.

El segundo objetivo está vinculado a aplicar el algoritmo *XLSR-Wav2vec2.0* a la base de datos AphasiaBank. En este estudio, se demostró como el algoritmo *XLSR-Wav2vec2.0* consiguió no solo aprender de la base de datos de AphasiaBank sino también, obtener muy buenos resultados en las métricas WER y CER.

Comparando los resultados de este estudio con los datos obtenidos en la publicación [22] se observa que, al tomar como referencia la severidad más leve, es el tipo de afasia que mejor resultados puede dar en un modelo de RAH. Esto se debe a que este tipo de pacientes son los que menos dificultad tienen al hablar y por tanto, el algoritmo reconoce mejor.

El mejor resultado obtenido en el WER y CER en esta investigación es de un 30.82 % y un 19.27 % respectivamente en el modelo basado en learning rate lineal, mientras que el mejor resultado obtenido en la publicación tomada como referencia ha sido la media de test que tiene como resultado 47,41$\pm$10,46 % de PER . Aunque las métricas no son directamente comparables, el hecho de que el WER medio obtenido sea mejor que el PER de referencia en el caso más leve

de afasias muestra que en principio se han obtenido mejoras considerables respecto de cualquier otro resultado reportado anteriormente.

De todos los experimentos que se han realizado en este trabajo, se han podido extraer una serie de conclusiones sobre los hiperparámetros. En primer lugar, se demostró que al utilizar el learning rate de tipo coseno este hacia que el hiperparámetro learning_rate tuviera una correlación e importancia muy alta sobre el modelo. Por otro lado, en la figura 5.1 se observa que al seleccionar un learning rate lineal, los hiperparámetros *activationactivation_dropout*, *attention_dropout*, *hidden_dropout*, *layer* y *feat_proj_dropout* tienen una importancia y un peso muy similar entre ellos. Por ello, se deben tener más en cuenta a la hora de ajustar los hiperparámetros del modelo. Una última conclusión a destacar es que un modelo con más épocas tiene una mejor precisión en los resultados.

En lo referente a la base de datos, una de las conclusiones extraídas es que cuánto mayor sea la cantidad de datos y mejor sea su limpieza y procesamiento, mejores resultados de WER y CER se obtendrán. Esto es debido a la necesidad de este tipo de arquitecturas de disponer de un gran volumen de datos para obtener un óptimo rendimiento y comparable a sistemas tradicionales que requieren de menos cantidad de datos.

6.2. Trabajo futuro

Este proyecto demuestra el enorme potencial que tiene el algoritmo *Wav2vec2.0* en datos vinculados con problemas en el habla. Sin embargo, se pueden realizar todavía mejoras en la precisión obtenida en el modelo.

De cara a trabajos futuros es la posibilidad de aplicar *Wav2vec2.0* a otras patologías como Alzheimer o demencia, y compararlo con investigaciones hechas anteriormente.

Otro de los trabajos a futuro podría realizarse entrenamiento con épocas mayores a 10 como, por ejemplo, 100, 200 o 500 épocas ya que se ha demostrado que a mayor épocas mejor precisión se puede obtener.

En relación al corpus utilizado se pueden hacer diferentes mejoras y trabajos. Un primer trabajo sería realizar una división por la severidad de afasia del paciente. De esta forma se podría hacer una comparación más precisa de los datos publicados en la referencia [22]. En esta línea, se pretende continuar investigando sobre este tema, reentrenar un modelo con PER y publicar los resultados obtenidos en una revista científica especializada.

Por otro lado, como se mostró en la publicación del propio algoritmo [9], este se puede aplicar a diferentes idiomas y obtener muy buenos resultados. De forma análoga se podría aplicar este algoritmo a otros corpus como Español, Alemán, Francés... que se encuentran dentro de AphasiaBank.

Además, para una posible mejora del PER se podría entrenar al algoritmo sobre un nuevo dataset al cual se le ha realizado una conversión de las transcripciones de palabras a fonemas.

Cuando en el estudio se ha realizado la limpieza del dataset de AphasiaBank se ha realizado un filtro de audios de menos de 25 segundos ya que había falta de memoria en el servidor. A estos audios se les podría realizar un preprocesamiento más especifico para obtener cortes de menos duración de estos y así que sea viable su utilización dentro del modelo. En relación a la la división del dataset una investigación futura posible sería hacer un dataset de test con datos de pacientes que no se encuentren en el dataset de train y observar si el modelo consigue aprender de la misma forma que si tuviera datos de pacientes tanto en train como en test.

Finalmente, dado que se ha podido demostrarlas múltiples posibilidades que tienen este tipo de algoritmos, una de las aplicaciones futuras sería utilizar el nuevo algoritmo no supervisado *Wav2vec-U*[5] a la base de datos de AphasiaBank.

Bibliografía

[1] Disartria: MedlinePlus enciclopedia médica.

[2] Afshine Amidi,Shervine Amidi. A detailed example of how to generate your data in parallel with pytorch. https://stanford.edu/~shervine/blog/pytorch-how-to-generate-data-parallel, Last accessed on 2021-05-23.

[3] O Abdel-Hamid, A Mohamed, H Jiang, L Deng, G Penn, and D Yu. Convolutional Neural Networks for Speech Recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(10):1533–1545, 10 2014.

[4] AphasiaBank. Aphasiabank. https://aphasia.talkbank.org, Last accessed on 2021-05-22.

[5] Alexei Baevski, Wei-Ning Hsu, Alexis Conneau, Michael Auli Facebook, and A I Google. Unsupervised Speech Recognition. Technical report.

[6] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. Technical report.

[7] Anna Bosch, Rué Jordi Casas, and Roma Toni Lozano Bagén. *Deep learning Principios y fundamentos*.

[8] Census of India : General Note. General note. https://www.censusindia.gov.in/Census_Data_2001/Census_Data_Online/Language/gen_note.html, Last accessed on 2021-05-26.

[9] Alexis Conneau, Alexei Baevski, Ronan Collobert, Abdelrahman Mohamed, and Michael Auli Facebook. Unsupervised cross-lingual representation learning for speech recognition. Technical report.

[10] Li Deng, Geoffrey Hinton, and Brian Kingsbury. New types of deep neural network learning for speech recognition and related applications: an overview. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8599–8603, 2013.

[11] Farhad Malik. What is grid search? explaining how to obtain optimal hyperparameter values, 2020-02-18. https://medium.com/fintechexplained/what-is-grid-search-c01fe886ef0a, Last accessed on 2021-05-29.

[12] FFmpeg. About ffmpeg. http://ffmpeg.org/about.html, Last accessed on 2021-05-23.

[13] Gábor Gosztolya, Réka Balogh, Nóra Imre, José Vicente Egas-López, Ildikó Hoffmann, Veronika Vincze, László Tóth, Davangere P. Devanand, Magdolna Pákáski, and János Kálmán. Cross-lingual detection of mild cognitive impairment based on temporal parameters of spontaneous speech. *Computer Speech and Language*, 69:101215, 9 2021.

[14] Awni Hannun. Sequence Modeling with CTC. *Distill*, 2(11):e8, 12 2017.

[15] J Huang, J Li, and Y Gong. An analysis of convolutional neural networks for speech recognition. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4989–4993, 2015.

[16] IBM Shoebox. Ibm archives: Ibm shoebox. https://www.ibm.com/ibm/history/exhibits/specialprod1/specialprod1_7.html, Last accessed on 2021-05-26.

[17] Navdeep Jaitly. Natural Language Processing with Deep Learning CS224N/Ling284 Lecture 12: End-to-end models for Speech Processing. Technical report.

[18] Norezmi Jamal, Shahnoor Shanta, Farhanahani Mahmud, and Mohd Nurul Al Hafiz Sha'Abani. Automatic speech recognition (ASR) based approach for speech therapy of aphasic patients: A review. In *AIP Conference Proceedings*, volume 1883, page 20028, 2017.

[19] B H Juang and Lawrence R Rabiner. Automatic Speech Recognition-A Brief History of the Technology Development. Technical report, 2004.

[20] S Kim, T Hori, and S Watanabe. Joint CTC-attention based end-to-end speech recognition using multi-task learning. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4835–4839, 2017.

[21] Lasse Borgholt. Ctc networks and language models: Prefix beam search explained, 2018-01-08. https://medium.com/corti-ai/ctc-networks-and-language-models-prefix-beam-search-explained-c11d1ee23306, Last accessed on 2021-03-18.

[22] Duc Le and Emily Mower Provost. Improving Automatic Recognition of Aphasic Speech with AphasiaBank. Technical report.

[23] MATLAB Simulink. Redes neuronales convolucionales. https://es.mathworks.com/discovery/convolutional-neural-network-matlab.html, Last accessed on 2021-03-10.

[24] Maxime. What is a Transformer?. An Introduction to Transformers and... — Inside Machine learning — Medium.

[25] Navdeep Jaitly. Lecture 12: End-to-End Models for Speech Processing - YouTube.

[26] Yilin Pan, Bahman Mirheidari, Markus Reuber, Annalena Venneri, Daniel Blackburn, and Heidi Christensen. Improving detection of Alzheimer's Disease using automatic speech recognition to identify high-quality segments for more robust feature extraction. 2020.

[27] Patrick von Platen. Hugging face – on a mission to solve nlp, one commit at a time. https://huggingface.co/blog/fine-tune-xlsr-wav2vec2, Last accessed on 2021-05-28.

[28] A Pompili, A Abad, D M de Matos, and I P Martins. Pragmatic Aspects of Discourse Production for the Automatic Identification of Alzheimer's Disease. *IEEE Journal of Selected Topics in Signal Processing*, 14(2):261–271, 2 2020.

[29] Vineel Pratap, Qiantong Xu, Anuroop Sriram, Gabriel Synnaeve, and Ronan Collobert. Mls: A large-scale multilingual dataset for speech research. *Interspeech 2020*, Oct 2020.

[30] Ying Qin, Tan Lee, and Anthony Pak Hin Kong. Automatic assessment of speech impairment in cantonese-speaking people with aphasia. *IEEE Journal of Selected Topics in Signal Processing*, 14(2):331–345, 2020.

[31] scikit. Gaussian mixture models, 2018. https://scikit-learn.org/stable/modules/mixture.html#gaussian-mixture-models, Last accessed on 2021-03-03.

[32] G. F. Simon and C. D. Fennig. M. P. Lewis. Ethnologue: Languages of the world, nineteenth edition. Online version: http://www.ethnologue.com, 2016.

[33] Sundar V. Language Model like Pre-Training for Acoustic Data — Towards Data Science.

[34] The National Aphasia Association. The national aphasia association, 2018. https://www.aphasia.org/aphasia-definitions/, Last accessed on 2021-05-20.

[35] Jordi Torres. Redes neuronales artificiales y Deep Learning. In *Deep Learning: Introducción práctica con Keras (primera parte)*, pages 54,55. 2018.

[36] Trainer - transformers 4.5.0.dev0 documentation. Trainer - transformers 4.5.0.dev0 documentation. https://huggingface.co/transformers/main_classes/trainer.html, Last accessed on 2021-03-23.

[37] Vanessa Barfod, BA. Western aphasia battery (wab), 2013-07-06. https://strokengine.ca/en/assessments/western-aphasia-battery-wab/, Last accessed on 2021-05-20.

[38] Ashish Vaswani, Google Brain, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. Technical report.

[39] S Watanabe, T Hori, S Kim, J R Hershey, and T Hayashi. Hybrid CTC/Attention Architecture for End-to-End Speech Recognition. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1240–1253, 2017.

[40] Feifei Xiong, Jon Barker, Zhengjun Yue, and Heidi Christensen. Source domain data selection for improved transfer learning targeting dysarthric speech recognition. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7424–7428, 2020.

[41] Cheng Yi, Jianzhong Wang, Ning Cheng, Shiyu Zhou, and Bo Xu. Applying Wav2vec2.0 to speech recognition in various low-resource languages. Technical report.

[42] Cheng Yi, Shiyu Zhou, and Bo Xu. Efficiently Fusing Pretrained Acoustic and Linguistic Encoders for Low-resource Speech Recognition. Technical report, 2021.

[43] Zhengjun Yue, Feifei Xiong, Heidi Christensen, and Jon Barker. Exploring appropriate acoustic and language modelling choices for continuous dysarthric speech recognition. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6094–6098, 2020.

Anexos

Anexo 1

En el siguiente repositorio se encuentran los scripts utilizados tanto para el preprocesamiento y limpieza de los datos como el modelo utilizado para el entrenamiento del algoritmo.

Repositorio GitHub - TFM: Aplicación de métodos de aprendizaje semi-supervisados para el reconocimiento del habla en personas con afasia

Anexo 2

En el siguiente repositorio se encuentran los experimentos utilizados para la Hugging Face - Fine-Tuning week.

Repositorio GitHub - TFM: Hugging Face - Fine-Tuning week