ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

PROCESAMIENTO ACÚSTICO DE LA VOZ APLICADO AL RECONOCIMIENTO DE FONEMAS SIMILARES EN EL HABLA INGLESA COMPARADOS CON SUS HOMÓLOGOS EN LENGUA HISPÁNICA.

EVALUACIÓN DE MODELOS BASADOS EN TRANSFORMER PARA LA DETECCIÓN DE ERRORES DE LECTURA A NIVEL DE PALABRAS.

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN CIENCIAS DE LA COMPUTACIÓN

CARLOS ENRIQUE CALVA GALEAS

carlos.calva@epn.edu.ec

DIRECTOR: PhD. JOSAFA DE JESUS AGUIAR PONTES

josafa.aguiar@epn.edu.ec

DMQ, agosto 2023

CERTIFICACIONES

Yo, CARLOS ENRIQUE CALVA GALEAS declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

_____

CARLOS ENRIQUE CALVA GALEAS

Certifico que el presente trabajo de integración curricular fue desarrollado por CARLOS ENRIQUE CALVA GALEAS, bajo mi supervisión.

_____

PhD. JOSAFA DE JESUS AGUIAR PONTES

DIRECTOR DEL TIC

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

CARLOS ENRIQUE CALVA GALEAS

PhD. JOSAFA DE JESUS AGUIAR PONTES

DEDICATORIA

A mis padres y hermanos que me motivaron a no rendirme y me apoyaron incondicionalmente a lo largo de la carrera.

AGRADECIMIENTO

Agradezco en primer lugar a Dios, por permitirme haber llegado a cumplir esta meta.

Agradezco a mi padre Manuel Calva por brindarme tanto su apoyo económico como moral para seguir adelante en mi carrera.

Agradezco a mi madre Lupe Galeas por darme consejos de vida y estar pendiente de mi cada día.

Agradezco a mi hermana Jenniffer Calva y a mi cuñado Luis Montaleza por haberme brindado su apoyo incondicional.

Agradezco a mi tutor PhD. JOSAFA DE JESUS AGUIAR PONTES por su guía en las tareas que debía de realizar.

Agradezco a si mismo a la Escuela Politécnica Nacional por darme la oportunidad de desarrollarme en mi carrera y brindarme sus espacios de conocimiento.

Por último, agradezco a mis amigos que estuvieron conmigo en el transcurso de mi carrera. En especial, a mis amigos Hugo, José, Daniel, Henry y William.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

RESUMEN

Este trabajo trata acerca de la evaluación de errores lectura a nivel de palabras en grabaciones de audio del idioma español (Noticieros NHK en el idioma español) utilizando el modelo basado en Transformer: Whisper-Large-V2. Los resultados de la evaluación demostraron que el modelo puede predecir la mayoría de los errores correctamente. Adicionalmente se realizaron dos experimentos: el primero fue un intento de entrenar un modelo basado en Transformer usando el modelo Whisper-Small a través de la técnica del transfer learning para predecir la pronunciación a nivel fonético para el idioma inglés y el segundo experimento fue la evaluación de los errores de pronunciación a nivel de fonemas utilizando el modelo “wav2vec2-xlsr-53-espeak-cv-ft” para el idioma inglés hablado por personas cuya lengua materna es el español. Los dos experimentos no tuvieron resultados satisfactorios.

PALABRAS CLAVE: Whisper, wav2vec, Transformer, ASR, palabras, fonemas.

ABSTRACT

This study focuses on evaluating word-level reading errors in audio recordings of the Spanish language (NHK newscasts for Spanish) using the Transformer-based model: Whisper-Large-V2. The evaluation results demonstrated the model's capacity to accurately predict most errors. Additionally, two experiments were conducted: the first involved an attempt to train a Transformer-based model using the Whisper-Small model through the transfer learning technique to predict phonetic-level pronunciation for the English language. The second experiment entailed assessing phoneme-level pronunciation errors using the "wav2vec2-xlsr-53-espeak-cv-ft" model for English, spoken by individuals whose native language is Spanish. Unfortunately, both experiments yielded unsatisfactory outcomes.

KEYWORDS: Whisper, wav2vec, Transformer, ASR, words, phonemes.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

El componente desarrollado trata acerca de la evaluación de errores de lectura a nivel de palabras usando un modelo basado en Transformer llamado Whisper. Un Transformer es un tipo de arquitectura de red neuronal basado en mecanismos de atención y que ha demostrado ser efectivo en tareas de procesamiento de lenguaje natural (NLP) [9]. Así mismo, Whisper es un ASR basado en la arquitectura de Transformer entrenado con 680 horas de audio de varios idiomas y permite realizar tareas de transcripción de un audio en los idiomas entrenados, tareas de traducción de un audio al inglés, además de poder detectar el idioma de un audio [13]. Así mismo, Whisper tiene varios tipos de modelos, que, para esta evaluación, se utilizó el Whisper-Large-V2. Por otro lado, se utilizó como data, los Noticieros NHK en el idioma español, los cuales tienen grabaciones de audio del idioma español en formato de noticias. Además, este corpus cuenta con las transcripciones de las noticias leídas por los locutores guardadas en archivos TextGrid como con las noticias que debió de leer el locutor guardadas en archivos HTML. Cada archivo de la data se almacenó en listas y se realizó normalizaciones en las mismas. Sobre las grabaciones del audio, sirvieron como entrada para el modelo Whisper-Large-V2 y así obtener la lista de Predicciones para la evaluación. Los errores de lectura que se analizaron fueron 3: eliminaciones, sustituciones e inserciones de palabras. Para obtener los errores de lectura, se realizó alineaciones en las listas. Una alineación es el proceso de comparar dos secuencias y obtener la mayor semejanza entre las mismas. Por lo tanto, la alineación permitió obtener las posiciones donde se encuentran las coincidencias entre las secuencias, así como los patrones o caracteres que no coincidieron, es decir, los errores (eliminaciones, sustituciones e inserciones). El tipo de alineación utilizado para la evaluación fue el alineamiento local a través del algoritmo Smith-Waterman. Las métricas utilizadas para la evaluación fueron: Precisión, Recall y F1-Score. Por último, estas métricas se calcularon para cada uno de los errores a través de las siguientes categorías: verdaderos positivos: son los casos donde el modelo predice correctamente el error, los falsos negativos: son los casos donde el modelo no detecto el error, los falsos positivos: son los casos donde el modelo predice que hay un error incorrectamente y por ultimo los verdaderos negativos: Son los casos no relacionados con el error que se está analizando que el modelo detectó correctamente.

1.1 Objetivo general

Evaluar el desempeño de modelos acústicos basados en Transformer para tareas relacionados con la pronunciación y lectura.

1.2 Objetivos específicos

1. Entrenar un modelo acústico basado en Transformer para predecir la pronunciación a nivel fonético para el idioma inglés.
2. Evaluar los errores de pronunciación a nivel fonético del idioma inglés hablado por personas cuya lengua materna es el español, usando el modelo “wav2vec2-xlsr-53-espeak-cv-ft”.
3. Evaluar los errores de lectura a nivel de palabras en grabaciones de audio del idioma español, mediante el modelo Whisper-large-V2.

1.3 Alcance

Como data se utilizó los corpus TIMIT, FLEURS, L2-ARCTIC y los Noticieros NHK en el idioma español.

En una primera instancia, se intentó entrenar un modelo usando la técnica de transfer learning con el modelo Whisper-Small para predecir la pronunciación a nivel fonético para el idioma inglés, cuyos resultados no fueron satisfactorios.

Posteriormente, se realizó la evaluación de los errores de pronunciación a nivel fonético utilizando el modelo “wav2vec2-xlsr-53-espeak-cv-ft” para el idioma inglés hablado por personas cuya lengua materna es el español, con resultados que no fueron satisfactorios.

Finalmente, se realizó la evaluación de lectura a nivel de palabras en grabaciones de audio del idioma español utilizando el modelo Whisper-Large-V2 con resultados que fueron satisfactorios.

Tras realizar los primeros dos experimentos, se observó que los resultados no eran satisfactorios para las tareas de pronunciación. Debido a los resultados previos, se realizó el tercer experimento con el cual se obtuvo mejores resultados.

1.4 Marco teórico

1.4.1 ASR

Un ASR es un sistema que permite convertir una señal de audio en texto, es decir, obtener la transcripción de audio [1].

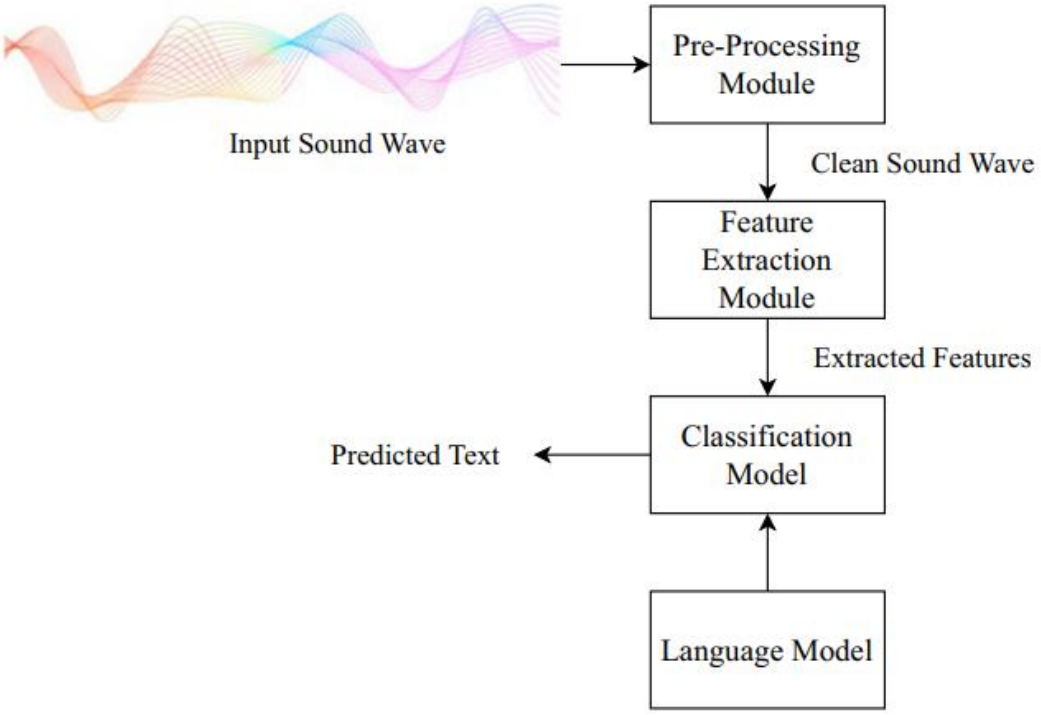

Figura 1. Arquitectura básica de un ASR [2].

Una arquitectura básica del ASR se muestra en la Figura 1. Un ASR tiene generalmente 4 componentes, los cuales se detallan a continuacion [2]:

1. **Preprocesamiento:** Este módulo toma la señal de audio y elimina ruidos y distorsiones que puede tener la misma.

- **Feature Extraction:** A este módulo se le entrega como entrada la señal de audio preprocesada y se la convierte del dominio del tiempo al dominio de la frecuencia a través de la transformada de Fourier, para así obtener vectores de características de la señal. El método de Feature Extracción más popular son los MFCCs [3].
- **Modelo de Clasificación:** El modelo toma como entrada los vectores de características y predice el texto.
- **Modelo de lenguaje:** Contiene las reglas y semántica de un lenguaje.

1.4.2 Transformer

Transformer es un tipo de arquitectura de red neuronal que fue presentada por el paper “All attention is you need” [5]. Transformer es muy utilizado en tares de procesamiento de lenguaje natural como: la transformación de texto y el desarrollo de modelos de lenguaje como chatGPT. Además, también se utiliza en tareas relacionadas con el reconocimiento del habla. Transformer permitió resolver el “problema de memoria” que sufrían las redes neuronales recurrentes (RNN), debido a que las RNN tienen memoria limitada y se pierde información entre secuencias alejadas entre sí [9]. Para resolver esto, Transformer utilizo el concepto de “mecanismos de atención”. Un mecanismo de atención permite añadir contexto al texto a través de cuatro partes que lo componen, las cuales son: Query, Keys, Values y Score Function y así producir predicciones más precisas (Revisar [12] para más detalles de los mecanismos de atención).

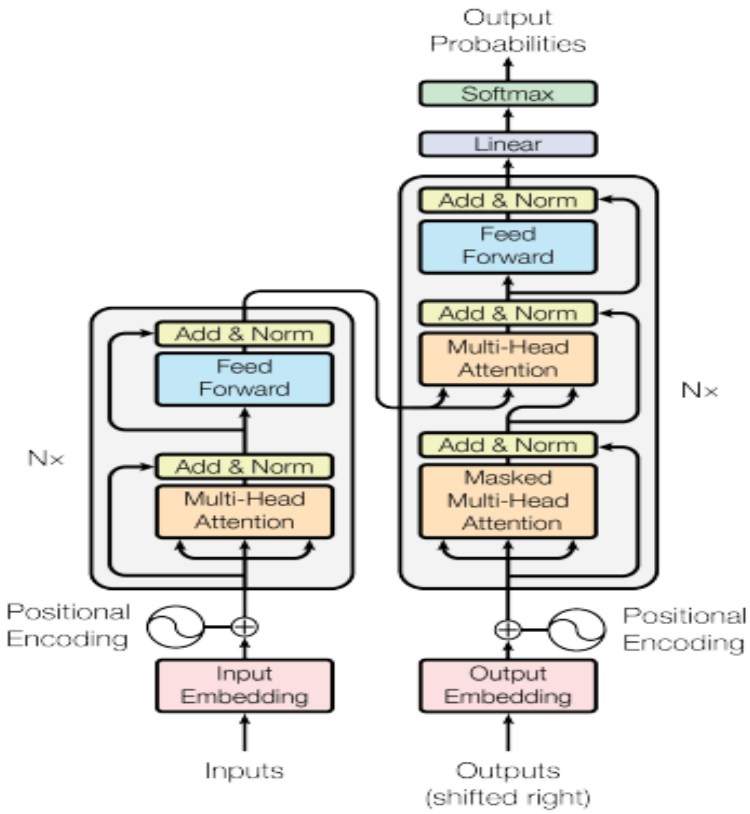

Figura 2. Arquitectura de Transformer [5].

La arquitectura de Transformer se muestra en la Figura 2, y una breve explicación sobre ella, se detalla a continuación. En primer lugar, se tiene los “inputs” (los cuales pueden ser palabras que forman determinada frase) y estos se transforman a vectores de dimensión n (input Embedding) y a estos vectores se les agrega una posición para que el modelo pueda comprender como se introdujo la secuencia de entrada o inputs. Los vectores con estas

posiciones pasan al encoder, el cual tiene 6 capas y en cada capa tiene una red neuronal Feed Forward y el mecanismo de atención de multi-cabeza. El encoder envía al decoder representación continuas de los vectores de entrada. Al decoder de igual forma se le pasa la salida que se desea predecir y esta salida se convierte en vectores de dimensión n y se le agregan las posiciones. Con esta salida y con la salida del encoder se genera una predicción (secuencia de salida). Al igual que el encoder, el decoder también cuenta con 6 capas [4],[6],[7],[8],[10], [11], [12].

1.4.2.1 Whisper

Whisper es un ASR basado en la arquitectura del Transformer, entrenado con 680000 horas de audio de 96 lenguajes. Cada audio se entrenó con su respectiva transcripción, es decir, se usó una data de audio etiquetada [14].

Así mismo, Whisper esta entrenado para realizar diferentes tareas de procesamiento del habla [13], las cuales se detallan a continuacion:

- **Tarea de transcripción:** Permite obtener la transcripción de un audio.
- **Tarea de traducción:** Permite traducir el audio de un idioma seleccionado al inglés.
- **Tarea de detección del idioma**: Permite detectar el idioma de un audio.
- También puede obtener los tiempos a nivel de frase del audio.

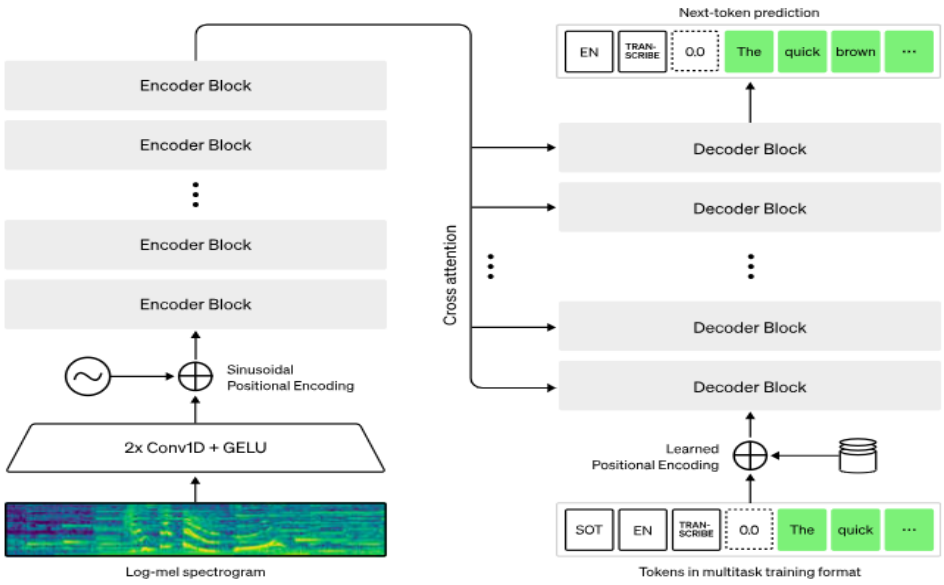

Figura 3. Arquitectura de Whisper [15].

La arquitectura del Whisper ilustrada en la Figura 3, se describe brevemente a continuación. En primer lugar, la señal de audio se transforma en espectrograma log-Mel

[16] y este espectrograma sirve como entrada al encoder del Transformer. El decoder del Transformer se entrenó para devolver la respectiva predicción sobre el audio y a través de determinados tokens especiales con los que fue entrenado (ver Figura 4), permite que se puedan realizar las tareas de procesamiento del habla explicadas anteriormente [14].

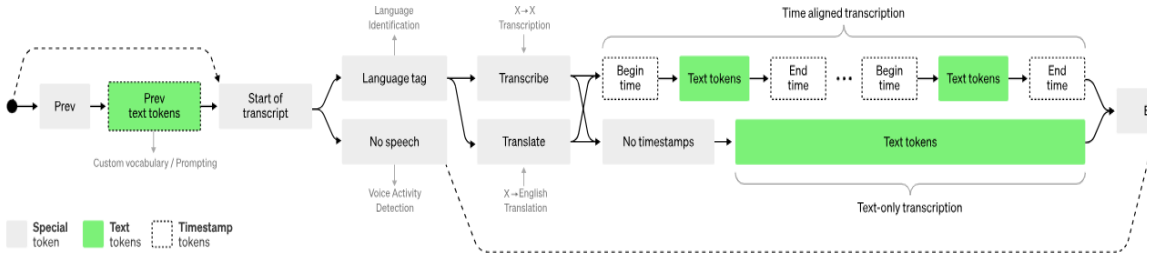

Figura 4. Tokens de Whisper [15].

Por otro lado, Whisper cuenta con 5 tipos de modelos. Todos los modelos fueron entrenados con data multilenguaje. Mas información sobre los modelos, se encuentran en la Figura 5.

Size	Layers	Width	Heads	Parameters	English-only	Multilingual
tiny	4	384	6	39 M	✓	✓
base	6	512	8	74 M	✓	✓
small	12	768	12	244 M	✓	✓
medium	24	1024	16	769 M	✓	✓
large	32	1280	20	1550 M	x	✓

Figura 5. Tipos de modelos de Whisper [14].

1.4.2.2 Wav2Vec 2.0

Wav2Vec 2.0 es otro ASR para el aprendizaje auto-supervisado basándose en la arquitectura de Transformer. Es antecesor del modelo Whisper. Wav2Vec 2.0 es el sucesor de Wav2Vec, el cual es un modelo que trabaja con audios sin etiquetar (sin transcripciones) para entrenar modelos ASR [17].

Wav2Vec 2.0 fue entrenado con 960 horas usando la data Librispeech sin las transcripciones de texto [20].

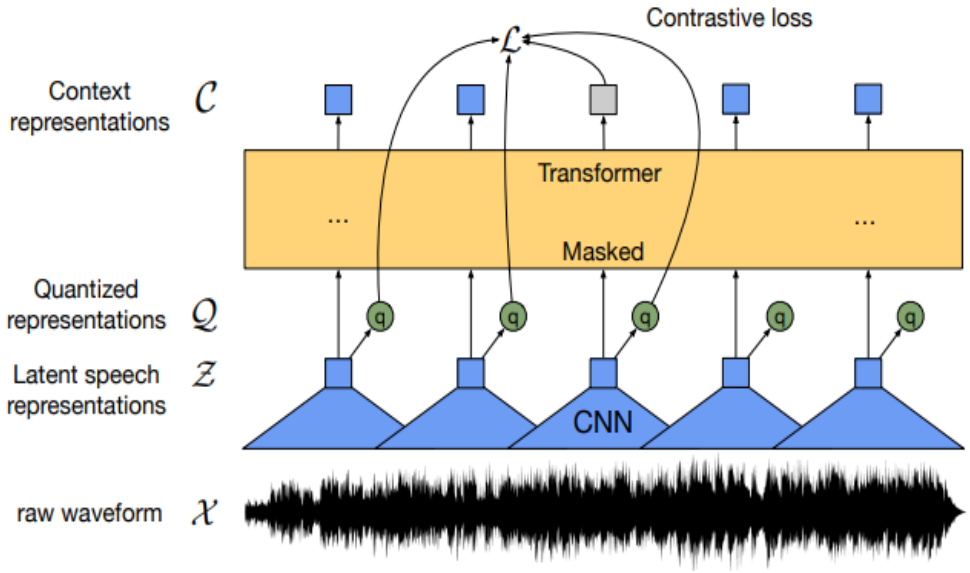

Figura 6. Arquitectura de Wav2Vec 2.0 [21].

La arquitectura del Wav2Vec 2.0 representada en la Figura 6, se describe de manera resumida a continuación. En primer lugar, utilizando redes convoluciones se procesa la señal de audio para obtener representaciones latentes del habla, las cuales son un conjunto de unidades del habla más pequeñas que los fonemas y que permiten describir una señal de audio. Estas representaciones sirven como entrada a un cuantificador. Un cuantificador permite la transformación de valores del espacio continuo a valores finitos en un espacio discreto [24]. Antes de agregar las representaciones de audio al Transformer, se realiza un proceso de enmascaramiento a la mitad de estas. A continuacion, estas representaciones ingresan al encoder del Transformer para construir representaciones contextualizadas. Por último, el decoder del Transformer devuelve la transcripción del audio a través de la resolución de una tarea contrastiva [18, 21, 24].

Así mismo, una versión multilenguaje del Wav2vec 2.0 fue entrenada, llamada XLSR. XLSR fue entrenado con un aproximado de medio millón de horas de audio de 128 lenguajes por medio de la data Common Voice [19].

A través de XLSR se entrenaron diferentes modelos para predecir fonemas, uno de ellos es el “wav2vec2-xlsr-53-espeak-cv-ft” [22]. Como se puede notar tiene “xlsr-53” en su nombre, es decir, que se usó la versión del modelo XLSR con 53 lenguajes en lugar de los 128. El modelo “wav2vec2-xlsr-53-espeak-cv-ft” es un transfer learning de XLSR que permite transcribir audios a fonemas en múltiples lenguajes usando el alfabeto IPA [23].

1.4.3 Alfabetos fonéticos

1.4.3.1 IPA

IPA (Alfabeto Fonético Internacional) es un alfabeto fonético diseñado para representar sonidos de los lenguajes del mundo. Es muy utilizado para enseñar idiomas y comúnmente se usa en diccionarios que indican como se pronuncian palabras en inglés. IPA tiene más de 100 símbolos fonéticos, de los cuales, en su mayoría, son símbolos provenientes del alfabeto romano y griego, pero no todos se utilizan en la transcripción fonética de un idioma. Para cada idioma se utiliza solo un subconjunto de este [25]. Revisar el Anexo ll para observar el alfabeto IPA.

1.4.3.2 Arpabet

Arpabet es un alfabeto fonético para el idioma inglés americano. Este alfabeto utiliza símbolos ASCII para la representación computacional de los fonemas, pero para la representación estándar, utiliza el formato de letras mayúsculas o minúsculas (decodificación de los símbolos ASCII), generalmente 2. Por otro lado, Arpabet comúnmente es utilizado en diccionarios de pronunciación como el CMU [26]. Revisar el Anexo lll para observar el alfabeto Arpabet.

1.4.4 Alineación de secuencias

Una alineación de secuencias permite comparar dos secuencias generalmente de diferentes tamaños, con el propósito de encontrar la mayor similitud entre las mismas.

Un ejemplo de una alineación se muestra a continuación:

Secuencia1: PTTHT-T

Secuencia2: GT-HTOT

Nótese que las secuencias presentan el símbolo “-” conocido como gap. Este símbolo en una alineación refleja que hubo una inserción o eliminación, dependiendo su ubicación. Tomando como referencia la Secuencia1, si el gap se encuentra en esta cadena, se tratará de una eliminación. En cambio, si el gap está en la Secuencia2, se tratará de una inserción. Así mismo, pueden existir diferentes alineaciones para alinear dos frases, pero se selecciona la que tenga la mayor “puntuación de alineación”, esta puntuación generalmente analiza cuantos caracteres coincidieron, cuantos fueron sustituidos y cuantos gaps hubo. A menudo se tiene la siguiente configuración para la “puntuación de alineación”: si

encuentra una coincidencia se sumaría 1, si se encuentra una sustitución, no se sumaría nada y si se encuentra un gap se restaría 1 [27].

También es importante recalar, que existen dos tipos de alineaciones: global y local. El alineamiento global trata de alinear las secuencias de inicio a fin, es decir, se incluyen todos los caracteres de las secuencias, generalmente se usa cuando las secuencias tienen la misma longitud y son muy similares mientras que el alineamiento local trata de encontrar las regiones de las secuencias con mayor similitud o coincidencias [28]. Para producir los alineamientos de las secuencias se utilizan algoritmos de programación dinámica como el algoritmo Smith-Waterman [29] (alineación local) o el algoritmo de Needleman-Wunsh (alineación global) [30].

1.4.5 Corpus

1.4.5.1 TIMIT

La data TIMIT es una data con grabaciones de audio del idioma inglés americano. Contiene un total de 6300 frases habladas por 63 personas de 8 dialectos diferentes de Estados Unidos, donde el 70% de los hablantes fueron hombres y el 30% fueron mujeres. Cada persona pronuncio 10 frases [31]. Cada frase tiene una transcripción fonética, además de la transcripción del audio. Sobre la transcripción fonética, la data TIMIT utilizó su propio alfabeto fonético comúnmente llamado “TIMITBET”, el cual es un alfabeto basado en Arpabet y contiene 61 fonemas. Mas información sobre este alfabeto se puede encontrar en [32].

Cada directorio de cada hablante contiene los archivos detallados a continuación:

- Las 10 grabaciones de audio en formato WAV.
- Las transcripciones de los audios guardadas con la extensión .txt.
- Las transcripciones de los audios a nivel de palabras guardadas con la extensión. WRD.
- Las transcripciones fonéticas de los audios con el alfabeto “TIMITBET” guardadas con la extensión. PHN.

1.4.5.2 FLEURS

La data FLEURS es una data multilenguaje con grabaciones de audio de más de 100 idiomas. Por idioma se tiene aproximadamente 12 horas de audio. Las sentencias que hablaron las personas de los diferentes lenguajes cubren temas como: la política, la

ciencia, deportes, etc., esto para tener mayor diversidad. Así mismo, la data FLEURS proporciona transcripciones de alta calidad de los audios [33].

Cada idioma tiene los siguientes archivos:

- **/audios:** Esta carpeta contiene las grabaciones de los audios y se encuentran comprimidos en formato tar.gz.
- Archivos con las transcripciones de los audios comprimidos en formato tar.gz.

La data FLEURS es una data online y se la puede encontrar en el repositorio de Hugginface [34].

1.4.5.3 L2-ARCTIC

L2-ARTIC es una data que contiene grabaciones de audios de personas cuya lengua nativa no es el inglés. Entre las lenguas maternas de los hablantes se tiene: coreano, español, chino, entre otras. Así mismo, el corpus posee las transcripciones fonéticas, no fonéticas y errores de pronunciación de los hablantes en los audios. La duración del corpus es de 27.1 horas, donde cada hablante grabó 1.13 horas en promedio [35].

La estructura de directorios para cada hablante se detalla a continuación [36]:

- **/wav:** Contiene las grabaciones de audios de los hablantes.
- **/transcript:** Contiene las transcripciones del audio guardados con la extensión .txt
- **/textgrid:** Contiene las transcripciones fonéticas usando el alfabeto Arpabet. Estas transcripciones se encuentran en formato TextGrid.
- **/annotation:** Contiene los errores de pronunciación (sustitución, eliminación e inserción) detectados por Anotadores (personas expertas en el idioma) usando los alfabetos Arpabet e IPA. Estos errores se guardaron en archivos TextGrid.

Para comprender de mejor manera los errores de pronunciación detectados por los anotadores, se utilizará un TextGrid de ejemplo del directorio de /annotation de un hablante al azar abierto con la herramienta Praat [37].

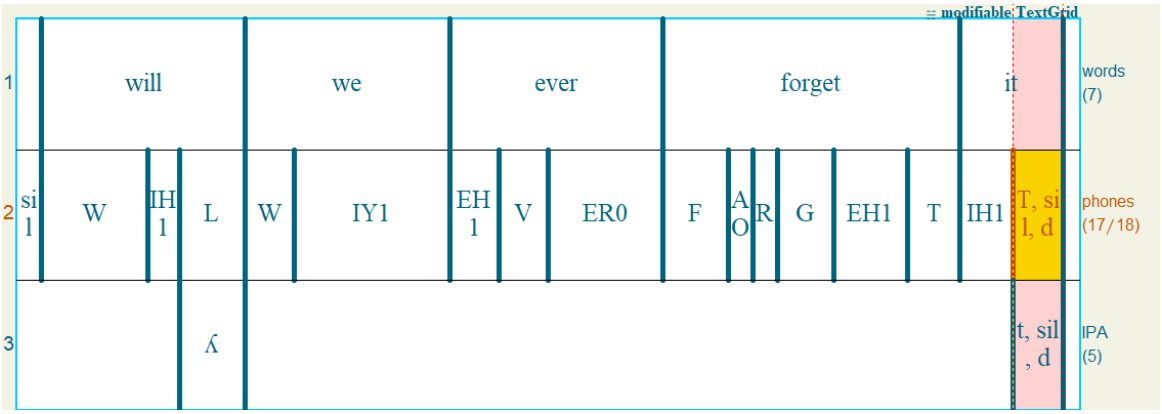

Figura 7. TextGrid del directorio annotation de un hablante seleccionado al azar.

Los niveles de anotación del TextGrid de la Figura 7, se detallan a continuación:

- **Nivel words:** En este nivel se encuentra la transcripción del audio.
- **Nivel phones:** En este nivel se encuentra la transcripción fonética del audio usando el alfabeto Arpabet, además se encuentran los errores de pronunciación detectados por los Anotadores. Los Anotadores utilizaron el formato: “fonema que se debió pronunciar, fonema pronunciado por el hablante, etiqueta con el tipo de error” para describir cada tipo de error. Esto se puede comprender de mejor manera observando el TextGrid, el cual tiene: “T, sil, d”, donde T es el fonema correcto, sil el incorrecto y d el tipo de error. Para el error de eliminación se utilizó la etiqueta “d” (eliminación), para el error de inserción se utilizó “a” (inserción) y para sustitución, se usó “s” (sustitución). Así mismo, es importante resaltar que se utilizó el fonema sil (silencio) cuando el hablante no debió pronunciar un fonema o cuando se olvidó de pronunciar un fonema**.** Para los fonemas que no tienen error, no se realizó ninguna anotación.
- **Nivel IPA:** En este nivel solo se encuentran las anotaciones de los errores usando el alfabeto IPA. Se utiliza el mismo formato que el nivel phones para describir los errores.

1.4.5.4 Noticieros NHK en el idioma español

NHK es un servicio que ofrece noticias en formato broadcast, TV y radio de lo relevante que pasa en el mundo, principalmente en Asia y cuya sede se encuentra en Japón [38]. La data Noticieros NHK en el idioma español contiene grabaciones de varios noticieros para el idioma español, así como las transcripciones de los audios.

Cada noticiero tiene los siguientes archivos:

- Archivos HTML con las noticias que los locutores debieron de leer en el noticiero. Las noticias se encuentran en la Sección “ArticleBody” de cada HTML.
- Grabación del noticiero en formato MP3.
- TextGrid que contiene la transcripción del noticiero y demás información.

Un ejemplo de un TextGrid de un noticiero al azar se muestra en la Figura 8.

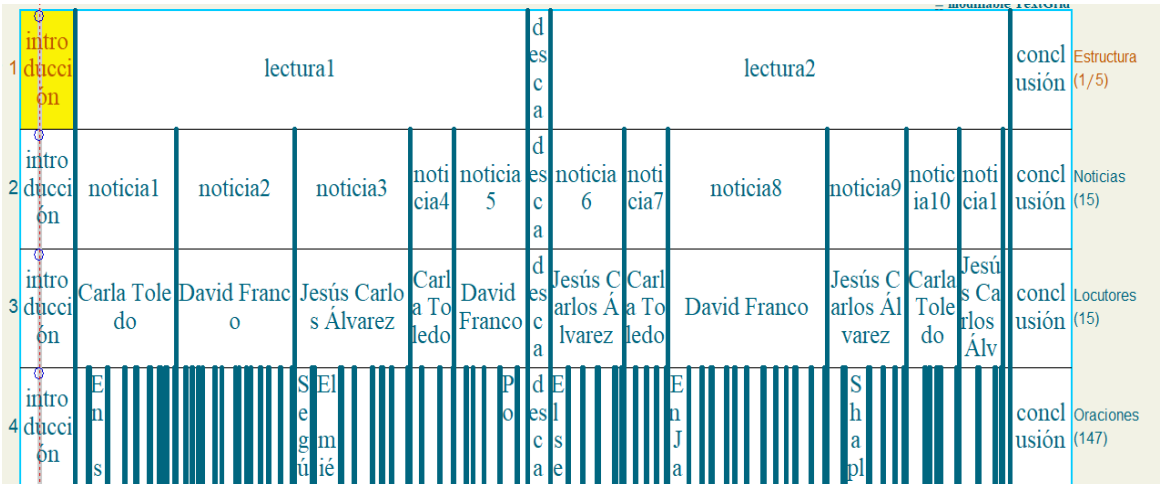

Figura 8. TextGrid de un noticiero seleccionado al azar.

Los niveles de anotación del TextGrid se detallan a continuación:

- **Nivel Estructura:** Refleja la división del noticiero antes y después del descanso (lectura1 y lectura2).
- **Nivel Noticias:** Refleja la división del noticiero por noticias. (noticia1, noticia2, etc.).
- **Nivel Locutores:** Contiene los nombres de los locutores que leen las noticias.
- **Nivel Oraciones:** Contiene cada oración hablada por el locutor con todos los fallos cometidos, los cuales pueden ser: eliminar, insertar o sustituir palabras. Así mismo, el número de oraciones de este nivel puede no ser igual a los archivos HTML, debido a que el locutor pudo saltarse o insertar oraciones.

Es importante recalar que cada nivel contiene introducción, descanso y conclusión, las cuales no tienen una transcripción.

2 METODOLOGÍA

2.1 Evaluación de los errores de lectura usando el modelo Whisper-Large-V2

Para realizar esta evaluación, se utilizó las siguientes herramientas:

- **Entorno:** Jupyter Notebook
- **Data:** Noticieros NHK en el idioma español
- **Lenguajes de programación:** Perl y Python

Además, la evaluación se la realizó a través del archivo: “Evaluación_errores_lectura. ipynb” en Perl (ver ANEXO I) y el procedimiento fue el siguiente:

2.1.1 Lectura de la data

La data que se usó fue los Noticieros NHK en el idioma español, donde cada noticiero se compone de archivos HTML, un archivo TextGrid y archivo un de audio.

La data se dividió en tres listas de la siguiente manera:

- **Referencia 1:** Esta lista se formó a partir de la Sección ArticleBody de los archivos HTML de cada noticiero. Estos archivos muestran lo que el locutor debería haber hablado correctamente.
- **Referencia 2:** Esta lista se formó a partir de los archivos TextGrid de cada noticiero, los cuales muestran lo que el locutor dijo, es decir, sería la transcripción del audio realizada por personas que hablan español.
- **Predicciones:** Esta lista se formó a partir de la predicción de cada archivo de audio de cada noticiero usando el modelo Whisper-Large-V2.

Tanto Referencia 1, Referencia 2 y Predicciones se dividieron por frases.

2.1.1.1 Obtención de Referencia 1

Para obtener las frases de Referencia 1, se realizó lo siguiente:

- En primer lugar, se obtuvo las ubicaciones de cada HTML de todos los noticieros.
- A continuación, se obtuvo el ArticleBody de cada HTML.
- Al obtener el ArticleBody de cada HTML, se dividió en frases utilizando los siguientes delimitadores:

- Punto (.): Se dividió una frase por este símbolo, solo si el siguiente carácter, saltándose espacios, fue una letra Mayúscula.
- Dos puntos, punto y coma o signo de exclamación (:;!): Se dividió una frase por estos símbolos, si el siguiente carácter, saltándose espacios, no fue un digito.

2.1.1.1.1 Segmentación de frases

Un punto importante que se analizó, es que el número de frases de Referencia 1 debió ser igual al número de frases de Referencia 2, pero esto no sucedió, debido a que existen frases en Referencia 1 que deben dividirse o deben convertirse en una sola frase para tener el mismo número de frases de Referencia 2, entonces para lograr esto, se realizó lo siguiente:

1. Para unir las frases que debían haber estado juntas, se verificó que el final de las frases coincidiese con las palabras de la columna 1 de la Tabla 1 y se unió esas frases con las siguientes frases verificando que el inicio de estas frases coincidiese con las palabras de la columna 2.

Tabla 1. Casos especiales de frases que deben convertirse en una sola.

Fin de la primera frase	Inicio de la segunda frase
Moon Jae-in	En su
ventanas	Fue necesario

2. Para separar las frases que debían haber estado separadas, se verificó que las frases coincidiesen con los elementos de las tres columnas de la Tabla 2 y se utilizó el delimitador de la columna 2 para separar en dos frases cada frase, donde la primera frase terminó con las palabras de la columna1 y la segunda frase comenzó con las palabras de la columna 3.

Tabla 2. Casos especiales de frases que debían de haber estado separadas.

Fin de la primera frase	Delimitador	Inicio de la segunda frase
2017	**.**	500 punto com

2.1.1.1.2 Normalización de Referencia 1

Debido a que debe de haber una similitud en cómo se presentan las frases entre cada lista, se realizó una normalización en Referencia 1, esto se puede observar en la Tabla 3.

Tabla 3. Normalización de Referencia 1.

Normalizaciones	Ejemplos
1. Cambió de códigos Unicode a UTF-8.	\\u00f3: ó
2. Reemplazo de "a. m" por "de la mañana", si se tiene dígitos separados por ":". Así mismo, reemplazo de "p. m." por "de la tarde" o "de la noche" dependiendo si el dígito del lado izquierdo separado por ":", es menor o igual que 6.	10:30 a. m: 10:30 de la mañana 5:30 p. m.: 5:30 de la tarde 8:00 p. m.: 8:00 de la noche
3. Reemplazo de ":" por "y", si el ":" se encuentra entre dígitos.	7:09: 7 y 09
4. Reemplazo "," por "coma" solo si "," se encuentra entre dígitos.	1,3: 1 coma 3
5. Cambio de números a texto. Así mismo, el formato "xxx-xxx" donde cada x es un digito, se cambió a texto digito por digito la primera parte antes del guion.	24: veinticuatro 737-800: siete tres siete ochocientos
6. Reemplazo de "%" por "por ciento" y "&" por "and"	10%: 10 por ciento S&N: S and N
7. Eliminación de símbolos especiales	"Le Monde": Le Monde ¡Tenemos!: Tenemos "parte, tiene": parte tiene
8. Eliminación espacios en blanco al principio y al final de cada frase	" El …": "El …" "… disturbio ": "… disturbio"
9. Eliminación de espacios adicionales separando cada palabra	"… de la tarde": "… de la tarde"
10. Cambio de todas las palabras a minúsculas	"…Corea del Norte…": "…corea del norte…"

2.1.1.2 Obtención de Referencia 2

Para obtener las frases de Referencia 2, se realizó lo siguiente:

- Se obtuvo las ubicaciones de cada TextGrid de todos los noticieros.

  A continuación, se leyó cada TextGrid y se fueron guardando las frases del nivel Oraciones en Referencia 2 (ver Figura 8 para observar la estructura del TextGrid). Las secciones de "Introducción", "descanso" y "conclusión", no se tomaron en cuenta para la evaluación, ya que no existe una transcripción en los archivos TextGrid.

2.1.1.2.1 Normalización de Referencia 2

Al igual que Referencia 1, debe de haber una similitud en cómo se presentan las frases entre cada lista, por tal motivo, se realizó una normalización en Referencia 2, esto se puede observar en la Tabla 4.

Tabla 4. Normalización de Referencia 2.

Normalizaciones	Ejemplos
1. Eliminación de cada frase de la lista Referencia 2 que coincida con “# _ %”	
2. Reemplazo de "veinte y numero" por "veintinumero", solo si el número estaba entre el uno y nueve	veinte y cuatro: veinticuatro
3. Reemplazó "diez y numero" por "diecinumero", solo si número estaba entre el seis y nueve.	diez y seis: dieciseis
4. Reemplazo “two” por “dos”	two: dos
5. También, se realizaron las normalizaciones 7, 8, 9 y10 y 11 de la Tabla1	

2.1.1.3 Obtención de las Predicciones

Para obtener las frases de Predicciones, se realizó lo siguiente:

- Se obtuvo las rutas de cada audio de cada noticiero de la data.

  A continuación, se guardaron los rangos de tiempo de cada noticia del nivel Noticias de cada TextGrid en una lista. Las secciones de “Introducción”, “descanso” y “conclusión”, no se tomaron en cuenta para la evaluación, ya que no existe una transcripción en los archivos TextGrid.

- Con los rangos de tiempos de cada noticia y las rutas de los audios, se cortó cada audio con respecto a los rangos de tiempo de cada noticia presentes en al audio y se guardó cada noticia cortada del audio en una carpeta llamada “AUDIOS” para cada noticiero.

- Los audios ya cortados se usaron para hacer las Predicciones con Whisper-large-v2 a través del archivo “Predicciones_Whisper.ipynb”, el cual es un notebook en Python.

El procedimiento para calcular las Predicciones fue el siguiente:

2.1.1.3.1 Calculó de las Predicciones usando Whisper-Large-V2

En primer lugar, se leyó cada audio cortado de la carpeta “AUDIOS” de cada noticiero.

A continuación, con las rutas de cada archivo de audio se realizó la predicción usando el modelo Whisper-Large-V2 y el idioma español para la transcripción.

2.1.1.3.1.1 Guardado de Predicciones

Para poder utilizar las Predicciones en el notebook de Perl con el que se venía trabajando, se guardó Predicciones en un archivo .txt de nombre "predicciones_tmp.txt".

2.1.1.3.2 Lectura de Predicciones

Se leyó el archivo "predicciones_tmp.txt"**,** el cual tiene las Predicciones de los audios cortados y se separó por frases utilizando el siguiente delimitador:

- Punto (.): Se dividió una frase por este símbolo, solo si el siguiente carácter, saltándose los espacios, fue una letra Mayúscula.

2.1.1.3.3 Segmentación de frases

Se realizó el mismo procedimiento que la Sección 2.1.1.1.1, se usó la Tabla 5 para unir las frases que debían haber estado juntas y la Tabla 6 para separar las frases que debían haber estado separadas.

Tabla 5. Casos especiales de frases que deben convertirse en una sola.

Fin de la primera frase	Inicio de la segunda frase
suerte	De ahí
tema	La televisión
puntos	Esto representa
Teherán	Grupos armados
Unidos	perpetró
mesa	La Asociación
regla	En algunos
invernadero	Si
desconocida	De ellos
ventanas	Fue necesario

Tabla 6. Casos especiales de frases que deben dividirse.

Fin de la primera frase	Delimitador	Inicio de la segunda frase
2011	,	terremoto
grupo	,	la aseguradora
2017	.	500.com
tipo	,	la última
mundo	,	Koki Niwa
evento	,	rojo
país	,	esto

festivos	.	7-Eleven
kilolitro	,	esto
siguiente	«	Estamos
ella	,	sin embargo

2.1.1.3.4 Normalización de Predicciones

Como se comentó en secciones anteriores, debe de haber una similitud en cómo se presentan las frases entre cada lista, por tal motivo, se realizó una normalización en Predicciones, esto se puede observar en la Tabla 9.

Tabla 7. Frases que añadió Predicciones incorrectamente.

Frases
"Para más información visite www.chaplanir.org"
"Gracias."
"Gracias por ver el video."
"Traducido por Marie Arias"

Tabla 8. Equivalencias de abreviaciones y palabras especiales de Predicciones.

Abreviaciones y palabras especiales (Predicciones)	**Equivalencias (Referencia 2)**
Jong-un	Jong un
Entre tanto	. Entretanto
EEUU	Estados Unidos
mal acostumbrada	malacostumbrada
Yamana Kako	Yamanakako
noche vieja	nochevieja
OEEA	Organismo Enternacional de Energía Atómica
siete-Eleven	seveneleven
Bangladesh	Bangladés
Piongyang	Pionyang
Hezbollah	Hizbulá
al-Shabaab	Alshabab

Tabla 9. Normalización de Predicciones.

Normalizaciones	Ejemplos
1. Eliminación de cada frase de la lista Referencia 2 que coincida con la Tabla 7	
1. Unión de la palabra "ex" con cualquier palabra, así como "mini" con cualquier palabra.	Exejecutivo: exejecutivo mini supermercado: mini supermercado
2. Reemplazo .com por punto com	.com: punto com
3. Reemplazo de abreviaciones y palabras especiales por sus equivalentes según la Tabla 8.	
4. También, se realizaron las normalizaciones 4,5,6 7, 8, 9 y 10 de la Tabla1.	

2.1.2 Alineaciones entre Referencia 1, Referencia 2 y Predicciones

Para obtener los errores de lectura a nivel de palabras, se realizó tres alineaciones entre las listas. El tipo de alineación utilizada fue la local a través del algoritmo Smith-Waterman.

Las tres alineaciones se detallan a continuación:

2.1.2.1 Alineación entre Referencia 1 y Referencia 2

Para calcular la alineación entre Referencia 1 y Referencia 2, se fue iterando cada elemento (frases) de las dos listas al mismo tiempo. Se calculó la distancia de Levenshtein [39] entre las dos frases actuales, esta distancia permitió conocer a través de un número entero, cual es la diferencia entre dos frases, mientras menor sea el número, menos diferencia hay entre las frases. A continuación, se obtuvo el tamaño de las frases y se seleccionó el mayor tamaño entre las dos y utilizando la fórmula de la Ecuación 1, se pudo analizar si las frases eran diferentes o no, si las frases tenían un porcentaje menor o igual a 65%, quería decir que eran similares, entonces se guardaron en una lista (Alineación 1) la alineación de las frases utilizando el algoritmo de Smith-Waterman.

$$\mathrm{PD} = \left(\frac{\text{Distancia de Levenshtein}}{\text{Tamaño de la frase más grande}}\right) * 100$$

Ecuación 1. Porcentaje de diferencia entre dos frases.

El motivo del calculó de este porcentaje (PD), es debido a que existen frases de la primera lista que no se encontraron en la segunda lista. En el caso de la alineación entre Referencia 1 y Referencia 2, el locutor se saltó frases. Por este motivo, si el porcentaje fue mayor a 65% (el porcentaje del 65% es debido a que hay frases que son similares, pero tienen un

PD cercano a ese número, se los podría considerar como casos especiales) se realizó lo siguiente:

- Se definió un número máximo de 10 iteraciones para iterar la primera lista, dejando a la frase actual de la segunda lista como referencia hasta encontrar una frase similar en la primera lista con un PD menor que 65%. Es decir, este número sirvió para saltarse las frases que están presentes en la primera lista, pero no en la segunda lista.
- Si se encontró antes o en las 10 iteraciones la frase similar en la primera lista, las frases antes de encontrar la frase similar se guardaron en Alineación 1 con la etiqueta <no-alineación>. La frase similar y la frase de la segunda lista se alinearon y se guardaron en Alineación 1 utilizando el algoritmo de Smith-Waterman

Un ejemplo de la Alineación 1 se muestra en la Figura 9:

```
una de las condiciones de su salida de la cárcel era que no podía abandonar japón
una de las condiciones de su salida de la cárcel era que no podía abandonar japón
```

Figura 9. Elemento de Alineación 1.

Cada elemento de Alineación 1 está formado por las frases alineadas de las listas Referencia 1 y Referencia 2**.** Además, esta alineación sirvió como referencia para conocer la ubicación de los errores de lectura que debió predecir el modelo.

2.1.2.1.1 Cambio de Alineación 1 a nivel de palabras

Debido a que se los errores de lectura se analizaron a nivel de palabras, las frases de cada elemento de Alineación 1 tuvieron que cambiarse a palabras. Para esto se realizó el siguiente procedimiento:

1. En primer lugar, se eliminaron todos los elementos que tenían <no-alineación> en la segunda frase de cada elemento de Alineación 1.
2. A continuación, si se encontró un guion (“-”) en la primera frase de cada elemento y en la segunda frase un espacio, se cambió el guion por el espacio en la primera lista. Se realizó este cambio para evitar que palabras no se hayan dividido correctamente. Este proceso también se realiza a la inversa, es decir, segunda frase con guion.

3. Con los anteriores cambios, se pudo cambiar a palabras las frases de cada elemento de Alineación 1, tomando las posiciones de los espacios de la primera frase de cada elemento y dividiendo por estas posiciones las dos frases.

Un elemento de Alineación 1 se puede observar en la Figura 10:

```
[['una', 'de', 'las', 'condiciones', 'de', 'su', 'salida', 'de', 'la', 'cárcel', 'era', 'que', 'no', 'podía', 'abandonar', 'j
apón'],
['una', 'de', 'las', 'condiciones', 'de', 'su', 'salida', 'de', 'la', 'cárcel', 'era', 'que', 'no', 'podía', 'abandonar', 'ja
pón']],
```

Figura 10. Elemento de Alineación 1 dividido por palabras.

2.1.2.1.2 Obtención de los tipos de errores de lectura a nivel de palabras para Alineación 1

Para el cálculo de los errores de lectura a nivel de palabras, se comparó cada palabra de la primera frase (Palabra 1) con cada palabra de la segunda frase (Palabra 2) de cada elemento de Alineación 1. Entonces se obtuvo que:

- **Error de eliminación:** Si Palabra 2 está compuesto solo de guiones, entonces en la segunda frase se guardó la etiqueta: <eliminación>: Palabra 1.
- **Error de inserción:** Si Palabra 1 está compuesto solo de guiones, entonces en la segunda frase se guardó la etiqueta: < inserción>: Palabra 2 en la misma posición que palabra 1.
- **Error de sustitución:** Si Palabra 1 y Palabra 2 son diferentes, entonces en la segunda frase se guardó la etiqueta: <sustitución>: Palabra 2.

La Alineación 1 muestra los errores como los no errores que el modelo tuvo que predecir. Es decir, es la referencia a la cual las Predicciones debieron de llegar. Un ejemplo se puede observar en la Figura 11.

```
[['la', 'llam---a', 'olímpica', 'recorrerá', 'las', 'cuarenta', 'y', 'siete', 'prefecturas', 'de', 'japón', 'a', 'lo', 'larg
o', 'de', 'ciento', 'veintiuno', 'días'],
['la', '<sustitución>: antorcha', 'olímpica', 'recorrerá', 'las', 'cuarenta', 'y', 'siete', 'prefecturas', 'de', 'japón',
'a', 'lo', 'largo', 'de', 'ciento', 'veintiuno', 'días']],
```

Figura 11. Elemento de Alineación 1 con los errores de lectura.

2.1.2.2 Alineación entre Referencia 2 y Predicciones

Se realizó el mismo procedimiento que la Sección 2.1.2.1, pero con las listas Referencia 2 y Predicciones.

A esta alineación, se la llamó Alineación 2 y en la Figura 12 se muestran un elemento de esta.

```
es la segunda vez que alberga los juegos la anterior fue en mil novecientos sesenta y cuatro
es la segunda vez que alberga los juegos la anterior fue en mil novecientos sesenta y cuatro
```

Figura 12. Elemento de Alineación 2.

2.1.2.2.1 Cambio de Alineación 2 a nivel de palabras

Se realizó el mismo procedimiento que la Sección 2.1.2.1.1. En la Figura 13 se muestra un elemento de Alineación 2.

```
[['es', 'la', 'segunda', 'vez', 'que', 'alberga', 'los', 'juegos', 'la', 'anterior', 'fue', 'en', 'mil', 'novecientos', 'sese
nta', 'y', 'cuatro'],
['es', 'la', 'segunda', 'vez', 'que', 'alberga', 'los', 'juegos', 'la', 'anterior', 'fue', 'en', 'mil', 'novecientos', 'sesen
ta', 'y', 'cuatro']],
```

Figura 13. Elemento de Alineación 2 dividido por palabras.

2.1.2.2.2 Eliminación de errores de inserción de Alineación 2 para realizar la alineación con Referencia 1

Con Alineación 2 a nivel de palabras, se calculó el error de inserción tomando como referencia la Sección 2.1.2.1.2, donde se explica como calcular el error de inserción. Pero para esta Alineación 2 el error de inserción que se obtuvo es una equivocación que el modelo predictor realizó, es decir, añadió una inserción incorrectamente, ya que Alineación 2 es una alineación entre Referencia 2 y Predicciones. por tal motivo, se guardó con la etiqueta <error-inserción> en cada segunda frase de cada elemento de Alineación 2 si existía el error.

A continuación, se contaron cuantos casos de <error-inserción> existieron y se guardó el conteo en una variable de nombre FP_inserción.

Por último, se eliminó estos casos de las segundas frases de cada elemento de Alineación 2 y se guardaron estas frases que fueron convertidas otra vez a una cadena con cada palabra separada por espacio, en una lista (Predicciones_a), la cual sirvió para realizar la alineación de la Sección 2.1.2.3 y permitió conocer si los errores de inserción predichos por el modelo fueron correctos o incorrectos en relación con Referencia 1.

2.1.2.3 Alineaciones entre Referencia 1 y Predicciones_a

Se realizó el mismo procedimiento que la Sección 2.1.2.1, pero con las listas Referencia 1 y Predicciones_a.

A esta alineación, se la llamó Alineación 3 y en la Figura 14 se muestran un elemento de esta.

```
la llam---a olímpica recorrerá las cuarenta y siete prefecturas de japón a lo largo de ciento veintiuno
la antorcha olímpica recorrerá las cuarenta y siete prefecturas de japón a lo largo de ciento veintiuno
```

Figura 14. Elemento de Alineación 3.

2.1.2.3.1 Cambio de Alineación 3 a nivel de palabras

Se llevó a cabo el mismo procedimiento que la Sección 2.1.2.1.1. En la Figura 15 se muestra un elemento de Alineación 3.

```
[['una', 'de', 'las', 'condiciones', 'de', 'su', 'salida', 'de', 'la', 'cárcel', 'era', 'que', 'no', 'podía', 'abandonar',
apón'],
['una', 'de', 'las', 'condiciones', 'de', 'su', 'salida', 'de', 'la', 'cárcel', 'era', 'que', 'no', 'podía', 'abandonar',
pón']],
```

Figura 15. Elemento de Alineación 3 dividido por palabras.

2.1.2.3.2 Obtención de los tipos de errores de lectura a nivel de palabras para Alineación 3

Se realizó el mismo proceso que la Sección 2.1.2.1.2. En la Figura 16 se muestra un elemento de Alineación 3.

La Alineación 3 muestra los errores como los no errores predichos por el modelo Whisper-Large-V2.

```
[['fiscales', 'de', 'japón', 'han', 'comenzado', 'a', 'registrar', 'la', 'casa', 'de', 'tokio', 'en', 'la', 'que', 'carlos',
'ghosn', 'expresidente', 'de', 'nissan', 'estuvo', 'viviendo', 'en', 'libertad', 'bajo', 'fianza', 'antes', 'de', 'su', 'fug
a'],
['fiscales', 'de', 'japón', 'han', 'comenzado', 'a', 'registrar', 'la', 'casa', 'de', 'tokio', 'en', 'la', 'que', 'carlos',
'<sustitución>: g-o-n', '<eliminacion>: expresidente', '<eliminacion>: de', '<eliminacion>: nissan', 'estuvo', 'viviendo', 'e
n', 'libertad', 'bajo', 'fianza', 'antes', 'de', 'su', 'fuga']],
```

Figura 16. Elemento de Alineación 3 con los errores de lectura

2.1.3 Métricas de evaluación para evaluar los errores de lectura

Las métricas utilizadas para la evaluación de los errores de lectura fueron las siguientes [40]:

- **Precision:** Responde a la pregunta: "¿Cuántas de todas las predicciones positivas que se obtuvieron, fueron verdaderas?". Esta métrica se calculó a través de la Ecuación 2.

$$\text{Precision} = \frac{\text{TP}}{(\text{TP} + \text{FP})}$$

Ecuación 2. Fórmula para el calculó de la Precisión.

- **Recall:** Responde a la pregunta: "¿Cuántos datos se predijeron como verdaderos de todos los datos que deberían haber sido verdaderos?". Esta métrica se calculó a través de la Ecuación 3.

$$\text{Recall} = \frac{\text{TP}}{(\text{TP} + \text{FN})}$$

Ecuación 3. Fórmula para el calculó de Recall.

- **F1-score:** Es una métrica que relaciona Precisión y Recall para obtener un valor más objetivo. Esta métrica se calculó a través de la Ecuación 4.

$$\text{F1} - \text{score} = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

Ecuación 4. Fórmula para el calculó de F1-score.

Para calcular estas métricas, se necesitó calcular los verdaderos positivos y negativos y falsos positivos y negativos [33], donde:

- **Verdaderos positivos (TP):** Existe el error y el modelo lo predijo correctamente.
- **Falsos negativos (FN):** Existe el error y el modelo no lo detectó.
- **Falsos positivos (FP):** Falsa predicción que hay el error.
- **Verdaderos negativos (TN):** Son los casos no relacionados con el error que se está analizando que el modelo detectó correctamente.

Para calcular TP, FN, FP y TN, se utilizó las segundas frases de Alineación 1, así como de la Alineación 3, debido a que ahí es donde se guardaron las etiquetas. Como se comentó en secciones anteriores, Alineación 1 sirvió para tener la referencia de los errores y no

errores que el modelo debió predecir y la Alineación 3 que muestra los errores y no errores que predijo el modelo.

2.1.3.1 Eliminación

Para calcular TP, FN, FP y TN se realizó lo siguiente:

- Para calcular los casos de TP se buscó en cada segunda frase de cada elemento de Alineación 1, si existían palabras con la etiqueta: <eliminación>: Palabra 1, si estas existían también en Alineación 3 en la misma posición y eran iguales, quería decir que el modelo predijo correctamente el error de eliminación, entonces era un TP.
- Para calcular los casos de FP se buscó en cada segunda frase de cada elemento de Alineación 1, si existían palabras con la etiqueta: <eliminación>: Palabra 1, si estas no existían en Alineación 3 en la misma posición, quería decir que el modelo no predijo el error de eliminación, entonces era un FP.
- Para calcular los casos de FN se buscó en cada segunda frase de cada elemento de Alineación 1, si existían palabras con la etiqueta: <eliminación>: Palabra 1, si estas existían, quería decir que el modelo hizo una falsa predicción de error de eliminación, entonces era un FN.
- Los casos de TN fueron las palabras que no están en ninguno de los tres casos anteriores. También en TN se añaden los casos eliminados del error de inserción de la Sección 2.1.2.2.2 (FP_inserción).

Los resultados de la métrica de evaluación y de los TP, FN, FP y TN se encuentran en la Sección 4.1.1.1.

2.1.3.2 Inserción

Para calcular los TP, FN, FP y TN del error de inserción, se llevó a cabo el mismo procedimiento que la Sección 2.1.3.1, pero se hicieron algunas modificaciones, las cuales fueron los siguientes:

- Intercambio de la etiqueta <eliminación>: Palabra 1 por <inserción>: Palabra 2.
- Los casos de FN se obtuvieron de la Sección de 2.1.2.2.2 (FP_inserción).
- No hay que tomar en cuenta en TN los casos de FP_inserción de la Sección 2.1.2.2.2.

Los resultados de la métrica de evaluación y de los TP, FN, FP y TN se encuentran en la Sección 4.1.1.2.

2.1.3.3 Sustitución

Para calcular los TP, FN, FP y TN del error de sustitución, se realizó el mismo procedimiento que la Sección 2.1.3.1 pero con la etiqueta <sustitución>: Palabra 2.

Los resultados de la métrica de evaluación y de los TP, FN, FP y TN se encuentran en la Sección 4.1.1.3.

3 EXPERIMENTOS PRELIMINARES

Se realizó dos experimentos anteriores al experimento de la Sección 2, los cuales fueron:

3.1 Experimento 1: Entrenamiento de un modelo acústico basado en Transformer para predecir la pronunciación para el idioma inglés

Para realizar este entrenamiento, se hizo uso de las siguientes herramientas:

- **Entorno:** Kaggle
- **Data:** TIMIT y FLEURS (la data TIMIT se puede encontrar en Kaggle Datasets con el nombre “Corpus_Experimentos”).
- **Lenguaje de programación:** Python

Además, para el entrenamiento se utilizó la técnica del transfer learning.

3.1.1 Transfer learning usando la data TIMIT

Para este primer transfer learning se utilizó solo la data TIMIT y el archivo con el que se realizó fue: “Transfer_learning_Timit.ipynb”. El procedimiento del transfer learning se describe a continuación:

3.1.1.1 Lectura de la data TIMIT

Para leer la data TIMIT se utilizó datasets de Hugginface [41].

El procedimiento para leer la data se detalla a continuacion:

1. En primer lugar, se creó un diccionario con las llaves “train” y “test”.

2. A continuación, con la ubicación de la data y un archivo en Python: “timit_asr.py” (este archivo se encuentra en el repositorio de Hugginface, ver Anexo IV si se desea descargar el archivo original), se pudo obtener las muestras de la data TIMIT a través de divisiones internas de carpetas de la data y configuraciones detalladas en el mismo. Sobre el archivo Python, se modificó para que pueda mostrar las transcripciones fonéticas de forma horizontal, además de cambiar los nombres de ciertas columnas de la data. Por otro lado, también se definió las divisiones de la data, las cuales pueden ser: Train, Test, Dev, etc. Para el caso de la data TIMIT, esta se compone solo de archivos de entrenamiento (train) y testeo (test) y, por último, se añadió cada muestra de la data al diccionario creado.
3. Como la data tiene diferentes metadatos como: el id del hablante, la ruta de cada archivo de audio, etc. Estas columnas de la data no son importantes al entrenar, por lo tanto, se eliminaron del diccionario para solo obtener las muestras de audio y las transcripciones fonéticas, columnas “audio” y “phoneme” respectivamente.

Después de realizar el procedimiento anterior, la data en formato de diccionario se puede observar a continuación:

```
DatasetDict({
    train: Dataset({
        features: ['audio', 'phoneme'],
        num_rows: 4620
    })
    test: Dataset({
        features: ['audio', 'phoneme'],
        num_rows: 1680
    })
})
```

Figura 17. Data TIMIT.

3.1.1.2 Preparación del Feature Extractor, Tokenizer y la Data

El flujo de trabajo de un ASR se puede detallar en tres etapas, basándose en la documentación del Hugginface [42]:

- Preprocesamiento: En esta etapa, se utiliza el Feature Extractor para transformar una señal de audio en espectrogramas log-Mel.
- En esta etapa intermedia, se utiliza un modelo de clasificación que recibe como entrada los espectrogramas log-Mel y devuelve Predicciones.

- Post-procesamiento: En esta etapa, se utiliza un Tokenizer para cambiar las Predicciones realizadas por el modelo a texto(tokens). Un Tokenizer se puede definir, como un proceso de transformar texto en unidades más pequeñas llamadas “tokens”, donde los tokens pueden ser: palabras, caracteres, fonemas, etc. [43]. Así mismo, el Tokenizer permite a cada token tener un identificador único que lo defina y estos identificadores (números), es lo que devuelve el modelo predictor.

3.1.1.2.1 Feature Extractor

Para este trabajo, se utilizó el Feature Extractor del Whisper.

3.1.1.2.2 Tokenizer

Whisper cuenta con un Tokenizer para usarse con más de 90 lenguajes.

Pero para este trabajo, se entrenó otro Tokenizer que permitió codificar y decodificar las transcripciones fonéticas. El Tokenizer del Whisper se utilizó para otras tareas.

Se utilizo el archivo: “Tokenizers_entrenamiento.ipynb” para entrenar el Tokenizer basado en el archivo: “py_3.1_models_WordLevel_test”. El procedimiento se detalla a continuación:

3.1.1.2.2.1 Entrenamiento del Tokenizer

1. En primer lugar, se guardó el campo “phoneme” de la data en una lista llamada Corpus.
2. A continuación, se definió la tokenizacion previa en división por terminos cada vez que encuentre un espacio en el Corpus, es decir, se dividió por fonemas.
3. Así mismo, se definió el tipo de Tokenizer a nivel de palabras y se utilizó el token “UNK” para fonemas desconocidos, es decir, que no estén en el vocabulario.
4. Se configuró un Trainer a nivel de palabras con tokens especiales descritos a continuación:
    - UNK: Representación para fonemas desconocidos.
    - CLS: Token especial de inicio de secuencia.
    - SEP: Token especial de separación de secuencias.
    - PAD: Token especial para el relleno de secuencias.
    - MASK: Token especial para enmascarar.
5. Por último, se entrenó el Tokenizer con el Trainer y el Corpus.

A este Tokenizer entrenado se lo guardo como tokenizer1.

3.1.1.2.2.1.1 Mapeo del Tokenizer entrenado con el Tokenizer del Whisper

El modelo Whisper está configurado con tokens especiales ya establecidos, además posee un vocabulario de más de 50000 símbolos, por tal motivo, para que no existan problemas al momento de entrenar el modelo, se tuvo que mapear tanto los tokens especiales del tokenizer1, así como su vocabulario con el vocabulario del Tokenizer del Whisper (TW).

El procedimiento fue el siguiente:

1. En primer lugar, se mapeo los tokens especiales del tokenizer1 (UNK Y MASK no se incluyeron) con los tokens especiales del TW, de acuerdo con los mostrado en la Tabla 10, cambiando así los identificadores del tokenizer1 por los del Whisper.

Tabla 10. Mapeo tokens especiales TW y tokenizer1.

Tokens especiales del TW	**Tokens especiales tokenizer1**	**Identificadores de los de los Tokens especiales del TW**
BOS	CLS	50258
EOS	SEP	50257
PAD	PAD	50257

2. Por otro lado, en lo que respecta al mapeo del vocabulario del tokenizer1 con el vocabulario del TW, este vocabulario se lo descargó del Anexo V y se lo exportó a Excel (“Mapeo_Whisper.xlsx”) para que sea más sencilla la búsqueda para el mapeo. En la Figura 18 se puede observar el proceso de mapeo que se realizó entre algunos símbolos del tokenizer1 y del TW para así obtener sus identificadores y modificar estos en el tokenizer1. Si no se pudo realizar el mapeo debido a que algún símbolo del tokenizer1 no se encontró en el vocabulario del TW, se cambió por un símbolo aleatorio del Whisper y se obtuvo su identificador. Así mismo, se analizó que no hubiese identificadores repetidos en el tokenizer1.

WHISPER (tokens)	IDENTIFICADORES DEL WHISPER	TIMITBET
"!"	0	
"\""	1	
"#"	2	
"$"	3	
"%"	4	
"&"	5	tcl
"'"	6	
"("	7	kcl
")"	8	dcl
"*"	9	ax-r
"+"	10	dh
","	11	dx
"-"	12	pcl
"."	13	gcl
"/"	14	bcl
"0"	15	
"1"	16	
"2"	17	
"3"	18	

Figura 18. Mapeo entre los vocabularios del tokenizer1 y Whisper.

Y este tokenizer1 fue el que se usó para codificar y decodificar las transcripciones fonéticas.

3.1.1.2.2.2 Post-procesamiento de tokenizer1

Debido a que el TW tiene un formato requerido para codificar las transcripciones, se le colocó este formato al tokenizer1. El formato es el siguiente:

Identificador del BOS + Identificadores de la transcripción + Identificador del EOS

3.1.1.2.3 Processor

Para simplificar el uso del Feature Extractor y del TW, se puede tener estos dos en uno solo, de nombre Processor. Esta es una característica que ofrece Hugginface (revisar [44]).

3.1.1.2.4 Preparación de la data

La preparación de la data se refiere a la conversión de los vectores de audio de la data a espectrogramas log-Mel, así como la conversión de las trascripciones fonéticas de la data a identificadores.

El procedimiento para preparar la data fue el siguiente:

- En primer lugar, se fue iterando por lotes la data.

- A continuación, se crearon dos nuevas columnas en la data: “input features” y “labels”.
- Se utilizó el Feature Extractor del Processor (FEP), el cual hizo que cada vector de audio (columna “audio”) tenga una longitud de 30 segundos, si el vector de audio tuvo una longitud mayor a 30 segundos, realizó un recorte del audio, y si el vector de audio fue menor a 30 segundos, rellenó el vector de audio con ceros hasta tener la duración máxima. A continuación, el FEP convirtió cada vector de audio en espectrogramas log-Mel usando una frecuencia de muestreo de 16KHz. Estas características se guardaron en “input_features”.
- En lo que respecta a las transcripciones fonéticas (columna “phoneme”) de los audios, se utilizó el tokenizer1 para cambiar cada fonema a su respectivo identificador. Estos identificadores se guardaron en “labels”.
- Por último, se eliminaron las anteriores columnas de la data (“audio” y “phoneme”).

El resultado del procedimiento anterior sobre la data se observa en la Figura 19:

```
DatasetDict({
    train: Dataset({
        features: ['input_features', 'labels'],
        num_rows: 4620
    })
    test: Dataset({
        features: ['input_features', 'labels'],
        num_rows: 1680
    })
})
```

Figura 19. Data TIMIT preprocesada.

3.1.1.3 Entrenamiento

Con la data preparada, se utilizó el Trainer del Hugginface para el entrenamiento [45], pero para poder usarlo, se necesitó definir los siguientes argumentos del Trainer:

- **Definición de un DataCollator:** Para entrenar un modelo, es necesario que los datos se encuentren en formato de tensores, por lo que, un colector de datos (DataCollator) permitió tomar la data preprocesada y convertir los datos en tensores [46] y agregarles un PAD si así lo necesitaban.
- **Definición de una métrica de evaluación:** Para evaluar el modelo, se necesita una métrica, por lo que, en este trabajo se utilizó WER.

- **Cargar un modelo preentrenado:** Para el proceso de entrenamiento, es necesario tener un modelo preentrenado, para lo cual, se usó el modelo Whisper-Small.

- **Definición de los hiperparámetros del entrenamiento:** Para entrenar el modelo, se necesitó definir hiperparámetros.

3.1.1.3.1 Definición del DataCollator

El DataCollator o colector de datos como se explicó anteriormente permite tomar los datos y convertirlos en tensores.

El colector de datos tomó cada columna de la data por separado, es decir, los “inputs features” fueron cambiados a tensores por el FEP y los “labels” fueron cambiados a tensores por el Tokenizer del Processor.

Los “inputs_features” ya fueron rellenados a una duración de 30 segundos en la Sección 3.1.1.2.4, por lo que no fue necesario aplicar un relleno, lo único que se realizó fue cambiarlos a tensores.

A los “labels” si fue necesario realizar un relleno con el identificador del token PAD a cada secuencia de identificadores a la longitud máxima entre todas las secuencias que existen en la data, así mismo, se realizó el cambio a tensores. A continuación, se reemplazó cada identificador del token PAD por -100, esto para cuando se calcule la perdida, esta no se vea afectada.

Por último, se eliminó el identificador del token de inicio de la transcripción (conocido como BOS) de todos los “labels”.

3.1.1.3.2 Definición de la métrica de evaluación

La métrica de evaluación tomada para este trabajo fue el WER, la cual es el número de errores dividido por el total de palabras y permite conocer cuál es el rendimiento del modelo, mientras menor sea el porcentaje mejor funciona el modelo como un ASR [47].

Para calcular WER se realizó lo siguiente:

- En primer lugar, se tomó las Predicciones del modelo para calcular el modelo. Estas Predicciones se componen de lo que el modelo debió predecir (“labels”) y lo predicho por el modelo.

- A continuacion, se reemplazó el -100 que se colocó a “labels” en la Sección 3.1.1.2.4 por el identificador del token PAD usando el Tokenizer del Processor.

- Así mismo, se decodificó las Predicciones para obtener la transcripción fonética en lugar de los Identificadores utilizando en tokenizer1.
- Por último, se calculó el WER utilizando estas Predicciones.

3.1.1.3.3 Cargar un modelo preentrenado:

El modelo preentrenado utilizado para realizar el transfer learning fue el Whisper-Small.

Para que se trate de un transfer learning, se congelaron los parámetros del codificador del modelo y se dejó libre los parámetros del decodificador para que el modelo pueda entrenarse.

3.1.1.3.4 Definición de los hiperparámetros del entrenamiento.

Algunos de los hiperparámetros relacionados con el entrenamiento se describen a continuación:

- Directorio de salida: Se llamo al modelo entrenado “Whisper-Small-TF-TIMIT” y se guardó localmente en Kaggle y se subió los resultados del entrenamiento a Hugginface con el mismo nombre.
- Tamaño del lote de entrenamiento (BT: Batch Train): Se definió en 32.
- Tamaño del lote de evaluación (BE: Batch Evaluation): Se definió en 8.
- Acumulación de gradiente por pasos (GA: Gradient Accumulation): Se definió el escalar en 1.
- Tasa de aprendizaje: Se uso una tasa de 1e-5.
- Estrategia de evaluación: La estrategia que se utilizó fue por “pasos” y se definió el número total de pasos de entrenamiento (MS: Max Steps) en 3000 y el número de épocas por estos pasos se puede calcular con la Ecuación 5. Además, cada 500 pasos se guardó un punto de control del modelo y se realizó la evaluación del modelo usando la métrica WER.

$$\textbf{número de épocas} = \left\lfloor \frac{\text{MS}}{\text{número de pasos por época}} \right\rfloor + \text{PE}$$

Ecuación 5. Número de épocas [48].

Donde:

$$\text{PE} = \begin{cases} 1, & (\text{MS } \mathbf{mod} \text{ número de pasos por época }) > 0 \\ 0, & (\text{MS } \mathbf{mod} \text{ número de pasos por época }) = 0 \end{cases}$$

Y,

$$\mathbf{número\ de\ pasos\ por\ época} = \left\lfloor \frac{\left\lfloor \frac{\mathbf{número\ de\ muestras\ de\ "train"\ de\ la\ data}}{\mathbf{BT}} \right\rfloor}{\mathbf{GA}} \right\rfloor$$

A modo de ejemplo para ilustrar el cálculo del número de épocas (Ecuación 5), se considerará los hiperparámetros que ya están establecidos. Entonces el cálculo se detalla a continuacion:

$$\text{número de pasos por época} = \left\lfloor \frac{\left\lfloor \frac{4620}{32} \right\rfloor}{1} \right\rfloor = 144$$

$$\text{PE} = (3000 \ \mathbf{mod}\ 144\ ) = (120 > 0) = 1$$

$$\mathbf{número\ de\ épocas} = \left\lfloor \frac{3000}{144} \right\rfloor + 1 = 21$$

Por otro lado, los resultados del modelo se pueden observar en la Sección 4.1.2.1.1.

3.1.2 Transfer learning usando la data TIMIT y FLEURS

Para este primer transfer learning, se utilizó la data TIMIT y FLEURS y se lo realizó a través del archivo: “Transfer_learning_Timit_Fleurs.ipynb”. El procedimiento fue similar al transfer learning de la Sección 3.1.1.

3.1.2.1 Lectura de Data

Se realizó el mismo procedimiento que la Sección 3.1.1.1 pero en la llave “train” del diccionario se guardaron todas las muestras de la data TIMIT (“train” y “test”) con el mismo archivo “timit_asr.py” y en llave “test” se guardaron las muestras de la data FLEURS (“train” y “test”) con el archivo en línea **"**google/fleurs**”**, debido a que la data FLEURS se encuentra online en el repositorio de Hugginface [34]. Las columnas que se dejaron en la data fueron “audio” y “transcription”.

La data en formato de diccionario se puede observar a continuación:

```
DatasetDict({
    train: Dataset({
        features: ['audio', 'transcription'],
        num_rows: 6300
    })
    test: Dataset({
        features: ['audio', 'transcription'],
        num_rows: 3249
    })
})
```

Figura 20. Datas TIMIT Y FLEURS.

3.1.2.2 Conversión de las transcripciones de la data a fonemas (IPA)

Como se puede observar en la Figura, el segundo campo que se tiene en la data es la transcripción del audio (“transcription”), debido a que, aunque la data TIMIT si tiene un campo con la transcripción fonética con el alfabeto TIMITBET (columna “phoneme” en la Sección 3.1.2.1), FLEURS no cuenta con esto, por tal motivo, se cambió la columna “transcription” de la data a un alfabeto comúnmente utilizado como IPA a través de “phonemizer”, el cuál es una herramienta para convertir texto en fonemas usando distintos backends [49]. Un backend es un software para cambiar el texto a fonemas en diferentes alfabetos fonéticos. El backend seleccionado fue “eSpeak” con lenguaje inglés para cambiar las transcripciones de la data a IPA. Además, se configuró para que las transcripciones fonéticas resultantes se separen cada fonema por un espacio y se guarden en una nueva columna creada de nombre “phoneme”. La columna “transcription” se eliminó.

La data quedo de la siguiente manera:

```
DatasetDict({
    train: Dataset({
        features: ['audio', 'phoneme'],
        num_rows: 6300
    })
    test: Dataset({
        features: ['audio', 'phoneme'],
        num_rows: 3249
    })
})
```

Figura 21. Datas TIMIT Y FLEURS con la transcripción fonética (columna “phoneme”).

3.1.2.3 Preparación del Feature Extractor, Tokenizer y la data

Se realizo el mismo procedimiento que la Sección 3.1.1.2. Sobre la Sección del Tokenizer (ver Sección 3.1.1.2.2), se entrenó otro Tokenizer (tokenizer2) siguiendo los mismos pasos y utilizando el mismo archivo Excel, solo creando otra columna: IPA.

La data quedo de la siguiente forma:

```
DatasetDict({
    train: Dataset({
        features: ['input_features', 'labels'],
        num_rows: 6300
    })
    test: Dataset({
        features: ['input_features', 'labels'],
        num_rows: 3249
    })
})
```

Figura 22. Data preparada para el entrenamiento.

3.1.2.4 Data normalizada

También se entrenó la data normalizada siguiendo los pasos que se detallan a continuación:

1. En primer lugar, se cambió a tensores todos los elementos de la data que se muestra en la Figura en la Sección 3.1.2.3. Para la normalización solo se utilizó la columna “input_features” de la data.
2. A continuación, se tomaron los elementos de la columna “input_features” del “train” de la data para calcular el promedio (mean) y desviación estándar (stdv).
3. Con el promedio y desviación estándar calculados, se utilizó la ecuación1 para realizar la normalización en todos los elementos de la columna “inputs_features” de la data.

$$Normalización = \frac{input\_features - mean}{stdv}$$

Ecuación 6. Normalización de la data.

4. Estos elementos normalizados se guardaron en una nueva columna llamada “input_features_n”.
5. Finalmente, se eliminó la columna “input_features” y los datos se los cambio a su forma original (sin tensores). La data se muestra en la Figura 23:

```
DatasetDict({
    train: Dataset({
        features: ['labels', 'input_features_n'],
        num_rows: 6300
    })
    test: Dataset({
        features: ['labels', 'input_features_n'],
        num_rows: 3249
    })
})
```

Figura 23. Data normalizada.

El archivo del transfer learning con la data normalizada se llama "Transfer_learning_Timit_Fleurs_Nornalizadas.ipynb".

3.1.2.5 Entrenamiento

Para la data normalizada y sin normalizar se realizó el mismo procedimiento que la Sección 3.1.1.4 pero se cambiaron algunos de los parámetros detallados en la Sección 3.1.1.4.4 por los siguientes valores:

- Tamaño del lote de entrenamiento: Se definió en 16.
- Tamaño del lote de evaluación: Se definió en 8.

El modelo entrenado con la data sin normalizar se llamó: "Whisper-Small-TF-Timit-Fleur"

Y al modelo entrenado con la data normalizada se llamó: "Whisper-Small-TF-Timit-Fleur-Normalizada"

Los resultados de los modelos se pueden observar en las secciones 4.1.2.1.2 y 4.1.2.1.3.

3.2 Experimento 2: Evaluación de los errores de pronunciación sobre la data "L2-ARTIC corpus" usando el modelo "wav2vec2-xlsr-53-espeak-cv-ft"

Para realizar este experimento, se hizo uso de las siguientes herramientas:

- **Entorno:** Kaggle
- **Data:** L2-ARTIC corpus (esta data se puede encontrar en Kaggle Datasets con el nombre "Corpus_Experimentos")
- **Lenguaje de programación:** Python

El archivo utilizado para esta evaluación fue el siguiente: “Transfer_learning_Timit_Fleurs_Nornalizadas.ipynb”. El procedimiento de la evaluación se detalla a continuación:

3.2.1 Lectura de la data

La data que se usó para esta evaluación fue “L2-ARCTIC corpus”, de la cual se seleccionaron personas cuya lengua materna es el español y pronunciaron frases en inglés. Los directorios que se usaron de los hispanohablantes fueron: /annotation, que se compone de archivos TextGrid (ver Figura 7 para ver el contenido del TextGrid), y /wav, que se compone de archivos de audios.

La data se dividió en tres listas de la siguiente manera:

- **Referencia 1:** Esta lista se formó a partir de la conversión a fonemas del nivel words de cada TextGrid, usando el alfabeto IPA.
- **Referencia 2:** Esta lista se formó a partir de Referencia 1 con la unión de los errores de pronunciación del nivel IPA de cada TextGrid.
- **Predicciones:** Esta lista se formó a partir de la predicción de cada archivo de audio usando el modelo “wav2vec2-xlsr-53-espeak-cv-ft”.

Tanto Referencia 1, Referencia 2 y Predicciones utilizaron IPA para representar los fonemas.

3.2.1.1 Obtención de Referencia 1

Para obtener las frases de Referencia 1, se realizó lo siguiente:

- Se obtuvo las ubicaciones de cada TextGrid de la carpeta /annotation de cada hispanohablante.
- A continuación, se utilizó el phonemizer para convertir el nivel words de cada TextGrid en fonemas (alfabeto IPA) y el resultado se fue guardando en Referencia 1.

3.2.1.2 Obtención de Referencia 2

Para obtener las frases de Referencia 2, se realizó lo siguiente:

- En primer lugar, se guardaron los errores de pronunciación del nivel IPA de cada TextGrid en una lista (“Errores”). Un ejemplo de los errores de pronunciación que cometió un hispanohablante de una frase al azar se muestra en la Figura 24. El primer elemento es el error de pronunciación que cometió el hablante, el segundo elemento es el tipo de error y el tercer elemento es la posición del error en la frase usando el nivel phones. En este nivel está la transcripción fonética con los errores de pronunciación, en formato Arpabet, pero sirve para obtener la posición en donde debe ubicarse cada error en Referencia 1

```
['sil,d,11', 'sil,d,12', 'ɪ,a,33', 'h,s,37']
```

Figura 24. Ejemplo de los errores de pronunciación que cometió un hispanohablante.

- Así mismo, como se puede observar en la Figura 24, el fonema que se coloca cuando existe un error de eliminación en el TextGrid es “sil”, pero este se cambió por el símbolo “|” para poder realizar la evaluación en etapas posteriores. Así mismo, se eliminaron símbolos relacionados con el acento y aspiración, denotados por los símbolos “*” y “ʰ” respectivamente, debido a que no son necesarios en la evaluación.
- A continuación, se colocaron los errores de cada frase de la lista Errores en Referencia 1, pero antes se copió el contenido de Referencia 1 a otra lista, para así no perder Referencia 1.

3.2.1.3 Obtención de las Predicciones

Para obtener las frases de Predicciones, se realizó lo siguiente:

- Se obtuvo las rutas de los audios de cada carpeta /wav del directorio de cada hispanohablante.
- Con las rutas de cada archivo de audio, se realizó la predicción usando el modelo “wav2vec2-xlsr-53-espeak-cv-ft” y el idioma inglés. Cada predicción se guardó en la lista Predicciones.

3.2.2 Eliminación de casos especiales de Referencia 1, Referencia 2 y Predicciones

Existen casos especiales en los archivos TextGrid que se leyeron para crear las listas Referencia 1 y Referencia 2, los cuales fueron:

- La anotación de un error de pronunciación del nivel phones no está presente en el nivel IPA, entonces no se sabe si existe un error o no. También puede suceder que en el nivel phones no esté presente la anotación del error.
- Frases de Referencia 1 y el nivel phones no tienen el mismo número de fonemas, entonces no se pudo colocar correctamente los errores.

Estos casos se eliminaron de las 3 listas para tener el mismo número de frases.

3.2.3 Normalización de Referencia 1, Referencia 2 y Predicciones

Debido a que debe de haber una similitud en cómo se presentan las frases entre cada lista, se realizaron normalizaciones para todas las listas, las cuales se detallan a continuación:

1. Se cambiaron fonemas de 2 o más caracteres a códigos (Ver Tabla 11) debido a que, al momento de realizar la alineación, se compararon carácter por carácter las frases de las listas.

Tabla 11. Códigos para fonemas con 2 o más caracteres.

Fonemas	Códigos
/aʊ/	1
/aɪ/	2
/eɪ/	3
/oʊ/	4
/ɔɪ/	5
/ts/	6
/t͡s/	6
/dz/	7
/d͡z/	7
/tʃ/	8
/t͡ʃ/	8
/dʒ/	9
/d͡ʒ/	9
/err/	0
/əɹ/	ə
/ɛɹ/	ɛ
/ɪɹ/	ɪ

2. Se cambiaron ciertos fonemas del inglés por fonemas similares (Ver Tabla 12) para reducir el número de fonemas y conseguir una mayor flexibilidad al momento de evaluar. Por un lado, los fonemas similares abarcan fonemas del inglés percibidos

como similares por hispanohablantes, por ejemplo, /æ/ como /a/. Por otro lado, los fonemas similares también abarcan alófonos, por ejemplo, /ɣ/ como /g/.

Tabla 12. Fonemas similares.

Fonemas	Fonemas similares
/ʔ/	/t/
/ɬ/	/ə/
/ʝ/	/j/
/β/	/b/
/ɣ/	/g/
/ɦ/	/h/
/ʎ/	/j/
/ʒ/	/z/
/ɟ/	/g/
/ʊ/	/u/
/i/	/ɪ/
/æ/	/a/
/ʌ/	/ə/

3. Por último, se convirtieron las frases de las listas en cadenas sin separaciones entre los fonemas para poder realizar la alineación. Además, se eliminó el símbolo ˈ de las listas, el cual no es necesario para la evaluación, por tal motivo, se descartó este símbolo.

3.2.4 Alineaciones entre Referencia 1, Referencia 2 y Predicciones

Para obtener los errores de pronunciación, se realizó tres alineaciones entre las listas. Se utilizó la alineación global a través del algoritmo Needleman-Wunsh. Las tres alineaciones se detallan a continuación:

3.2.4.1 Alineación entre Referencia 1 y Referencia 2

Esta alineación entre Referencia 1 y Referencia 2 sirvió como referencia para conocer donde deben estar los errores de pronunciación.

Para calcular la alineación entre Referencia 1 y Referencia 2, se fue iterando cada elemento (frases) de las dos listas al mismo tiempo y se utilizó el algoritmo de Needleman-Wunsh para la alineación.

El resultado de un elemento de la alineación entre Referencia 1 y Referencia 2 se muestra en la Figura 25.

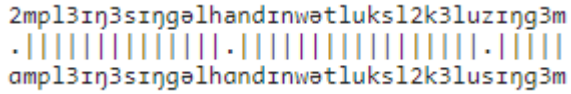

Figura 25. Ejemplo de un elemento de la alineación entre Referencia 1 y Referencia 2.

A esta alineación se la llamó Alineación 1.

3.2.4.1.1 Eliminación del caso especial de Alineación 1

El símbolo "|" significa que hubo un error de eliminación, por tal motivo, si la segunda frase de la Alineación 1 tiene el símbolo "-", eso significa que hubo un error en la alineación de esas frases. Por tal motivo, se eliminaron estas frases tanto de Alineación 1 como de las listas Referencia 1, Referencia 2 y Predicciones, para poder realizar la evaluación de forma correcta.

3.2.4.1.2 Cambio de Alineación 1 a fonemas individuales

Como se pudo observar en la Figura 26, Alineación 1 tiene las frases con los fonemas como una solo cadena, sin ninguna separación, por tal motivo, se dividió cada fonema como un elemento individual, para poder obtener los errores de pronunciación.

Entonces el ejemplo de la Figura. se vería así:

```
[['2', 'm', 'p', 'l', '3', 'ɪ', 'ŋ', '3', 's', 'ɪ', 'ŋ', 'g', 'ə', 'l', 'h', 'a', 'n', 'd', 'ɪ', 'n',
'ə', 't', 'l', 'u', 'k', 's', 'l', '2', 'k', '3', 'l', 'u', 'z', 'ɪ', 'ŋ', 'g', '3', 'm'], ['a', 'm',
'l', '3', 'ɪ', 'ŋ', '3', 's', 'ɪ', 'ŋ', 'g', 'ə', 'l', 'h', 'a', 'n', 'd', 'ɪ', 'n', 'w', 'ə', 't', 'l
'k', 's', 'l', '2', 'k', '3', 'l', 'u', 's', 'ɪ', 'ŋ', 'g', '3', 'm']]
```

Figura 26. Ejemplo de un elemento de Alineación 1 con fonemas separados.

3.2.4.1.3 Obtención de los tipos de errores de pronunciación para Alineación 1

Para el calculó de los errores de pronunciación, se comparó cada fonema de la primera frase (Fonema 1) con cada fonema de la segunda frase (para fines prácticos se llamará Fonema 2) de cada elemento de Alineación 1. Entonces se obtuvo que:

- **Error de eliminación:** Si Fonema 2 es igual al símbolo "|", entonces en la segunda frase se guardó la etiqueta: <eliminación>: Fonema 1.
- **Error de inserción:** Si Fonema 1 es un guion (-), entonces en la segunda frase se guardó la etiqueta: <inserción>: Fonema 2.
- **Error de sustitución:** Si Fonema 1 y Fonema 2 son diferentes, entonces en la segunda frase se guardó la etiqueta: <sustitución>: Fonema 2.

La Alineación 1 muestra los errores como los no errores que el modelo tuvo que predecir. Es decir, es la referencia a la cual las Predicciones debieron de llegar. Un ejemplo de un elemento de la Alineación 1 con las etiquetas de los errores se puede observar en la Figura 27.

```
[['a', 'n', 'd', 't', 'w', 'ɛ', 'n', 't', 'ɪ', 'm', 'ɛ', 'n', 'k', 'u', 'd', 'h', '4', 'l', 'd', 'ɪ', 't', 'w', 'ɪ', 'ð',
's', 'p', 'ɪ', 'ɹ', 'z', 'a', 'n', '-', 'd', 'a', 'ɹ', '-', '4', 'z'], ['a', 'n', '<eliminación>:d', 't', 'w', 'ɛ', 'n', 't',
'ɪ', 'm', 'ɛ', 'n', 'k', 'u', '<eliminación>:d', 'h', '4', 'l', '<eliminación>:d', 'ɪ', 't', 'w', 'ɪ', '<sustitución>:t',
's', 'p', 'ɪ', 'ɹ', 'z', 'a', 'n', '<inserción>:ɛ', 'd', '<sustitución>:d', '<sustitución>:ɑ', '<inserción>:ɔ', '4', 'z']]
```

Figura 27. Ejemplo de un elemento de Alineación 1 con las etiquetas de los errores.

3.2.4.2 Alineaciones entre Referencia 2 y Predicciones

Se realizó el mismo procedimiento que la Sección 3.2.4.1 pero con las listas Referencia 2 y Predicciones.

A esta alineación, se la llamo Alineación 2 y en la Figura 28 se muestran un elemento de esta.

```
ampl3ɪŋ3sɪŋgəlhandɪnwətluksl2k3lusɪŋg3m
.|||.||.||-|.||.|.|||.|||||||.||||||||
ampl2ɪŋesɪ-gulhantɪnwətluksl2kelusɪŋg3m
```

Figura 28. Ejemplo de un elemento de la alineación entre Referencia 2 y Predicciones.

3.2.4.2.1 Cambio de Alineación 2 a fonemas individuales

Se realizó el mismo proceso que la Sección 3.2.4.1.2. Un ejemplo de un elemento de la Alineación 2, se puede observar en la Figura 29.

```
[['ɑ', 'm', 'p', 'l', '3', 'ɪ', 'ŋ', '3', 's', 'ɪ', 'ŋ', 'g', 'ə', 'l', 'h', 'ɑ', 'n', 'd', 'ɪ', 'n', 'w', 'ə', 't', 'l',
'u', 'k', 's', 'l', '2', 'k', '3', 'l', 'u', 's', 'ɪ', 'ŋ', 'g', '3', 'm'], ['a', 'm', 'p', 'l', '2', 'ɪ', 'ŋ', 'e', 's',
'ɪ', '-', 'g', 'u', 'l', 'h', 'a', 'n', 't', 'ɪ', 'n', 'w', 'a', 't', 'l', 'u', 'k', 's', 'l', '2', 'k', 'e', 'l', 'u', 's',
'ɪ', 'ŋ', 'g', '3', 'm']]
```

Figura 29.- Ejemplo de un elemento de Alineación 2 con fonemas separados.

3.2.4.2.2 Eliminación de errores de inserción de Alineación 2 para realizar la alineación con Referencia 1

Con Alineación 2 se calculó el error de inserción tomando como referencia la Sección 3.2.4.1.3, donde se explica como calcular el error de inserción. Pero para esta Alineación 2 el error de inserción que se obtuvo es una equivocación que el modelo predictor realizó, es decir, añadió una inserción incorrectamente, ya que Alineación 2 es una alineación entre

Referencia 2 y Predicciones. por tal motivo, se guardó con la etiqueta <error-inserción> en cada segunda frase de cada elemento de Alineación 2 si existía el error.

A continuación, se contaron cuantos casos de <error-inserción> existieron y se guardó el conteo en una variable de nombre FP_inserción.

Por último, se eliminó estos casos de las segundas frases de cada elemento de Alineación 2 y se guardaron estas frases, que fueron convertidas otra vez a una cadena de fonemas sin separación, en una lista de nombre Predicciones_a, la cual sirvió para realizar la alineación de la Sección 3.2.4.3 y permitió conocer principalmente si los errores de inserción predichos por el modelo fueron correctos o incorrectos en relación con Referencia 2. Así mismo, Predicciones_a cambió el símbolo"-" por "|" para poder realizar la alineación con Referencia 1.

3.2.4.3 Alineaciones entre Referencia 1 y Predicciones_a

Se realizó el mismo procedimiento que la Sección 3.2.4.1 pero con las listas Referencia 1 y Predicciones_a.

A esta alineación, se la llamo Alineación 3 y en la Figura 30, se muestra un elemento de esta.

```
2mpl3ɪŋ3sɪŋgəlhandɪnwətluksl2k3luzɪŋg3m
.|||.||.||.|.||||.|||.||||||||.||.|||||
ampl2ɪŋesɪ|gulhantɪnwatluksl2kelusɪŋg3m
```

Figura 30. Ejemplo de un elemento de la alineación entre Referencia 2 y Predicciones_a.

3.2.4.3.1 Cambio de Alineación 3 a fonemas individuales

Se realizó el mismo procedimiento que la Sección 3.2.4.1.2 Un ejemplo de un elemento de la Alineación 3, se puede observar en la Figura 31.

```
[['2', 'm', 'p', 'l', '3', 'ɪ', 'ŋ', '3', 's', 'ɪ', 'ŋ', 'g', 'ə', 'l', 'h', 'a', 'n', 'd', 'ɪ', 'n', 'w', 'ə', 't', 'l',
'u', 'k', 's', 'l', '2', 'k', '3', 'l', 'u', 'z', 'ɪ', 'ŋ', 'g', '3', 'm'], ['a', 'm', 'p', 'l', '2', 'ɪ', 'ŋ', 'e', 's',
'ɪ', '|', 'g', 'u', 'l', 'h', 'a', 'n', 't', 'ɪ', 'n', 'w', 'a', 't', 'l', 'u', 'k', 's', 'l', '2', 'k', 'e', 'l', 'u', 's',
'ɪ', 'ŋ', 'g', '3', 'm']]
```

Figura 31. Ejemplo de un elemento de Alineación 3 con fonemas separados.

3.2.4.3.2 Obtención de los tipos de errores de pronunciación para Alineación 3

Se llevó a cabo el mismo procedimiento que la Sección 3.2.4.1.3. Un ejemplo de un elemento de la Alineación 3, se puede observar en la Figura 32.

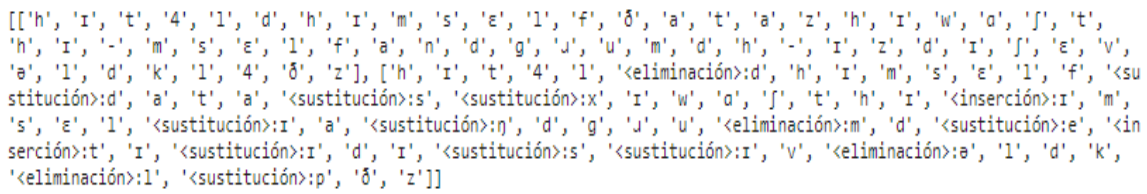

Figura 32. Ejemplo de un elemento de Alineación 3 con los errores de pronunciación.

3.2.5 Métricas de evaluación para evaluar los errores de pronunciación a nivel fonético

Se calcularon las mismas métricas que la Sección 2.1.3 y así mismo se calcularon los TP, FN, FP y TN entre Alineación 1 y Alineación 3, donde Alineación 1 sirvió para tener la referencia de los errores y no errores que el modelo debió predecir y la Alineación 3 que muestra los errores y no errores que predijo el modelo.

3.2.5.1 Eliminación

Para calcular TP, FN, FP y TN se realizó el mismo procedimiento de la Sección 2.1.3.1 pero enfocado a nivel fonético y con la etiqueta <eliminación>: Fonema 1.

Los resultados de las métricas de evaluación y de los TP, FN, FP y TN se encuentran en la Sección 4.1.2.2.1.

3.2.5.2 Inserción

Se realizó el mismo procedimiento que la Sección 2.1.3.2, pero enfocado a nivel fonético y se realizaron algunos cambios, los cuales fueron los siguientes:

- Intercambio de la etiqueta <inserción>: Palabra 2 por <inserción>: Fonema 2.
- Los casos de FN se obtuvieron de la Sección de 3.2.4.2.2 (FP_inserción).
- No hay que tomar en cuenta en los casos de TN, los casos de FP_inserción de la Sección 3.2.4.2.2.

Los resultados de las métricas de evaluación y de los TP, FN, FP y TN se encuentran en la Sección 4.1.2.2.2.

3.2.5.3 Sustitución

Se realizó lo mismo que la Sección 2.1.3.3, pero enfocado a nivel fonético y se intercambió la etiqueta <sustitución>: Palabra 2 por <sustitución>: Fonema 2.

Los resultados de las métricas de evaluación y de los TP, FN, FP y TN se encuentran en la Sección 4.1.2.2.3.

3.2.6 Cálculo del número de fonemas predichos correctamente por el modelo "wav2vec2-xlsr-53-espeak-cv-ft"

Además de las métricas anteriormente calculadas, también se realizó el cálculo del número de fonemas que el modelo predijo correctamente a través de los siguientes pasos:

1. En primer lugar, se utilizó la Alineación 2, pero sin haber quitado los errores de inserción cometidos por el modelo, debido a que si son necesarios en el cálculo.

2. A continuación, se calculó la matriz de confusión de la Alineación 2, la cual se guardó en un CSV de nombre: "Matriz_confusion_predicciones.csv". La matriz de confusión permite comparar las clases originales a las que pertenecen los datos con las predicciones a través de una distribución por clases [50]. En este caso, las clases serían los fonemas, en las columnas estarían las "clases predichas" y en las filas las "clases esperadas".

3. Entonces para cada fonema se calculó el Accuracy, para lo cual, se obtuvo el número de aciertos que tuvo la predicción sobre el fonema (elemento actual en la diagonal principal de la matriz) y se dividió para la suma de todos los valores de su columna. La fórmula de Accuracy se muestra en la Ecuación 7.

$$Accuracy\ de\ un\ fonema = \frac{Numero\ de\ aciertos\ del\ modelo\ sobre\ el\ fonema}{Suma\ de\ los\ valores\ de\ la\ columna\ del\ fonema}$$

Ecuación 7. Formula de Accuracy de un fonema.

4. Con el Accuracy de cada fonema se armó una matriz y a través de esta matriz se pudo conocer cuántos fonemas fueron predichos correctamente con cierto porcentaje de error. Los resultados de la matriz de Accuracy y el número de los fonemas correctamente predichos se analizaron en la Sección 4.1.2.2.4.

4 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

4.1 Resultados

4.1.1 Resultados de la evaluación de los errores de pronunciación usando el modelo Whisper-Large-V2

Los resultados de las métricas de evaluación de la Sección 2.3.4 para cada tipo de error se describen a continuación:

4.1.1.1 Error de eliminación

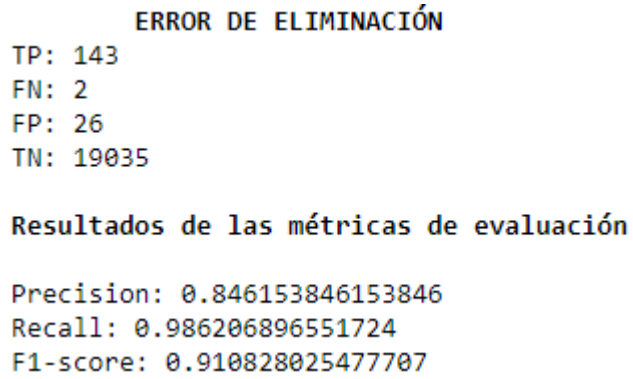

Figura 33. Resultados de las métricas para el error de eliminación.

Como se puede observar en la Figura 33, la Precisión del error de eliminación es del 84%, debido a que el modelo tiene pocos casos de FP, y en cuanto a los TP, el modelo predijo la mayoría de ellos correctamente. Por otro lado, en cuanto al Recall, se obtuvo el 98%, debido a que el modelo presenta pocos casos de FN y predijo la mayoría de TP correctamente. Por último, la métrica F1-score tiene el 91% de desempeño. Los resultados de estas métricas demuestran que el modelo detecta la mayoría de los errores de eliminación correctamente.

4.1.1.2 Error de inserción

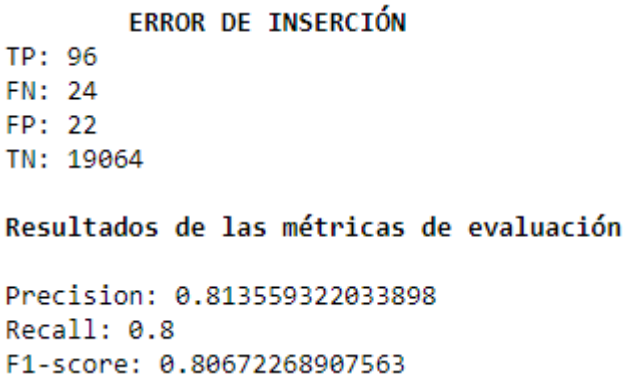

Figura 34. Resultados de las métricas para el error de inserción.

Como se puede observar en la Figura 34, la precisión del error de inserción es del 81%, debido a que el modelo tiene muy pocos casos de FP y los TP los predijo correctamente la mayoría de ellos. Por otro lado, en cuanto al Recall, se obtuvo el 80% de acierto, debido a que el modelo presenta pocos casos de FN y una gran cantidad de TP. Por último, la métrica F1-score tiene el 80% de desempeño. Los resultados de estas métricas demuestran que el modelo detecta la mayoría de los errores de inserción correctamente.

4.1.1.3 Error de sustitución

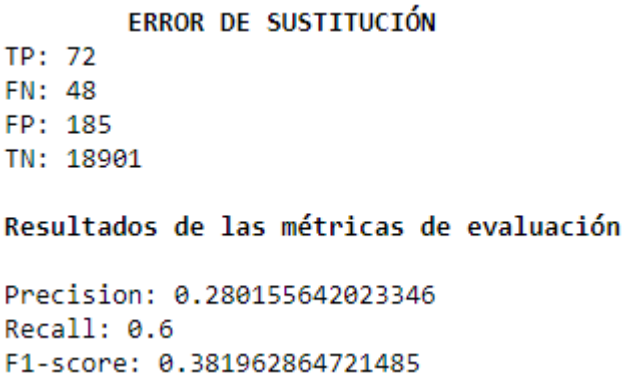

Figura 35. Resultados de las métricas para el error de sustitución.

Como se puede observar en la Figura 35, la precisión del error de sustitución es del 28%, debido a que el modelo tiene muy muchos casos de FP y sobrepasa el número de TP. Por otro lado, en cuanto al Recall, se obtuvo el 60%, debido una cantidad considerable de FN. Por último, la métrica F1-score tiene el 38% de desempeño. Los resultados obtenidos de

estas métricas sugieren que el modelo no logra identificar la mayoría de los errores de sustitución correctamente.

Por esta razón, se realizó el histograma de la Figura 36, para observar el número de casos de FN y FP, si se suma todas las barras se tendrá el número de FN+FP. A través de este gráfico, se puede conocer en qué se equivocó el modelo. Si se observa la primera barra se tiene 150 palabras con el caso 0-guiones, el cual significa que el modelo cambió una letra (generalmente) o más letras de una palabra. El siguiente caso es 1-guión, este caso implica que el modelo eliminó una letra de la palabra que debía de predecir. Los casos de 2, 3 y 4-guiones funcionan como el caso de 1-guión, donde se eliminaron de las palabras 2, 3 y 4 letras respectivamente. Así mismo, como se puede observar en el gráfico, existen muy pocas predicciones que cumplan estos últimos 3 casos, lo que quiere decir que el modelo se equivocó en su mayoría en el cambio de letras de una palabra (caso 0-guiones) y la eliminación de una letra de una palabra (caso 1-guion). A continuación, se mostrarán algunos ejemplos de los casos 0-guiones y caso 1-guión en la Tabla 13.

Tabla 13. Ejemplos de los casos 0-guiones y 1-guión.

EJEMPLOS		
Referencia	**Predicción**	**Caso**
Carole	Carol-	1-guion
trescientas	trescientos	0-guiones
dieciseis	dieciséis	0-guiones

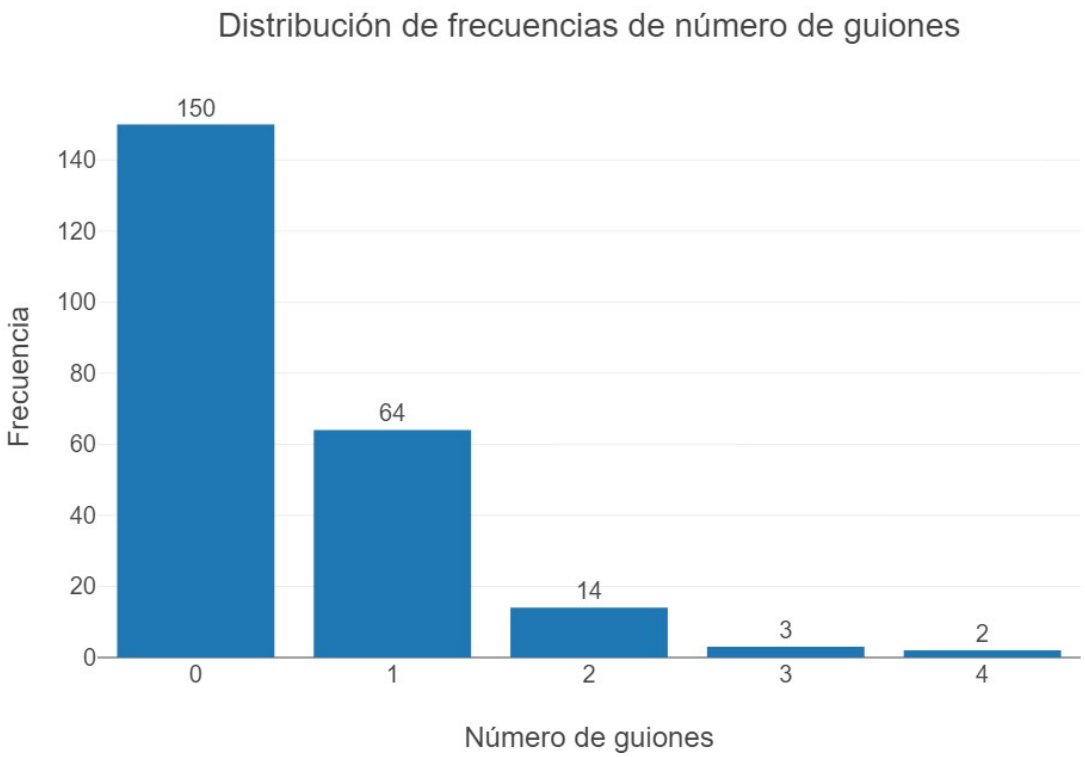

Figura 36. Equivocaciones de las Predicciones sobre el error de sustitución.

4.1.2 Resultados de los experimentos preliminares

4.1.2.1 Resultados del Experimento 1

4.1.2.1.1 Resultados del transfer learning con la data TIMIT

Training Loss	Epoch	Step	Validation Loss	Wer
0.3408	3.45	500	0.3994	83.6838
0.2057	6.9	1000	0.4079	92.3470
0.0616	10.34	1500	0.5076	94.2053
0.023	13.79	2000	0.5998	95.3184
0.0043	17.24	2500	0.6825	97.1284
0.0023	20.69	3000	0.7104	98.0856

Figura 37. Resultados del transfer learning con la data TIMIT.

Como se puede observar en la Figura 37, el “training loss” disminuye al avanzar el número de épocas, lo que refleja que el modelo se ajusta a los datos del entrenamiento. Por otro lado, el “test loss” empieza con un los relativamente bajo, pero al avanzar el número de épocas, comienza a aumentar, lo que quiere decir que el modelo no se ajusta a los datos de testeo. Por último, el WER también comienza a aumentar al pasar las épocas, obteniendo al final un WER del 98%.

4.1.2.1.2 Resultados del transfer learning con la data TIMIT y FLEURS

Training Loss	Epoch	Step	Validation Loss	Wer
0.4965	1.27	500	0.9304	37.3857
0.1668	2.54	1000	0.8561	32.7384
0.069	3.81	1500	0.8093	52.7441
0.0152	5.08	2000	0.9021	54.9437
0.0083	6.35	2500	0.8471	57.3611
0.0021	7.61	3000	0.8885	35.0461

Figura 38. Resultados del transfer learning con la data TIMIT y FLEURS.

Debido a los resultados del transfer learning anterior (ver Sección 4.1.2.1.1), los cuales reflejan que podría haber un overfitting sobre la data TIMIT, se entrenó otro modelo utilizando otra data para el testo (Data FLEURS) y como se puede observar en la Figura 38, el training loss disminuye al avanzar el número de épocas, lo que refleja que el modelo se ajustó a los datos del entrenamiento. Por otro lado, el “test loss” empieza con un loss alto y al avanzar el número de épocas, comienza a fluctuar y se mantiene siempre con un loss alto, lo que quiere decir que el modelo no se ajustó a los datos de testeo. Por último, el WER hasta el step 1000 (2.54 épocas) disminuye, pero comienza a aumentar al avanzar el número de épocas, aunque al final se obtenga un WER de 35% no refleja que el modelo sea adecuado para la tarea de evaluar la pronunciación.

4.1.2.1.3 Resultados del transfer learning con la data TIMIT y FLEURS normalizadas.

Training Loss	Epoch	Step	Validation Loss	Wer
0.5923	1.27	500	0.9379	98.7612
0.1823	2.54	1000	0.6721	89.3262
0.0852	3.81	1500	0.6534	86.1141
0.0327	5.08	2000	0.6794	84.4019
0.0106	6.35	2500	0.7170	82.5587
0.0064	7.61	3000	0.7395	85.3796

Figura 39. Resultados del transfer learning con la data TIMIT y FLEURS normalizada.

Debido a los resultados del transfer learning anterior (Sección 4.1.2.1.2), se normalizó la data de TIMIT Y FLEURS utilizando el promedio y la desviación estándar. Como se puede observar en la Figura 39, el "training los" disminuye al avanzar el número de épocas, lo que refleja que el modelo se ajusta a los datos del entrenamiento. Por otro lado, el "test loss" empieza con un loss alto y al avanzar el número de épocas comienza a disminuir hasta el step 1500 (3.81 épocas) pero después comienza a aumentar hasta el final del entrenamiento, lo que quiere decir que el modelo no se ajustó a los datos de testeo. Por último, el WER hasta el step 2500 (6.35 épocas) disminuye, pero comienza a aumentar al final del entrenamiento, obteniendo un porcentaje muy alto de WER.

Estos resultados se comprenden debido a que hubo dificultades con la normalización debido a que la duración de los vectores de audio eran diferentes entre sí.

4.1.2.2 Resultados del Experimento 2

Los resultados de las métricas de evaluación de la Sección 2.3.4 para cada tipo de error se describen a continuación:

4.1.2.2.1 Error de eliminación

```
        ERROR DE ELIMINACIÓN
TP: 95
FN: 133
FP: 426
TN: 12220

Resultados de las métricas de evaluación
Precision: 0.18234165067178504
Recall: 0.4166666666666667
F1-score: 0.2536715620827771
```

Figura 40. Resultados de las métricas para el error de eliminación.

Como se puede observar en la Figura 40, la precisión del error de eliminación es del 18%, debido a que el modelo tiene muchos casos de FP y sobrepasa a los casos de TP. Por otro lado, en cuanto al Recall, se obtuvo el 41%, debido a que el modelo tiene muchos casos de FN y una menor cantidad de TP. Por último, la métrica F1-score tiene el 25% de desempeño. Los resultados de estas métricas demuestran que el modelo no detecta la mayoría de los errores de eliminación correctamente.

4.1.2.2.2 Error de inserción

```
        ERROR DE INSERCIÓN
TP: 11
FN: 86
FP: 309
TN: 12468

Resultados de las métricas de evaluación
Precision: 0.034375
Recall: 0.1134020618556701
F1-score: 0.052757793764988015
```

Figura 41. Resultados de las métricas para el error de inserción.

Como se puede observar en la Figura 41, la precisión del error de inserción es del 3%, debido a que el modelo tiene muchos casos de FP y los casos de TP son casi nulos comparado a los de FP. Por otro lado, en cuanto al Recall, se obtuvo el 11% de acierto, debido a que el modelo tiene bastantes casos de FN y una cantidad diminuta de casos de TP. Por último, la métrica F1-score tiene el 5% de desempeño. Los resultados de estas métricas demuestran que el modelo no detecta la mayoría de los errores de eliminación correctamente.

4.1.2.2.3 Error de sustitución

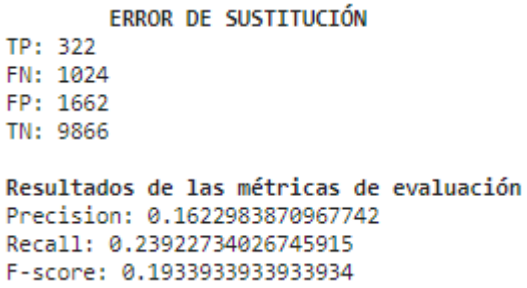

Figura 42. Resultados de las métricas para el error de sustitución.

Como se puede observar en la Figura 42, la precisión del error de sustitución es del 16%, debido a que el modelo tiene muchos casos de FP y los casos de TP son muy pocos comparado a los de FP. Por otro lado, en cuanto al Recall, se obtuvo el 23%, debido a que el modelo tiene muchos casos de FN y una cantidad pequeña de casos de TP. Por último, la métrica F1-score tiene el 19% de desempeño. Los resultados de estas métricas demuestran que el modelo no detecta la mayoría de los errores de eliminación correctamente.

4.1.2.2.4 Resultados del calculó del número de fonemas predichos correctamente por el modelo Wav2Vec.

	Fonemas	Accuracy
0	-	nan
1	0	nan
2	1	81.6092
3	2	90.3614
4	3	90.4215
5	4	53.0303
6	5	nan
7	6	0.0000
8	8	84.7059
9	9	62.6667
10	a	66.4783
11	b	86.4734
12	d	78.4499
13	e	8.6957
14	f	91.7293
15	h	86.0656
16	i	0.0000
17	j	74.4186
18	k	96.3855
19	l	93.5547
20	m	93.6869
21	n	88.7077
22	o	16.3462
23	p	90.7563
24	r	3.8462
25	s	92.9091
26	t	80.1214
27	u	82.9508
28	v	75.8197
29	w	96.0674
30	x	56.4103
31	y	0.0000
32	z	33.9286
33	|	17.8788
34	ð	47.3389
35	ŋ	79.4872
36	ɐ	25.0000
37	ɑ	60.2041
38	ɔ	50.0000
39	ə	75.4636
40	ɚ	52.7344
41	ɛ	86.5385
42	ɜ	nan
43	ɡ	84.1667
44	ɨ	0.0000
45	ɪ	82.3744
46	ɹ	84.6154
47	ɾ	34.9398
48	ʃ	91.9118
49	ʒ	28.5714
50	θ	70.2703

Figura 43. Accuracy de cada fonema.

Debido a los pobres resultados obtenidos de las métricas de evaluación para cada tipo de error de pronunciación, se calculó el Accuracy de cada fonema que el modelo debía de predecir. Los resultados se muestran en la Figura 43 y como se puede observar hay fonemas que no tienen un valor numérico. Entre ellos se tiene el “-” y el “0” y esto es correcto, debido a que “-” se cambió por “|” y el “0” hace referencia al fonema /err/, este fonema refleja los fonemas que los anotadores no pudieron detectar y es correcto que no

se encuentre en las predicciones. Los otros fonemas (ver Tabla 11 para conocer cuales fonemas están relacionados con los números 0,1,2, etc.) son fonemas que el modelo no pudo detectar.

Para obtener cuantos fonemas el modelo predijo correctamente, de los 50 fonemas que se obtuvieron, se filtraron los fonemas que tuvieran un Accuracy mayor o igual al 90%. Los fonemas que cumplieron esta condición se pueden observar en la Figura 44. Como se puede observar, el modelo solo pudo detectar 10 de 50 fonemas correctamente.

	Fonemas
3	2
4	3
14	f
18	k
19	l
20	m
23	p
25	s
29	w
48	ʃ

Figura 44. Fonemas con un Accuracy mayor o igual a 90%.

4.2 Conclusiones

Los resultados del Experimento 2 demostraron que el modelo “wav2vec2-xlsr-53-espeak-cv-ft” no es útil para evaluar la pronunciación a nivel fonético del idioma inglés hablado por personas cuya lengua materna es el español (L2-ARTIC CORPUS). Así mismo, este modelo no puede detectar la mayoría de los fonemas correctamente, solo pudo detectar el 20% de los fonemas con un Accuracy igual o mayor al 90%. Es decir, este modelo no se lo puede utilizar para dar una retroalimentación a una persona que este aprendiendo inglés.

Por otro lado, sobre el entrenamiento basado en Transformer para predecir la pronunciación a nivel fonético para el idioma inglés a través de la técnica de transfer learning (Experimento 1), de los 3 transfer learning que se entrenaron, ninguno obtuvo resultados satisfactorios, con lo cual los modelos resultantes del entrenamiento no pueden usarse como un tutor de pronunciación, debido a que no se puede obtener una retroalimentación adecuada por los resultados obtenidos.

Por último, sobre la evaluación de los errores de lectura a nivel de palabras en grabaciones de audio del idioma español usando el modelo Whisper-Large-V2, se obtuvieron resultados satisfactorios en la detección de los errores de eliminación e inserción con porcentajes mayores o cercanos al 80% en todas las métricas. Sobre el error de sustitución, si se realizará los cambios detallados en la Sección 4.3, se obtendrían resultados en sus métricas, parecidos a los errores comentados anteriormente. Por lo tanto, se puede afirmar que el modelo Whisper-Large-V2 permite detectar la mayoría de los casos de los errores de lectura a nivel de palabras correctamente.

El modelo de Whisper-Large-V2 podría utilizarse en tareas similares y observar los resultados que se obtienen, este trabajo es una base para próximos estudios.

4.3 Recomendaciones

Sobre el error de sustitución de la Sección 4.1.3.3, en su mayoría, las palabras que se encuentran en los casos 0-guiones y 1-guion son como los ejemplos mostrados en la Tabla 13, lo que quiere decir que, si se normalizara el género de las palabras para masculino, así mismo, si se considerara como similares las variaciones de nombres extranjeros, lugares, ciudades, entre otros, y si se eliminaran las tildes de las palabras, se obtendrían resultados similares en las métricas de los tipos de errores de sustitución e inserción. Los “errores” que comete el modelo Whisper-Large-V2 sobre el error de sustitución, son errores de forma no de fondo y se debe a las inflexiones de las palabras.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] Levis, J.; Suvorov, R. Automatic speech recognition. In The Encyclopedia of Applied Linguistics; Springer: Berlin, Germany, 2012

[2] M. Malik, M. K. Malik, K. Mehmood, and I. Makhdoom, “Automatic speech recognition: a survey,” Multimedia Tools and Applications, vol. 80, pp. 9411–9457, 2021.

[3] E. Deruty. "Intuitive understanding of MFCCs". Medium. https://medium.com/@derutycsl/intuitive-understanding-of-mfccs-836d36a1f779 (accedido el 16 de agosto de 2023).

[4] J. Alammar. "The illustrated transformer". Github. https://jalammar.github.io/illustrated-transformer/ (accedido el 16 de agosto de 2023).

[5] Vaswani, A.; Shazeer, N. M.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; Polosukhin, I. Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing System, 6000–6010, 2017.

[6] Datagen Tech. "What is the transformer architecture and how does it work?" Datagen. https://datagen.tech/guides/computer-vision/transformer-architecture/ (accedido el 16 de agosto de 2023).
[7] Stefania Cristina. "The Transformer Model". Machine learning mastery. https://machinelearningmastery.com/the-transformer-model/ (accedido el 16 de agosto de 2023).
[8] Zhang Aston and Lipton Zachary C. and Li Mu and Smola Alexander J., "Attention Mechanisms and Transformer", en DIVE INTO DEEP LEARNING. arXiv preprint arXiv:2106.11342, 2021.
[9] G. Giacaglia. "How Transformer Work". Towardsdatascience. https://towardsdatascience.com/Transformer-141e32e69591 (accedido el 16 de agosto de 2023).
[10] Cohere. What are Transformer Models and How do they Work? (15 de junio de 2023). Accedido el 16 de agosto de 2023. [Video en línea
[11] Cohere. What Is Attention in Language Models? (13 de abril de 2023). Accedido el 16 de agosto de 2023. [Video en línea
[12] Turing. "Understanding Transformer Neural Network Model in Deep Learning and NLP". Turing. https://www.turing.com/kb/brief-introduction-to-Transformer-and-their-power (accedido el 16 de agosto de 2023).
[13] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, “Robust speech recognition via large-scale weak supervision,” arXiv preprint arXiv:2212.04356, 2022.
[14] S. Gandhi. "Fine-Tune Whisper For Multilingual ASR with Transformer". Huggingface. https://huggingface.co/blog/fine-tune-whisper (accedido el 16 de agosto de 2023).
[15] OpenAi. "Introducing Whisper". OpenAi.com. https://openai.com/research/whisper (accedido el 16 de agosto de 2023).
[16] L. Roberts. "Understanding the Mel Spectrogram". Medium. https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53 (accedido el 16 de agosto de 2023).
[17] Meta. "Wav2vec: State-of-the-art speech recognition through self-supervision". AI Meta. https://ai.meta.com/blog/wav2vec-state-of-the-art-speech-recognition-through-self-supervision/ (accedido el 16 de agosto de 2023).
[18] Meta. "Wav2vec 2.0: Learning the structure of speech from raw audio". AI Meta. https://ai.meta.com/blog/wav2vec-20-learning-the-structure-of-speech-from-raw-audio/ (accedido el 16 de agosto de 2023).
[19] Meta. "XLS-R: Self-supervised speech processing for 128 languages". AI Meta. https://ai.meta.com/blog/xls-r-self-supervised-speech-processing-for-128-languages/ (accedido el 16 de agosto de 2023).
[20] P. v. Platen. "Fine-Tune Wav2Vec2 for English ASR with Transformer". Huggingface. https://huggingface.co/blog/fine-tune-wav2vec2-english (accedido el 16 de agosto de 2023).
[21] Baevski A., Zhou H., Mohamed A., Auli M., “wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations,” arXiv preprint arXiv:2006.11477, 2020

[22] Q. Xu, A. Baevski y M. Auli. "Wav2Vec2-Large-XLSR-53 finetuned on multi-lingual Common Voice". Huggingface. https://huggingface.co/facebook/wav2vec2-xlsr-53-espeak-cv-ft (accedido el 16 de agosto de 2023).
[23] Q. Xu, A. Baevski y M. Auli., “SIMPLE AND EFFECTIVE ZERO-SHOT CROSS-LINGUAL PHONEME RECOGNITION” arXiv preprint arXiv:2109.11680v1, 2021
[24] Ł. Sus. "Wav2Vec 2.0: A Framework for Self-Supervised Learning of Speech Representations". Neurosys. https://neurosys.com/blog/wav2vec-2-0-framework (accedido el 16 de agosto de 2023).
[25] A. Brown, ‘‘International phonetic alphabet,’’ in The Encyclopedia of Applied Linguistics. 2012.
[26] D. Jurafsky y James H. Martin, "Advanced Dialog Systems", en Speech and Language Processing. 2018. Accedido el 16 de agosto de 2023. [En línea
[27] J. Blanca. "Significado biológico". Bioinformatics at COMAV. https://bioinf.comav.upv.es/courses/intro_bioinf/alineamientos.html#:~:text=Hay%20dos%20tipos%20de%20alineamientos,sólo%20las%20zonas%20más%20parecidas (accedido el 16 de agosto de 2023).
[28] Pevsner, 2009: Ch 3 Pairwise Sequence Alignment
[29] Amrita. "Smith-Waterman Algorithm - Local Alignment of Sequences". Vlab Amrita. https://vlab.amrita.edu/?sub=3&brch=274&sim=1433&cnt=1 (accedido el 16 de agosto de 2023).
[30] Amrita. "Global alignment of two sequences - Needleman-Wunsch Algorithm". Vlab Amrita. https://vlab.amrita.edu/?sub=3&brch=274&sim=1431&cnt=1 (accedido el 16 de agosto de 2023).
[31] John S. Garofolo, Lori F. Lamel, William M. Fisher, Jonathon G. Fiscus, David S. Pallett, and Nancy L. Dahlgren. The DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus CDROM. Linguistic Data Consortium, 1993b.
[32] C. Lopes and F. Perdigao, "Phone recognition on the timit database", Speech Technol./Book, vol. 1, pp. 285-302, 2011.
[33] A. Conneau, M. Ma, S. Khanuja, Y. Zhang, V. Axelrod, S. Dalmia, J. Riesa, C. Rivera, and A. Bapna, “Fleurs: Few-shot learning evaluation of universal representations of speech,” arXiv preprint arXiv:2205.12446, 2022.
[34] P. V. Platen y Aconneau. "FLEURS". Huggingface. https://huggingface.co/datasets/google/fleurs (accedido el 16 de agosto de 2023).
[35] G. Zhao, S. Sonsaat, A. O. Silpachai, I. Lucic, E. ChukharevKhudilaynen, J. Levis, and R. Gutierrez-Osuna, “L2-arctic: A non-native english speech corpus,” Perception Sensing Instrumentation Lab, 2018.
[36] Tamu. "L2-ARCTIC documentation". Psiengr tamu. https://psi.engr.tamu.edu/l2-arctic-corpus-docs/ (accedido el 16 de agosto de 2023).
[37] P. Boersma y D. Weenink. "Praat: doing phonetics by computer". Fon Hum. https://www.fon.hum.uva.nl/praat/ (accedido el 16 de agosto de 2023).
[38] "FAQ NHK WORLD". NHK. https://www3.nhk.or.jp/nhkworld/en/faq/ (accedido el 16 de agosto de 2023).
[39] E. Nam. "Understanding the Levenshtein Distance Equation for Beginners". Medium. https://medium.com/@ethannam/understanding-the-levenshtein-distance-equation-for-beginners-c4285a5604f0 (accedido el 16 de agosto de 2023).

[40] T. Tigerschiold. "What is Accuracy, Precision, Recall and F1 Score?" Labelf. https://www.labelf.ai/blog/what-is-accuracy-precision-recall-and-f1-score#:~:text=Recall%20is%20also%20called%20true,we%20correctly%20predict%20as%20true?”&text=As%20you%20can%20see%20from,recall%20they%20are%20tightly%20connected.&text=F1%20Score%20is%20a%20measure%20that%20combines%20recall%20and%20precision (accedido el 16 de agosto de 2023).

[41] Huggingface, Load audio data, Distributed by Huggingface. [Online

[42] S. Gandhi. "Fine-Tune Whisper For Multilingual ASR with Transformer". Huggingface. https://huggingface.co/blog/fine-tune-whisper (accedido el 16 de agosto de 2023).

[43] A. Menzli. "Tokenization in NLP: Types, Challenges, Examples, Tools". Neptune AI. https://neptune.ai/blog/tokenization-in-nlp (accedido el 16 de agosto de 2023).

[44] A. Zucker. "Whisper". Huggingface. https://huggingface.co/docs/Transformer/v4.31.0/en/model_doc/whisper#Transformer.WhisperProcessor (accedido el 16 de agosto de 2023).

[45] Hugging face. "Trainer". Huggingface. https://huggingface.co/docs/Transformer/main_classes/trainer (accedido el 16 de agosto de 2023).

[46] Redacción KeepCoding. "Tensores: todo lo que necesitas saber". KeepCoding. https://keepcoding.io/blog/tensores-todo-lo-que-necesitas-saber/#:~:text=Los%20tensores%20son%20una%20generalización,bidimensional%20o%20de%20segundo%20orden (accedido el 16 de agosto de 2023).

[47] Hugging face. "Metric: wer". Huggingface Spaces. https://huggingface.co/spaces/evaluate-metric/wer (accedido el 16 de agosto de 2023).

[48] HuggingFace Inc. team. (2020). Source code for Transformer.trainer. Huggingface. https://huggingface.co/Transformer/v4.6.0/_modules/Transformer/trainer.html

[49] Mathieu Bernard and Hadrien Titeux. Phonemizer: Text to phones transcription for multiple languages in python. Journal of Open Source Software, 6(68):3958, 2021.

[50] R. Kundu. "Confusion Matrix: How To Use It & Interpret Results". V7 Platform. https://www.v7labs.com/blog/confusion-matrix-guide (accedido el 16 de agosto de 2023).

6. ANEXOS

ANEXO I

Los códigos y las datas de los experimentos se encuentran en el siguiente enlace de OneDrive:

https://epnecuador-my.sharepoint.com/:f:/g/personal/carlos_calva_epn_edu_ec/EpWBeEmJdW9KjUxXX7ecdwIBD45iWTXBEFo1bjPaUvvUgg?e=IiINK4

ANEXO II

El alfabeto IPA se lo puedo observar a través de este enlace:

https://www.internationalphoneticassociation.org/sites/default/files/IPA_Kiel_2015.pdf

ANEXO III

El alfabeto Arpabet se lo puedo observar a través de este enlace:

http://courses.csail.mit.edu/6.345//notes/IPA/P001.html

ANEXO IV

El archivo “timit_asr.py” se lo puedo descargar a través de este enlace:

https://huggingface.co/datasets/timit_asr/tree/main

ANEXO V

El vocabulario del Tokenizer del Whisper se lo puedo descargar a través de este enlace:

https://huggingface.co/openai/whisper-base/resolve/main/vocab.json