Generación de una base de datos para reconocimiento de voz en español mediante alineamiento de audio y texto

Luis Miguel Martínez Antolín

Máster en Ingeniería de Telecomunicación

MÁSTERES DE LA UAM

2019 - 2020

Escuela Politécnica Superior

UAM Universidad Autónoma de Madrid

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

Generación de una base de datos para reconocimiento de voz en español mediante alineamiento de audio y texto

Máster Universitario en Ingeniería de Telecomunicación

Autor: Martínez Antolín Luis Miguel
Tutor: Diego de Benito Gorrón
Ponente: Doroteo Torre Toledano

JUNIO 2020

Trabajo Fin de Máster

GENERACIÓN DE UNA BASE DE DATOS PARA RECONOCIMIENTO DE VOZ EN ESPAÑOL MEDIANTE ALINEAMIENTO DE AUDIO Y TEXTO

AUTOR: Luis Miguel Martínez Antolín
DIRECTOR: Diego de Benito Gorrón
PONENTE: Doroteo Torre Toledano

AUDIAS
Dpto. de Tecnología electrónica y de las comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
JUNIO 2020

Resumen

Resumen

Los sistemas de reconocimiento automático del habla son aquellos sistemas encargados de decodificar la señal de voz obteniendo como salida una transcripción en forma de texto. Este tipo de tecnología está avanzando a gran velocidad en la actualidad, estando presente en la mayoría de las grandes empresas a pesar de algunas limitaciones, como la ausencia de grandes volúmenes de datos de libre acceso necesarias de cara a su aprendizaje y entrenamiento.

Este Trabajo de Fin de Máster se plantea la elaboración de una base de datos en español con una cantidad considerable de audio y texto. Se plantea tomar como referencia la construcción de la base de datos **Librispeech** adaptando el problema a datos en español en lugar de en inglés. Se recogen los datos de las páginas **LibriVox** y **Project Gutenberg** y se realizan técnicas de procesamiento de los datos hasta obtener un total de 207 horas de voz junto a sus transcripciones. Después, estos datos se alinean a partir de un sistema de alineamiento forzado estudiado en el primero de los Trabajos de Fin de Máster, mostrado en [1]. Por último, se utiliza esta base de datos para el entrenamiento y evaluación de un reconocedor de voz de libre acceso de los más avanzados en la actualidad, implementado mediante el uso de redes neuronales *end-to-end* en la herramienta **ESPnet**.

Palabras Clave

bases de datos, alineamiento forzado, sistemas de reconocimiento de voz, audio, texto, voz, procesamiento de datos, redes neuronales, espnet, librispeech, librivox

Abstract

Automatic Speech Recognition is a technology that decodes a voice signal and ouputs it in text form. This type of technology is currently advancing at great speed, despite some limitations such as the absence of large volumes of freely accessible data, necessary for learning and training. This Master's thesis aims to create a database in Spanish with a considerable amount of audio and text. It is proposed to take as a reference the construction of the **Librispeech** corpora, but adapting the problem to data in Spanish. Data is collected from **LibriVox** and **Project Gutenberg** projects and data processing techniques are performed until 207 voice hours are obtained along with their transcripts. This data is then aligned from a forced alignment system studied in the first of the Master's Degree Thesis, shown in [1]. Finally, this database is used for the training and evaluation of one of the most advanced voice recognition systems available today, implemented through the use of end-to-end neural networks in the **ESPnet tool**.

Key words

speech corpus, forced alignment, automatic speech recognition, asr, audio, text, data preparation, neuronal networks, espnet, librispeech, voice, librivox

Agradecimientos

A mi tutor y mi ponente, Diego y Doroteo, por ayudarme en todo momento y por todo lo que he aprendido con ellos durante la realización de este trabajo.

A todo el grupo de AUDIAS, y en especial a Beltrán, por estar siempre que lo he necesitado disponible para echarme una mano.

A Mar y Patri, por ayudarme en la costosa tarea de la construcción de la base de datos.

Índice general

Índice de figuras

Indice de tablas

Capítulo 1

Introducción

1.1. Motivación del proyecto

Los sistemas de reconocimiento automático del habla o *Automatic Speech Recognition* (ASR) son tecnologías que nos permiten transformar una señal de voz en texto. Este tipo de sistemas está en auge en los últimos años, teniendo gran relevancia en infinidad de tecnologías, como los teléfonos móviles o la gestión de documentos (búsquedas por voz). A la hora de implementar un sistema de reconocimiento automático del habla serán necesarias dos informaciones fundamentales. La primera de ellas, será la correspondiente a la información relacionada con la señal de voz que se reconocerá, y se encontrará en el llamado Modelo Acústico. La segunda de ellas, será la información sobre el vocabulario utilizado en el tipo de habla, y estará caracterizada en el Modelo de Lenguaje.

Para poder generar de forma fiable los dos modelos mencionados es necesario tener tener una base de datos que contenga los ficheros de audio a reconocer y las transcripciones correspondientes, así como los datos de alineamiento temporal entre cada uno de los ficheros de audio y texto. Por tanto, una de las principales dificultades a la hora de implementar un sistema de ASR es la ausencia de bases de datos de libre acceso que permitan el entrenamiento de estos sistemas.

En los últimos años ha aumentado considerablemente la cantidad de contenido multimedia de libre acceso disponible para los usuarios, como por ejemplo los libros en formato electrónico y los audiolibros. Sin embargo, para la construcción de una base de datos es necesario obtener la información acerca de la correspondencia temporal (alineamiento) entre cada audio y su fichero de texto correspondiente, para lo cual se utilizan los alineadores forzados de audio y texto.

En el primero de los Trabajos de Fin de Máster se evaluó una serie de alineadores forzados de uso libre, y además se implementó uno propio basado en un reconocedor de voz.

Para el segundo Trabajo de Fin de Máster, una vez visto cuál de los alineadores forzados se ajusta más al problema que se plantea, es interesante utilizarlo de cara a crear una base de datos de libre acceso en español, mediante la recopilación de datos de audio y texto y la posterior aplicación del alineador forzado sobre ellos.

Por tanto, el haber profundizado sobre el alineamiento entre ficheros de audio y texto nos facilita la construcción de una base de datos disponible para el público, que nos servirá, a su misma vez, para entrenar un sistema de reconocimiento de voz **(ASR)**, ya que la cantidad de datos de habla y texto en español de la que se dispone en la actualidad no es muy abundante.

1.2. Objetivos y enfoque

Los objetivos principales de este trabajo serán los siguientes. En primer lugar, una vez se ha evaluado el alineador forzado óptimo en la evaluación del primero de los trabajos [1], utilizarlo para generar etiquetas entre los audiolibros y los textos que se recopilarán para la creación de una base de datos en español. Una vez se haya elaborado la base de datos propia, se utilizará de cara a entrenar un reconocedor de voz basado en redes neuronales end-to-end mediante la herramienta de ASR **ESPnet**, definida en [3].

Los objetivos parciales referentes a este Trabajo de Fin de Máster son los siguientes.

En primer lugar, será necesario implementar un método que nos permita ver qué audiolibros de la página web **LibriVox** [4] se encuentran en la página de **Project Gutenberg**, ya que son las dos páginas que más contenido de audio y texto con licencia **Creative Commons** tienen disponible. Además, la base de datos **Librispeech** [5] utiliza estas dos páginas, y será el ejemplo que se tomará para construir la base de datos, pero con la diferencia en el objetivo de que sea en español en lugar de en inglés.

El segundo objetivo será el de descargar una cantidad considerable (>200 horas) de audio junto a sus correspondientes transcripciones de texto, a partir de las coincidencias entre libros y audiolibros de **Project Gutenberg** y **LibriVox**.

Una vez se hayan obtenido todos los datos, será necesaria una etapa de pre-procesamiento en la que se realizará un filtrado de los textos y los audios para que no haya partes del texto que no se encuentren en su audiolibro correpondiente o viceversa (introducción, información complementaria...).

Después, el objetivo será alinear todos los ficheros de audio y texto mediante el alineador forzado elegido en la evaluación del primer Trabajo de Fin de Máster, y una etapa de post-procesamiento en la que se eliminen aquellas partes de los textos que no coincidan con ninguna parte del audio correspondiente. Una vez se realicen los objetivos anteriores se estructurará la base de datos tomando como referencia los pasos llevados a cabo en **Librispeech**.

Para la segunda parte, se realizará un pre-procesado de los datos, necesario para poder llevar a cabo un entrenamiento de un sistema de reconocimiento de voz, implementado mediante redes neuronales en la librería **ESPnet** [3]. La forma de los datos de entrada de las herramientas de esta librería es similar a la utilizada en **Kaldi** [6], y se verá más adelante en profundidad.

El último objetivo del trabajo será el de evaluar el reconocedor de voz tras aplicar diferentes modalidades de entrenamiento, tanto utilizando la pequeña base de datos creada en el primer trabajo, como con la que se creará en éste último, de tamaño mucho mayor.

1.3. Metodología y plan de trabajo

- **Capítulo 1. Introducción**

  Este capítulo contextualiza el trabajo en el marco de los sistemas de reconocimiento de habla y su entrenamiento, así como en el de la construcción de bases de datos.

- **Capítulo 2. Estado del arte**

  En este capítulo se tratará sobre todos los conceptos previos necesarios para la comprensión de los sistemas de reconocimiento de voz, además de la base de datos que se tomará como referencia para la elaboración de una propia (**Librispeech**).

- **Capítulo 3. Descripción y diseño de los experimentos**

  En este capítulo se verá el proceso de creación de la base de datos, así como los pasos necesarios para entrenar un sistema de reconocimiento de voz.

- **Capítulo 4. Resultados**

  Una vez se realice el entrenamiento del reconocedor de voz, se realizará una evaluación de su rendimiento y se tratará sobre los resultados obtenidos a partir de él. Además, se expondrá el resultado final de la elaboración de la base de datos.

- **Capítulo 5. Conclusiones y trabajo futuro**

  Tras observar los resultados obtenidos, este capítulo resumirá las conclusiones obtenidas durante todo el trabajo, así como las posibles líneas de trabajo futuro dentro de esta área.

Capítulo 2

Estado del arte

2.1. Bases de datos para sistemas de reconocimiento de voz

Una base de datos de habla, **speech corpus**, o **ASR Corpus** es un conjunto de ficheros de audio y transcripciones de texto. En las tecnologías del habla, estas bases de datos se utilizan, entre otras cosas, para crear modelos acústicos que permitan implementar sistemas de reconocimiento de voz. En la lingüística de corpus se usan para investigar en campos como la fonética, análisis de conversaciones o el estudio de dialectos. Dentro de las **Speech Corpus** se tienen dos variantes en función de la información de los ficheros de audio que contienen. El primer tipo corresponde a las bases de datos de habla leída, cuyo contenido se obtiene de extractos de libros, noticias, listas de palabras o secuencias de números. El segundo tipo incluye datos de habla espontánea, que se pueden encontrar en ficheros de diálogos de conversación entre una o varias personas y en narraciones (una persona cuenta una historia).

Es interesante tener acceso a una cantidad de datos de audio y texto muy grande en cualquier idioma, y bajo acceso libre, que permita investigar en las tecnologías relacionadas con los sistemas de reconocimiento del habla (ASR). Este trabajo será enfocado hacia el primero de los dos tipos, en concreto en la construcción de una base de datos creada a partir de audiolibros y transcripciones de texto de dominio público, tal y como se hizo en **Librispeech**, cuya creación se abordará en el siguiente punto.

2.1.1. Librispeech

Dentro de las bases de datos de habla leída disponibles de forma gratuita, una de las que más contenido ofrece y de mayor calidad es **Librispeech**, y es por ello que se tomará como referencia de cara a la elaboración de la base de datos en español.

El proyecto **LibriVox** es un conjunto de audiolibros de libre acceso, leídos por voluntarios, que constituyen un conjunto de más de 8000 horas de audio en diferentes idiomas, principalmente en inglés [4]. La mayor parte de estos está sacada de libros que se encuentran en el Proyecto Gutenberg [7], también de dominio público, lo que facilita la obtención de las transcripciones de texto. Una vez se obtienen los datos de estos proyectos, el proceso de construcción de **Librispeech** consta de las siguientes etapas o fases.

- **Alineamiento del audio**. En primer lugar se realizó una normalización de los textos de cada libro, se obtuvo un léxico basado en **CMUdict** generando las palabras mediante una herramienta de **G2P** (Grapheme-to-phoneme) y se entrenó un modelo de lenguaje

para cada libro. Además, se reconoció el audio mediante un reconocedor de la herramienta **Kaldi** [6] y se cortaron los audiolibros en segmentos de 30 minutos de duración máxima.

En la primera fase del alineamiento se utilizó el algoritmo *Smith-Waterman* [8] para encontrar aquellas regiones óptimas de alineamiento entre los segmentos de audio reconocidos y cada texto. A partir de esto se tomaron las regiones de texto en las que la similitud (confianza) era muy alta y se descartaron el resto. Para terminar esta fase se volvieron a cortar los segmentos de audio en fragmentos de 35 segundos o menos, utilizando un algoritmo de programación dinámico.

En la segunda fase del alineamiento se filtraron aquellos segmentos donde los textos candidatos alineados en la primera fase tenían una alta probabilidad de ser incorrectos.

En la última fase se realizó una segmentación de los datos finales. Para los datos de entrenamiento se segmentaron utilizando como límites los silencios mayores de 0.3 segundos. Para los datos de test solo se segmentaron en dichos límites si coincidían en un final de frase en el texto de referencia.

- **Selección de los datos y estructuración de la base**. Para la selección de los datos a incluir en la base de datos se utilizó la API de **LibriVox** para recoger información acerca de los lectores y los capítulos de libros que había leído cada uno. Las URLs de descarga de los audiolibros se obtuvieron a partir del matching entre la información obtenida y registros de metadatos del archivo de **Project Gutenberg**. Se acotó el número de lectores por audiolibro, para lo que se descartaron aquellos de género dramático, pues incluyen más de un lector por cada obra en la mayoría de casos. Además se utilizó la herramienta de diarización **LIUM** [9] para detectar aquellos libros con más de un lector y descartarlos. Esta herramienta también aporta información de género, lo que sirvió para que haya un equilibrio en la cantidad de datos de cada uno.

  Para algunos usuarios descargar un único archivo de tamaño tan grande puede ser un inconveniente, por lo que se dividió la base en tres subconjuntos de 100, 360 y 500 horas respectivamente, en función de la calidad de las grabaciones y de la similitud en el acento al inglés norteamericano. Para realizar las métricas de calidad se reconoció la base y se estimaron medidas de *Word Error Rate* (WER) entre las transcripciones y los textos originales.

- **Modelos de lenguaje**. Para realizar el entrenamiento de los modelos de lenguaje se utilizaron libros seleccionados cuidadosamente de **Project Gutenberg** [7]. Estos libros se filtraron eliminando aquellas partes que tenían fragmentos compartidos en otros libros, así como tablas o secuencias numéricas u otro tipo de documentos inapropiados para el entrenamiento del modelo de lenguaje. Después del filtrado quedaron unos 14500 libros, con 803 millones de tokens y 900 000 palabras únicas. Para seleccionar el léxico las palabras de la base fueron ordenadas en función de su frecuencia, eligiendo las 200 000 más frecuentes.

- **Pruebas y experimentos**. Una vez se terminó la base de datos se hicieron pruebas en la decodificación de los datos de test de la base de datos de Wall Street Journal (WSJ). Se usaron modelos de lenguaje entrenados en conjunto con WSJ y Librispeech para reconocer estos datos y se hicieron pruebas para dos modelos acústicos diferentes, uno aproximado mediante GMMs adaptadas al hablante, y otro mediante redes neuronales profundas (DNNs). Por último, se midieron los valores de error, obteniendo mejores resultados que con los modelos entrenados mediante la base de datos de WSJ.

Por último, Librispeech se subió a internet y se introdujeron junto a ella las recetas y scripts de Kaldi utilizados para el reconocimiento de voz, de forma que facilite a los usuarios replicar estos resultados. [5]

2.1.2. LibriVox Spanish

En plena elaboración del trabajo se publicó un trabajo similar al que se estaba realizando, sobre una base de datos en español para entrenar sistemas de reconocimiento de voz. Este trabajo, sin embargo, se diferencia del que se plantea en este trabajo en que es de pago y está revisado manualmente, por lo que el tiempo de elaboración será mucho mayor y no será de libre acceso para los usuarios, tal y como lo es **Librispeech**.

El corpus **LibriVox Spanish** contiene aproximadamente 73 horas de voz en español junto a sus transcripciones. El audio fue obtenido del proyecto **LibriVox** [4], al igual que se hizo en Librispeech, y las transcripciones se generaron manualmente para crear la base de datos.

Las 73 horas de voz están divididas en frases correspondientes a 300 audiolibros leídos por 154 hablantes diferentes (77 hombres y 77 mujeres). Los segmentos de audio se segmentaron de forma manual y su duración es de entre 3 y 10 segundos de voz. Las transcripciones fueron generadas por hablantes nativos de español.

Los segmentos de voz tienen un sample rate de 16 kHz y su codificación es de 16-bit, en formato .flac. Al descomprimirse, se obtienen ficheros PCM de extensión WAV. Los ficheros de texto se encuentran en codificación UTF-8. Esta base de datos se asemeja al problema planteado en el presente TFM, pero su construcción está basada en la generación de las transcripciones manualmente, en lugar de obtenerse de páginas web de libre acceso. [10]

2.2. Los sistemas de reconocimiento de voz

El proceso de reconocimiento automático del habla es un proceso de decodificación y transcripción de la señal de voz. Los sistemas de ASR reciben como entrada una señal de audio procedente de un hablante y capturada a partir de un micrófono, y se encargan de analizarla a partir de determinados patrones, modelos o algoritmos, dando como resultado de salida la información de la señal de voz en forma de texto. [11]

Estos sistemas requieren dos principales modelos que permiten asociar probabilidades para cada una de las frases y palabras que aparecen en la señal de voz, con la finalidad de obtener aquellas que más probabilidad tienen de haberse dicho, consiguiendo así la transcripción final en formato de texto.

El primero de los modelos es el **Modelo Acústico**. Este se encarga de predecir los fonemas que más probabilidad tienen de haber sido pronunciados para cada segmento de la señal de voz.

El segundo modelo, correspondiente al **Modelo de Lenguaje**, es el encargado de predecir qué frases son más probables de pronunciarse dadas las palabras reconocidas anteriormente. Por tanto, ayuda a dar información contextual y permite construir aquellas frases que tengan más sentido, ya que si no se tendría una sucesión de palabras reconocidas, válidas morfológicamente, pero no sintácticamente. Para construir el **Modelo de Lenguaje** es necesario un texto de entrenamiento, así como el **Léxico**, que contiene la información fonética de cada una de las palabras que se pronuncian en el lenguaje a reconocer. Es decir, en él se encuentra la información de pronunciación de cada una de las palabras conocidas por el reconocedor.

Para generar estos modelos se utilizan principalmente los **Modelos Ocultos de Markov** o *Hidden Markov Models* (**HMMs**), aunque en los últimos años se han estado sustituyendo progresivamente por el uso de sistemas basados en redes neuronales recurrentes o *Recurrent Neuronal Networks* (**RNNs**). En el próximo punto se tratará sobre estos modelos más detalladamente. [12]

2.3. Modelos Ocultos de Markov (HMMs)

Los Modelos Ocultos de Markov o *Hidden Markov Models* (**HMMs**) han sido hasta hace poco tiempo los modelos más utilizados en reconocimiento del habla. Se encargan de modelar de forma estadística la acústica de la señal de voz, es decir, cómo suenan los diferentes fonemas y palabras. Estos modelos están siendo sustituidos progresivamente por las redes neuronales, aunque también se pueden utilizar en modelos híbridos que combinan ambos sistemas.

El objetivo es predecir parámetros de la señal de voz desconocidos a partir de otros conocidos observables, para lo que se utiliza un diagrama de estados que se define mediante el número de estados, la matriz de probabilidades de transición entre estados, el número de parámetros observables y las probabilidades iniciales de cada estado. De este modo, se toman como parámetros o estados del diagrama los diferentes fonemas de la señal de voz, con el objetivo de predecir aquellos fonemas cuya probabilidad de haber sido pronunciados haya sido mayor.

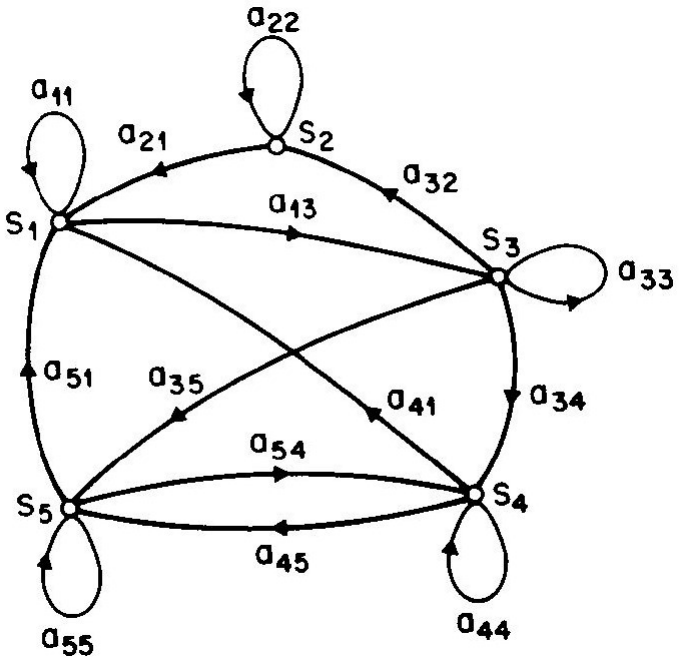

Figura 2.1: Diagrama de estados de un HMM.

El algoritmo más utilizado para estimar los parámetros de los HMMs, que maximiza la probabilidad de la observación dado el modelo es el algoritmo *Baum-Welch*, un caso particular del algoritmo *Expectation-Maximization* [13] aplicado a los **HMMs**.

El algoritmo utilizado para ver qué secuencia de estados (fonemas) es más probable que suceda a partir de unos parámetros observables es el algoritmo de *Viterbi*. [14]

2.4. HTK: Herramienta de Modelos Ocultos de Markov

HTK es un conjunto de herramientas de software de código abierto desarrolladas por la Universidad de Cambridge en 1994 para construir y manejar Modelos Ocultos de Markov. Consiste en una serie de módulos de librerías y un conjunto de más de 20 herramientas (programas). Está implementada en **ANSI C** y comenzó funcionando principalmente en sistemas **UNIX**, aunque en la actualidad puede ejecutarse en cualquier sistema operativo moderno. Se utiliza principalmente en sistemas de reconocimiento de voz y análisis de la señal de audio, basándose en modelos HMM.

La librería de HTK contiene diez módulos que proveen una interfaz entre las herramientas a utilizar y el usuario, así como una variedad de funciones de apoyo de gran utilidad. Los módulos de dicha librería son los mostrados en la siguiente tabla. [15]

Nombre	**Módulo**
HDbase	Base de datos de tokens de entrenamiento
HGraf	Control gráfico de señales de audio
HLabel	Control de etiquetas de entrada y salida
HMem	Administración de memoria
HModel	Interpreta las definiciones de HMM de entrada/salida
HParse	Soporte gramatical
HShell	Interfaz del sistema operativo
HSigP	Rutinas de procesamiento de señal
HSpIO	Control de datos de voz de entrada y salida

Tabla 2.1: Modulos de la librería de HTK

Estos módulos se utilizan mediante una herramienta de entrenamiento que lee una serie de recetas de HMM y unos datos de entrenamiento asociados, produciendo nuevos archivos de HMM. En los últimos años se ha ampliado el número de estos módulos, entre los que destacan los siguientes ejemplos. **HAudio**, encargado de controlar la captura de señales de audio. **HMath**, que se encarga de operar con estructuras de alto nivel que sirven para manejar los datos con mayor facilidad (vectores, matrices, etc). Por último, **HML** es un modelo dedicado únicamente a los modelos de lenguaje.

Nombre	**Función**
HAlign	Realiza alineamientos forzados
HCode	Análisis de voz (LPC,MFCC, etc)
HDEd	Editor de diccionario en modo batch
HERest	Re-estimación de Baum-Welch integrada
HHEd	Editor de HMM en modo batch
HInit	Unidad asilada segmentada de K-means para la inicialización del modelo
HLEd	Editor de archivos de etiquetas en modo batch
HList	Lista de contenidos de un fichero de datos
HRest	Re-estimación Baum-Welch aislada
HResults	Análisis de resultados
HSLab	Editor de ficheros simple e interactivo
HSource	Generación de datos utilizando una fuente estadística de HMM
HVite	Decodificador Viterbi (Aislado y conectado)

Tabla 2.2: Principales funciones de HTK

A parte de estos módulos, **HTK** se construyó con más de 25 herramientas de partida que sirven para construir sistemas basados en HMMs, y se utilizan unas u otras dependiendo del tipo de HMMs requeridos. En la tabla 2.2 se pueden ver las principales herramientas con las que cuenta HTK.

A partir de estas funciones se realizan tareas como la parametrización de los datos (**HCode**), o el entrenamiento de sistemas de **ASR** con gran facilidad (**HERest**).

A partir de los diferentes módulos y herramientas definidas, **HTK** se ha utilizado para una cantidad muy grande de aplicaciones y sistemas. En 1994 su licencia se utilizaba en más de 100 laboratorios del habla en todo el mundo. Originalmente su uso se limitaba al desarrollo de sistemas ASR, sin embargo en la actualidad se utiliza en cualquier sistema que pueda modelarse como un HMM. Ha resultado ser una herramienta de gran rendimiento en sistemas de reconocimiento y síntesis de voz, pero también en otras áreas como el etiquetado automático de texto y audio para infinidad de bases de datos, o incluso campos no relacionados con el procesamiento de la señal de voz, como el Proyecto Genoma (secuenciación del ADN) o análisis de imagen y reconocimiento de patrones. [16]

En cuanto a su licencia es algo limitado, ya que permite su uso libre para investigación y enseñanza, pero establece una serie de restricciones para su uso comercial y distribución, en comparación con otras herramientas más modernas, como **Kaldi**, que se verá a continuación.

2.5. Kaldi: Herramienta para sistemas de ASR

Kaldi es una herramienta de código abierto utilizada principalmente para sistemas de ASR, implementada en **C++** y bajo la licencia **Apache v2.0**, una de las menos restrictivas dentro de las disponibles. Aparte de su licencia, es más moderno, flexible y tiene un mejor soporte que sus competidores, HTK y RASR (RWTH ASR) [17].

Su estructura se conforma de la siguiente manera. En primer lugar, para la infraestructura se utiliza la librería **OpenFST**. Como librerías para el álgebra numérico se utilizan las librerías **BLAS** y **LAPACK**. En función de cuál de las dos clases de librería se utilicen se conformarán dos subgrupos de librerías. A cada una de las funcionalidades de estas herramientas se accede a través de una terminal de comandos creada en **C++**. Cada una de estas herramientas tiene funciones específicas de reconocimiento de voz. Por ejemplo, hay ejecutables diferentes para acumular estadísticas, sumar acumulaciones de éstas, y, por último, generar un modelo acústico mediante GMMs utilizando las estimaciones de máxima verosimilitud. En la tabla de la figura 2.2 se puede ver la jerarquía de librerías con las que cuenta Kaldi, desde los comandos de la terminal o scripts hasta las librerías internas que utiliza para los diferentes procesos. [6]

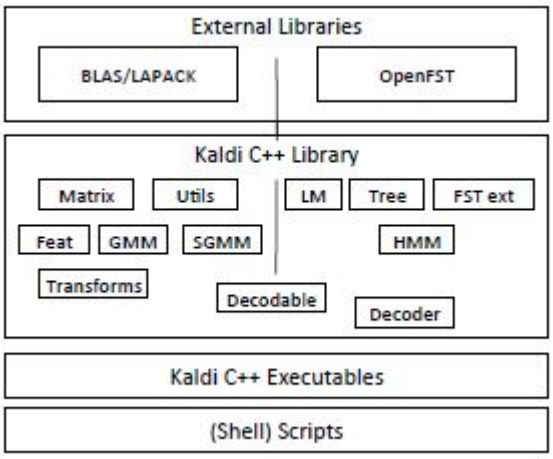

Figura 2.2: Jerarquía de librerías de Kaldi.

Dentro de las opciones que tiene para extraer características de la señal de voz, hay una gran variedad, como por ejemplo *Mel Frequency Cepstral Coefficients* **MFFC**, *Cepstral Mean and Variance Normalization* **CMVN**, etc. Además, se tiene libertad para modificar las diferentes opciones de cada uno de estos coeficientes, como por ejemplo, el número de bins de los **MFCCs**.

En cuanto al modelado y entrenamiento de sistemas ASR, ofrece gran variedad de posibilidades, desde el uso de GMMs/HMMs hasta redes neuronales profundas o recurrentes (**DNN**, **RNN**). Estas dos últimas han sido añadidas recientemente, ya que Kaldi es una herramienta en constante desarrollo, y permite ir actualizándose a medida que nacen nuevos modelos para entrenamiento de sistemas de reconocimiento de voz. [6]

2.6. ESPnet: End-to-End Speech Processing Toolkit

ESPnet es una herramienta de código abierto de procesamiento de voz end-to-end presentada en 2018 en [3]. Está especializada en los sistemas de reconocimiento de voz (ASR) *end-to-end*, para lo que utiliza una gran variedad de herramientas de redes neuronales, así como las librerías de Python **Chainer** y **PyTorch**. Esta herramienta sigue el mismo formato que Kaldi para el procesamiento de datos y extracción de características, así como en las recetas de configuración de los experimentos de procesamiento de voz a implementar.

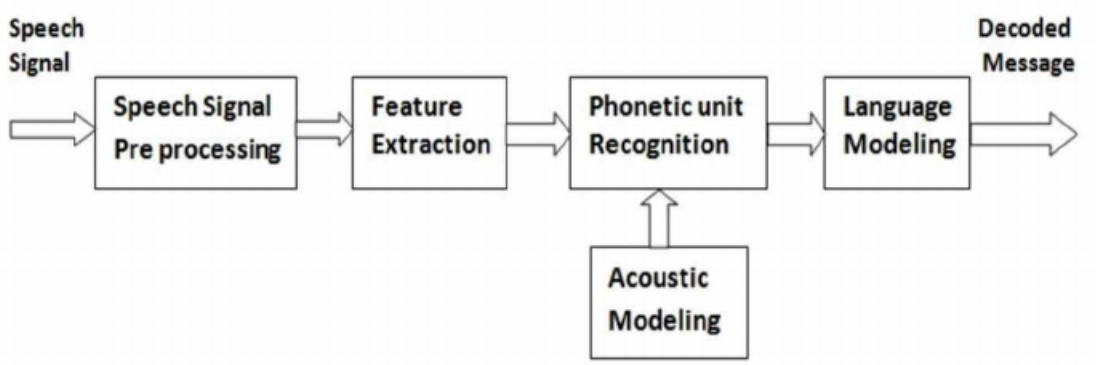

Figura 2.3: Diagrama de bloques de un sistema de ASR [2]

El método clásico en el desarrollo de un sistema de reconocimiento de voz está basado en los HMMs, descritos en la sección 2.3. Como se ha visto en 2.2, estos sistemas requieren un desarrollo muy complejo del modelo acústico, un diccionario de pronunciación, un modelo de lenguaje y un decodificador de estados, además de fuentes lingüísticas. En el diagrama de bloques de 2.3 se puede ver su estructura.

Sin embargo, la utilización de una única red neuronal como estructura, tal y como se muestra en la figura 2.4, simplifica mucho el proceso de creación, obteniendo las siguientes ventajas.

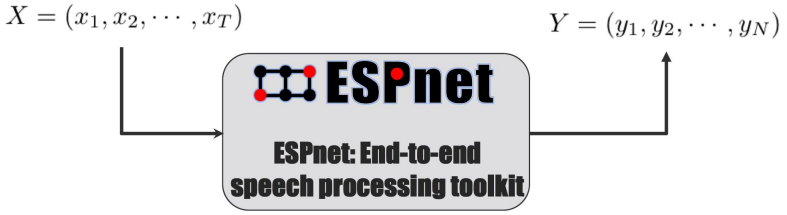

Figura 2.4: Diagrama de bloques de ESPnet

En primer lugar, simplifica enormemente el proceso de construcción de modelos acústicos y de lenguaje, integrándolos directamente con más facilidad. Además, puede entrenar directamente una red neuronal con una secuencia de palabras/frases objetivo. Por último, se pueden integrar módulos adicionales a la red con facilidad, en cuyo caso cada módulo constituiría una red neuronal, y se combinarían entre sí, entrenándose todos los módulos en conjunto.

La integración de varios módulos de redes neuronales optimiza el sistema para minimizar el error de salida, reduce los desajustes entre redes mediante la optimización conjunta y no requiere etiquetas de objetivos intermedios para el entrenamiento del sistema. Como aspecto en contra, las redes son específicas para cada problema, por lo que no pueden utilizarse para otras tareas, algo que ya ocurría en los modelos híbridos de **HMMs-RNNs** de **Kaldi**.

La receta o script estándar de ESPnet consta de las siguientes fases, dentro de un script principal que se denomina habitualmente **run.sh**.

- **Fase 0**. Preparación de los datos mediante el formato de directorios y rutas de Kaldi, por lo se se puede usar directamente el script de Kaldi de preparación de datos.
- **Fase 1**. Extracción de características: De nuevo se puede utilizar el script de Kaldi.
- **Fase 2**. Preparación de los datos para ESPnet. Se convierte la entrada del script de preparación de datos de Kaldi en un fichero JSON.
- **Fase 3**. Entrenamiento del modelo de lenguaje mediante una red neuronal basada en caracteres (Character-based RNNLM) implementada en Chainer o PyTorch. Esta fase es opcional, también se podría utilizar sin modelos de lenguaje, pero no sería lo óptimo.
- **Fase 4**. Entrenamiento del ASR End-to-End. Se entrena un codificador-decodificador de un sistema híbrido de CTC/attention-based mediante Chainer o PyTorch.
- **Fase 5**. Reconocimiento del audio mediante la RNNLM creada en la fase 3 y el ASR End-to-End entrenado en la fase 4.

En cuanto a su rendimiento, es comparable a la principal aproximación en Kaldi. En experimentos de menos de 100 horas (utilizando el corpus Wall Street Journal (WSJ)) se obtuvieron valores de WER superiores a los obtenidos mediante el uso de **HMM/DNN**, pero similares o menores en experimentos de más de 100 horas. Sin embargo, su velocidad de entrenamiento para este corpus es mucho mayor que en la mayoría de sistemas, teniendo un tiempo medio de procesamiento o *Average Batching Processing Time* de 125 segundos aproximadamente, respecto de los 1400 segundos de promedio que tiene Kaldi.

En resumen, esta herramienta simplifica mucho el entrenamiento de un sistema de reconocimiento de voz mediante la base de datos en español que se creará, y dado que esta tiene una duración mayor a 100 horas, se cree que el rendimiento de ESPnet puede ser superior al de los clásicos sistemas basados en HMM/DNN.

2.7. Sistemas de alineamiento de audio y texto

Un alineador forzado es un sistema que permite realizar alineamientos entre el habla y su transcripción ortográfica, obteniendo como salida un mapa de sincronización que asocia cada intervalo de texto a un intervalo de tiempo (dentro del archivo de audio).

Por tanto, los sistemas de alineamiento de audio-texto permiten encontrar un nexo entre la información de la señal de voz y su transcripción. Dentro de su uso se encuentran los sistemas de indexado multimedia (para búsquedas de voz o texto), así como sistemas de síntesis de voz, y, finalmente, sistemas ASR.

Dentro de los alineadores se pueden distinguir dos principales tipos. El primero de ellos está basado en el reconocimiento de la señal de voz, mediante un sistema ASR, para después alinear la transcripción obtenida con el texto original. Este alineamiento consiste en obtener qué fonemas se parecen más a cada parte de la señal de audio, a su forma de onda y espectrograma, y dependerán de los modelos acústicos, construidos, en la mayoría de casos, mediante el uso de HMMs, implementados en Kaldi o HTK. Estos modelos acústicos pueden estar pre-entrenados (con bases de datos alineadas previamente e incorporadas al alineador) o entrenados mediante

un proceso de entrenamiento/alineamiento simultáneo en el que los datos de entrenamiento serán los mismos para los que se generarán las etiquetas de alineamiento. [18]

En el segundo tipo, mucho menos común, se utiliza un sintetizador de voz para convertir el texto en una señal de audio, que se comparará con la señal de voz original para así obtener una correspondencia temporal entre ambas.

Además, dentro de los alineadores forzados de la actualidad, la gran mayoría requiere una serie de condiciones para obtener un alto rendimiento. Lo primero y más importante, es necesario que se acompañe la señal de voz junto a su transcripción de texto. Muchos de ellos son capaces de corregir errores en las transcripciones incorrectas, para alinear de nuevo con las correcciones aplicadas. También es importante que posibles errores durante el alineamiento no afecten a alineamientos de etiquetas temporales posteriores. En caso de haber errores en la transcripción, muchos de los modelos no los corregirán pero serán capaces de realizar alineamientos con las partes que correspondan. Todos deben tener una capacidad de memoria suficiente para guardar todas las posibles ponderaciones sobre qué etiquetas generar, sobre todo en modelos probabilísticos como HMMs o GMMs. Por último, en la mayoría de ellos consta de un sistema de ASR interno, que permite alinear audio y texto para idiomas para los que no se tienen modelos acústicos y de lenguaje entrenados. [19]

Dentro de las licencias de los alineadores disponibles, la mayoría de ellos están protegidos comercialmente, con precios muy elevados, que van desde los 1000 euros anuales hasta los 10k anuales por la licencia del programa completo.

2.7.1. Aeneas Alignment Tool.

Dentro de las herramientas de licencia libre de la actualidad para realizar alineamientos entre audio y texto se encuentra el alineador Aeneas. Este modelo fue implementado en 2015 con una gran acogida, ya que ofrecía muy buenos resultados a pesar de su licencia no restrictiva. Previamente a la creación de Aeneas, se encontraban disponibles algunos modelos de alineadores gratuitos que habían desarrollado grupos de investigación, todos ellos con alguna de las siguientes limitaciones.

En primer lugar, todos carecían de modelos acústicos y de lenguaje pre-entrenados correspondientes a idiomas que no fueran el inglés.

En segundo lugar, su instalación y ejecución era demasiado compleja para que la mayoría de los usuarios fueran capaces de utilizarlo con facilidad. Además, si el usuario no tiene conocimientos relacionados con el procesamiento de lenguaje, su uso se dificulta aún más.

Por último, la licencia de muchos de ellos no permitía utilizarlos con fines comerciales.

La principales ventajas de Aeneas de cara a al objetivo final del trabajo, que tal y como se explica en 1.1, trata de la construcción de una base de datos de licencia libre en español, son las siguientes.

Tiene un alto rendimiento en idiomas diferentes al inglés, como por ejemplo, el español en este caso. Los buenos resultados de Aeneas en general, y en concreto en español, se conocen dado que a simple vista los alineamientos que realiza son precisos, y además se demostró analíticamente en su evaluación durante el primero TFM.

Aeneas no utiliza herramientas complejas de ASR como **Kaldi** o **HTK**, sino que está basado en una estructura mucho más sencilla. En primer lugar, utiliza un sistema de **TTS (*Text-to-speech*)** para transformar el texto en audio sintético. Este audio no necesita ser de gran calidad, sino que siendo inteligible es suficiente para realizar la tarea de alineamiento que se realiza a posteriori. Una vez se cuenta con el audio original y el sintético, se aplica el algoritmo **DTW**

(*Dynamic Time Warping*) a los **MFCCs** de cada una de las señales, obteniendo así las mejores correspondencias entre las dos señales de audio. Dado que el audio sintetizado va asociado a cada una de las partes del texto, se tendrá un alineamiento entre el audio original y dicho texto asociado.

En cuanto al código, se desarrolla en **Python**, lo que supone una mayor libertad a la hora de manipular ficheros de configuración, escribir y leer ficheros de texto, etc. Dentro de Python se utilizan programas como **eSpeak** para la síntesis del audio o **FFMpeg** para la conversión de formatos de los ficheros para el procesamiento de la señal, así como módulos que permiten recoger las llamadas a estos programas y recoger los resultados. Además, su instalación, aunque compleja en algunos sistemas operativos, es muy sencilla en Windows o Mac, con un ejecutable 'todo en uno' que permite instalarlo con facilidad.

Por último, se distribuye con la licencia **AGPL** (*Affero General Public License*) que ofrece las siguientes ventajas. Permite descargar el código fuente, utilizarlo, así como modificarlo de forma gratuita. También permite su uso con fines comerciales, incluyendo su venta tras aplicar modificaciones o vender servicios basados en él, siempre que no se modifique, y se referencie a la licencia original, de cara a contribuir a la comunidad de Aeneas. En los resultados del primero de los trabajos [1] se describe la evaluación realizada en Aeneas para datos de etiquetas fiables en español. [20]

Capítulo 3

Descripción y diseño de los experimentos

3.1. Elaboración de la base de datos Librispeech Spanish

En este capítulo se tratará sobre el proceso de creación de la base de datos de voz y texto que se ha construido basándose en el corpus **Librispeech**, cuyo desarrollo se describe en 2.1.1.

El objetivo final de los Trabajos de Fin de Máster era el de poder entrenar un sistema de reconocimiento de voz en español. Para ello, en el primer trabajo se evaluaron una serie de alineadores forzados que han servido para la construcción de la base de datos, y que servirán a su vez para el entrenamiento del sistema de ASR.

3.1.1. Obtención y descarga de ficheros de audio y texto

El objetivo era el de conseguir una cantidad considerable de horas (>100h) de ficheros de voz en español y de licencia libre, de cara al posterior entrenamiento de un reconocedor de voz.

Para ello, el primer paso fue documentarse acerca de las páginas de las que se pueden obtener los datos, en concreto **LibriVox** [4] para los ficheros de audio (audiolibros), y **Project Gutenberg** [7] para las transcripciones (libros de texto).

El siguiente paso fue el de obtener todas las correspondencias entre aquellos libros y audiolibros que se encontraban en las respectivas páginas. El objetivo a priori fue el de, a partir de los códigos fuente de las dos páginas web, filtrar todos los títulos de las obras de cada una y ver cuáles coincidían. Debido a que la estructura de LibriVox está basada en hipervínculos, no se pudo hacer, y se obtuvieron los datos de los títulos manualmente. Mediante un script de Python se filtraron aquellos datos que no correspondían con títulos de obras y se obtuvieron las correspondencias entre obras de ambas páginas. El resultado fue un número mucho menor que el obtenido en la construcción de **Librispeech**, debido a que la cantidad de audiolibros disponibles en español es muy inferior a la que hay en inglés, y a que algunos de los títulos variaban ligeramente a pesar de ser la misma obra.

Después, ya que el número de horas de voz del que se disponía era insuficiente, se decidió escoger aquellas obras de LibriVox cuya duración fuera mayor, y, ya que se encuentran bajo la licencia **Creative Commons**, se pudieron buscar en otras páginas diferentes a Project Gutenberg sin dificultad.

La idea inicial fue la de descargar cada uno de estos ficheros de forma automática mediante comandos de Linux como **wget**, sin embargo, debido a la estructura de LibriVox no fue posible

hacerlo. Finalmente, se descargaron manualmente cada una de las obras, obteniendo un total de 18 audiolibros divididos en capítulos (217 horas de voz) y sus respectivas transcripciones, divididas en 18 carpetas con sus respectivos ficheros de texto.

3.1.2. Pre-procesado de audio y texto

Una vez se han obtenido los datos, es necesario revisarlos manualmente para ver que coinciden con cada uno de los textos. Tras hacerlo, se vio que en cada uno de los audiolibros se encuentra una introducción y una conclusión en la que se especifica la web de referencia, el nombre del lector, el nombre del libro y el número de capítulo. Además, se tuvo que anotar manualmente cuál es la duración de estos fragmentos de inicio y final, para después poder borrarlos de cada audio.

En primera instancia, para evitar tener que anotar estos instantes de tiempo manualmente y eliminar los fragmentos sobrantes, se utilizó un detector de actividad de voz o Voice Activity Detector (VAD) implementado por **AUDIAS** en el marco de un proyecto del grupo de investigación, que permite ver en qué instantes de cada audiolibro se encuentra hablando un lector y en cuáles hay silencio. Sin embargo, el carácter ruidoso de algunos de los audiolibros y la ausencia de pausas en el estilo de algunos de los lectores impedían que funcionase con un alto rendimiento.

Por tanto, se implementó un script de Python que permite introducir el directorio de la carpeta en la que se encuentran los capítulos, así como los segundos del inicio y del final a recortar, eliminando las partes que no interesan de todos los capítulos de una obra a la vez.

Ya que cada texto correspondía a una obra diferente, y cada audiolibro estaba dividido en capítulos, fue necesario dividir cada texto en el número de capítulos de cada audiolibro. Esta parte conllevó una carga de tiempo muy grande, ya que muchos de los capítulos estaban descuadrados respecto al texto, y fue necesario revisar manualmente uno a uno cada uno y crear un fichero de texto para cada uno de audio. En total, se pasó de tener 18 ficheros de texto, uno por cada obra, a un total de 601, uno por cada fichero de audio.

Por último, se realizó una normalización de todos los textos mediante el programa de Python **TextFilter** del grupo **AUDIAS**, que permite eliminar signos de puntuación, así como tener todos los textos en mayúsculas y sustituir números por palabras, eliminar guiones, paréntesis,etc.

3.1.3. Alineamiento forzado de audio y texto

En el primero de los trabajos de fin de máster se realizó una evaluación de los alineadores forzados más avanzados de la actualidad, con el objetivo de encontrar uno que se adaptase al problema que se plantea, de alinear una gran cantidad de ficheros de audio y texto de gran tamaño.

Tras evaluar los resultados de cada uno, se optó por utilizar el alineador Aeneas, detallado en 2.7.1, para realizar esta tarea.

Para ello, se nombraron los ficheros de la base de datos de la siguiente forma. En primer lugar, se nombraron las carpetas de cada obra mediante números de dos dígitos (00, 01, 02...). Del mismo modo, se nombró cada capítulo de cada obra añadiendo el número de cada uno (00_1, 00_2, 00_3...).

Una vez se renombraron los 601 capítulos, se implementó un programa de Python de forma que se pudiera alinear de forma automática toda la base de datos en una sola ejecución, obteniendo un fichero de alineamientos por cada fichero de voz, y que servirá para segmentar la base

de datos en ficheros de tamaño mucho menor, aptos para utilizarse como entrada en sistemas de reconocimiento de voz.

Una vez se realizó esta parte, la base de datos contaba con 601 ficheros de audio, 601 de texto y otros 601 de alineamientos entre cada uno de los anteriores, para un total aproximado de entre 1.7 y 2.5 de millones de palabras pronunciadas.


```
{
 "fragments": [
  {
   "begin": "1.840",
   "children": [],
   "end": "8.920",
   "id": "f000001",
   "language": "spa",
   "lines": [
    "Como todas las tardes, la barca-correo anuncio su llegada al Palmar con varios
        toques de bocina."
   ]
  {
   "begin": "8.920",
   "children": [],
   "end": "26.560",
   "id": "f000002",
   "language": "spa",
   "lines": [
    "El barquero, un hombrecillo enjuto, con una oreja amputada, iba de puerta en puerta
         recibiendo encargos para Valencia, y al llegar a los espacios abiertos en la
        unica calle del pueblo, soplaba de nuevo en la bocina para avisar su presencia a
         las barracas desparramadas en el borde del canal."
   ]
  },
```


Figura 3.1: Alineamiento de Aeneas en formato .JSON

Los alineamientos obtenidos se encuentran en formato JSON, y se procesarán en el siguiente capítulo para especificar al sistema de reconocimiento en qué partes debe segmentar la base de datos durante la fase de entrenamiento del sistema.

3.2. Preparación de datos de entrenamiento: Kaldi

Una vez se ha construido la base de datos es necesario generar los ficheros de texto que utiliza el reconocedor para obtener la información necesaria de los ficheros de voz y texto, de cara al entrenamiento del modelo, en el formato de recetas de Kaldi.

El primero de los ficheros se llama **text** y contiene cada una de las frases de cada uno de los textos correspondientes a cada capítulo de cada obra. Cada una de estas frases irá junto a un identificador o `utterance_id` diferente.

```
F_00_LBSPA_0_00_00000000_00007560 EL HOMBRE PISO ALGO BLANCUZCO Y ENSEGUIDA SINTIO LA
    MORDEDURA EN EL PIE
F_00_LBSPA_0_00_00007560_00019320 SALTO ADELANTE Y AL VOLVERSE CON UN JURAMENTO VIO UNA
    YARACACUSU QUE ARROLLADA SOBRE SI MISMA ESPERABA OTRO ATAQUE
F_00_LBSPA_0_00_00019320_00029120 EL HOMBRE ECHO UNA VELOZ OJEADA A SU PIE DONDE DOS
    GOTITAS DE SANGRE ENGROSABAN DIFICULTOSAMENTE Y SACO EL MACHETE DE LA CINTURA
F_00_LBSPA_0_00_00029120_00040880 LA VIBORA VIO LA AMENAZA Y HUNDIO MAS LA CABEZA EN EL
    CENTRO MISMO DE SU ESPIRAL PERO EL MACHETE CAYO DE PLANO DISLOCANDOLE LAS VERTEBRAS
F_00_LBSPA_0_00_00040880_00049120 EL HOMBRE SE BAJO HASTA LA MORDEDURA QUITO LAS GOTITAS
     DE SANGRE Y DURANTE UN INSTANTE CONTEMPLO
F_00_LBSPA_0_00_00049120_00055120 UN DOLOR AGUDO NACIA DE LOS DOS PUNTITOS VIOLETAS Y
    COMENZABA A INVADIR TODO EL PIE
F_00_LBSPA_0_00_00055120_00062400 APRESURADAMENTE SE LIGO EL TOBILLO CON SU PANUELO Y
    SIGUIO POR LA PICADA HACIA SU RANCHO
F_00_LBSPA_0_00_00062400_00081680 EL DOLOR EN EL PIE AUMENTABA CON SENSACION DE TIRANTE
    ABULTAMIENTO Y DE PRONTO EL HOMBRE SINTIO DOS O TRES FULGURANTES PUNTADAS QUE COMO
    RELAMPAGOS HABIAN IRRADIADO DESDE LA HERIDA HASTA LA MITAD DE LA PANTORRILLA
```

Figura 3.2: Texto en formato de entrada text.txt

El siguiente de los ficheros es **utt2spk**. Éste contiene el mismo utterance_id junto a un identificador de locutor para cada una de las frases.

```
F_00_LBSPA_0_00_00000000_00007560 F_00
F_00_LBSPA_0_00_00007560_00019320 F_00
F_00_LBSPA_0_00_00019320_00029120 F_00
F_00_LBSPA_0_00_00029120_00040880 F_00
F_00_LBSPA_0_00_00040880_00049120 F_00
F_00_LBSPA_0_00_00049120_00055120 F_00
F_00_LBSPA_0_00_00055120_00062400 F_00
F_00_LBSPA_0_00_00062400_00081680 F_00
F_00_LBSPA_0_00_00081680_00093880 F_00
F_00_LBSPA_0_00_00093880_00100200 F_00
F_00_LBSPA_0_00_00100200_00106920 F_00
```

Figura 3.3: Utt2spk con el utterance_id e identificador de locutor

La ruta de cada uno de los segmentos de audio junto a cada identificador de frase se encuentran en el fichero **wav.scp**. Dado que los ficheros de voz son demasiado grandes es necesario realizar una segmentación previa, para lo que utilizará el comando **sox** dentro del wav.scp. Los límites entre los cuales se dividieron los ficheros de voz vienen dados por los límites entre frases obtenidos en el capítulo anterior, en el alineamiento de audio y texto.

```
F_00_LBSPA_0_00_00000000_00007560 sox /data/test_luis/00.wav -t wav - trim 0 7.56 |
F_00_LBSPA_0_00_00007560_00019320 sox /data/test_luis/00.wav -t wav - trim 7.56 11.76 |
F_00_LBSPA_0_00_00019320_00029120 sox /data/test_luis/00.wav -t wav - trim 19.32 9.8 |
F_00_LBSPA_0_00_00029120_00040880 sox /data/test_luis/00.wav -t wav - trim 29.12 11.76 |
F_00_LBSPA_0_00_00040880_00049120 sox /data/test_luis/00.wav -t wav - trim 40.88 8.24 |
F_00_LBSPA_0_00_00049120_00055120 sox /data/test_luis/00.wav -t wav - trim 49.12 6 |
F_00_LBSPA_0_00_00055120_00062400 sox /data/test_luis/00.wav -t wav - trim 55.12 7.28 |
F_00_LBSPA_0_00_00062400_00081680 sox /data/test_luis/00.wav -t wav - trim 62.4 19.28 |
F_00_LBSPA_0_00_00081680_00093880 sox /data/test_luis/00.wav -t wav - trim 81.68 12.2 |
F_00_LBSPA_0_00_00093880_00100200 sox /data/test_luis/00.wav -t wav - trim 93.88 6.32 |
F_00_LBSPA_0_00_00100200_00106920 sox /data/test_luis/00.wav -t wav - trim 100.2 6.72 |
```

Figura 3.4: Fichero de entrada del reconocedor wav.scp

También fue necesario obtener un fichero llamado **segments** que proporcionase la información sobre los intervalos temporales entre los cuales se encuentra cada segmento de voz de cada fichero de audio. Este fichero permite al sistema de reconocimiento de voz saber en qué partes

puede dividir cada uno de los capítulos durante el reconocimiento de voz, ya que la duración de cada uno demasiado alta. Para ello se utilizó el detector de actividad de voz de **AUDIAS**. Este programa ya se utilizó en el primero de los trabajos, durante la evaluación de los alineadores forzados.

```
00_1_000000000-000000600 00_1 0.000 0.600
00_1_000003000-000009450 00_1 3.000 9.450
00_1_000009450-000015810 00_1 9.450 15.810
00_1_000017580-000022740 00_1 17.580 22.740
00_1_000022830-000025710 00_1 22.830 25.710
00_1_000025710-000028320 00_1 25.710 28.320
00_1_000029550-000033540 00_1 29.550 33.540
00_1_000033600-000041820 00_1 33.600 41.820
```

Figura 3.5: Fichero segments con los intervalos de voz de cada fichero

Además, fue necesario generar un modelo de lenguaje a partir de los 601 ficheros de texto de las obras. Para ello se generaron varios archivos. Para el primero de ellos se utilizó un script de Python que permite obtener, a partir de una carpeta de ficheros de texto, la lista de palabras totales en todas las obras, junto a un indicador de frecuencia de cada palabra.

A partir de este fichero, se utilizó el programa de bash **Audias_STT_G2P**, que utilizando la lista de palabras y frecuencias genera una transcripción fonética para cada una de ellas, y que sirvió como base del modelo de lenguaje.

Sin embargo, tras valorar las diferentes opciones y, tras observar que en el grupo **AUDIAS** se estaba trabajando con un reconocedor de rendimiento mucho mayor y mucho más rápido, se optó por realizar el entrenamiento y reconocimiento de la base de datos mediante el sistema ASR de **ESPnet**, descrito en 2.6.

3.3. Preparación de datos de entrenamiento: ESPnet

El proceso de entrenamiento del reconocedor de voz, así como la evaluación de dicho reconocedor mediante la decodificación de los datos de prueba, se llevó a cabo en colaboración con **Beltrán Labrador**, investigador del grupo **AUDIAS** que se encontraba trabajando como parte de su tesis doctoral en este sistema de ASR.

Los datos necesarios para el entrenamiento del reconocedor de ESPnet son los mismos que se utilizan en la receta del reconocedor de Kaldi, por lo que no supuso un gran cambio a la hora de preparar los datos.

Para preparar los datos, fue necesaria la implementación de un script que sacara los datos que requiere el sistema para el entrenamiento, para lo que se utilizaron diferentes herramientas de Matlab.

En primer lugar, este programa debía acceder a cada uno de los ficheros de alineamientos (figura 3.1) y obtener los intervalos entre los que se encontraba cada frase, así como las frases asociadas a cada intervalo.

A partir de éstos, se recorre la base de datos asociando un locutor a cada una de las obras, y un `utterance_id`, correspondiente a cada una de las frases de cada obra, escribiendo así el **utt2spk.txt** (figura 3.3).

A partir del `utterance_id` y cada frase cargada de los ficheros de alineamiento se escribe el fichero **text.txt**, tal y como se muestra en la figura 3.2.

A partir de los intervalos de alineamiento y las rutas de cada capítulo de audio se genera el fichero **wav.scp**, mostrado en la figura 3.4.

3.4. Entrenamiento y decodificación mediante ESPnet

Después de generar la base de datos descrita en 3.1, el siguiente paso es realizar el entrenamiento y reconocimiento de voz mediante el sistema de ASR de **ESPnet**.

Sin embargo, tras revisar los alineamientos entre ficheros de voz y texto, se vio que algunos de los alineamientos tenían valores de error mucho mayores a los obtenidos durante la evaluación de **Aeneas** en el primero de los Trabajos de Fin de Máster. Esto se debe a que algunos de los ficheros de texto correspondientes a determinados capítulos no coinciden exactamente con los ficheros de voz asociados, ya que, a pesar de que la base de datos se revisó manualmente para dividir cada obra en capítulos, fue imposible revisar que las 207 horas de voz coincidieran con cada uno de los textos de la base. Dentro de las líneas de trabajo futuro entraría revisar aquellos capítulos cuyo error de alineamiento es demasiado grande, para corregir aquellos errores en la transcripción que lleven a ello y poder realizar un entrenamiento del sistema con esta base de datos. Por ello, se decidió reenfocar los objetivos en los experimentos del reconocedor de voz, estableciendo los siguientes:

- Entrenamiento del reconocedor mediante la base de datos **LibriVox Spanish**, mostrada en 2.1.2.
- Reconocimiento de la base de datos creada en el primer trabajo, tal y como se describe en [1].
- Reconocimiento de la base de datos **Librispeech Spanish** mostrada en 3.1, con el objetivo de utilizar el resultado de la decodificación como entrenamiento del reconocedor de ESPnet.
- Entrenamiento del reconocedor de ESPnet con la salida obtenida en el punto anterior. Reconocimiento de la base de datos del primer trabajo y comparación de resultados.

El modelo utilizado para entrenar el modelo está basado en una red de codificación-decodificación basada en atención (Attention-based). Para la decodificación ESPnet hace uso de un modelo híbrido de CTC/Attention, como se explicó en 2.6 en la fase 4 del script **run.sh**.

El codificador está formado por una red neuronal convolucional VGG estándar, cuya salida entra en 5 capas BLSTM (Bidirectional Long Short-Term Memory) de 1024 neuronas cada una. En la segunda y tercera capas del encoder se hace un muestreo a la mitad, es decir, se eliminan la mitad de los frames. Esto se realiza ya que a la entrada del sistema se tiene un frame cada 10 ms, y en la salida se obtiene un carácter a intervalos de tiempo mayores. Esto ayuda a la red a aprender mejor durante el entrenamiento.

El decodificador está formado por una red de 2 capas LSTM de 1024 neuronas. En este sistema todo está integrado como una red neuronal y no es necesario el entrenamiento previo de un modelo acústico o de lenguaje. El entrenamiento realizado fue de 25 épocas con el corpus LibriVox Spanish, y de 5 épocas adicionales para el re-entrenamiento con Librispeech Spanish, sin observar cambios sustanciales en el aprendizaje de la red.

En el siguiente capítulo se mostrarán los resultados a nivel de palabra y de token, que es una unidad en la que se dividen las palabras. La división en tokens o **Byte Pair Encoding** (BPE) garantiza que las palabras más comunes se representen como un solo token, miestras que las palabras menos comunes se dividirán en dos o más tokens. Los tokens se generan en función de la frecuencia de cada una de las palabras.

Capítulo 4

Resultados

4.1. Librispeech Spanish

El resultado obtenido tras la elaboración de la base de datos mostrada en 3.1 consta de las siguientes obras, divididas en sus respectivos capítulos.

Obra	**Capítulos**	**Duración**	**Hablantes**
El 19 de Marzo y el 2 de Mayo	32	7h:30min	1
El Aprendiz de Conspirador	35	6h:45min	1
Bailén	35	7h	1
El Buscón	23	4:49h	1
Cádiz	35	8h:28min	1
Cañas y Barro	12	8h30min	1
El Doncel de don Enrique el doliente	40	12h:20min	1
Los Jinetes del Apocalipsis	22	13h	1
Juanita la Larga	47	8h:10min	1
La Odisea	24	14h	1
La Regenta	54	33h	1
Las Mil y Una Noches	49	23h	1
La Nariz de un Notario	6	3h:7min	1
Niebla	35	7h	1
El Romancero del Cid	24	3h:28min	1
El Quijote de la Mancha	79	43h	>30
Trafalgar (Episodios Nacionales)	17	5h	1
Zaragoza	32	7h:24min	1
TOTAL	601	207h	>45

Tabla 4.1: Base de datos Librispeech Spanish

En total se obtuvieron un total de 207 horas de voz segmentadas en 601 capítulos, una cantidad mucho mayor a la que se obtuvo en el corpus de LibriVox Spanish, descrito en 2.1.2.

Para contar el número de palabras de la base de datos se ha creado un programa de Python que recorre cada uno de los ficheros de texto segmentándolos en palabras y frases. El total de palabras obtenido es de 1.896.600. Además, se calculó el número de frases totales a partir del número de puntos ('.') y signos de puntuación ('?','!') que determinan el final de cada frase, obteniendo un total de 122.241 frases.

Finalmente, se puede determinar que la duración media de cada frase es de 6.01 segundos, y que cada frase tiene de media de 15.5 palabras por frase.

4.2. Entrenamiento y decodificación mediante ESPnet

En este capítulo se expondrán los resultados tras las pruebas realizadas con el reconocedor de voz de ESPnet. En primer lugar, se realizó un entrenamiento mediante la base de datos **LibriVox Spanish**, de 77 horas de duración, y se reconoció la base de datos elaborada en el primero de los trabajos [1]. Esta base de datos está formada por 20 ficheros de audio de 5 minutos de duración junto a sus transcripciones revisadas manualmente, y se le hará referencia a partir de ahora como **LBSPA20**.

En experimentos previos del grupo AUDIAS se realizó un entrenamiento y decodificación del reconocedor de voz de ESPnet utilizando la base de datos LibriVox Spanish, descrita en 2.1.2. Los resultados obtenidos fueron mejores que los correspondientes al entrenamiento y reconocimiento de la base LBSPA20, por lo que ésta se reviso manualmente, detectando y corrigiendo numerosos errores ortográficos, que llevaron a una notable disminución del error.

	Frases	Tokens	Corr	Sust	Borr	Ins	CER
Pre-revisión	533	22686	79.2	14.8	5.9	3.5	24.3
Post-revisión	531	22093	82.2	14.1	3.3	3.3	20.7

Tabla 4.2: Resultados decodificación LBSPA20 con ESPnet a nivel de token

	Frases	Palabras	Corr	Sust	Borr	Ins	WER
Pre-revisión	533	14679	78.3	18.9	2.8	4.3	26.1
Post-revisión	531	14672	81.1	16.2	2.7	4.0	22.9

Tabla 4.3: Resultados decodificación LBSPA20 con ESPnet a nivel de palabra

El las tablas de 4.2 y 4.3 se muestran los resultados del reconocimiento a nivel de token y palabra, antes y después de realizar la corrección manual de las erratas que contenía LBSPA20. En concreto, se tienen los porcentajes de tokens o palabras correctas (Corr), de sustituciones (Sust), de borrados (Borr), inserciones (Ins) y el **Character Error Rate** o error de carácter (CER) y **Word Error Rate** o error de palabra. Además de éstos se obtuvo un error de frase del 91.6 %, es decir, un 8.4 % de las frases se reconocieron sin ningún error.

Como se puede observar el CER y WER disminuyeron de forma considerable tras la corrección, de un 24.3 y 26.1 a un 20.7 y 22.9. Además, el error de frase bajó de un 91.6 % a un 80.6 %. Esto da una idea de que parte de los errores de reconocimiento en realidad eran errores de etiquetado. También refuerza la idea de que un corpus oral requiere una gran cantidad de refinado para conseguir que esté libre (o casi libre) de errores de etiquetado.

Durante la revisión de los errores se observaron alrededor de 45 errores de palabra debidos al acento en la pronunciación del hablante. Un ejemplo de estos errores es, por ejemplo, letras como la C pronunciadas como una S o B confundida con la V. Este error se debe a la falta de datos de entrenamiento del modelo de lenguaje, por lo que aumentar esta cantidad de texto es una parte interesante dentro de las líneas de trabajo a investigar.

Otra forma de mejorar los resultados del reconocimiento involucrando los resultados obtenidos en 4.1 es el re-entrenamiento del modelo de ESPnet mediante la base de datos Librispeech Spanish, cuyos pasos se verán en la siguiente sección.

4.2.1. Re-entrenamiento y decodificación mediante ESPnet

El primer paso para re-entrenar el reconocedor fue el de preparar y adaptar los ficheros de entrada del reconocedor tal y como se muestra en 3.2.

Una vez se obtuvieron estos ficheros se inició la decodificación de la base de datos Librispeech Spanish (207h) en una de las GPUs del grupo AUDIAS, tardando alrededor de 3 días en realizar el reconocimiento de los datos. Se obtuvieron un total de 177942 frases reconocidas para las 207 horas de voz.

Después de obtener como salida las transcripciones de cada audio de la base se revisaron manualmente para ver de forma subjetiva el grado de precisión del sistema, observando que algunas de las frases reconocidas tenían niveles de error del 100 %. Estos errores se deben a la precisión del VAD utilizado durante la obtención del fichero *segments* mostrado en la figura 3.5, ya que en ocasiones marca intervalos temporales demasiado cortos para los cuáles el reconocedor es incapaz de obtener ninguna transcripción.

En primer lugar se procesaron los ficheros de salida del reconocedor para el primer experimento (decodificación de LBSPA20) obteniendo de este modo el número de palabras por frase y el número de errores por cada frase reconocida. Esto se hizo con el objetivo de ver si los errores grandes se encontraban únicamente en frases muy largas o no dependían de ello.

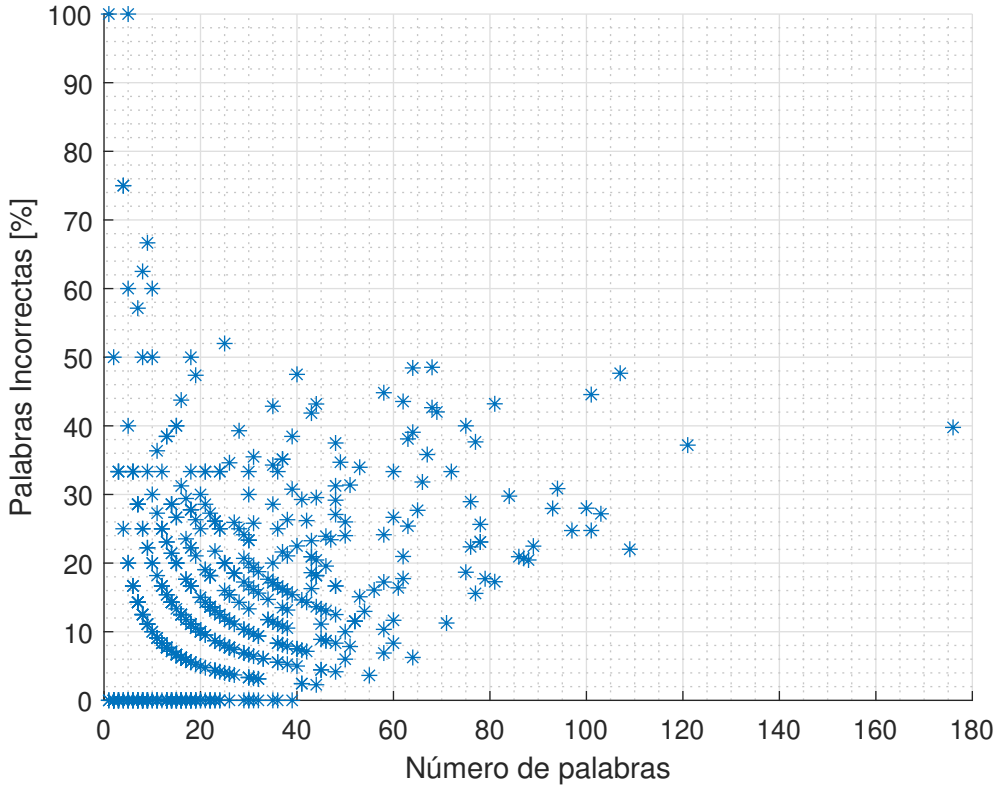

Figura 4.1: Errores de ESPnet según tamaño de frase en la base LBSPA20

Como se puede ver en la figura 4.1, la mayoría de frases con un porcentaje de error por encima del 10 % se encuentra en aquellas cuyo número de palabras es mayor. Las frases cuyo número de palabras es muy pequeño y cuyo error es muy elevado se corresponden a errores en el alineamiento forzado de los ficheros. En estos casos una o más palabras de una frase corta se alinean con la frase anterior o posterior, produciendo un porcentaje de error muy elevado en esa frase, debido a su corta duración.

Además, ninguna de las frases con un 0 % de error sobrepasa las 40 palabras. Sin embargo, eliminar todas las frases cuya duración sea muy alta llevaría a reducir considerablemente el número de horas de voz, por lo que se realizó un cambio en el enfoque.

Se decidió utilizar la información de confianza normalizada o **Normalized Likelihood Ratio** que proporciona la salida del reconocedor para cada frase decodificada.

El objetivo era el de obtener un umbral óptimo que permitiera escoger aquellas frases que tenían mayor probabilidad de haber sido bien reconocidas, de cara a utilizarlas en el re-entenamiento del sistema, ya que si los datos de entrada tienen errores demasiado grandes la red podría no entrenarse adecuadamente y su nivel de error aumentaría considerablemente.

El siguiente paso fue el de buscar una relación entre el nivel de confianza de cada frase y el porcentaje de error de cada una, obteniendo el siguiente resultado.

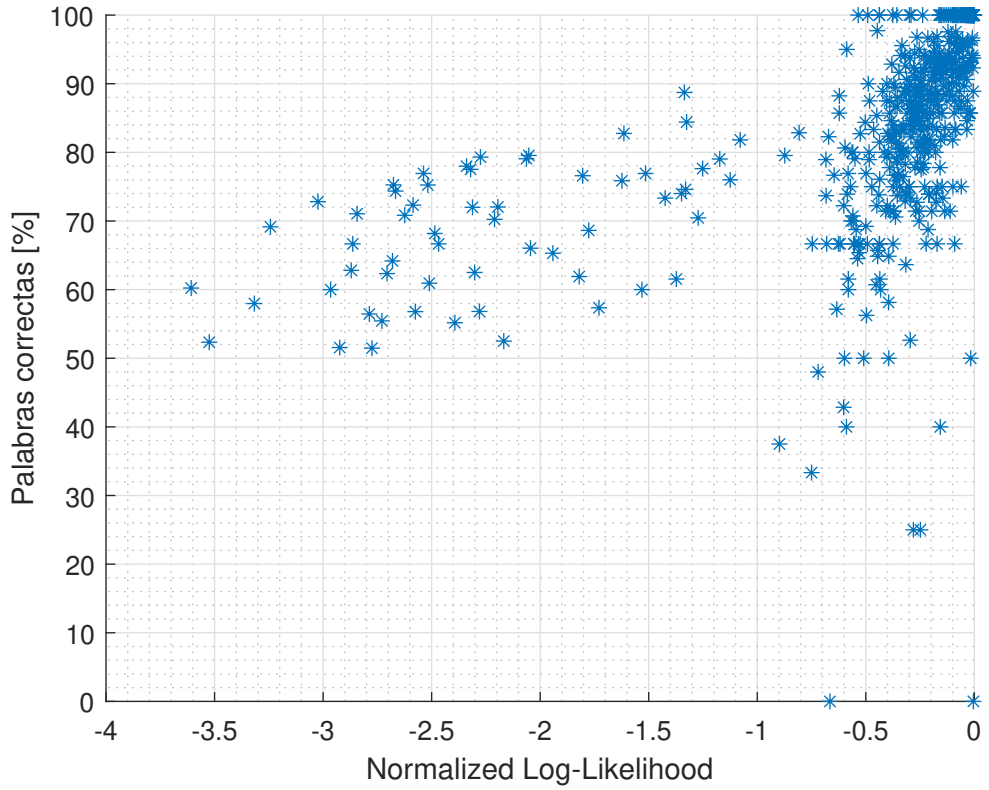

Figura 4.2: Relación entre confianza y palabras correctas en la base LBSPA20 (%)

Como se puede observar en la figura 4.2 el nivel de error por frase tiene una clara relación con el nivel de confianza que aporta el reconocedor, y por tanto, con la longitud de la frase reconocida. Sin embargo, el utilizar un umbral de Likelihood-Ratio en lugar de longitud de frase nos permite quedarnos con aquellas frases largas cuya decodificación ha sido buena.

Una vez se supo que el nivel de confianza es un buen indicador para ver de qué frases deshacerse el siguiente paso fue procesar y representar la distribución los valores de confianza para el reconocimiento de LBSPA20 y Librispeech Spanish.

En el histograma de distribución de valores de confianza por frase mostrado en la figura 4.3 se puede determinar que la distribución de los valores es similar en el reconocimiento de ambas bases de datos. La distribución de los valores de Librispeech Spanish es más suavizada debido a que se tienen 207 horas de audio en comparación con los 100 minutos que contiene LBSPA20.

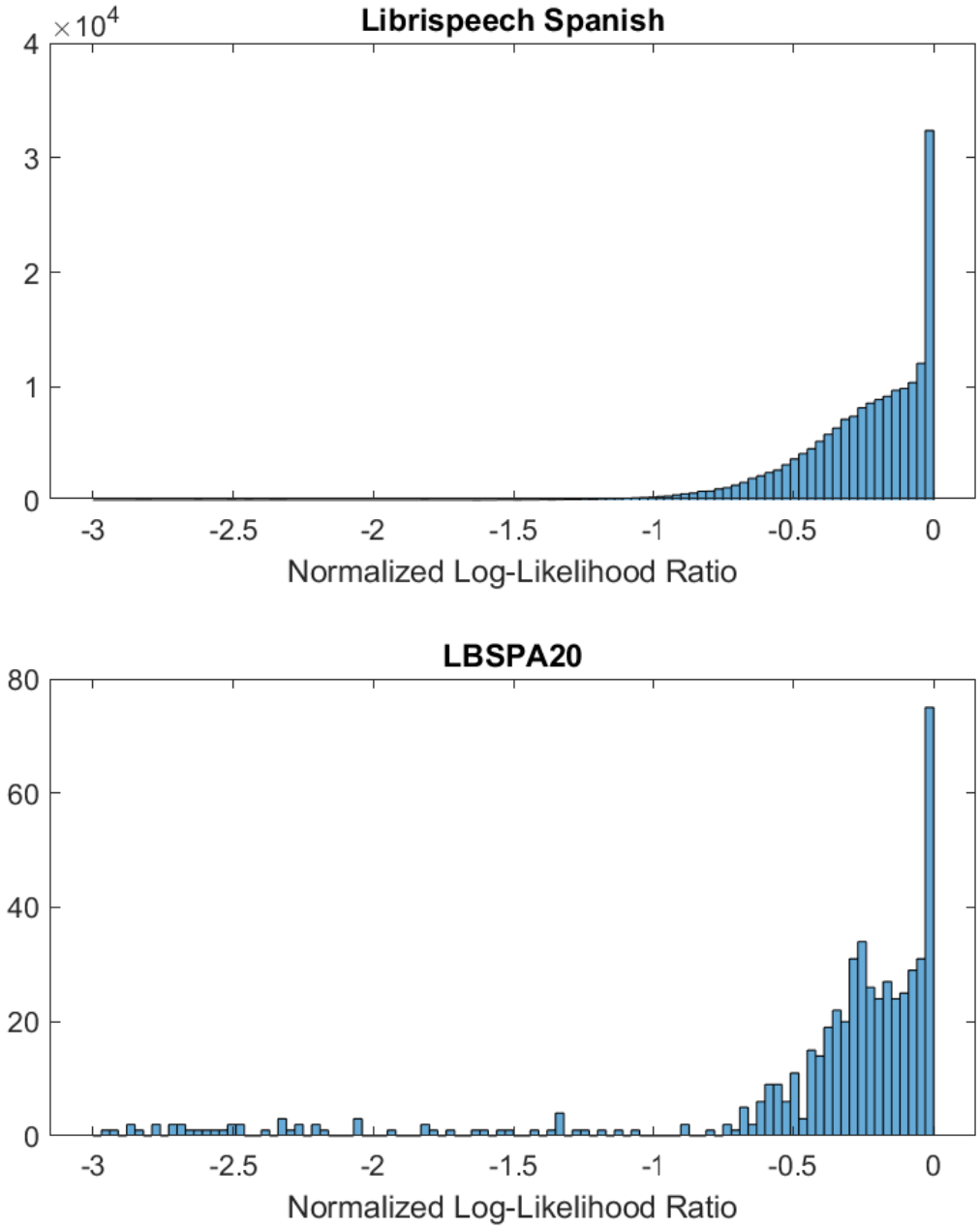

Figura 4.3: Distribución de valores de confianza por frase

La interpretación de las dos figuras llevó a plantear varios umbrales diferentes, teniendo en cuenta que cuanto mayor sea el umbral menor será el error de los datos de entrenamiento, pero también menor el número de horas.

Si uno se atiene únicamente a los histogramas se puede ver que aplicando un umbral de entre -0.5 y -0.4 se eliminarían todas las frases cuyo error fuera muy grande, sin perder demasiadas horas de voz.

La cantidad de horas restante después de quedarnos solamente con aquellas frases que sobrepasan los determinados umbrales de confianza es la que se muestra en la figura 4.4.

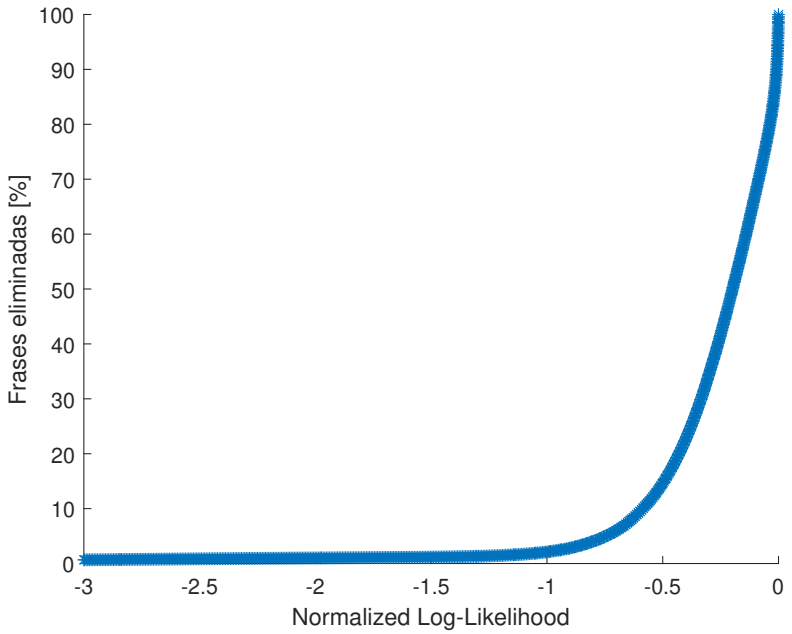

Figura 4.4: Frases eliminadas aplicando distintos umbrales de confianza

A partir de estos resultados se plantea re-entrenar el reconocedor para 4 umbrales diferentes, mostrados en la tabla 4.4.

Umbral	-0.1	-0.4	-0.5	-0.9
Frases entrenamiento (%)	32.66	77.26	85.36	97.18

Tabla 4.4: Porcentaje de frases restantes tras aplicar distintos umbrales de confianza

Para entrenar el reconocedor de voz con las frases que superen cada uno de los umbrales de confianza se decidió implementar un script específico para ello. A partir de los datos de reconocimiento de la base Librispeech Spanish se extrajeron los valores de confianza asociados a cada `utterance_id` de cada frase. A partir de los valores de confianza se establecieron las `utterance_id` de aquellas frases que se encontraban por encima del umbral. Dado que el orden de las frases de salida del reconocedor es diferente a la que se le introdujo como entrada, fue necesario comparar cada `utterance_id` obtenido con los de los datos de entrada del reconocedor, para determinar qué frases eliminar y cuáles no, obteniendo así los ficheros **text** 3.2, **utt2spk** 3.3 y **wav.scp** 3.4 específicos para cada uno de los umbrales utilizados.

A partir de estos ficheros, se entrenó el reconocedor de ESPnet con la base de datos LibriVox Spanish, así como con el 85.36 % de las frases del corpus Librispeech Spanish, correspondientes a aplicar un umbral de -0.5 para eliminar aquellas frases que tienen más probabilidad de haberse reconocido con un error muy elevado. Posteriormente, se decodificaron dos bases de datos diferentes: LBSPA20, y LibriVox Spanish.

Los resultados de la decodificación a nivel de palabra y token para cada una de las bases de datos mediante el reconocedor entrenado, en primer lugar con LibriVox Spanish (**LVS**) y para el segundo experimento con LibriVox Spanish + Librispeech Spanish (**LVS + LSS**) fueron los siguientes. Para el entrenamiento y test de LibriVox Spanish se utilizaron 33463 frases (aproximadamente 67 horas de voz) para entrenar el modelo y 2875 frases (6 horas de voz) para realizar la fase de test, para un total de 36338 frases y 73 horas de audio que componen el corpus.

Entrenamiento	Frases	Tokens	Corr	Sust	Borr	Ins	CER
LVS	531	22093	82.2	14.1	3.3	3.3	20.7
LVS + LSS	531	22093	82.4	14.2	3.2	3.4	20.8

Tabla 4.5: Resultados decodificación LBSPA20 con ESPnet a nivel de token

Entrenamiento	Frases	Palabras	Corr	Sust	Borr	Ins	WER
LVS	531	14672	81.1	16.2	2.7	4.0	22.9
LVS + LSS	531	14672	81.1	16.3	2.6	4.1	23.0

Tabla 4.6: Resultados decodificación LBSPA20 con ESPnet a nivel de palabra

Los resultados obtenidos en el reconocimiento del corpus **LBSPA20** se muestran en las tablas 4.5 y 4.6.

Los valores de WER Y CER son prácticamente iguales en ambos experimentos. Los resultados obtenidos son mejores que los esperados. La base de datos con la que se re-entrenó el modelo no está revisada y en este tipo de sistemas este hecho repercute muy negativamente en los resultados. Esto muestra que la red de ESPnet es capaz de entrenarse sin aumentar el error en el reconocimiento a pesar de que los datos añadidos extra no tengan la misma calidad que de los de partida. Es decir, el añadir más datos de entrenamiento de menor calidad no mejora pero tampoco empeora el rendimiento del reconocedor.

Entrenamiento	Frases	Tokens	Corr	Sust	Borr	Ins	CER
LVS	2875	78746	88.8	8.7	2.5	1.6	12.7
LVS + LSS	2875	78746	88.5	8.9	2.5	1.6	13.0

Tabla 4.7: Resultados decodificación LibriVox Spanish con ESPnet a nivel de token

Entrenamiento	Frases	Palabras	Corr	Sust	Borr	Ins	WER
LVS	2875	54776	87.7	10.1	2.2	1.7	14.0
LVS + LSS	2875	54776	87.4	10.4	2.2	1.7	14.3

Tabla 4.8: Resultados decodificación LibriVox Spanish con ESPnet a nivel de palabra

En las tablas 4.7 y 4.8 se muestran los resultados tras el re-entrenamiento del sistema y la decodificación del corpus LibriSpeech Spanish.

Los resultados obtenidos son similares, aunque muy ligeramente inferiores a los del entrenamiento únicamente mediante LibriVox Spanish. Por tanto, se puede afirmar que es necesaria una limpieza y filtrado adicional de los datos de forma manual para que se reduzca el número de errores, y volver a entrenar el reconocedor con ella para determinar si mejoran o no los resultados.

Capítulo 5

Conclusiones y trabajo futuro

En este Trabajo de Fin de Máster se ha detallado el proceso de construcción de una base de datos de acceso libre para los usuarios en español de cara a entrenar sistemas de reconocimiento de voz. En primer lugar, se obtuvieron los ficheros de los proyectos **LibriVox** y **Gutenberg**. Posteriormente, se procesaron los datos a través de numerosos scripts que automatizaron los procesos hasta obtener el resultado final. En total se obtuvieron un total de 207 horas de voz junto a sus transcripciones, no revisadas de forma manual, pero con una cantidad de errores muy pequeño. La base de datos generada se utilizó para entrenar un sistema de ASR, todo ello mediante la herramienta **ESPnet**, también de acceso libre y cuyos resultados mejoran a los modelos anteriores, implementados en **Kaldi** o **HTK**. Los resultados obtenidos fueron buenos, con niveles de *Word Error Rate* y *Character Error Rate* de un 14 % y un 23 % para la decodificación de dos bases de datos revisadas manualmente. Sin embargo, no hubo diferencia entre el entrenamiento del reconocedor de voz con una base de datos sin errores (**LibriVox Spanish**) y el re-entrenamiento mediante ésta última y la elaborada durante este trabajo. El hecho de que los niveles de error se mantengan prácticamente igual determina que la base de datos no tiene niveles de error muy elevados, pero que no son suficientemente pequeños para entrenar un sistema de ASR y mejorar su rendimiento.

Los valores mayores de error se obtuvieron utilizando la base de datos creada para el primer Trabajo de Fin de Máster, que contiene algunos errores a nivel ortográfico así como en los alineamientos de audio y texto que utiliza el reconocedor para segmentarla durante la decodificación. Por ello, se ha visto que es de gran relevancia tener una base de datos de un número considerable de horas, pero que es igual o más importante que estos corpus estén revisados manualmente y que los ficheros de voz coincidan exactamente con las transcripciones, así como que los alineamientos tengan el menor nivel de error posible, que será nulo si se realizan de forma manual. En caso de no tener una base de datos revisada de este modo, a pesar de que la mayoría de datos coincidan, se puede ver que hay un salto del 23 % del WER en la decodificación del corpus **LBSPA20** (con algunos errores) respecto de los valores de error de en torno al 14 % en el corpus revisado a mano detalladamente (**LibriVox Spanish**).

Como líneas de trabajo futuro se proponen varios proyectos. Es interesante revisar las 207 horas de voz y texto para que la base de datos construida carezca de ningún error y sea de utilidad para entrenar sistemas de ASR del mismo modo que se hizo en Librispeech o LibriVox Spanish. En segundo lugar, en el primero de los TFMs se implementó un alineador forzado cuyos resultados fueron buenos, a pesar de que se utilizó para ello un reconocedor de voz mucho menos avanzado que el de ESPnet. Por tanto un posible trabajo sería implementar un alineador forzado mucho más efectivo mediante este reconocedor mucho más moderno y potente, una vez que se ha estudiado su uso y se han realizado experimentos con él.

Bibliografía

[1] Luis Miguel Martínez Antolín. Estudio e implementación de técnicas de alineamiento de grandes volúmenes de voz y texto. 2020.

[2] Kandarpa Sarma and Mousmita Sarma. *Acoustic Modelling of Speech Signal using Artificial Neural Network: A Review of Techniques and Current Trends*, pages 287–303. 06 2015.

[3] Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, et al. Espnet: End-to-end speech processing toolkit. *arXiv preprint arXiv:1804.00015*, 2018.

[4] Jodi Kearns. Librivox: Free public domain audiobooks. *Reference Reviews*, 2014.

[5] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

[6] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukáš Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlíček, Yanmin Qian, Petr Schwarz, Jan Silovský, Georg Stemmer, and Karel Vesel. The kaldi speech recognition toolkit. *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, 01 2011.

[7] Project gutenberg. (n.d.). retrieved february 21, 2016, from www.gutenberg.org. 2020.

[8] Temple F Smith, Michael S Waterman, et al. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

[9] Sylvain Meignier and Teva Merlin. Lium spkdiarization: an open source toolkit for diarization. 2010.

[10] Carlos Daniel Hernández. Mena. Librivox spanish ldc2020s01. web download. philadelphia: Linguistic data consortium. 2020.

[11] John Levis and Ruslan Suvorov. *Automatic Speech Recognition*. 11 2012.

[12] R. Schmidt and R. Neumann. Automatic text-to-speech alignment: Aspects of robustification. In Václav Matousek, Pavel Mautner, Jana Ocelíková, and Petr Sojka, editors, *Text, Speech and Dialogue*, pages 72–76, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[13] Todd K Moon. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996.

[14] Biing Hwang Juang and Laurence R Rabiner. Hidden markov models for speech recognition. *Technometrics*, 33(3):251–272, 1991.

[15] Steve J Young and Sj Young. *The HTK hidden Markov model toolkit: Design and philosophy*. University of Cambridge, Department of Engineering Cambridge, England, 1993.

[16] Roberto Carrillo Aguilar. Diseño y manipulación de modelos ocultos de markov, utilizando herramientas htk. una tutoría. *Ingeniare. Revista chilena de ingeniería*, 15(1):18–26, 2007.

[17] David Rybach, Stefan Hahn, Patrick Lehnen, David Nolden, Martin Sundermeyer, Zoltan Tüske, Simon Wiesler, Ralf Schlüter, and Hermann Ney. Rasr-the rwth aachen university open source speech recognition toolkit. In *Proc. ieee automatic speech recognition and understanding workshop*, 2011.

[18] Simon Gonzalez, James Grama, and Catherine E Travis. Comparing the performance of major forced aligners used in sociophonetic research.

[19] Paul Lamere, Philip Kwok, Evandro Gouvea, Bhiksha Raj, Rita Singh, William Walker, Manfred Warmuth, and Peter Wolf. The cmu sphinx-4 speech recognition system. 1:2–5, 2003.

[20] Aeneas alignment tool. `http://www.readbeyond.it/aeneas/docs/`. Accessed: 2019-09-30.