UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

Deep learning aplicado al reconocimiento de locutor dependiente de texto

Máster Universitario en Ingeniería de Telecomunicación

Autor: Palomo Sánchez, Álvaro
Tutor: Torre Toledano, Doroteo

FECHA: Junio, 2019

Deep learning aplicado al reconocimiento de locutor dependiente de texto

AUTOR: Álvaro Palomo Sánchez
TUTOR: Doroteo Torre Toledano

AUDIAS- Audio, Data Intelligence And Speech
Dpto. Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2019

Resumen (castellano)

Este trabajo de fin de master tiene como objetivo el estudio, investigación e implementación de un sistema de reconocimiento de locutor dependiente de texto, para ello trabajaremos con la base de datos RSR2015, una base de datos destinada al estudio y desarrollo de estos sistemas.

En muchos de los sistemas de reconocimiento de locutor dependiente de texto se parte de un modelo general de locutores y se adapta este modelo a un modelo de locutor con las características específicas del mismo, por lo que hay que entrenar un modelo de locutor por cada uno de los locutores que tengamos. La idea de este trabajo es crear un sistema que generalice el reconocimiento de locutor sin un entrenamiento previo específico, vamos a crear un sistema innovador que nunca antes se había probado. Como veremos, se ha obtenido unos resultados más que aceptables, mejorando incluso resultados obtenidos en trabajos anteriores.

Para construir nuestro sistema primero extraeremos las características acústicas y fonéticas de los audios, como son los MFCCs o los posteriorgramas, para después combinarlos y extraer la correlación de Pearson frame a frame. Una vez que tenemos estas matrices de correlaciones usaremos redes neuronales para extraer las características locales de estas matrices de correlaciones.

Trabajaremos sobre un escenario en el que el locutor tiene que pronunciar una frase para autenticarse y el sistema decide si es un locutor genuino o un impostor. Trataremos de mejorar los resultados obtenidos en el TFG *Mejoras en un sistema de reconocimiento de locutor dependiente de texto.*

Abstract (English)

This TFM aims to study, research and implement a text-dependent recognition system, we will work with the RSR2015 database, a database created for the study and development of these systems.

In many of text-dependent speaker recognition systems, we start with a general model of speakers and adapt this model to a speaker model with the specific characteristics of the speaker, so we have to train a speaker model per speaker. The idea is to create a system that generalizes the recognition without specific prior training, we will create an innovative system that has been never before tested and obtaining more than acceptable results, improving even results obtained in previous jobs.

To build our system we will first extract the acoustic and phonetic characteristics of the speech, such as MFCCs or posteriorgrams, and then combine them and extract the Pearson correlation frame by frame, once we have these correlation matrices we will use neural networks to extract local characteristics of these correlation matrices.

We will work on a scenario in which the speaker has to pronounce a phrase to authenticate and the system decides if it is a genuine speaker or an imposter. We will try to improve the results obtained in the TFG Mejoras en un sistema de reconocimiento de locutor dependiente de texto.

Palabras clave (castellano)

Reconocimiento del locutor dependiente de texto, Verificación de locutor, Redes Neuronales, Reconocimiento de voz, RSR2015., , ,

Keywords (inglés)

Speaker recognition, RSR2015, Neural Networks, Text-dependent speaker recognition, Speaker verification, UBM, ASR, Speaker adaptation, Speaker models.

Agradecimientos

Me gustaría en primer lugar agradecer este TFM a mi tutor Doroteo con el que tanto he aprendido como profesor y como tutor a lo largo de mis años en la universidad.

También me gustaría dedicárselo a todas esas amistades que me llevo de la universidad y con las que he pasado tanto tiempo.

Por último me gustaría agradecérselo a mi familia que tanto ha confiado en mí y me han apoyado durante todos estos años de universidad.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 Introducción

1.1 Motivación

Este TFM tiene como objetivo investigar en la tarea de reconocimiento de locutor dependiente de texto. La tarea de reconocimiento de locutor dependiente de texto tiene como base la tarea de reconocimiento de la voz.

La voz es un rasgo biométrico característico de la persona, a partir de la voz somos capaces de identificar muchos rasgos de la persona, entre ellos la identidad del hablante, sexo, edad e idioma entre otros, es por eso que dentro del ámbito de reconocimiento de voz hay muchas tareas como puede ser el reconocimiento de idioma.

En el caso de este trabajo se corresponde con la verificación de locutor, donde un usuario reclama una identidad y el sistema se encarga de verificar si es genuina esa persona, estos sistemas pueden servir como sistemas de acceso biométrico. Son muchas las empresas que están desarrollando sistemas de acceso biométrico para llevar al mundo del gran público o electrónica de consumo, como pueden ser el acceso con huella dactilar o reconocimiento facial.

Los sistemas de acceso biométrico son mucho más seguros que los accesos con sistemas tradicionales como contraseñas, que en muchos casos es poco robusta, es cierto que se están implementando sistemas con doble factor de autentificación pero no son tan seguros como los accesos biométricos donde las características únicas de cada persona son las que nos dan el acceso.

En concreto este trabajo se basará en el rasgo biométrico de la voz, donde existen ya muchas aplicaciones en este ámbito, algunas de ellas pueden ser:

- Autentificación o verificación por voz: Consiste en determinar si el hablante es verdaderamente quién dice ser.
- Reconocedor de voz: Se encarga de reconocer lo que se ha dicho en un audio, se puede transformar segmentos de audio en texto (speech to text) o simplemente reconocer lo que se ha dicho para otros fines.
- Identificación de locutor: Consiste en identificar al locutor de entre muchos locutores posibles, esto es distinto a verificación donde ya partimos con la premisa de quién es el locutor.
- Diarización de locutores: Consiste en identificar a locutores dentro de audios con diversos locutores, con el objetivo de segmentar ese audio por cuando interviene ese locutor en concreto.

De estas aplicaciones que hemos visto, nuestro trabajo consiste en el primero de ellos, la autentificación o verificación de locutores, uno de los escenarios típicos de estos sistemas es el siguiente, un usuario que intenta acceder a su cuenta a través de su móvil o tablet y el sistema le pide introducir una contraseña mediante la voz, este sistema se encargará por lo tanto de verificar al locutor de manera positiva o negativa.

Con el fin de evitar posibles grabaciones del locutor, en muchos sistemas se implementa una frase generada aleatoriamente por el sistema.[13]

Una de las principales motivaciones de este TFM es el de continuar el TFG realizado en 2017, *Mejoras en un sistema de reconocimiento del locutor dependiente de texto*. Este TFG tenía como objetivo mejorar los resultados previos de otro TFG, es cierto que conseguimos mejorar los resultados, pero aun había margen de mejora. Es aquí donde está la principal motivación de este TFM, mejorar esos resultados y seguir investigando en este ámbito con la base de datos RSR2015 y así, contribuir en la medida de lo posible en la comunidad científica.

1.2 Objetivos

Como hemos comentado, una de las principales motivaciones de este TFM es también uno de los principales objetivos, la mejora de los resultados conseguidos en el TFG: *Mejoras en un sistema de reconocimiento del locutor dependiente de texto.*

Para conseguir estos objetivos vamos a seguir trabajando con la base de datos RSR2015.

El objetivo principal de este trabajo es la creación de un sistema innovador de reconocimiento de locutor dependiente de texto que sea capaz de generalizar la decisión de si un locutor es genuino o es un impostor, en muchos sistemas de reconocimiento de locutor dependiente de texto se partía de un modelo general de locutores y este modelo se adaptaba a las características particulares de un hablante, esto es lo que se llama modelo de locutor.

En este trabajo vamos a crear mediante el uso de redes neuronales un sistema que nunca antes se había probado, este sistema pretende que el reconocimiento de locutor sea independiente del modelo de locutor, es decir, que sea capaz sin previo entrenamiento particular de locutor de decidir si un locutor es genuino o impostor, únicamente necesitaremos una frase previa de ese locutor. Para ello extraeremos previamente las características de los audios de la base de datos RSR2015 y compararemos audio a audio, la red neuronal se encargará de encontrar los patrones dentro de los MFCCs o de los posteriorgramas para su decisión, en los siguientes capítulos entraremos en detalle sobre que son los MFCCs, posteriorgramas y cómo será la construcción del sistema.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Capítulo 1: Introducción**

  En este capítulo se hará una breve introducción al tema relacionado con el trabajo y se expondrá la motivación y los objetivos del TFM.

- **Capítulo 2: Estado del arte**

  En este capítulo hablaremos sobre el estado del arte y las técnicas punteras que se utilizan para el reconocimiento de voz, reconocimiento de locutor y de las redes neuronales.

- **Capítulo 3: Diseño**

  Aquí hablaremos sobre las herramientas que hemos utilizado y nuestro esquema de trabajo, para dar una idea del sistema de reconocimiento que queremos montar.

- **Capítulo 4: Desarrollo**

  En el desarrollo entraremos en el detalle de nuestro trabajo, explicaremos paso a paso como hemos utilizado las herramientas anteriormente descritas y como hemos ejecutado el trabajo.

- **Capítulo 5 Integración pruebas y resultados**

  En este capítulo mostraremos los resultados obtenidos en nuestro trabajo y haremos un breve análisis.

- **Capítulo 6: Conclusiones y trabajo futuro**

  Por último haremos una conclusión y un análisis global del trabajo explicando y dando una visión general del mismo, hablaremos también sobre las posibles líneas de investigación futura.

2 Estado del arte

En esta sección hablaremos sobre el estado del arte de las técnicas para reconocimiento de locutores, en concreto nos centraremos en el reconocimiento de locutor dependiente de texto, aunque también hablaremos sobre el reconocimiento de locutor independiente de texto, por último hablaremos sobre las redes neuronales centrándonos en las que hemos usado en este trabajo.

2.1 Reconocimiento de locutor

El reconocimiento de locutor consiste en la tarea de reconocer la identidad del hablante a partir de su voz. Para ello existen diferentes técnicas que veremos más adelante.

Dentro de la tarea de reconocimiento de locutor podemos encontrar dos tipos de dependencias con el contenido léxico, el reconocimiento de locutor dependiente de texto y el reconocimiento de locutor independiente de texto, ambos utilizan técnicas muy diferentes.

En el reconocimiento de locutor dependiente de texto, los sistemas utilizan el contenido léxico de la frase para reconocer al locutor. Podemos clasificar los sistemas dependiente de texto en 2 grupos, los sistemas con texto fijo y los sistemas con texto variable.

En los sistemas con texto fijo el contenido léxico tanto en el entrenamiento como en la tarea de reconocer al locutor siempre es igual, por el contrario, en los sistemas con texto variable el contenido léxico es diferente en el entrenamiento y en el test, además, en cada prueba de test, la frase a pronunciar es distinta. Estos sistemas de tipo variable son mucho más robustos que los de texto fijo ya que no pueden ser atacados mediante grabación de frase.[13]

Por otro lado tenemos los sistemas de reconocimiento de locutor independiente de texto, estos sistemas tratan de minimizar la influencia del contenido léxico, que es considerado como desconocido para el reconocimiento de locutor.

Este trabajo trata sobre un sistema de reconocimiento de locutor dependiente de texto, es por ello que nos centraremos en este sistema y sus técnicas.

A continuación vamos a explicar cómo funciona un sistema de reconocimiento de locutor dependiente de texto.

2.2 Reconocimiento de locutor dependiente de texto

Los sistemas de reconocimiento de locutor dependiente de texto pueden ser muy variados, pero todos se basan en una misma estructura:

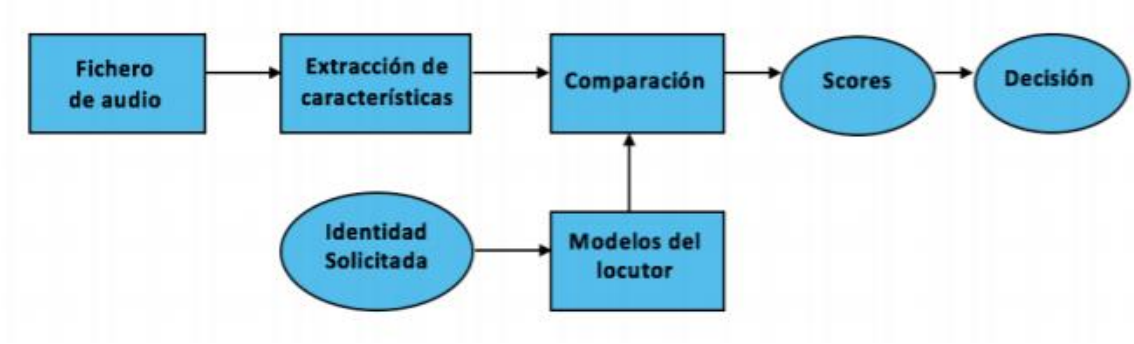

Figura 1: Esquema reconocimiento de locutor [2]

Como se puede ver en el esquema anterior tenemos 2 entradas, por un lado la entrada donde el locutor reclama una identidad, esta es la entrada etiquetada con identidad solicitada, y por otro lado tenemos el fichero de audio que es la grabación del locutor pronunciando una contraseña de acceso al sistema, por ejemplo.

El segundo paso del sistema es la extracción de características, en este paso el sistema se encarga de analizar el audio, que es la grabación del locutor, y extraer la información, existen varias técnicas para extraer información de los audios pero comúnmente se utilizan los MFCCs (Mel Frequency Cepstrum Coefficients), que explicaremos en la siguiente sección, y también existen los LPCC (Linear Prediction Cepstrum Coeficients) pero no se usan en este tipo de tareas.

Una vez extraídas estas características ya tenemos una primera entrada para el bloque de comparación, la otra entrada en el bloque de comparación es el modelo de locutor, este modelo de locutor es diferente según reclamemos una identidad u otra.

Los modelos de locutor son previamente entrenados, para ello se parte de un modelo general de locutores llamado UBM (Universal Background Model), este modelo UBM está entrenado por muchísimos locutores distintos. Trata de recoger unas características generales de un grupo muy grande, como pueden ser los locutores de un país. Una vez que tenemos el UBM aplicamos técnicas de adaptación de locutor, que veremos en una sección posterior, para adaptar las características particulares del locutor al modelo general, de esta manera hacemos un modelo especifico por locutor.

Una vez que tenemos las dos entradas preparadas el bloque de comparación hace una comparación (pattern matching) que devuelve una puntuación en base a la similitud.

La decisión finalmente se hace en base a las puntuaciones extraídas del bloque de comparación, por cada intento de acceso se hacen 2 comparaciones que nos devuelven 2 scores distintos, por un lado tenemos la comparación de la grabación con el modelo UBM, que nos devuelve el $Score_{SI}$ y por otro lado una comparación con el modelo de locutor especifico de la identidad que estamos reclamando, $Score_{SD.}$ Los subíndices SI y SD se corresponde con Speaker Independent y Speaker Dependent.

Estos scores representan la probabilidad que tiene la locución de pertenecer a ese modelo.

$$ScoreSI = P(X|\lambda_{SI})$$
$$ScoreSD = P(X|\lambda_{SD})$$

En base a estas probabilidades asociadas y a un umbral, Ω, que se puede modificar, calculamos nuestra decisión de aceptar o rechazar al locutor que intenta acceder.[2]

$$\frac{P(X|\lambda sd)}{P(X|\lambda si)} = \theta \begin{cases} \theta \geq \Omega \textit{ Aceptamos al locutor} \\ \theta < \Omega \textit{ Rechazamos al locutor} \end{cases}$$

Alternativamente:

$$\theta = Log\big(P(X|\lambda sd)\big) - Log(P(X|\lambda si))$$

El umbral Ω lo controlaremos nosotros en base a 2 estadísticas, la FA (Falsa aceptación) y la FR (Falso Rechazo), la FA se considera cuando se acepta a un locutor impostor ya que está por encima del umbral y el caso contrario, el FR, es cuando se rechaza a un locutor genuino ya que está por debajo del umbral. Este umbral por lo tanto es un parámetro modificable en nuestro sistema, por lo general se ajusta a un umbral donde la FA = FR, esto se le llama el EER (Equal Error Rate), aunque se puede ajustar el sistema para que sea más restrictivo y reducir las falsas aceptaciones aumentando los falsos rechazos.[5]

2.2.1 Extracción de características

Una de las tareas más importantes en estos sistemas es la extracción de las características acústicas de la voz.

Como hemos comentado anteriormente la voz contiene mucha información del locutor, no solo tenemos la información del mensaje, sino también su identidad, edad, y sexo entre otros.

El proceso por el cual los humanos somos capaces de hablar y comunicarnos mediante el habla ha sido objeto de estudio durante años, una vez que el cerebro humano tiene el mensaje que quiere pronunciar se produce una serie de procesos articulatorios en el aparato fonador del emisor que hace posible emitir el mensaje, este mensaje no solo contiene el mensaje lingüístico, también la identidad del hablante. Estos procesos en el aparato fonador del emisor quedan reflejados en la forma de onda del mensaje codificado en una combinación de características espectro-temporales. [6]

Estas características espectro-temporales son las que nos interesan decodificar para interpretar el mensaje y la identidad del locutor, para la extracción de características bajaremos al nivel más bajo, esto son las características espectrales de la señal. En las características espectrales podemos encontrar las particularidades de como se ha articulado el tracto vocal del locutor.

Para extraer las características espectrales del mensaje se utiliza el análisis short-term o a corto plazo. Para hacer un análisis de la señal lo más preciso posible necesitamos que la señal tenga propiedades estacionarias que no son fáciles de encontrar en una señal que tiene

continuos cambios, para obtener una señal lo más estacionaria posible lo que se hace es analizar la señal en ventanas de pequeñas longitudes de 20 ms o 40 ms.[6]

Una vez que tenemos un pequeño frame de 20 ms o 40 ms, analizamos esta ventana y la parametrizamos mediante técnicas como LPC o MFCCs.

La técnica más empleada para extraer las características acústicas son los MFCCs (Mel Frequency Cepstral Coefficients), los MFCCs se extraen aplicando la escala de Mel como escala ajustada en frecuencia a las características espectrales de la señal.

El proceso para la extracción de los MFCCs es el siguiente [6]:

1. **Pre-énfasis:** Aplicamos una ganancia a la señal con el objetivo de compensar la atenuación en altas frecuencias.
2. **Extracción de tramas:** Esta extracción consiste en dividir la señal en frames de 20 ms a 40 ms para conseguir una señal lo más estacionaria posible.
3. **Enventanado:** Aplicamos un enventanado, esto es, multiplicar en tiempo una función de tipo ventana, como puede ser la ventana Hamming.
4. **Cálculo de los parámetros MFCC:**
    a. **Cálculo de la DFT:** Calculamos la DFT (Discrete Fourier Transform) mediante la FFT (Fast Fourier Transform) para pasar la señal a frecuencia y nos quedamos con el módulo del espectro.
    b. **Cálculo de energía:** Calculamos la energía en cada una de las bandas de la escala de Mel aplicando un banco de filtros, estos bancos de filtros están centrados en las bandas críticas en la escala de Mel.

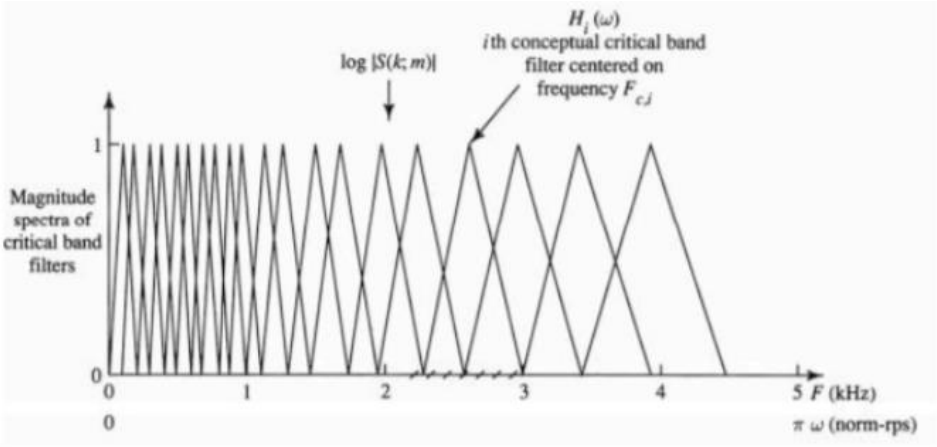

Figura 2: Banco de filtros Mel [6]

    c. **Transformación Cepstral:** Al resultado obtenido tras aplicar el banco de filtros aplicamos la DCT (Discrete Cosine Transform), estos coeficientes obtenidos son los MFCC, habitualmente nos quedamos con los 13 primeros coeficientes MFCC, aunque en algunas tareas que se requiere mayor precisión se utilizan hasta 19 coeficientes.

Una vez que hemos obtenido los MFCCs también se pueden extraer los coeficientes Delta y Delta-Delta que nos dan información sobre velocidad y aceleración. Estos coeficientes nos

dan información sobre como es la coarticulación y se calculan teniendo en cuenta como varían los coeficientes en ventanas adyacentes.
Para evitar el efecto del canal se aplican técnicas de normalización de los coeficientes como pueden ser CMS (Cepstral Mean Substraction) o CMVN (Cepstral Mean and Variance Normalization).

2.2.2 Reconocimiento de voz: Modelos de lenguaje, léxicos y acústico-fonéticos

Todos los sistemas de reconocimiento de voz tienen 3 modelos básicos sobre los que se asienta el reconocimiento, el primero de ellos es el modelo de lenguaje, el modelo de lenguaje modela como se forman las frases, palabras y cuáles son sus fonemas, los modelos de lenguaje están fuertemente ligados al idioma sobre el que estamos trabajando, básicamente calcula las probabilidades de que un fonema vaya seguido de otro. Estos modelos de lenguaje están creados a partir de N-gramas.

Como segundo modelo tenemos el modelo léxico, este modelo está formado por las palabras que nuestro reconocedor es capaz de identificar, para ello partimos de un diccionario base, este diccionario tendrá una transcripción palabra – fonemas, esto es así ya que nuestro reconocedor solo reconoce fonemas, que es la unidad mínima de lenguaje, aunque para reducir combinaciones posibles los reconocedores utilizan tri-fonemas.

El último modelo es el acústico-fonético, este modelo es el más complejo de construir. El modelo acústico-fonético se encarga de modelar como suenan los distintos fonemas dentro de un idioma, estos modelos están formados por HMM (Hidden Markov Model) y son modelos probabilísticos entrenados a partir de muchísimas locuciones de distintos locutores de una misma lengua.[12]

Estos tres modelos juntos forman un reconocedor de voz, los tres son iguales de importantes ya que se apoyan unos a otros.

En la actualidad, los reconocedores de voz modernos tienden a sustituir algunos o todos estos modelos por redes neuronales profundas.

2.2.3 Adaptación de locutor

Debido a la variabilidad dentro del conjunto de locutores existentes se hace necesario adaptar este modelo acústico-fonético a las características específicas, de un locutor en cuestión o de un subgrupo de locutores.

Una vez que tenemos construido un modelo acústico-fonético vamos a adaptarlo, ya que este ha sido creado para generalizar un conjunto de locutores de una misma lengua. Para adaptar este modelo general, que llamaremos UBM (Universal Background Model), es necesario técnicas específicas, algunas de estas técnicas de adaptación son MLLR, fMLLR o MAP.

La adaptación MLLR (Maximum Likelihood Linear Regresion) se basa en entrenar una transformación lineal para transformar un conjunto de parámetros con una sola transformación [12], con esta técnica conseguimos reducir los datos necesarios para la adaptación.

Esta adaptación ajusta parámetros como la media y varianza de las gaussianas del modelo (GMM).

Otra adaptación muy utilizada es la adaptación fMLLR (Feature-Space MLLR), que consiste en una transformación afín parecida a la adaptación MLLR [12].

Estas dos técnicas consiguen hacer una adaptación de locutor con pocos datos, por lo que es ideal cuando necesitamos la interacción del usuario, ya que un sistema que pide muchas grabaciones al locutor puede no ser una buena opción. Estas dos técnicas son las más utilizadas en adaptación de locutor.

También existe la adaptación MAP (Maximum a Posteriori) o también conocida como Adaptación Bayesiana, esta adaptación opera parámetro a parámetro sobre los HMM y no es tan eficiente como las vistas anteriormente, además esta técnica necesita mucha cantidad de locuciones para adaptar correctamente el modelo de locutor.[12]

2.2.4 Bases de datos

Durante mucho tiempo no existía una base de datos destinada específicamente a la tarea de reconocimiento de locutor dependiente de texto, se utilizaban bases de datos destinadas a otras tareas como pueden ser TI-DIGITS [17] y TIMIT [18].

Una de las primeras bases de datos destinadas a esta tarea fue la base de datos YOHO [19], esta base de datos es de las más conocidas y de las más utilizadas a la hora de medir el rendimiento de un sistema dependiente de texto pero tiene algunas limitaciones como por ejemplo que solo ha sido grabada con el mismo micrófono y no ha sido creada para simular impostores.

Otras bases de datos más reciente son MIT Mobile Device Speaker Verification Corpus [20], BIOSEC Baseline Corpus [21] y la RSR2015 [4].

Esta última base de datos, sobre la que entraremos en detalle más adelante, es el estado del arte en los sistemas de reconocimiento de locutor dependiente de texto, no solo es la base de datos más amplia y con más variabilidad de las existentes sino que también proporciona un protocolo de evaluación para los sistemas de reconocimiento de locutor dependiente de texto. Actualmente es de las bases de datos más modernas y utilizadas en este tipo de tareas.

La base de datos RSR2015 es la que utilizaremos en este trabajo.

2.3 Situación actual en los sistemas de reconocimiento de voz

El reconocimiento vocal es una tarea complicada, la señal acústica de la voz, que es nuestra fuente de entrada y que es la que procesamos, contiene mucha información, dentro de esta señal tenemos múltiples fuentes de información, entre ellos el lenguaje, la identidad de la persona o el sexo entre otros. Además tiene una alta variabilidad debido a factores externos del hablante como puede ser el dispositivo con el que grabamos o el ruido ambiente, o también a factores propios del hablante como su estado físico o su estado anímico.
Todos estos factores hacen que la tarea de reconocimiento vocal no sea una tarea sencilla aunque en los últimos años se haya avanzado mucho en resolver estos problemas.

Tradicionalmente el estado del arte en el reconocimiento vocal se ha basado en una extracción de características y una fase de modelado. La extracción de características es básicamente extraer una serie de vectores en cada segmento de audio, típicamente son los MFCCs.

Una vez que tenemos los vectores de características, estos los utilizamos en nuestra fase de modelado, durante años el modelo más utilizado ha sido "Total Variability subspace" [22] en el cual usamos unos vectores de tamaño fijo llamados i-vectors que mantiene la información relevante de la señal además de la variabilidad de la misma, estos i-vectors son la entrada a un sistema llamado PLDA (Probabilistic Linear Discriminant Analysis) [23] que es entrenado para extraer únicamente las características relevantes de los i-vectors y poder hacer una decisión.

Estos sistemas basados en la técnica i-vectors-PLDA son los principales en este tipo de aplicaciones de procesado de voz. Aunque en los últimos años estos sistemas se están mezclando con sistemas de Deep Learning mejorando aún más los resultados. Uno de estos sistemas híbridos trata de mejorar la fase de extracción de características, para ello se utilizan un tipo de DNNs (Deep Neural Network) en las que a la entrada utilizamos los MFCCs y cuya salida son unos vectores de tamaño fijo, de esta manera utilizamos estos vectores para caracterizar el audio. Este tipo de DNNs se llaman Bottleneck [27]. Este tipo de capas se combinan con modelos de subespacios vectoriales como los i-vectors-PLDA. Otros tipos de arquitecturas tratan de utilizar DNNs para la fase de modelado, el objetivo es obtener un vector parecido al i-vector que recoja las características discriminantes del locutor, este tipo de DNNs se conocen como embedding [28] o d-vector. [29]

En los sistemas de Deep Learning, utilizando redes neuronales, somos capaces de conseguir diferentes niveles de abstracción en la representación de la señal de entrada en cada capa de nuestra red neuronal. Conceptualmente, con una red neuronal somos capaces de discriminar la información útil de la no deseada sin necesidad de un procesado o extracción de características previa gracias a la gran capacidad de aprendizaje que tienen estos sistemas, aunque en la práctica se hace necesario un pre-procesado previo para reducir la dimensionalidad de la entrada y reducir los tiempos de entrenamiento obteniendo un rendimiento similar.

Uno de los tipos de redes más interesante en el reconocimiento de voz son las RNN (Recurrent Neural Networks) [24] o redes recurrentes, en especial las LSTM (Long Short-Term Memory) [30] con las que somos capaces de modelar señales temporales como la voz. En los últimos años se han hecho mejoras en las redes LSTM, algunas de estas mejoras son por ejemplo las redes LSTM multidimensionales [25] que expanden las LSTM a problemas con múltiples dimensiones permitiendo explorar las dimensiones de tiempo y frecuencia tratándolas como secuencias conjuntas. Una mejora de estas redes multidimensionales son las Grid LSTM [26], son como las anteriores pero añade una conexión entre las capas y no se limita solo a la dimensión espacio-temporal, este tipo de redes son más robustas que las multidimensionales.

2.4 Redes neuronales

En la última década son muchos los logros que se han conseguido utilizando redes neuronales, dejando a un lado las técnicas convencionales, que solían basarse en extracción de características seguido de un posterior algoritmo que permitía clasificar la entrada.

Las redes neuronales no son nada nuevo, tienen ya muchos años, pero ha sido en estos últimos años donde gracias a la potencia de cálculo de las máquinas actuales junto con una cantidad masiva de datos disponibles cuando se ha sabido darle utilidad.

Las redes neuronales son un modelo matemático inspirado en el comportamiento de una red neuronal biológica y en cómo se estructuran.

Las redes neuronales están diseñados para reconocer patrones, partiendo de una entrada con los datos, estos se procesan en la red y van pasando de capa en capa, cada capa se encarga de abstraer un nivel más los datos y por último se llega a la salida, donde dependiendo del problema tiene una salida u otra, se pueden utilizar para muchos problemas, ya sea de clasificación o regresión.

El uso de una red neuronal es el siguiente, explicado de una manera sencilla:

1- Establecer una arquitectura de la red, esto implica:
    a. Unidades de entrada: Depende de la dimensión de los datos del problema.
    b. Unidades de salida: Depende de la tarea del problema, como norma general tendremos tantas unidades de salida como clases posibles en el problema.
    c. Numero de capas: El número de capas depende del problema sobre el que estamos trabajando, normalmente cuanto más complejo es el problema mayor número de capas tenemos que poner.
    d. Tipos de capas: Dependiendo del problema, algunos tipos de capas son más recomendables que otros tipos, también depende de los datos del problema.
    e. Conexiones entre las capas: Las conexiones entre las capas puede ser un parámetro modificable, se pueden conectar todas con todas, conexiones recurrentes o saltos de capas.
2- Una vez que tenemos la red definida, las conexiones entre las distintas neuronas que tenemos en la red se inicializan con un peso aleatorio.
3- Los datos se introducen en la red por las capas de entrada y se realiza el paso de Forward.
4- Una vez que los datos llegan a la salida se calcula la función de coste, esto es la diferencia que hay entre lo que debería ser la salida y el resultado del paso de Forward.
5- Con el coste calculado se procede a calcular el paso de Backward.
6- Por último se actualizan los pesos de las conexiones entre neuronas en función del resultado del paso de Backward.

Los pasos del 3 al 5 se ejecutan en bucle, para ello se introduce en la red lo que se llama batches, esto es el número de muestras que se introducen en la red en cada paso de Forward. La duración del entrenamiento de la red se mide en epochs, un epoch es el tiempo que tarda la red en procesar todos los datos de entrenamiento.

Las redes más simples son las que se llaman Feed-Forward, estas redes tienen una única dirección de salida, a diferencia de las recurrentes (RNN), que tienen una salida que se

realimenta a la entrada. Un ejemplo de redes Feed-Forward son las Fully-Connected, son redes en el que todas las neuronas están conectadas a la siguiente capa:

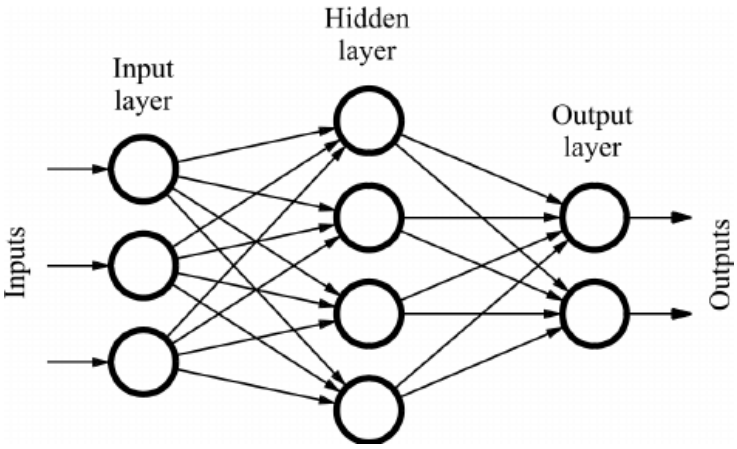

Figura 3: Ejemplo de red neuronal Fully-Connected [14]

2.4.1 Convolutional Neural Network

Las redes neuronales convolucionales (CNN) son un tipo de redes neuronales que están especialmente diseñadas para tratar con imágenes a su entrada. Si utilizásemos redes Fully-Connected para una imagen tendríamos muchos parámetros, por lo que no son eficientes para este tipo de problemas.

Típicamente una red convolucional tiene la siguiente estructura:

1- En la entrada tendremos una o varias capas convolucionales.
2- Seguido tendremos una o varias capas de Pooling, explicaremos más adelante que es.
3- Por ultimo tendremos una o varias capas Feed-Forward que nos dará la salida.

Las capas convolucionales consisten en una ventana deslizante, llamada Kernel, que se encarga de realizar operaciones lineales con los datos que enventana ese Kernel, por lo tanto cada pixel de salida se corresponde a una operación lineal de los pixeles de entrada y el Kernel, el Kernel se desplaza por toda la imagen, pudiendo configurar este desplazamiento del Kernel.

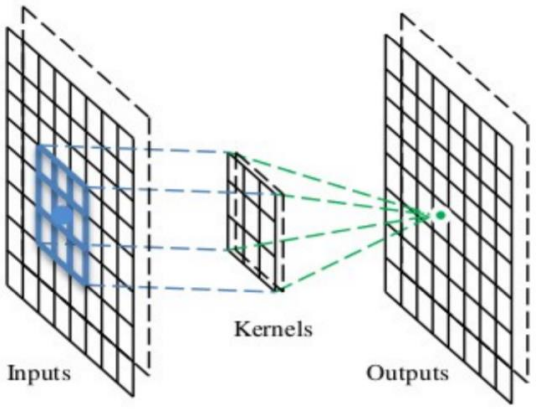

Figura 4: Ejemplo red convolucional [15]

Las capas de pooling se encargan de reducir la dimensionalidad de los datos que se van generando, esto ayuda a reducir la complejidad computacional, esta reducción se puede calcular en función del máximo, como muestra la siguiente imagen, pero también se pueden seguir criterios como la media del kernel o el mínimo, entre otros. El kernel recorre la imagen deslizándose y calculando el máximo de cada zona que cae en el kernel.

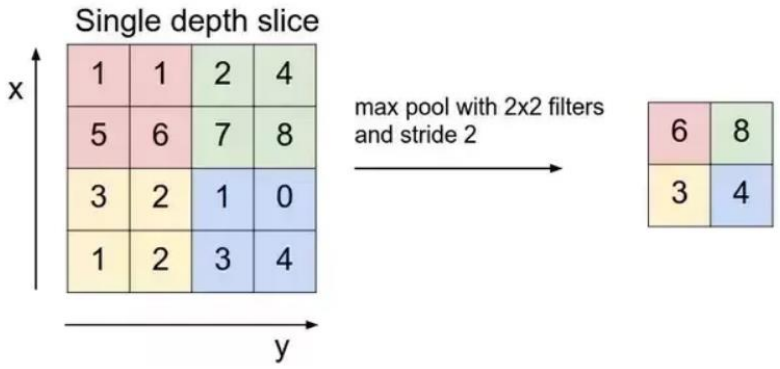

Figura 5: Ejemplo de pooling con criterio de máximo [15]

Por último se conecta una capa de tipo Fully-Connected, previo a esta capa hay que reducir las imágenes para adaptarlas a la entrada de la capa Fully-Connected, este redimensionado hace que perdamos la información espacial, pasamos de una matriz de 2 dimensiones a un vector de 1 dimensión.

La siguiente imagen refleja la estructura de una red convolucional completa:

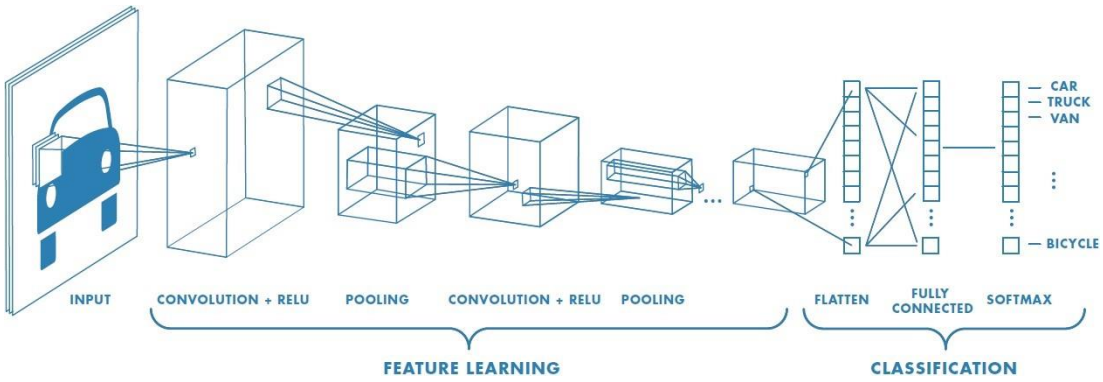

Figura 6: Esquema red convolucional completa [16]

3 Diseño

3.1 Herramientas Utilizadas

En las siguientes subsecciones vamos a explicar las distintas herramientas que hemos utilizado en este trabajo y para qué las hemos usado.

3.1.1 Kaldi

Kaldi es un toolkit diseñado para tareas de reconocimiento de voz, está diseñado para ser más flexible y moderno que el toolkit HTK, otra herramienta muy utilizada en reconocimiento de voz.

Kaldi está escrito en C++ y distribuido bajo licencia apache v2.0. Al estar diseñado en código C++ tiene la ventaja de que es fácilmente modificable, además, incluye integración y soporte con diversas librerías como OpenFST, BLAS y LAPACK que nos permiten cálculos más optimizados. [7]

En nuestro trabajo vamos a usar Kaldi para extraer los MFCCs (Mel Frequency Cepstral Coefficients), estos MFCCs se extraerán de los audios que nos proporciona la base de datos RSR2015, sobre la que entraremos en detalle más adelante.

3.1.2 PHNrec

El toolbox PHNrec (Phoneme Recognizer) es una herramienta de reconocimiento fonético desarrollada por la universidad tecnológica de Brno.

PHNrec nos permite extraer las probabilidades a posteriori (posteriors en adelante) de cada fonema. Es una herramienta pre-entrenada. Su salida se corresponde con los posteriors de los 40 posibles fonemas (para el caso del inglés) en cada frame del audio. La salida de esta herramienta está en formato legible para HTK. [9]

Esta herramienta la usaremos para extraer todos los posteriors por frame de cada audio de la base de datos RSR2015. Por lo tanto tenemos a la entrada un audio y a la salida los posterior de cada fonema en cada frame.

3.1.3 HTK

La herramienta HTK (Hidden Markov Model Toolkit) está diseñada para trabajar con modelos de Markov. HTK principalmente es usada para tareas de reconocimiento de voz aunque también se puede usar en otros ámbitos como por ejemplo reconocimiento de caracteres, entre otros. [8]

La librería HTK está diseñada en C y es fácilmente modificable para adaptar su uso a la tarea que queramos, como hemos dicho, esta herramienta es parecida a Kaldi aunque tiene sus diferencias.

En nuestro trabajo, usaremos esta herramienta para ser capaces de interpretar las salidas de los ficheros de PHNrec, lo usaremos por tanto para convertirlos a ficheros legibles, nos dará una salida a los ficheros de PHNrec que contienen las probabilidades por fonema.

3.1.4 Keras

Keras es una API de alto nivel para trabajar con redes neuronales, está escrita en Python y en nuestro caso trabaja sobre Tensorflow, aunque también es capaz de trabajar sobre Theano o CNTK. Keras nos proporciona la capacidad de crear redes neuronales y trabajar con ellas de una manera muy sencilla, abstrayéndose de los detalles de bajo nivel, además es capaz de trabajar tanto sobre CPU como GPU. [10]

En nuestro trabajo Keras será la principal librería sobre la que trabajaremos para entrenar y hacer pruebas, estará montada sobre Anaconda, una plataforma que recopila diversos programas y librerías fácilmente instalables y modificables dentro de los entornos que nos proporciona la herramienta.

3.2 Base de Datos: RSR 2015

La base de datos utilizada en este trabajo y que será la base de los experimentos es la RSR2015.

La base de datos RSR2015 está diseñada para evaluar sistemas de verificación de locutor dependiente de texto bajo diferentes duraciones y contenidos léxicos, esta base de datos ha sido creada por el HLT (Human Language Technology), departamento del I^2R de Singapur.

La base de datos está grabada por hablantes ingleses con diferentes acentos que se pueden encontrar en Singapur.

Algunas de las características de la base de datos RSR2015 son las siguientes:

- Tiene un total de 300 locutores, de esos 300 locutores 143 son femeninos y 157 masculinos, con edades comprendidas entre 17 y 42 años.
- Tiene un total de 151h que han sido grabadas utilizando dispositivos móviles.
- Tiene 3 partes diferenciadas con distinto contenido léxico y distintas duraciones para probar distintos escenarios de los sistemas.
- Nos proporciona un protocolo de evaluación de la base de datos para que la comparación de los resultados sea lo más justa posible.

La base de datos tiene la siguiente organización:

	Female	*Male*	*Total*
Chinese	118	119	79%
Malay	14	28	14%
Others	11	10	7%

Figura 7: Distribución locutores RSR2015 [2]

Como se puede ver la mayoría de los hablantes son Chinos, además se puede ver que los datos están equilibrados entre hablantes masculinos y femeninos.

La base de datos tiene un total de 151h de grabación que están repartidas entre los 3 escenarios distintos que existen. Cada escenario tiene un total de 9 sesiones de grabación, de esas 9 sesiones de grabación 3 están destintadas a la fase de train y 6 a la fase de test.

La parte I de la base de datos está compuesta por frases cortas que sirven a modo de contraseña para autenticarse, en cada sesión el locutor pronuncia 30 frases extraidas de la base de datos TIMIT, la duración media de cada frase es 3,20 segundos. Las frases son del tipo:

A good attitude is unbeatable

La parte II de la base de datos son comandos para un sistema de control doméstico, la duración media es de 1,99 segundos, un ejemplo es el siguiente:

Play music

La parte III de la base de datos son frases de 10 y 5 dígitos, comprendidos entre 0 y 9, en cada sesión se pronuncian 3 frases de 10 dígitos o 10 frases de 5 dígitos. Las secuencias de dígitos son distintas entre sesiones con lo cual nunca se repiten, pero si que son iguales entre los diferentes locutores que se pueden encontrar en la base de datos.

6-0-8-4-7-5-9-1-3-2
9-7-3-1-4

En nuestro trabajo vamos a utilizar la parte III de la base de datos, en concreto con las secuencias de 10 dígitos, por lo tanto para nuestros experimentos tenemos un total de 150 locutores, 9 sesiones de grabación por locutor, 3 de train y 6 de test, y en cada sesión tenemos 3 frases de 10 dígitos que nunca se repiten entre sesiones.

3.3 Esquema de trabajo

Una vez explicadas las herramientas que utilizaremos en este trabajo, vamos a explicar cómo va a ser nuestro proceso para crear nuestro sistema de reconocimiento de locutor dependiente de texto. Nuestro objetivo es determinar si un locutor es quién dice ser a partir de una frase que sabemos, esto es un sistema de reconocimiento de locutor dependiente de texto, en este caso partimos de frases que conocemos previamente que son las frases de 10 dígitos, nuestro procedimiento será el siguiente:

- Por un lado tenemos que extraer las características de los audios de la base de datos, tendremos 2 tipos de características, los MFCCs y los posteriorgramas, estas características las extraemos con los programas de Kaldi y PHNrec.
- Estas características las cruzaremos con los distintos locutores evaluando punto a punto la correlación existente, creando así matrices rectangulares.
- Estas matrices rectangulares serán la entrada de una red neuronal que analizará los patrones de correlaciones con el objetivo de que encuentre aquellos puntos clave para reconocer a un locutor como verdadero o como impostor.

En la siguiente sección entramos al detalle de cómo hemos ido siguiendo este procedimiento para conseguir los resultados.

4 Desarrollo

En esta sección nos vamos a centrar en el desarrollo de nuestro trabajo y los experimentos, entrando más en detalle sobre los pasos que hemos seguido. Los primeros pasos de nuestro trabajo han consistido en la extracción de las características a partir de los audios, estas características son la base de nuestro trabajo, a partir de aquí usamos estas características para analizarlas. Hemos extraído dos tipos de características, los MFCCs y los posteriorgramas.

Antes de empezar el desarrollo de nuestro trabajo vamos a detallar cual es protocolo de pruebas que la base de datos propone.

4.1 Protocolo de pruebas

El protocolo de pruebas que la base de datos RSR2015 propone es el siguiente [7]:

- Usaremos la parte III de la base de datos RSR2015, en concreto las frases con 10 dígitos comprendidos entre el 0-9.
- Tenemos 9 sesiones de grabación por locutor.
- Tenemos dos fases, una fase de train y otra de test, la fase de entrenamiento del sistema está formado por las sesiones 1, 4 y 7, y para la fase de test, las sesiones de grabación 2, 3, 4 ,5 ,8 y 9.
- Cada sesión de grabación tiene 3 frases de 10 dígitos que nunca se repiten entre sesiones, pero sí que son iguales entre los distintos locutores.

La base de datos también especifica los locutores que se usan para entrenamiento ya que no se usan todos los locutores. Para cada locutor se usan sus 3 sesiones con sus 3 frases por sesión. Cada sesión de entrenamiento correspondería a un modelo de locutor, por lo tanto tenemos 3 modelos de locutor por locutor.

Para la fase de test es igual, tenemos un fichero propio de la base de datos donde se indican todos los trials o comparaciones que se tienen que hacer para probar el sistema. En concreto este archivo hace todas las comparaciones sea correcta o no la frase pronunciada. Es decir se hace una comparación modelo vs frase, cuyo modelo ha sido entrenado previamente.

En nuestros experimentos va a ser algo distinto ya que no tenemos modelos de locutor a entrenar. Queremos desarrollar un sistema que sea independiente del entrenamiento de un modelo específico de un locutor, es decir que generalice la detección de un locutor genuino de un impostor. El detalle de nuestra propuesta para este sistema lo detallaremos en la sección 4.4.2.

4.2 Extracción de MFCCs

Uno de los primeros pasos de nuestro trabajo era la extracción de las características de los locutores a partir de los audios de la base de datos RSR2015. Los MFCCs (Mel frequency Cepstral Coefficients) son unos coeficientes para la representación del habla basados en la percepción auditiva humana, consisten en unos bancos de filtros, en concreto filtros de la escala Mel, que se aplican sobre la señal en frecuencia.

Para extraer estos coeficientes MFCCs vamos a utilizar la herramienta de Kaldi, como hemos explicado anteriormente, Kaldi es un toolkit diseñado para tareas de reconocimiento de voz. Kaldi nos proporciona herramientas automatizadas para la extracción de características de audios, en concreto vamos a utilizar el script de *make_mfcc.sh*, esta herramienta nos extrae los MFCC del audio por frames, es decir, en tramas de duración de 20ms donde se analiza el audio. Dado que la salida de estos ficheros están en formato de Kaldi hay que convertirlos a ficheros de texto legibles, para ello utilizamos el siguiente comando:

Copy-feats ark: fichero_entrada.ark ark,t: fichero_salida.txt

El fichero de salida tendrá dimensión de X filas, donde X será el número de frames, que depende de la duración total del audio, y 13 columnas, que representan los coeficientes MFCC del frame. Usaremos la configuración por defecto del script sin modificar ningún parámetro, aunque se puede ajustar según las necesidades.

Esta extracción la haremos tanto para los ficheros de train como para los ficheros de test.

4.3 Extracción de posteriorgramas

Otra de las bases de nuestros experimentos son los posteriorgramas, los posteriorgramas son representaciones de los posteriors de cada posible fonema de un audio en cada instante de tiempo.

Para extraer los posteriors vamos a utilizar la herramienta de PHNrec, esta herramienta está pre-entrenada sobre la base de datos TIMIT para extraer los posteriors.
La entrada de audio para la herramienta de PHNrec tiene que ser de 16KHz en formato WAV, para hacer esta conversión desde los ficheros que nos proporciona la base de datos RSR2015 es necesario ejecutar el siguiente comando:

Sox fichero_in.sph –r 16000 fichero_out.wav

Una vez que tenemos todos nuestros ficheros con frecuencia de muestreo a 16Khz y en formato WAV procedemos a calcular los ficheros con las probabilidades a posteriori de los audios, para ello ejecutamos el siguiente comando:

phnrec -c directorio_configuracion -t post -i fichero.wav -o fichero.post

Donde nuestro *directorio_configuracion* es *PHN_EN_TIMIT_LCRC_N500* para utilizar el reconocedor fonético en inglés.

La salida de este fichero está en formato HTK, no podemos interpretarlo tal cual sale de la herramienta de PHNrec, por lo tanto, es necesario utilizar HTK.

HTK está compuesto por varias herramientas dentro del toolkit, para convertir el fichero .post resultante de la salida de PHNrec vamos a utilizar HList. Con el comando siguiente convertimos la salida en formato legible.

HList -r fichero.post > fichero.post.txt

El archivo de salida es un archivo txt donde tenemos X filas correspondientes a los X frames del audio y 120 columnas, correspondientes a las probabilidades por estado.

Como cada fonema tiene 3 estados con sus probabilidades asociadas, vamos a sumar estas probabilidades por estado para calcular la probabilidad total del fonema, de esta manera conseguimos 40 columnas correspondientes a los 40 fonemas posibles.

Como resultado, obtenemos un fichero con las probabilidades por fonema en cada instante de tiempo, esto son los posteriorgramas. Hay que calcular los posteriorgramas de todos los ficheros de la base de datos.

4.4 Construcción del sistema

Una vez que hemos extraído las características de los audios, ya sea MFCCs o posteriorgramas, vamos a proceder a la fase de train, para ello vamos a explicar en qué va a consistir nuestros experimentos y nuestro sistema.

Nuestro sistema es un reconocedor de locutores dependiente de texto, es decir pretende discernir entre un locutor genuino y un impostor conociendo la frase previamente.

En trabajos anteriores utilizábamos los modelos de locutor, estos modelos de locutor eran modelos que a partir de un modelo general de hablantes en lengua inglesa se adaptaba mediante una técnica llamada fMLLR con audios particulares del locutor en cuestión.

Nuestro objetivo en este trabajo es crear un modelo con redes neuronales que sea capaz de generalizar el reconocimiento de locutor a partir de las características del locutor, ya sean las características fonéticas mediante los posteriorgramas o las características espectrales mediante los MFCCs.

Para construir este sistema hemos extraído previamente las características de los audios, con estas características vamos a hacer una comparación, por un lado audio original y por otro el audio trial. Frame a frame vamos a ir calculado la correlación de Pearson:

$$r_{xy} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\,\sqrt{n\sum y_i^2 - (\sum y_i)^2}}$$

Figura 8: Fórmula correlación de Pearson

Esta correlación está acotada entre [-1,1], siendo 1 altamente correlado y siendo 0 incorrelado.

El resultado son matrices rectangulares, donde nuestra hipótesis es que para un locutor genuino tendremos zonas con alta correlación mientras que para un impostor, aunque diga la misma frase y tenga una correlación alta no será igual que para el locutor genuino. Aquí es donde es de vital importancia que nuestra red neuronal sea capaz de analizar y extraer las zonas de alta correlación y los patrones para dar alta puntuación a los locutores genuinos y una baja puntuación a los impostores.

4.4.1 Train

Para la fase de train nuestro primer paso es establecer el protocolo de entrenamiento que nos da la base de datos.

La base de datos nos da una serie de locutores que sirven de entrenamiento, cada locutor tiene 3 sesiones con 3 frases cada sesión para entrenar, como en nuestro sistema no trabajamos con modelos de locutor vamos a utilizar todos los audios del mismo locutor para entrenamiento, esto nos da 9 audios por locutor para combinar entre ellos.

Tendremos dos tipos de reconocimiento, es una decisión binaria, o reconocimiento positivo o reconocimiento negativo, para el caso de reconocimiento positivo enfrentaremos cada uno de los 9 audios de un locutor con otro audio de esos 9 del mismo locutor, esto nos da un total de 45 comparaciones únicas positivas. Por cada comparación única positiva vamos a generar otra comparación única negativa contra un audio de otro locutor.

Una vez que tenemos las comparaciones definidas, con las características de los MFCCs o de los posteriorgramas vamos a generar la comparación, para ello nuestros ficheros de entrada son archivos de X filas (frames) y 13 columnas en el caso de los MFCCs y 40 columnas en el caso de los posteriorgramas. Nuestra comparación será tal y como se muestra en la siguiente ilustración. En cada punto del grid calcularemos la correlación Pearson.

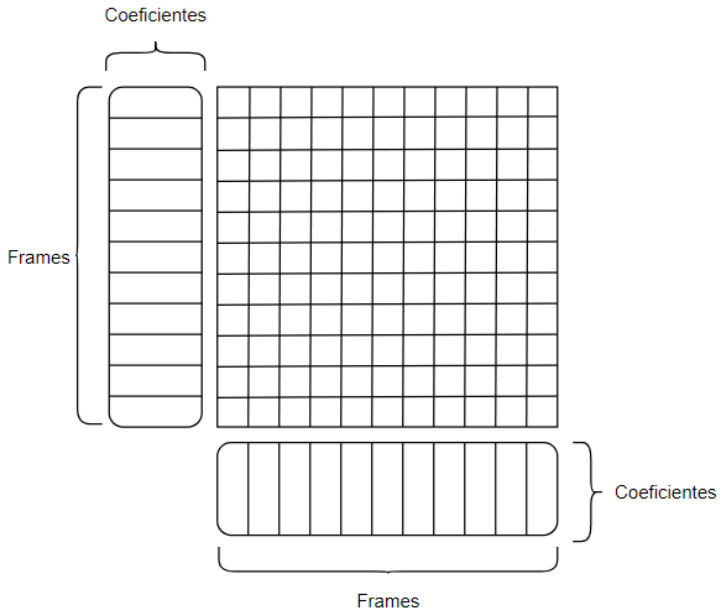

Figura 9: Esquema de comparación frase original vs frase target

Necesitamos que nuestro ficheros de entrada tengan el mismo número de frames para que la salida sean matrices cuadradas y tengan una entrada con un tamaño fijo para la red neuronal que irá después, por ello es necesario ver cual será el tamaño que fijaremos para nuestro audios.

Analizando los audios de la base de datos para los ficheros de train vemos la siguiente distribución de longitudes:

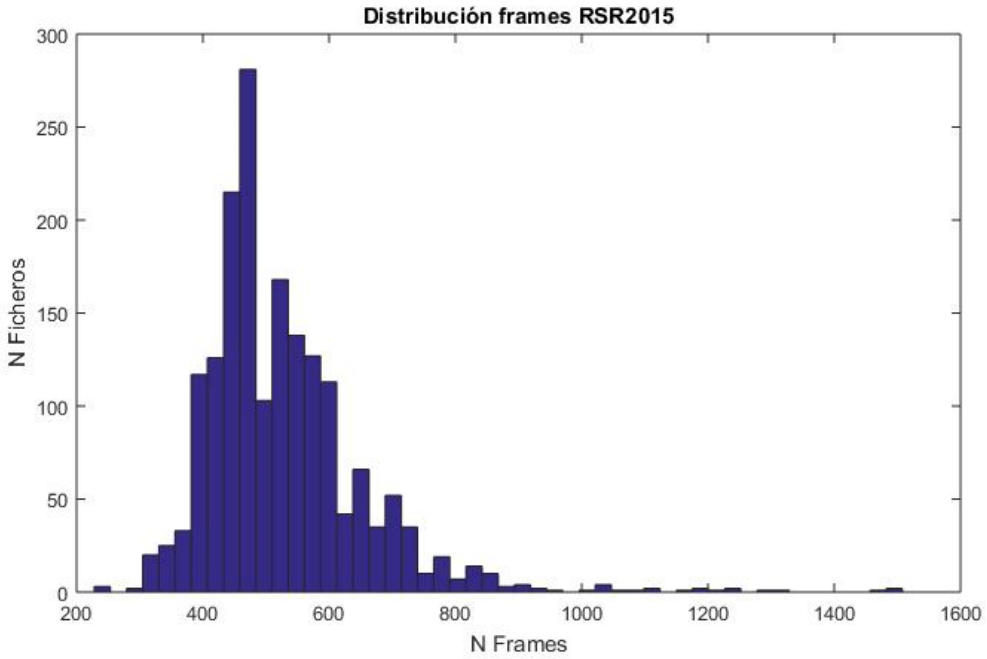

Figura 10: Distribución de frames RSR2015

Por requisitos computacionales y dado que estamos por encima de la media vamos a elegir nuestro tamaño fijo de frames a 600, por lo tanto el resultado de comparar nuestros audios serán matrices de tamaño 600x600. El resto de frames a partir del frame 600 los ignoramos o los ponemos a 0 en el caso de que el audio tenga menos frames.

Una vez que tenemos todas nuestras matrices de comparación creadas el siguiente paso es entrenar la red neuronal, para ello vamos a hacer uso de Keras. Para ello seleccionamos un conjunto dentro del conjunto de entrenamiento de manera que equilibremos los datos de entrenamiento, nuestro siguiente paso es crear la red neuronal.

Nuestra red neuronal tendrá en la primera capa una CNN de tipo Conv2D con tamaño de entrada de archivos 600x600x1 y kernel 3x3, estas capas están seguidas de capas de Maxpooling, Dropout y otras Conv2D, por ultimo tenemos una capa flatten y capas Dense hasta la salida que es de tipo sigmoid. El detalle de la red usada queda reflejado en el siguient código de Keras:

```
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(600, 600, 1)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(48, kernel_size=(3, 3), activation='relu'))
model.add(Dropout(0.25))
model.add(MaxPooling2D(pool_size=(3, 3)))
model.add(Conv2D(80, kernel_size=(3, 3), activation='relu'))
model.add(Dropout(0.25))
model.add(MaxPooling2D(pool_size=(3, 3)))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(256, activation='relu'))
```

model.add(Dropout(0.25))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(256, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

En el entrenamiento de la red se usará como función de coste, binary crossentropy (Entropía cruzada binaria), y como optimizador, SGD (Stochastic gradient descent, descenso de gradiente estocástico) con parámetros:

- Learning rate: 0.01
- Decay: 1e-6
- Momentum: 0.9
- Nesterov: True

Usamos este optimizador dado que con el optimizador Adam no conseguimos grandes resultados.

Por último asignamos un Checkpoint o punto de control a nuestro entrenamiento, esto sirve para que Keras guarde el modelo dado un indicador que nosotros elegimos, en nuestro caso vamos a asignar el indicador de Validation Accuracy, por lo tanto se guardará el modelo en cada epoch siempre y cuando mejore el mejor Validation Accuracy de todos los epoch anteriores, de esta manera nos aseguramos que no sobreentrenamos el modelo y nos quedamos con el que mejor rendimiento nos aporte.

4.4.2 Test

Una vez que nuestra fase de train está completada, calculamos el rendimiento de nuestro sistema con los ficheros de test.

Para ello usamos el protocolo de pruebas que nos proporciona la base de datos RSR2015, en este protocolo de pruebas se enfrenta un modelo contra una frase, es decir, modelo vs frase, si recordamos, en el apartado anterior indicábamos que cada modelo estaba entrenado por 3 frases, pero en nuestro caso no tenemos un modelo por locutor si no que queremos generalizar este sistema sin crear un modelo por locutor.

Como solución a esto calcularemos dos escenarios, uno en el que para un enfrentamiento modelo vs frase haremos 3 enfrentamientos por modelo (cada modelo tiene 3 frases) como si fueran independientes y otro caso en el que calcularemos la media de las puntuaciones de los 3 enfrentamientos de un modelo contra una frase:

Tabla 1: Esquema protocolo de evaluación

Base de Datos	1 Modelo (3 frases) vs 1 Frase
Caso 1	3 puntuaciones: 3 frases vs 1 frase
Caso 2	1 puntuación: Media 3 frases vs 1 frase

Es decir, el caso 2 es calcular la media de las puntuaciones del caso 1.

Por ultimo una vez que tenemos todos nuestros scores de los locutores genuinos y de los locutores impostores, usamos la herramienta DETware del NIST para calcular nuestras curvas DET del sistema. En estas curvas DET nos quedaremos con el punto de la gráfica donde tenemos el EER (Equal Error Rate), es decir, el punto donde tenemos igual tasa de error de falsas aceptaciones y falsos rechazos.

5 Integración, pruebas y resultados

5.1 Resultados previos

Antes de mostrar nuestros resultados, vamos a ver primero los resultados que obtuvimos en el TFG *Mejoras en un sistema de reconocimiento de locutor dependiente de texto,* en este trabajo utilizamos la base de datos RSR2015 y nuestro sistema estaba basado en modelos HMMs de locutor, es decir, se adaptaban los modelos UBM a cada locutor.

El resultado obtenido en el trabajo fue el siguiente:

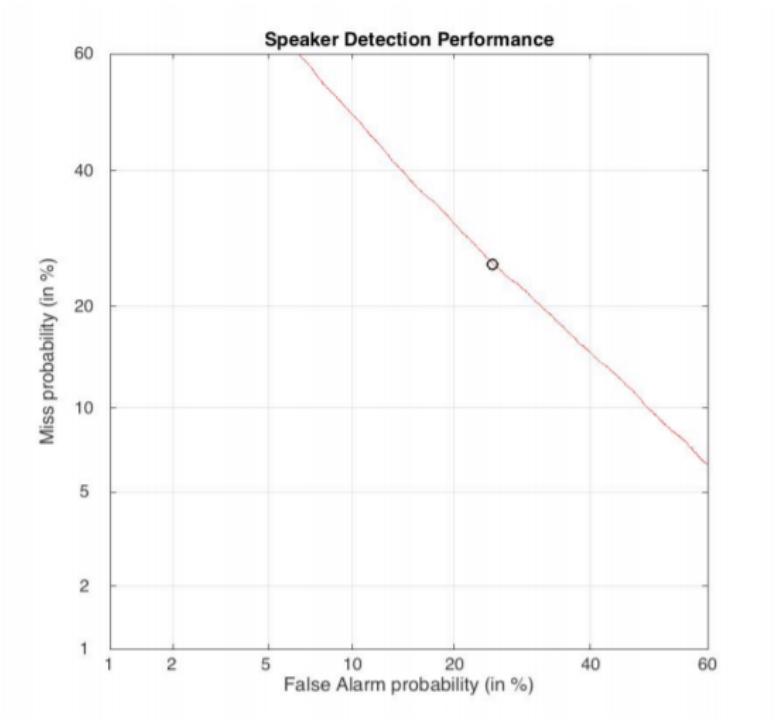

Figura 11: Resultado obtenido en el TFG con la base de datos RSR2015 [1]

Como se puede ver el EER está sobre el 28%, en una prueba posterior a este resultado le aplicamos una Z-Normalización de Scores y se llegó a mejorar al 19% en el mejor de los casos. Un resultado que aunque mejoró los resultados previos aún tenía margen de mejora.

5.2 Resultados actuales

Nuestro sistema, como hemos mencionado en capítulos anteriores, no necesita de un entrenamiento previo específico de ningún modelo de locutor, simplemente necesitamos un audio original del locutor a comparar y una frase test.

En la siguiente gráfica podemos ver el resultado obtenido para el conjunto de locutores femenino de la base de datos utilizando las características de los MFCCs para la comparación:

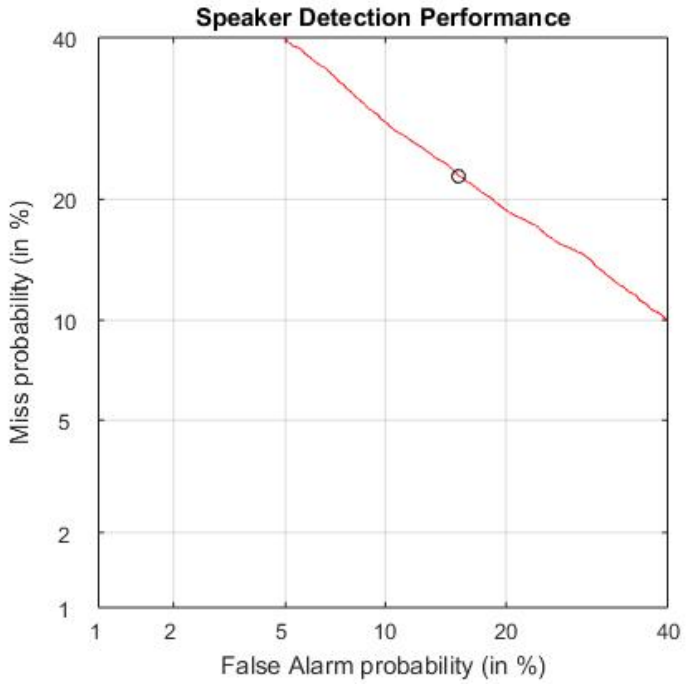

Figura 12: MFCCs DET conjunto femenino RSR2015 en el caso 1

Como se puede ver el EER está sobre el 19%, esta prueba está basada en el Caso 1 que hablábamos en la sección 4.4.2, es decir, agrupa todos los scores de todas las frases.
Con este sistema ya estamos igualando el rendimiento que obtuvimos en el TFG con los modelos de locutor.

En la siguiente imagen mostramos los resultados del Caso 2, este caso consistía en hacer una comparación más justa, si para los modelos de locutor teníamos 3 frases para entrenar al modelo, en nuestro sistema ahora haremos una media de los 3 scores obtenidos para esas tres frases del modelo.

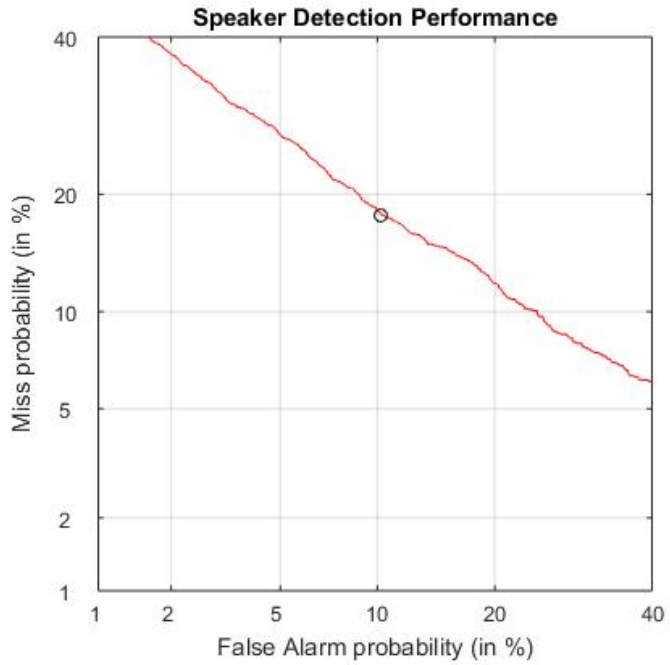

Figura 13: MFCCs DET conjunto femenino RSR2015 en el Caso 2

Como se puede ver el EER se reduce considerablemente hasta el 15%, este método mejora notablemente los resultados obtenidos en trabajos anteriores, además mejora el Caso 1, esto es debido a que puede haber alguna frase que se le dé mayor puntuación debido a su similitud pero es difícil que se dé la misma situación con las otras dos frases.

En las siguientes gráficas mostramos los resultados de nuestro sistema con las características MFCCs para el conjunto de locutores masculinos.

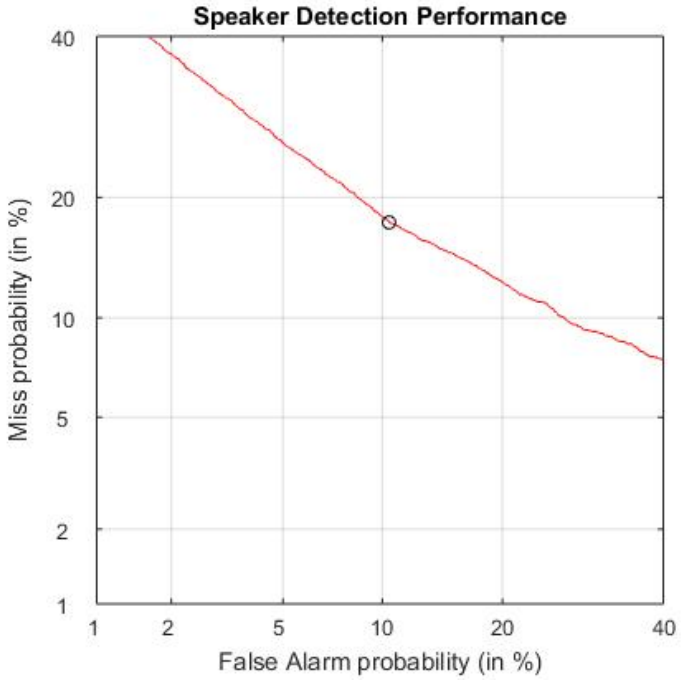

Figura 14: MFCCs DET conjunto masculino RSR2015 caso 1

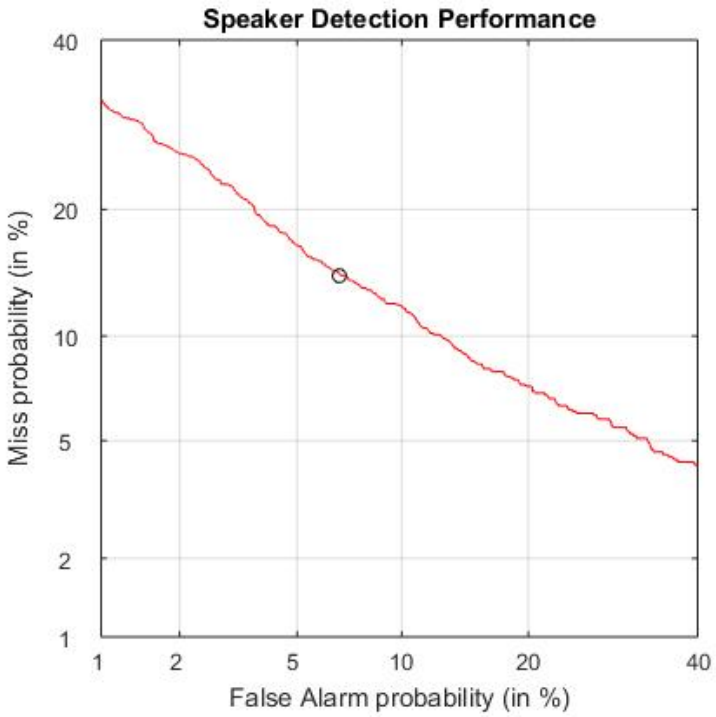

Figura 15: MFCCs DET conjunto masculino RSR2015 caso 2

Para el conjunto de locutores masculino el rendimiento del sistema mejora más aun, para el caso 1 obtenemos un 15% de EER mientras que para el caso 2 obtenemos un 11% de EER.

Hemos visto los resultados con MFCCs, ahora veremos los resultados obtenidos con las características de los posteriors. La siguiente figura muestra los resultados para el caso 1 en el conjunto de locutores femeninos.

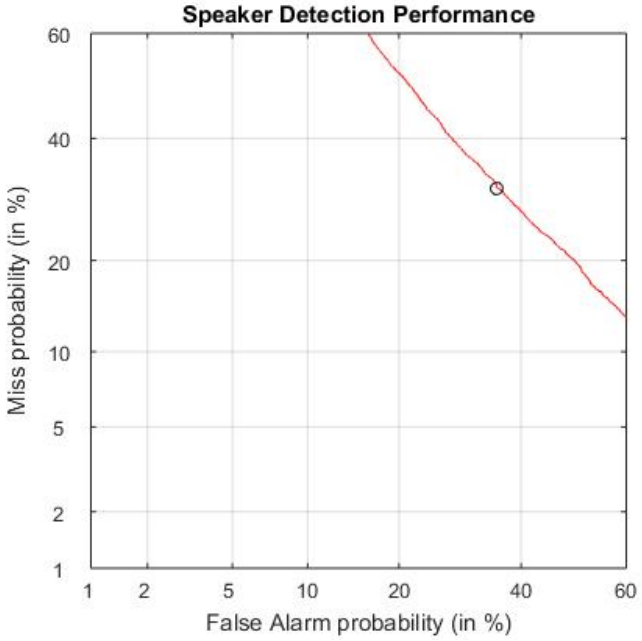

Figura 16: Posteriorgramas DET conjunto femenino RSR2015 caso 1

Como se puede ver, los resultados obtenidos con los posteriorgramas no son nada buenos, el EER aumenta hasta un 38%, esto nos hace indicar que las características de las probabilidades a posteriori de los fonemas no es una buena característica para discriminar a un locutor de otro, al menos en nuestro sistema. A continuación mostramos los resultados del caso 2 para los locutores femeninos.

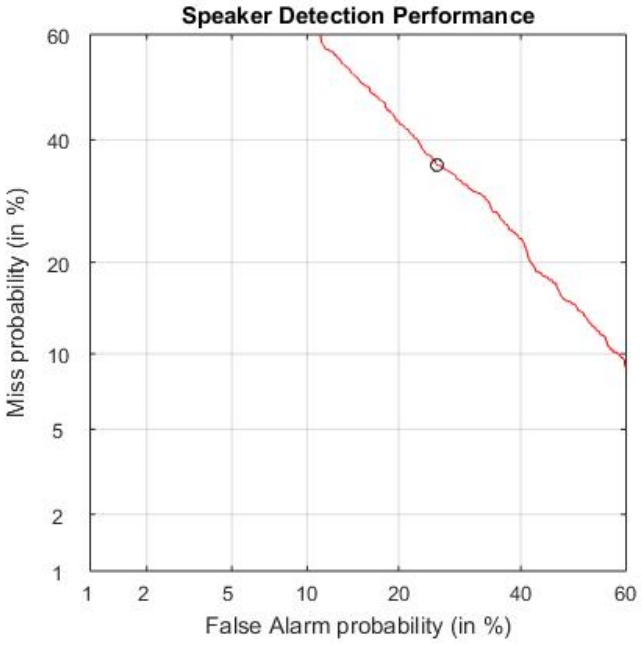

Figura 17: Posteriorgramas DET conjunto femenino RSR2015 caso 2

Aunque tenemos cierta mejora en el caso 2, sigue siendo insuficiente, el EER se sitúa en torno al 30%.

En la siguiente gráfica mostramos los resultados para el conjunto masculino de la base de datos y el caso 1.

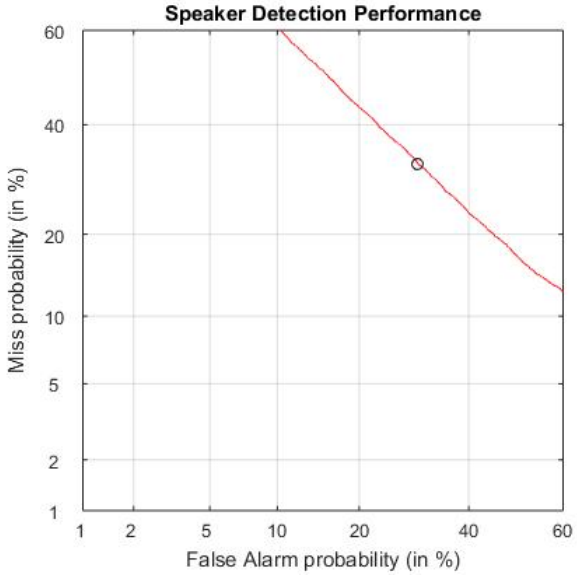

Figura 18: Posteriorgramas DET conjunto masculino RSR2015 caso 1

El resultado mejora ligeramente respecto al conjunto de locutores femenino en el caso 1, el EER se sitúa en el 34%.

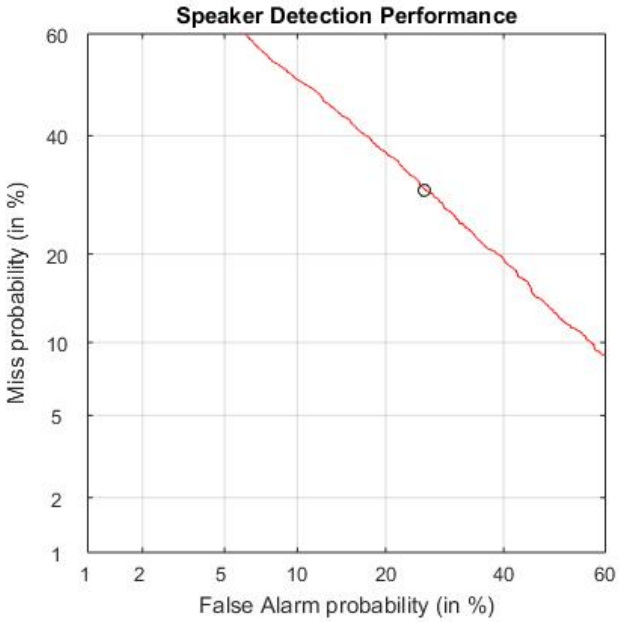

Figura 19: Posteriorgramas DET conjunto masculino RSR2015 caso 2

Para el conjunto de locutores masculino en el caso 2 podemos ver que de igual manera se reduce respecto al caso 1, bajamos el EER hasta el 27%. De igual manera estos resultados están muy lejos del 11% obtenido con los MFCCs. Como hemos comentado anteriormente, las características de los posteriorgramas empeoran bastante respecto a los MFCCs en nuestro sistema.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

En los sistemas tradicionales de reconocimiento de locutor dependiente de texto como el construido en el TFG *Mejoras en un sistema de reconocimiento de locutor dependiente de texto* se necesita pasar por una fase de adaptación del locutor, es decir, como hemos visto en las secciones del estado del arte, necesitamos un modelo general de locutores que generalice lo mejor posible el conjunto de hablantes de una lengua en particular, esto es lo que llamamos UBM, pero por otro lado necesitamos adaptar este UBM a las características particulares de un locutor. Posteriormente mediante las probabilidades de pertenecer a un modelo u otro hacemos nuestra decisión.

Para esto necesitamos crear dos modelos, uno de ellos nos serviría para cualquier locutor mientras que el modelo específico necesitamos crearlo para cada locutor.

En este trabajo de fin de master hemos construido un sistema de reconocimiento de locutor dependiente de texto completamente innovador que intenta generalizar el reconocimiento sin una adaptación especifica del locutor, obteniendo unos resultados más que aceptables.

Con nuestro sistema lo único que necesitamos son dos frases, la frase del locutor genuino que será nuestro requisito previo y una frase que será la que enfrentaremos en el sistema, posteriormente nuestro sistema se encargará de extraer las características espectrales del audio y una red neuronal entrenada se encargará de darle una puntuación en base a la similitud que tiene con el original.

En nuestro trabajo hemos experimentado con dos tipos de características, los MFCCs y los posteriorgramas, como hemos visto, para los MFCCs hemos obtenido unos muy buenos resultados mejorando nuestro sistema de partida basado en HMMs.
Por otro lado los experimentos con las características de los posteriorgramas no hemos conseguido buenos resultados, esto es debido a que no nos aporta suficiente información del locutor.

En los resultados también hemos comprobado que para el caso 1 obtenemos peores resultados que para el caso 2, el caso 2 sería el que más se ajusta al protocolo de evaluación de la base de datos empleada, la RSR2015. Esto es debido a que hacemos media de los tres reconocimientos de las frases y eliminamos el efecto de un posible reconocimiento fallido. De media mejoramos un 5% en el caso 2 respecto del caso 1.

Por otro lado en nuestro sistema podemos ver que el reconocimiento sobre el conjunto de locutores masculino es ligeramente mejor que para el conjunto de locutores femenino, esto puede ser a que nuestro sistema está desbalanceado en el entrenamiento o a que somos más capaces de extraer información relevante del locutor en los locutores masculinos que en los femeninos.

6.2 Trabajo futuro

De este trabajo de fin de master salimos con varias líneas de investigación.

Por un lado, partiendo de nuestro sistema construido para este trabajo, podemos experimentar con distintas características de los audios, hemos utilizado MFCCs y posteriorgramas, pero también podemos combinarlos para tratar de quedarnos con las mejores características que cada uno nos aporta, podemos utilizar MFCCs con más coeficientes de los utilizados en este trabajo, recordemos que hemos utilizado 13 coeficientes pero podemos utilizar más coeficientes para tratar de conseguir mejores resultados.

Utilizando nuestro sistema como base, también podemos experimentar con otro tipo de correlaciones, hemos utilizado la correlación de Pearson pero podemos utilizar cualquier otra y buscar cual de ellas nos funciona mejor. También podemos utilizar otro tipo de arquitectura de red neuronal, por ejemplo añadiendo más capas y haciéndola más compleja o variando detalles de cada capa como puede ser número de neuronas u otros parámetros que Keras u otra librería como Tensorflow nos permita hacer.

También otra línea de investigación aunque parecida no es igual, podemos utilizar las redes LSTM multidimensionales o GridLSTM que en recientes experimentos han tenido muy buenos resultados, basándonos en la idea original de enfrentar dos audios, un original y un test.

Por último podemos utilizar nuestro sistema en combinación con otro sistema tradicional en el que utilizando los dos consigamos mejores resultados gracias a las virtudes de cada uno.

Como podemos ver, gracias a que este sistema es novedoso y gracias a que hemos conseguido resultados más que aceptables en la tarea de reconocimiento de locutor dependiente de texto, nos ha permitido abrir muchas líneas de investigación sobre las que podemos experimentar.

Referencias

[1] Á. Palomo Sánchez y D. Torre Toledano, Mejoras en un sistema de reconocimiento del locutor dependiente de texto, Trabajo de Fin de Grado, EPS-UAM, 2017.

[2] Á. Mesa Castellanos y D. Torre Toledano, Reconocimiento del locutor dependiente de texto: Experimentos con la base de datos RSR2015, Trabajo de Fin de Grado, EPS-UAM, 2016.

[3] J. S. M. M. &. H. Y. Benesty, Springer handbook of speech processing, Chapter 37-Text-dependent speaker recognition, Springer Science & Business, 2007.

[4] A. Larcher, K. Aik Lee, B. Ma y H. Li, «Text-dependent speaker verification: Classifiers, databases and RSR2015,» *Spech Communication*, pp. 56-77, 2014.

[5] D. Ramos Castro, «Transparencias de la asignatura: Tratamiento de Señales de Voz y Audio: Introduccion a la evaluacion de sistemas de reconocimiento,» pp. 16-21, 2017.

[6] J. Ortega García, «Transparencias de la asignatura: Tratamiento de Señales de Voz y Audio: Sistema Auditivo, Sensación Sonora y Parametrización Perceptual,» pp. 41-59, 2017.

[7] «Documentación online de Kaldi,» [En línea]. Available: http://kaldi-asr.org/doc/about.html. [Último acceso: 10 02 2019].

[8] «Documentación online de HTK,» [En línea]. Available: http://htk.eng.cam.ac.uk/. [Último acceso: 10 02 2019].

[9] «Herramienta PHNrec,» [En línea]. Available: https://speech.fit.vutbr.cz/software/phoneme-recognizer-based-long-temporal-context. [Último acceso: 10 02 2019].

[10] «Libreria Keras,» [En línea]. Available: https://keras.io/. [Último acceso: 10 02 2019].

[11] «Herramienta DETware v2.1,» [En línea]. Available: https://www.nist.gov/itl/iad/mig/tools. [Último acceso: 10 02 2019].

[12] D. Torre Toledano y J. González Rodríguez, «Transparencias de la asignatura: Tratamiento de Señales de Voz y Audio: HMMs,» 2017.

[13] J. González Rodríguez, D. Torre Toledano y J. Ortega-García, «Voice biometrics,» de *Handbook of biometrics*, Springer US, 2008, pp. 151-170.

[14] «Imagen red neuronal fully-connected,» [En línea]. Available: https://www.researchgate.net/figure/Sample-of-a-feed-forward-neural-network_fig1_234055177.

[15] «Redes neuronales convolucionales,» [En línea]. Available: http://cs231n.github.io/convolutional-networks/.

[16] «Imagen red convolucional,» [En línea]. Available: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148.

[17] R. G. Leonard and G. Doddington. Tidigits (ldc93s10). http://www.ldc.upenn.edu.

[18] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallet, N. L. Dahlgren, and V. Zue. Timit acoustic-phonetic continuous speech corpus (ldc93s1). http://www.ldc.upenn.edu.

[19] J. Campbell and A. Higgins. Yoho speaker verification (ldc94s16). http://www.ldc.upenn.edu.

[20] R. Woo, A. Park, and T. J. Hazen. The mit mobile device speaker verification corpus: data collection and preliminary experiments. In Proceedings of IEEE Odyssey, 2006.

[21] J. Fierrez-Aguilar, J. Ortega-Garcia, D. T. Toledano, and J. Gonzalez-Rodriguez. Biosec baseline corpus: A multimodal biometric database. Pattern Recognition, 40:1389–1392, 2007.

[22] Dehak N. et al. “Front-End Factor Analysis for Speaker Verification”. Audio, Speech, and Language Processing, IEEE Transactions on, 19(4):788 – 798, Feb. (2011).

[23] Prince, S. J. D. & Elder, J. H., “Probabilistic linear discriminant analysis for inferences about identity”. Proc. 11th Int. Conference on Computer Vision, pp. 1–8. (2007).

[24] Graves, A. et al. “Speech recognition with deep recurrent neural networks”. Acoustics, Speech and Signal Processing, IEEE International Conference on. IEEE, (2013).

[25] Li, J., et al. "Exploring multidimensional LSTMs for large vocabulary ASR." Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE Int. Conference on. IEEE, 2016.

[26] Kalchbrenner, N. et al., "Grid long short-term memory." arXiv:1507.01526, 2015.

[27] Grezl F., et al., “Investigation into bottle-neck features for meeting speech recognition”, Proc. Interspeech, 2009.

[28] Snyder, D., et al., “Deep neural network embeddings for text-independent speaker verification”, Proc. Interspeech, 2017.

[29] Variani E. et al., “Deep neural networks for small footprint text-dependent speaker verification”, Proc. ICASSP, 2014.

[30] Hochreiter, S. & Schmidhuber, J. “Long short-term memory”. Neural Comput. 9,1735–1780 (1997).

Glosario

HMM: Hidden Markov Model

MLLR: Maximum Likelihood Linear Regresion

fMLLR: feature-space Maximum Likelihood Linear Regresion

MAP: Maximum A Posteriori

MFCC: Mel-Frecuency Cepstral Coefficients

LPCC: Linear Prediction Cepstrum Coefficients

CMVN: Cepstral Mean and Variance Normalization

CMS: Cepstral Mean Substraction

EM: Expectation-Maximization

EER: Equal Error Rate

FA: False Acceptance

FR: False Recognition

DNN: Deep Neural Network

DFT: Discrete Fourier Transform

FFT: Fast Fourier Transform

UBM: Universal Background Model

TFG: Trabajo de Fin de Grado

ASR: Automatic Speech Recognition

TFM: Trabajo de fin de Master

PLDA Probabilistic Linear Discriminant Analysis

RNN Recurrent Neural Network

LSTM Long-Short Term Memory

CNN Convolutional Neural Network

HTK Hidden Markov Model Toolkit

CPU Central Processing Unit

GPU Graphics Processing Unit

WAV WAVEform audio format

SGD Stochastic Gradient Descent

DET Detection Error Tradeoff