UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

PROYECTO FIN DE CARRERA

DESARROLLO DE UN SISTEMA DE RECONOCIMIENTO DE HABLA NATURAL BASADO EN REDES NEURONALES PROFUNDAS

Ingeniería de Telecomunicación

Carolina Camacho Costumero
Septiembre 2016

DESARROLLO DE UN SISTEMA DE RECONOCIMIENTO DE HABLA NATURAL BASADO EN REDES NEURONALES PROFUNDAS

AUTOR: Carolina Camacho Costumero
TUTOR: Daniel Tapias Merino
PONENTE: Doroteo Torre Toledano

Área de Tratamiento de Voz y Señales
Dpto. de Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Septiembre 2016

Resumen

Resumen

El objetivo de este proyecto es diseñar un sistema de reconocimiento de habla continua de gran vocabulario (LVCRS) utilizando modelos ocultos de Markov (HMM) y redes neuronales profundas (DNN).

Una vez conseguido un sistema de referencia con HMM, se procederá a optimizarlo usando, en primer lugar, diversas transformadas y en segundo lugar, las redes neuronales. Para el caso de las redes neuronales, se harán diferentes experimentos variando una serie de parámetros hasta conseguir mejoras en el sistema.

Para ello, se realizarán pruebas de reconocimiento sobre contenidos audiovisuales obtenidos de Internet, para así ver cómo reacciona el sistema en un entorno menos controlado.

Una vez conseguido el sistema mejorado, se integrará en un producto comercial el cual permite la búsqueda avanzada de contenidos dentro de vídeos de Internet, utilizando el texto reconocido por el sistema para añadir posibilidades de búsqueda avanzada sobre los vídeos.

Palabras Clave

Sistema de reconocimiento de habla continua de gran vocabulario, modelos ocultos de Markov, redes neuronales profundas, Kaldi, capas, neuronas, contenidos audiovisuales.

Abstract

Abstract

The goal of this project was to implement a large vocabulary continuous speech recognition system (LVCRS) using hidden Markov models (HMM) and deep neural networks (DNN).

Once a reference system with HMM is implemented, the next step will be to improve it using different transforms and neural networks. In regards to the neural networks, different experiments will be performed while tuning a series of parameters in order to achieve a better system.

Therefore more tests were performed on audiovisual content obtained from the Internet in order to verify how the system works in a less controlled environment.

Finally, its abilities were tested so that it can be integrated in a comercial product, used for speech recognition on Internet videos with the aim of using the recognized text to add new advanced search capabilities to these videos.

Key words

Large Vocabulary Continuous Speech Recognition, Hidden Markov Models, Deep Neural Networks, Kaldi, layers, neurons, audiovisual content indexing.

Agradecimientos

Agradecimientos

Teleco es una carrera dura, una de esas que la gente se debería pensar dos veces antes de comenzarla o de salir corriendo a tiempo al ver las notas generales de los primeros exámenes. Sin embargo, aquí me encuentro yo después de estos años, rozando el final de esta etapa, y si miro hacia atrás podría contar más cosas buenas que malas de todos estos años. Mentiría si dijera que nunca estuve a punto de tirar la toalla, pero no fue así, seguí adelante y me alegro de ello, gracias a la gente que siempre estuvo ahí y me apoyó para seguir adelante. Y volvería a mentir si dijera que todo ha sido un camino de rosas, nadie dijo que fuera fácil, muchas veces lo sencillo es aburrido, así que hay que darle un poco de emoción.

Al final todo este trabajo y esfuerzo han sido recompensados y como dijo el maestro Ludwing Van Beethoven "el genio se compone del 2 % de talento y del 98 % de perseverante aplicación", y creo que si hay algo que nos caracteriza a los telecos es la perseverancia, a cabezotas no nos gana nadie.

Por supuesto todo esto no habría sido posible, ni siquiera llevadero, sin la cantidad de experiencias vividas que me llevo dentro de las cuatro paredes de una clase, un laboratorio, la cafetería o incluso fuera de la universidad. La gente que he ido conociendo y de la que me he ido rodeando estos años, ha sido de lo mejor que me podría haber pasado y no lo cambiaría por nada. Así que de antemano gracias a todos los que habéis formado parte de esta etapa de una forma u otra.

En primer lugar, quiero dar las gracias a mi tutor, Daniel Tapias, por haberme dado la oportunidad de realizar este proyecto, y por permitirme dar mis primeros pasos en el mundo profesional. Gracias por tu apoyo y dedicación durante todo este tiempo. Extiendo el agradecimiento a profesores como Chema, Doroteo, Ángel, Joaquín, José y Daniel R., que en este último mes de estrés vivido, han sido comprensivos con la situación y después de tanto lío, me han ayudado.

Javier, has sido mi apoyo durante toda la carrera y para colmo me has tenido que aguantar también en la oficina. Decirte que sin tu ayuda tanto durante la carrera, fuera de la universidad, en el proyecto y en la oficina no hubiera sido posible llegar donde estoy hoy. Nunca sabré cómo agradecerte tanto eres un gran amigo y como tú ya hay pocos.

Bruno, mi italiano favorito, gracias por tu saber estar y por conseguir tranquilizarme en los momentos de estrés. Gracias por tu ayuda y por ser como eres.

Juanma, compañero de carrera que prácticamente desconocía hasta llegar a sigma y que me alegro de haber podido conocer. Gracias por la ayuda, dedicación y las risas en la oficina.

Por supuesto no puedo dejar sin mencionar a Jorge, mi otro compañero de trabajo, que

tantas cosas me ha enseñado durante este tiempo y que todavía me sigue sorprendiendo con todo lo que sabe.

Me gustaría dar las gracias también a las demás personas que forman Sigma Technologies y Tax Planning, porque habéis hecho que desde el principio me sintiera una más. Martín, Cristina, Lidia, Mayte, Mari Luz, Hamada, Elena R., Elena M y Kawika.

Mi niña Almudena, esa chica que el primer día de clase me encontré apoyada en una columna, mientras yo estaba en otra. Estábamos solas y nos miramos y lo mejor que pudimos hacer fue presentarnos y ya todo fue fácil. Fuiste mi compañera de prácticas muchos años y compañera, y ahora mi amiga, mi apoyo en los momentos malos y las carcajadas en los buenos y por supuesto nos quedan muchos años más de todo esto.

No puedo olvidarme de mis chicas, con las que he compartido mucho los últimos años de carrera y que sin ellas la cuesta final habría sido más difícil: Cris, Pili, Guada, Mónica, Ana. Gracias a todas.

Mis compis de batalla desde el primer día de universidad: Raquel, Celia, Tamara, Alicia, Marta A., Marta M, Eva, Maya, Gila, Casas, Pascu, Garri, Brande, Suja, Álvaro, Txiki, Eslava, Cubillo, Rosely, Gustavo.

No podría dejar sin mencionar a mi mejor amiga desde la infancia, Sandra, hemos crecido juntas y seguiremos creciendo. Además de mis compañeros del colegio y posteriormente amigos: Polchi, Kun, Elena, Antonio, Fer, Riki, Rober etc.

Agradecer a mis profesores, amigos y compañeros del conservatorio de música, en el que he pasado tan buenos años, duros, pero muy gratificantes a la vez. La música es mi otra mitad, y siempre ha formado y formará parte de mi vida. En especial, gracias a mi primer profesor de música, Serafín, por descubrirme este mundo tan bonito, insistirme para que me apuntara al conservatorio y hacerme ver que sería algo tan importante para mí en el futuro. A mi profesor de flauta travesera, Miguel, por intentar sacar siempre lo mejor de mí y a todos esos músicos que he conocido, con los que me he cruzado o compartido escenario, gracias por cada granito de arena aportado tanto personal como musicalmente. Y por supuesto seguiremos haciendo lo que mejor se no da, que es música.

A mis compañeros de batallas y demás: Sonia, Marta, David, Ángela, Sara, Ángel, este año sin vosotros habría sido muy poco llevadero.

Dar las gracias al apoyo y cariño de Ruben y su familia en estos años, siempre me hicieron sentir parte de esa increíble familia.

Mi más sincero agradecimiento a mis padres por haberme enseñado tantas cosas durante toda mi vida y por aguantarme en lo bueno, en mis épocas de estrés durante estos años y por darme todo sin pedir nada a cambio. Cómo no agradecer a mis hermanos por hacerme crecer como persona. Sois lo mejor que tengo.

No podría olvidarme de mis abuelos que han formado parte de mi vida de una manera especial y, que allá donde estén, sé que estarán orgullosos.

En general a toda mi familia, que cada uno de una forma u otra, ha influido en estos años así que gracias por estar ahí siempre. En especial quería agradecer a mi primo David, al que creo que le debo parte de responsabilidad de haber estudiado esta carrera, aunque intentaste que hiciera informática no coló, gracias por todos tus consejos y experiencia.

Como reflexión una frase de la película en busca de la felicidad “nunca dejes que nadie te

diga que no puedes hacer algo. Si tienes un sueño debes protegerlo. Si alguien no puede hacer algo te dirá que tú tampoco puedes. Si quieres algo ve tras ello. Punto.” Creo que resume bastante bien mi experiencia en teleco, en la música y en la vida en general.

Quiero expresar el más sincero agradecimiento al Banco Santander por la concesión de la "Beca de Prácticas Santander CRUE-CEPYME", que ha servido como complemento a mi formación y contribuido a mi acercamiento al ámbito profesional.

También quiero mostrar el agradecimiento a Telefónica I+D por la cesión de los derechos de las bases de datos que se han empleado en el desarrollo del sistema realizado como objetivo de este proyecto.

Además quiero dar las gracias al Biometric Recognition Group - ATVS por ofertar desde su grupo este proyecto final de carrera y haberme concedido la oportunidad de realizarlo.

Por último, y no menos importante, agradecer a Sigma Technologies la oportunidad brindada de realizar este proyecto, y por supuesto, de la acogida por parte de todos sus miembros y el conocimiento adquirido de cada uno de ellos.

Índice general

Índice de figuras

Índice de cuadros

Acrónimos

Acrónimos

ARPA	Advanced Research Projects Agency
ASR	Automatic Speech Recognition
BLSTM	Bidirectional Long Short-Term Memory
bMMI	boosting Maximun Mutual Information
CMS	Cepstral Mean Substraction
CMU	Carnegie Mellon University
CMVN	Cepstral Mean and Variance Normalization
DCT	Discrete Cosine Transformate
DFT	Discrete Fourier Transformate
DLSTM	Deep Long Short-Term Memory
DNN	Deep Neural Networks
DTW	Dynamic Time Warping
FFT	Fast Fourier Transformate
fMMLR	Feature-space Maximum Likelihood Linear Regression
FST	Finite State Transducers
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model (Modelos Ocultos de Markov)
HTK	HMM ToolKit
LDA	Linear Discriminant Analysis
LDC	Linguistic Data Consortium
LPC	Linear Predictive Coding
LSTM	Long Short-Term Memory
VCSR	Large Vocabulary Continuous Speech Recognition
MFCC	Mel Frequency Cepstral Coefficient
MLLT	Maximum Likelihood Linear Transform
MMI	Maximun Mutual Information
MPE	Minimum Phone Error
PLP	Perceptual Linear Prediction
RAH	Reconocimiento de habla automático
RNN	Recurrent Neural Networks
SNR	Signal to Noise Ratio
SRILM	Stanford Research Institute Language Modelling

1 Introducción

1.1. Motivación del proyecto

El reconocimiento automático de voz, también denominado conversión de voz a texto, ha ido avanzando a lo largo de los últimos años y ha experimentado una gran evolución debido a su gran utilidad, a la facilidad que aporta a determinadas tareas y al incremento en la potencia de los sistemas informáticos necesarios. Esto permite la liberación de las manos de cualquier teclado o dispositivo de entrada y por consiguiente se convierte en una gran ventaja a la hora de usar estos sistemas en nuestro día a día.

Para poder convertir la información vocal pronunciada por el hablante en formato texto, es necesario obtener una representación simbólica discreta de una señal de voz continua. Esto es realizado por los sistemas de reconocimiento automático de habla (Automatic Speech Recognition - ASR). En este ámbito, se tiende a aumentar la complejidad en los modelos, para mejorar la precisión del reconocimiento en distintas condiciones acústicas y vocabularios extensos.

La variabilidad que afecta al reconocimiento de voz viene dada por varios factores como son: los diferentes tractos o cuerdas vocales, canales (como condiciones acústicas del entorno, micrófonos, ancho de banda), el ruido, algunos factores congénitos (por ejemplo frecuencias de las cuerdas vocales, capacidad pulmonar) y otros adquiridos (dialectos, velocidad del habla) entre otros. Al existir tanta variabilidad, es necesario conseguir un modelo acústico robusto. Además, será necesario un modelo estadístico de lenguaje adaptado, para obtener, con ambos, mejores resultados en el reconocimiento de contenidos audiovisuales en Internet.

Los avances en los nuevos modelos de aprendizaje automático y en el hardware, han dado lugar a métodos más eficientes en el reconocimiento automático del habla. Por ello, la motivación principal de este proyecto será el estudio del reconocimiento de contenidos audiovisuales de Internet, utilizando Redes Neuronales Profundas (Deep Neural Network - DNN), para conseguir una mejora respecto a los Modelos Ocultos de Markov (Hidden Markov Models - HMM).

1.2. Objetivos y enfoque

El objetivo fundamental de este proyecto es el desarrollo de un sistema de reconocimiento de habla natural en español y para el cual se usarán contenidos multimedia de Internet, utilizando el software Kaldi. Para ello, se llevará a cabo una evaluación de un reconocedor basado en DNNs que será comparado con uno basado en HMMs. Se podrá observar si las DNNs son o no más robustas frente a las fuentes de variabilidad antes mencionadas, si proporcionan mejores tasas de reconocimiento y analizar los distintos parámetros de configuración que poseen los reconocedores de voz basados en DNNs, para optimizar su comportamiento. La evaluación del sistema se llevará a cabo de la siguiente manera:

- En primer lugar, se tiene una base de datos de entrenamiento, que será con la que se realizarán las pruebas de Auto-test.
- En segundo lugar, se cuenta con una base de datos de test, distinta a la de entrenamiento, la cual no ha sido usada para entrenar el sistema. Para test, con una parte de los archivos de Auto-test que no han sido entrenados. La base de datos de test está compuesta por tres conjuntos de datos:
  - Una base de datos grabada en las mismas condiciones que la base de datos de entrenamiento.
  - Una base de datos clasificada por el pitch del locutor.
  - Una base de datos de vídeos obtenidos a través de Internet.

Para poder realizarlo se fijarán una serie de objetivos, que se enuncian a continuación:

1. Se usarán las herramientas de Kaldi [1] tanto para el proceso de entrenamiento de HMMs, como para la etapa de reconocimiento.
2. A continuación, se adaptará el software (Kaldi) para usar los mismos parámetros y ficheros de audio utilizados en el sistema de referencia desarrollado en los proyectos fin de carrera de Javier Antón Martin [2] y de Juan Manuel Perero Codosero [3]. En ellos se utilizó HTK para el proceso de entrenamiento y Julius para el de reconocimiento. Con el fin de hacer una comparativa entre dicho sistema y el propuesto para este proyecto, se usarán HMMs, que fueron los usados en el sistema de referencia..
3. Más tarde, se estudiarán las diferentes mejoras obtenidas al repetir este proceso con Kaldi implementando DNNs.
4. Finalmente, se evaluarán todos los resultados obtenidos.

1.3. Estructura de la memoria

La memoria de este proyecto está dividida en los siguientes capítulos:

- Capítulo 1. Introducción: Motivación del Proyecto, Objetivos y Enfoque.
- Capítulo 2. Estado del Arte: Sistemas de Reconocimiento de Voz y Arquitectura.
- Capítulo 3. Bases de Datos: Descripción y Caracterización.
- Capítulo 4. Generación del Sistema Base Kaldi y Comparativa con HTK.
- Capítulo 5. Optimización del Sistema Base y Resultados.
- Capítulo 6. Optimización del Sistema usando Redes Neuronales.
- Capítuo 7. Aplicación del Sistema.
- Capítulo 8. Conclusiones y Trabajo Futuro.
- Referencias y anexos.

2

Estado del Arte

En este capítulo se dará una visión general de la evolución en el área del reconocimiento de voz a lo largo de los años hasta nuestros días (sección 2.1). En la siguiente sección, se dará esta evolución enfocada al uso de las redes neuronales en el reconocimiento de voz (sección 2.2). A continuación, se explicarán los elementos que forman una red neuronal y sus principales características (sección 2.3). Por último, se explicará la arquitectura utilizada en los sistemas de reconocimiento y sus diferentes bloques (sección 2.4).

2.1. Introducción general del reconocimiento de voz

El reconocimiento de habla natural ha experimentado un gran desarrollo gracias a los avances que ha habido en el procesamiento de señal, en los algoritmos, las arquitecturas y las plataformas de computación.

Desde 1940, los laboratorios Bell de AT&T desarrollaron un dispositivo rudimentario para reconocer voz, que se basaba en los principios de la fonética acústica. El éxito de esta tecnología dependería de su capacidad para percibir información verbal compleja con alta precisión.

En la década de los 50, el sistema anterior desarrollado, hizo posible la identificación de dígitos mono-locutor basada en la medición de las resonancias espectrales del tracto vocal para cada dígito. En esta misma línea, RCA Labs trabajó en el reconocimiento de 10 sílabas, pero no es hasta finales de la década, cuando tanto la University College de Londres como el MIT Lincoln Lab, intentaron desarrollar un sistema de reconocimiento limitado de vocales y consonantes. Esto parecía algo novedoso por el uso de información estadística y su objetivo era conseguir mejorar el rendimiento en palabras que tuvieran dos o más fonemas.

Pero fue en los 60, cuando estos sistemas electrónicos utilizados hasta el momento, dieron paso a los sistemas con hardware específico, en los NEC Labs de Japón. En esta etapa, hay que destacar tres proyectos importantes en la investigación de esta disciplina:

- RCA Labs tenía como objetivo desarrollar soluciones para los problemas con la falta de uniformidad que existen en las escalas de tiempo en el habla. Para ello, diseñaron unos métodos de normalización en el dominio temporal, que detectaban de manera fiable el inicio y fin de discurso.

- En la Unión Soviética, T. K. Vintsyuk, propone el uso de métodos de programación dinámica para obtener el alineamiento temporal de parejas de realizaciones. De aquí surge la técnica DTW (Dynamic Time Warping).

- Por último, en el área del reconocimiento de habla continua, D. R. Reddy de la Universidad de Stanford, desarrolló el seguimiento dinámico de fonemas, consiguiendo un reconocedor de oraciones de amplio vocabulario.

En los años 70, surgieron críticas sobre la viabilidad y utilidad del reconocimiento automático de habla. Aún así, los principales campos de estudio fueron los siguientes: el reconocimiento de palabras aisladas fundamentado en el encaje de patrones, programación dinámica, y más adelante, técnicas LPC (Linear Predictive Coding). Esta última se empleó con gran éxito en la codificación y compresión de la voz, usando medidas de distancias sobre el conjunto de parámetros LPC.

Los primeros intentos de reconocedores de habla continua y grandes vocabularios fueron llevados a cabo por IBM, con el dictado automático de voz, ARPA Speech Understanding Research, y la Universidad de Carnegie Mellon, con el sistema de gran éxito Hearsay I. Finalmente, en los AT&T Labs, se investigaron los reconocedores independientes del locutor para aplicaciones telefónicas, terminando con la realización de sistemas ASR (Automatic Speech Recognition), favorecidos por la aparición de tarjetas microprocesador.

La década de los 80 comienza con una buena base en la construcción de sistemas de reconocimiento. En las etapas anteriores solo se reconocían vocablos aislados, ahora se pueden reconocer palabras encadenadas fluidamente. El avance más importante fue el paso de métodos basados en comparación de plantillas a otros basados en modelos estadísticos, ampliando el uso de los Modelos Ocultos de Markov o HMMs. Estos experimentaron muchas mejoras y fueron considerados los mejores modelos que capturaban y modelaban la variabilidad del habla. En esta etapa también, comenzaron a tener importancia las redes neuronales gracias al desarrollo de algoritmos de aprendizaje más eficaces.

Además, se llevaron a cabo los siguientes avances:

- El diseño de unidades de decodificación fonética a partir de la experiencia de fonetistas en la interpretación de espectrogramas.

- La grabación de grandes bases de datos como TIMIT, que permitieron comparar los resultados de los diferentes grupos de trabajo.

- El programa DARPA (Defence Advance Research Projects Agency), ayudó en Estados Unidos a impulsar el desarrollo de sistemas de reconocimiento para habla continua y vocabularios de gran tamaño independientes del locutor.

- El desarrollo, por parte de la CMU, de su sistema SPHINX [4].

En los años 90, se ampliaron los tamaños de vocabularios y se separaron los campos de aplicación. Fue de gran importancia su aplicación sobre la línea telefónica, así como en entornos con condiciones adversas y ruido, donde dio buenos resultados.

Los avances que se han ido produciendo en el ámbito de las tecnologías del habla cada día, son más significativos. En el campo del reconocimiento automático de voz, los reconocedores actuales manejan vocabularios cada vez más grandes y reducen las tasas de error. Esto es debido al uso de algoritmos más eficientes, al uso de equipos más potentes, al aumento de la complejidad de estos sistemas, con modelados más sofisticados y sobre todo, a que hay muchos más datos para entrenar que anteriormente. El amplio grado de aplicación en función de los usuarios y los distintos entornos, hacen que no haya un sistema de reconocimiento de voz universal y sea necesaria su adaptación a las condiciones de funcionamiento y al tipo de aplicación que se requiera.

2.2. Historia de las redes neuronales

La redes neuronales artificiales comenzaron a tener una mayor importancia a partir de la década de los 80, sin embargo mucho tiempo atrás, ya se habían estudiado y modelado algún tipo de esas redes. A continuación se muestra su evolución a lo largo de los años:

En 1936 Alan Turing, fue el primero que estudió el cerebro como una posible forma de ver la computación. Sin embargo, los primeros teóricos que formalizaron los fundamentos de la computación neuronal fueron el neurofisiólogo Warren McCulloch y el matemático Walter Pitts. Los cuales en 1943, produjeron la primera neurona artificial y modelaron una red neuronal simple mediante circuitos eléctricos.

La aparición de las ANNs (Artificial Neural Network) en el reconocimiento de patrones se da en los años 40, pero no es hasta estos últimos años en los que la mejora y avances en la computación permiten su uso de una forma extendida.

En 1949 Donald Hebb, fue el primero que consiguió explicar los procesos del aprendizaje (que es el elemento fundamental de la inteligencia humana), el cual según él, ocurría cuando ciertos cambios eran activados en una neurona. Además, intentó buscar similitudes entre la actividad nerviosa y el aprendizaje. Los estudios de Hebb formaron parte de las bases de la Teoría de las Redes Neuronales.

En 1950 Karl Lashley, descubrió que la información no se almacenaba de forma centralizada en el cerebro, si no que se distribuía por encima de él.

Se puede considerar que la inteligencia artificial nació en el Congreso de Dartmouth, que tuvo lugar en 1956.

El perceptrón fue creado por Frank Rosenblatt en 1958, y es la red neuronal más antigua, que hoy en día se sigue usando como identificador de patrones. Este modelo se puede generalizar ya que, es capaz de reconocer señales similares a una serie de patrones aprendidos con anterioridad, aunque dichas señales no se hayan usado en el entrenamiento de los patrones. Pero tenía una serie de limitaciones por el momento, por ejemplo, no era capaz de resolver el problema de la función OR-exclusiva ni clasificar clases no separables linealmente.

En 1960 Bernard Widroff y Marcian Hoff, desarrollaron el modelo Adaline, que fue la primera red neuronal que se aplicó a un problema real (filtros adaptativos para eliminar ecos en las líneas telefónicas) y que se siguió usando durante décadas.

En 1969 se produjo el final momentáneo de las redes al descubrir, Marvin Minsky y Seymour Papert, que los perceptrones no eran capaces de resolver problemas más o menos fáciles como el aprendizaje de una función no lineal. Con ello quedó demostrado que el perceptrón era muy débil. En 1974 Paul Verbos, desarrolló la idea del algoritmo de propagación hacia atrás (backpropagation). Pero estos avances se vieron interrumpidos por la falta de financiación y la existencia de otros métodos alternativos.

En 1986 fue John Hopfield el que provocó el resurgimiento de las redes neuronales y David Rumelhart y G.Hinton quienes redescubrieron el algoritmo de backpropagation. Y a partir de aquí siguieron las investigaciones y el desarrollo de las redes neuronales.

En la década de los 2000, el interés por la investigación en esta área creció debido a las mejoras en la tecnología computacional y los avances en los algoritmos de aprendizaje. Debido a esto aparecieron en 2006 las DBN (Deep Belief Neural) redes compuestas por múltiples capas, con conexiones entre las diferentes capas, pero no entre las unidades de una misma capa. Usan un pre-entrenamiento no supervisado para la actualización de los parámetros de la red.

Hoy en día, las ANNs son redes neuronales artificiales con multitud de capas que son entrenadas con una gran cantidad de datos. Ya el entrenamiento no supervisado ha caído en desuso y la nueva tendencia es hacia el aprendizaje supervisado [5].

2.3. Redes Neuronales

La mayoría de los sistemas de reconocimiento de voz actuales, utilizan modelos ocultos de Markov (HMM). Pero en los últimos años los avances en los algoritmos de aprendizaje automático y en la capacidad de los equipos informáticos, han dado lugar a métodos más eficientes para el entrenamiento de redes neuronales artificiales (ANN), y en particular de redes neuronales profundas (DNN) que serán las usadas en la última parte de este proyecto.

2.3.1. Definición y caracterización

Las redes neuronales artificiales (ANN) son una forma de simular características propias de los humanos, como es la capacidad para memorizar y asociar hechos. El hombre es capaz de resolver problemas acudiendo a la experiencia acumulada. Por lo que, se intenta construir sistemas capaces de reproducir esta característica humana.

En definitiva, una red neuronal es un método de reconocimiento de patrones que intenta imitar, de forma artificial y simplificada, la forma en la que el cerebro humano procesa la información, siendo capaz de adquirir conocimiento a través de la experiencia. La unidad básica de procesamiento, que está inspirada en la célula fundamental del sistema nervioso, es la neurona.

Son redes interconectadas de forma masiva, en las que su organización jerárquica se realiza de la misma forma en la que lo hacen los sistemas nerviosos biológicos [6]. Se cree que el cerebro se compone de miles de millones de neuronas interconectadas en muchas capas. Todas estas neuronas están especializadas en el aprendizaje automático de la información y por esta razón, los seres humanos interpretan muy bien el mundo que los rodea, aunque tampoco estamos exentos de cometer errores. Las ANNs abordan todo esto de una manera

similar al cerebro humano, usando una función de coste para determinar el error entre la salida deseada y la estimada. Por lo tanto las redes neuronales:

- Consisten en unidades de procesamiento que intercambian datos o información.
- Se utilizan para reconocer patrones.
- Tienen capacidad de aprender y mejorar su funcionamiento.

2.3.1.1. Neurona

La neurona biológica es estimulada o excitada a través de sus entradas (inputs) y cuando alcanza un determinado umbral, la neurona se dispara o activa, pasando una señal hacia el axón.

Las neuronas biológicas transmiten la información (estímulo) entre ellas a través de la sinapsis. Las ANNs tratan de modelar los estímulos de una neurona (entradas) entre las conexiones (sinapsis) de las neuronas, dando ciertos pesos a las entradas. Cada neurona, excepto las que se encuentran en las capas de entrada, recibe información de la capa anterior y produce una salida para las neuronas de la capa siguiente.

Cada entrada se pondera con un parámetro que asigna un valor, y a esto se añaden los productos resultantes anteriores. A continuación, se produce una salida por parte de cada neurona, que viene determinada por una función escalonada llamada función de activación, y es por la que se diferencian las neuronas.

- **Perceptrón:** fue creado por Frank Rosenblatt en 1958 [6] y fue uno de los tipos más utilizados para neuronas artificiales. Un perceptrón lleva varias entradas x_i con las que calcula una suma ponderada de ellas junto con los pesos w_i y da una salida en función del sesgo b. La salida del perceptrón se calcula de la siguiente forma:

$$y(x) = \begin{cases} 1, \sum_{i=1}^{n} w_i x_i + b > 0 \\ 0, \sum_{i=1}^{n} w_i x_i + b \leq 0 \end{cases}$$

donde n es el número de entradas al perceptrón.

El perceptrón utiliza una función escalón como función de activación. Aunque la función escalón es simple e intuitiva, puede causar un comportamiento inestable. No sería posible modificar los pesos de las entradas con esta función ya que tiene una salida discreta. Así que los cambios en los pesos y en los sesgos durante el aprendizaje, se realizarían de forma más eficaz en otro tipo de funciones de activación, como son las diferenciables.

Ahora se muestra una figura de un perceptrón simple:

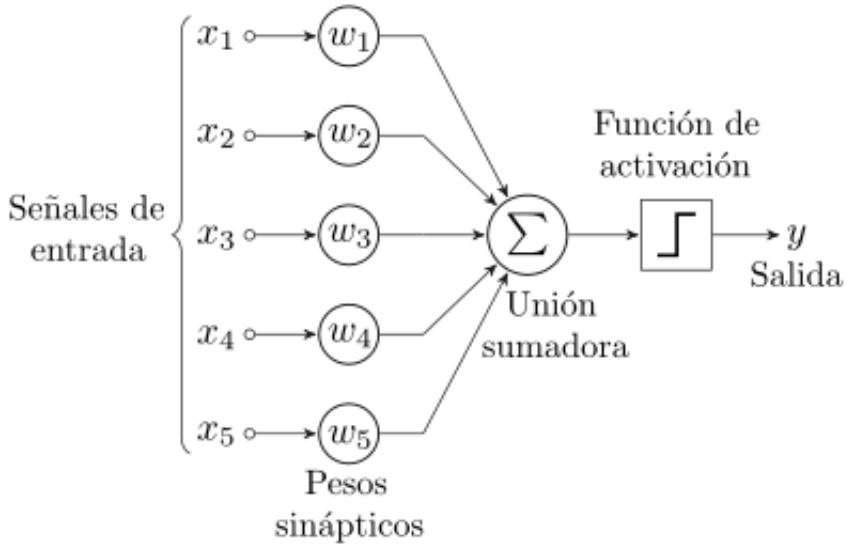

Figura 2.1: Perceptrón simple

2.3.1.2. Función de entrada

La neurona tiene muchos valores de entrada que trata como si fuera uno solo y recibe el nombre de entrada global. Para poder combinar todas estas entradas, se usa la función de entrada que se calcula mediante un vector de entrada.

El vector de entrada necesario, viene dado por la multiplicación de cada valor de entrada por el peso asignado anteriormente a la neurona. A continuación se muestra un ejemplo de una neurona con dos entradas y una salida:

1. Sumatorio: se realiza el sumatorio de todas las entradas multiplicadas por sus correspondientes pesos.

$$\sum_j n_{ij} w_{ij} \ \ con \ \ j = 1, 2, \ldots, n \tag{2.1}$$

2. Producto: se realiza el producto de todas las entradas multiplicadas por sus correspondientes pesos.

$$\prod_j n_{ij} w_{ij} \ \ con \ \ j = 1, 2, \ldots, n \tag{2.2}$$

3. Máximo: solamente se considera el valor de entrada más fuerte, con su previa multiplicación por su peso.

$$\max_j (n_{ij} w_{ij}) \ \ con \ \ j = 1, 2, \ldots, n \tag{2.3}$$

2.3.1.3. Función de activación

Una neurona biológica puede estar activa (excitada) o inactiva (no excitada). Las neuronas artificiales también tienen distintos estados de activación: pueden tener dos como las biológicas o pueden tomar un cierto valor dentro de un conjunto acotado.

La función de activación calcula el estado de actividad en el que se encuentra la neurona y transforma la entrada en un valor (estado) de activación, cuyo rango suele estar entre 0 y 1 o entre -1 y 1.

A continuación se detallan las funciones de activación más utilizadas:

- **Función sigmoide:** la neurona sigmoide es un tipo muy popular de neurona artificial que utiliza la función de activación sigmoide σ. La salida viene dada por:

$$\sigma(z) = \frac{1}{1 + e^{-az}}$$

donde "a" es el valor que modifica la pendiente de la función, y "z" el valor de entrada global menos el umbral.

Cuando se utiliza la función sigmoidea logística para determinar la salida de la neurona, sigue la siguiente función:

$$y(x) = \frac{1}{1 + e^{-z(x)}} = \frac{1}{1 + \exp\left(\sum_{i=1}^{n} w_i x_i + b\right)}$$

$$\text{donde } \; z(x) = \sum_{i=1}^{n} w_i x_i + b$$

La salida de esta neurona tiene el mismo límite superior e inferior que la salida del perceptrón, 1 y 0. Cuando "z" es muy grande y positivo $e^{-z} \to 0$ y σ(z)≈1 y cuando "z" es muy negativa $e^{-z} \to \infty$ y σ(z)≈0.

La función sigmoide es una versión suavizada de una función escalonada. Esto es beneficioso, ya que la función es continua y diferenciable y se puede usar para realizar pequeños cambios en los pesos o sesgos para conseguir pequeños cambios en la salida de la neurona.

- **Función Lineal:** algunas redes neuronales usan esta función de activación por su eficiencia y facilidad.

$$f(x) = \begin{cases} -1, x \leq \frac{-1}{a} \\ a * x + b, \frac{-1}{a} < x < \frac{1}{a} \\ 1, x \geq \frac{1}{a} \end{cases}$$

donde a es la pendiente, x es la entrada de la neurona y b es el umbral. Variando la pendiente se puede conseguir una mejora en la adaptación de las salidas de la red neuronal.

Los valores de salida estarán comprendidos en el rango $\left(\frac{-1}{a}, \frac{1}{a}\right)$, por encima o debajo de esa zona se fija la salida a 1 o -1. Cuando a=1 las salida es igual a la entrada.

- **Función tangente hiperbólica:** las redes que usan esta función son aquellas que tienen las salidas continuas, ya que su algoritmo de aprendizaje necesita una función que sea derivable.

$$f(x) = \frac{e^{ax} - e^{-ax}}{e^{ax} + e^{-ax}}$$

donde x es el valor de entrada menos el umbral y a afecta al valor de la pendiente si se varía. Los valores de salda de esta función están comprendido entre -1 y 1.

2.3.1.4. Función de salida

Es el último componente necesario en una neurona y su valor de salida es el que se transmite a las neuronas vinculadas. Si la función de activación está por debajo del umbral determinado, ningún valor de salida se pasa a la neurona siguiente. Esto es debido a que no todos los valores de entrada son válidos para una neurona, por lo que los valores de salida están comprendidos entre unos rangos [0, 1] o [-1, 1] y también pueden ser binarios 0, 1 o -1, 1.

Las dos funciones de salida más comunes son las siguientes:

- **Identidad:** es la función más sencilla posible, ya que la salida es la misma que la entrada.
- **Binaria:** es el caso en el cual si el valor de la función de activación es mayor que el umbral, este se pasa a la siguiente neurona, en caso contrario no.

2.3.2. Elementos básicos de una red neuronal

Una red neuronal está formada por diferentes capas o niveles, los cuales están formados por neuronas artificiales agrupadas. Las neuronas de cada capa están conectadas con neuronas de la capa anterior y de la posterior. Cada neurona obtiene la información. A partir de la situación que ocupen en la red, se pueden distinguir tres tipos de capas:

- **De entrada:** es la primera capa y la que recibe de forma directa la información que proviene de fuentes externas a la red. Las neuronas de esta capa son pasivas, es decir, no modifican los datos que reciben. A las neuronas de esta capa no se les asigna ningún peso y su función de activación es lineal.
- **Ocultas:** son las capas internas a la red, que se encuentran entre la capa de entrada y la de salida y que no tienen un contacto directo con el exterior. El número de capas ocultas puede ser muy variado, desde ninguna hasta un número elevado. Cuando la red está formada por dos o más capas ocultas, esta es considerada como una red neuronal profunda.

Las neuronas de estas capas pueden estar interconectadas de diferentes maneras, lo que nos lleva, junto con su número, a tener una gran variedad de topologías de redes neuronales.

El diseño de estas capas ocultas, tiene una gran importancia en el proceso de aprendizaje, ya que determinan la complejidad del sistema. Y la forma de encontrar el diseño correcto es variando los parámetros de las capas ocultas.

- **De salida:** es la capa encargada de transferir la información hacia la red exterior.

2.3.3. Topologías de las redes neuronales

La topología o arquitectura de una red neuronal viene dada por la forma en la que las neuronas se organizan y disponen dentro de la misma red, formando capas o grupos de neuronas en las capas intermedias (ocultas). Para las diferentes tipologías los parámetros importantes son: el número de capas, el de neuronas, el grado de conectividad y el tipo de conexiones entre neuronas. A continuación se detallan estas topologías:

- **Redes monocapa:** estas redes se caracterizan por tener únicamente una capa, en la cual se establecen las conexiones entre sus neuronas. Se suelen usar para regenerar la información incompleta o distorsionada que les llega a la entrada.

- **Redes multicapa:** son redes que disponen de un conjunto de neuronas que se agrupan en varios niveles o capas. Para poder distinguir a la capa que pertenece cada neurona, hay que fijarse en el origen de las señales que recibe de entrada y el destino de la señal de salida. Podemos diferenciar dos tipos de redes multicapa:

  1. **Conexiones hacia adelante o feedforward:** son aquellas en las que la señal que reciben de entrada proviene de la salida de una capa anterior y después envían su salida a una capa posterior.
  2. **Conexiones hacia atrás o feedback:** son aquellas en las que se conecta la salida de las neuronas de capas posteriores a la entrada de las capas anteriores.

- **Redes de propagación hacia atrás (backpropagation)**

  El nombre de este tipo de red viene dado por la forma en la que el error se propaga hacia atrás en la red neuronal desde la capa de salida. Esto permite que los pesos dados a las diferentes conexiones entre las neuronas que están en las capas ocultas puedan variar durante el entrenamiento.

  Este cambio en los pesos influye en la entrada, en la función de activación y por lo tanto en la salida de cada neurona. En este caso se deben mirar los cambios que producen en la función de activación un cambio en los pesos. Esto se conoce como sensibilidad de la función de activación [5].

  Las redes de propagación hacia atrás que tienen lazos cerrados son llamados sistemas recurrentes.

  A continuación se presenta la topología de las ANNs:

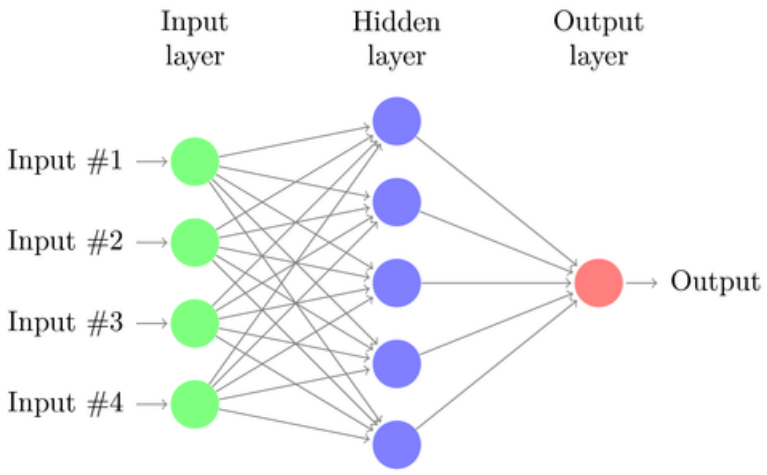

Figura 2.2: Topología red neuronal artificial

2.3.4. Métodos de aprendizaje

Como se ha visto anteriormente, los datos de entrada a la red neuronal son procesados para lograr una salida. Por lo tanto, una red neuronal debe aprender a conseguir una salida correcta para cada vector de entrada. Es lo que se conoce como proceso de entrenamiento y el conjunto de datos sobre el cual se realiza se denominan datos de entrenamiento.

Durante el aprendizaje, tanto la topología de red como las diferentes funciones de cada neurona (entrada, activación y salida) no pueden variar. La única forma de poder adaptarse a la información de entrada, es realizando una modificación de los pesos sobre cada una de las conexiones, que se conoce con el nombre de adaptación de los pesos. Es decir, el aprendizaje es el proceso por el cual una red neuronal modifica sus pesos en función de la entrada. En realidad lo que se produce durante este proceso es la destrucción, modificación y creación de las conexiones existentes entre neuronas. Al crearse una nueva conexión, el peso de la misma pasa a tener un valor distinto de cero, si se destruyera, pasaría a tener valor cero. Una vez que estos pesos permanecen estables, significa que la red ha aprendido.

Lo siguiente es conocer cómo se modifican los valores de los pesos y cuáles son los criterios que siguen para asignar esos valores a nuevas conexiones, a la hora de que la red aprenda nueva información. Hay dos métodos de aprendizaje:

- **Aprendizaje Supervisado:** se caracteriza por ser un proceso de aprendizaje en el cual el entrenamiento es controlado por un agente externo (supervisor, maestro) que determina la respuesta que debería tener la red en función de una entrada determinada. El agente externo controla la salida y si no coincide con el valor deseado, se procede a modificar los pesos de las conexiones, para que se aproxime a la salida deseada. Hay tres formas de llevarlo a cabo: aprendizaje por corrección de error, aprendizaje por refuerzo y aprendizaje estocástico.

- **Aprendizaje no supervisado:** también conocido como autosupervisado, no requiere un agente externo para ajustar los pesos, la red no recibe ninguna información que le

indique si la salida generada a una entrada es la correcta o no. Estas redes deben encontrar las correlaciones necesarias entre los datos de entrada y hay varias interpretaciones de la salida en función de su estructura y aprendizaje empleado.

A veces, la salida representa la similitud entre la información de entrada y las que se le habían mostrado hasta entonces. Y otras, se realiza una clusterización donde la red indica, a la salida, a qué categoría pertenece la información dada a la entrada, siendo la red la que debe encontrar estas correlaciones. Hay dos tipos:

- Aprendizaje hebbiano: mide el parecido o extrae características de los datos de entrada. El fundamento es que si tenemos dos neuronas con distinto peso y toman el mismo estado, el peso de la conexión entre ellas aumenta.
- Aprendizaje competitivo y comparativo: está orientado a la clusterización o clasificación de los datos de entrada. Si un patrón nuevo pertenece a una clase existente previamente se incluye en él, si no pertenece a ninguna de las clases reconocidas, entonces la red neuronal se estructura y los pesos cambian para reconocer la nueva clase.

Hay otro criterio para distinguir el método de aprendizaje que consiste en considerar si la red puede aprender mientras está funcionando, que se trataría de un aprendizaje on-line o aprendizaje off-line si para el aprendizaje se necesita una desconexión de la red hasta que el proceso haya finalizado.

2.3.5. Tipos de Redes Neuronales Artificiales

Las ANNs se pueden clasificar en diferentes grupos dependiendo de su topología y los algoritmos de aprendizaje. El tipo de red neuronal elegida influye en el rendimiento del sistema en determinadas tareas. A continuación se describen los más utilizados [6]:

- **Feed-Forward Neural Network (FFNN):** las neuronas de esta red se disponen en capas. La primera capa recibe las entradas y la última genera la salida. Además cuenta con unas capas intermedias ocultas. La información se propaga hacia adelante, es decir, de la capa de entrada a la de salida.

  Pueden tener una estructura totalmente conectada, en la cual cada neurona tiene conexión con cada neurona de la capa anterior y con cada una de la capa siguiente. Si no tiene estructura totalmente conectada, se considera como si alguna de las conexiones tuvieran peso 0.

  Utilizan un aprendizaje supervisado y se suelen utilizar para tareas de clasificación. Ejemplo de una FFNN totalmente conectada:

- **Radial Basis Function Network (RBF):** tienen un enfoque parecido a la regla de los K-vecinos más cercanos, que es un clasificador supervisado basado en reconocimiento de patrones con criterios de vecindad (cercanía). Se calcula la distancia Euclídea entre las entradas y los datos y ese valor se da como entrada a una función no lineal (normalmente gaussiana):

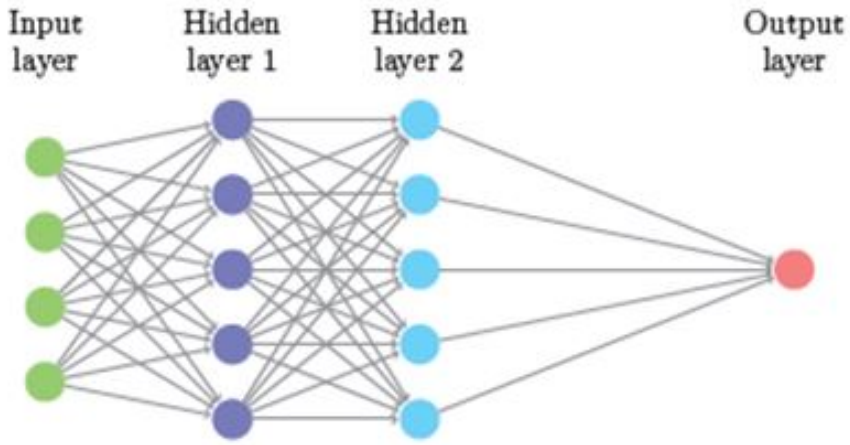

Figura 2.3: Ejemplo de Feed-Forward Neural Network (FFNN)

$$f(x) = \lambda_0 + \sum_{i=1}^{N_c} \lambda_i \phi(\|x - C_i\|) \tag{2.4}$$

donde $\|\|$ representa la norma Euclídea, C_i son los centros de datos y N_c es el número de centros de datos.

- **Self-Organizing Map (SOM):** utiliza un aprendizaje no supervisado para sintonizar las neuronas y los pesos. Se utilizan para convertir las entradas de altas dimensiones en bajas dimensiones. Son útiles en aplicaciones de clúster.

  Son redes que no tienen capas ocultas, dependiendo de las entradas y los pesos se calcula la salida de cada neurona y a menudo se elige la función de la distancia Euclídea.

- **Convolutional Neural Network (CNN):** se compone de una o más capas convolucionales, en las cuales cada neurona de una capa no recibe conexiones entrantes de todas las neuronas de la capa anterior, sino solo de algunas. Esto favorece que una neurona se centre en una región de la capa anterior y disminuyan tanto los pesos como las multiplicaciones necesarias. Lo más habitual es que dos neuronas consecutivas se especialicen en regiones solapadas de la capa anterior.

  A continuación les siguen una o más capas totalmente conectadas, como en una red multicapa estándar.

  Son fáciles de entrenar y tienen muchos menos parámetros que las redes totalmente conectadas con el mismo número de unidades ocultas. Realizan propagación hacia atrás [7].

- **Recurrent Neural Network (RNN):** son redes que permiten que sus neuronas compartan sus salidas con las neuronas de la capa anterior, creando ciclos de realimentación. Esto implica que las neuronas pueden mantener su estado de activación temporal, incluso sin una entrada. Por lo tanto, son sistemas dinámicos con memoria dinámica. Esto tiene sus ventajas, en cuanto a modelar de forma más fiable la activación del cerebro humano, sin embargo implica una mayor complejidad en su estructura.

Son utilizadas en neurociencia computacional, en el aprendizaje automático y en las aplicaciones de procesado de señal no lineales. A continuación se muestra su topología:

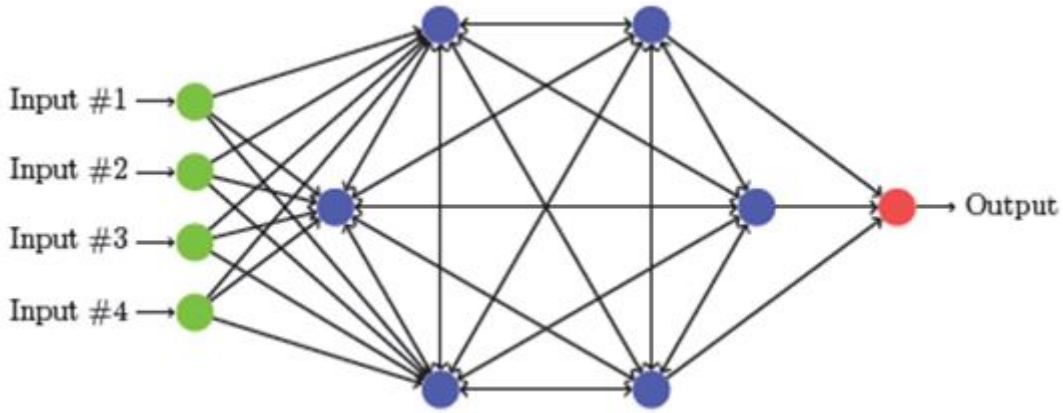

Figura 2.4: Ejemplo de Recurrent Neural Network (RNN)

Dentro de este tipo de red neuronal se encuentran las redes LSTM (Long Short-Term Memory), que serán las usadas en este proyecto y definidas en el capítulo 6.

2.3.6. Ventajas de una red neuronal

Las redes neuronales artificiales presentan un elevado número de características semejantes a las del cerebro. Esto hace que aporten numerosas ventajas y que se aplique esta tecnología a múltiples tareas. Entre ellas destacan [5]:

- **Aprendizaje adaptativo:** es la capacidad de aprender a realizar tareas basadas en un entrenamiento o experiencia inicial. Una red neuronal no precisa de un algoritmo para resolver un problema, ya que ella puede generar su propia distribución de los pesos en los enlaces mediante el aprendizaje.

  Es necesario que desarrolle un buen algoritmo de aprendizaje que proporcione la red la capacidad de discriminar, mediante un entrenamiento con patrones.

- **Auto-organización:** la red neuronal puede crear su propia organización a través de la información que recibe de la etapa de aprendizaje. Durante el entrenamiento se estiman unos pesos que determinan esta organización de la red. Esto provoca la generalización de las redes, para que puedan responder bien cuando se les presentan datos o situaciones a las que no se habían sometido antes.

- **Tolerancia a fallos:** normalmente la destrucción de una parte de la red lleva a una degradación en la estructura de esta, sin embargo algunas redes tienen la capacidad de retener algunas capacidades. La razón por la que son tolerantes a fallos es que tienen su información distribuida en las conexiones entre neuronas, existiendo cierto grado de redundancia en el almacenamiento.

- **Operación en tiempo real:** Una de las prioridades es la necesidad de realizar procesos con datos de forma muy rápida. La computación de las redes neuronales se puede

realizar en paralelo, para ello se diseñan máquinas con hardware especializado para conseguirlo, como pueden ser las tarjetas gráficas.

- **Fácil integración:** existen chips especializados para redes neuronales que mejoran su capacidad y facilita la integración modular en los sistemas ya existentes. Así se pueden usar para mejorar sistemas y pueden ser evaluados por separado antes de ser utilizadas en un desarrollo más grande.

2.4. Arquitectura de un reconocedor automático de habla

Los sistemas de reconocimiento automático de habla (RAH) han sido enfocados desde diferentes perspectivas. Los que mejores resultados han dado son aquellos que emplean la "Teoría de la Decisión de Bayes", la "Teoría de la Información" y las "Técnicas de Comparación de Patrones".

Este tipo de sistemas tiene como finalidad extraer en formato texto, de la información acústica que se encuentra en la señal de voz, una representación del conjunto de sonidos que han sido pronunciados.

Para poder llevar a cabo esta decodificación, hay diferentes técnicas que parten de un conjunto de patrones semejantes al del mensaje de entrada, y devuelven una secuencia con los patrones que tienen una probabilidad mayor de representar al mensaje.

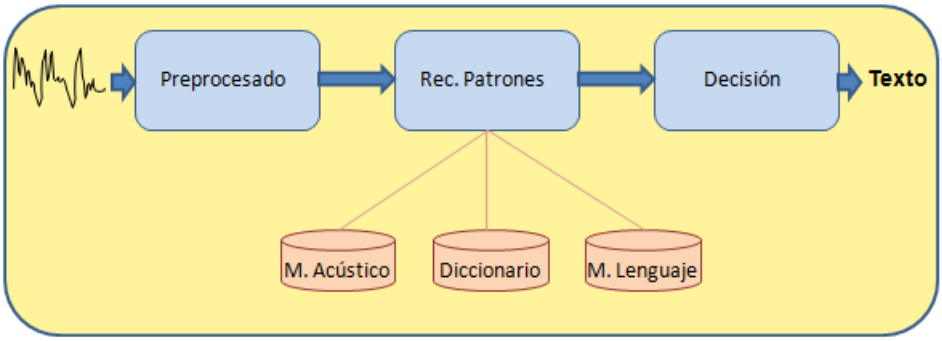

Figura 2.5: Esquema básico de la arquitectura de un RAH.

Después de dar una visión general de los sistemas de reconocimiento, se va a centrar la atención de este apartado, en los sistemas de reconocimiento de habla continua y de vocabulario extenso (Large Vocabulary Continuous Speech Recognition - LVCSR), y su arquitectura está compuesta por los siguientes bloques:

2.4.1. Preprocesado

En este bloque se agrupa la extracción de características y transformación, procesado de características de robustez al ruido y la estimación de características adaptativas y discriminativas [8].

Extracción de características: la función de este módulo es la extracción de una secuencia de vectores de características acústicas, a partir de la señal de voz. Esto se realiza gracias

a la transformada rápida de Fourier (Fast Fourier Transform - FFT) de la señal de voz en una ventana de análisis, la que se desplaza un intervalo de tiempo fijo.

Las energías de las frecuencias vecinas, dentro de cada trama, se descartan por medio de un banco de filtros en la escala Mel, estando sus características inspiradas en el proceso auditivo humano. A la salida de los filtros, se aplica un logaritmo y los coeficientes son decorrelados mediante la transformada discreta del coseno, dando lugar a un vector de coeficientes cepstrales de frecuencia Mel (Mel Frequency Cepstral Coeficiente - MFCC).

A continuación, estos coeficientes se reemplazan obteniendo una mayor robustez frente al ruido, basado en coeficientes perceptibles de predicción lineal (Perceptual Linear Prediction - PLP).

En este ámbito, hay dos técnicas importantes que han aprovechado la extracción de características: la primera, es el uso de la media basada en el locutor, y la normalización de la varianza de los coeficientes cepstrales. La sustracción de la media cepstral basada en la pronunciación (Cepstral Mean Substraction - CMS) es una técnica muy conocida, la normalización de la varianza cepstral (Cepstral Variance Normalization - CVN) se ha empezado a utilizar recientemente. La segunda, consiste en la introducción de contexto temporal entre las tramas, calculando los coeficientes dinámicos o de velocidad y aceleración, también llamados coeficientes delta o delta-delta respectivamente). Estos se calculan a partir de las tramas que están próximas dentro de una ventana de unas 4 tramas de media. Para formar el vector final se añaden los coeficientes dinámicos a los estáticos.

Características robustas al ruido: el ruido ambiente siempre degrada la calidad de la señal de voz, y que después afecta de forma negativa al reconocimiento. Por ello hay que tratar este efecto en la etapa de preprocesado. El algoritmo SPLICE ("Stereo-based piecewise linear compensation for enviroments") [9] propuesto para lugares con ruido no estacionario. Se utiliza para eliminar el ruido a través de la diferencia entre la voz limpia y la voz corrupta, en la región con más probabilidad del espacio acústico. Otro algoritmo QE (Quantile-based histogram equalization) se desarrolló para compensar la desalineación de los datos de entrenamiento y de test. Ambos se evaluaron con un corpus de The Wall Street Journal [8], modificando el tipo y los niveles de ruido, y se comprobó que mejoraba en entornos limpios y multicondición.

Estimación de características adaptativas y discriminativas: la variación en las características acústicas puede ser entre locutores diferentes o en un mismo locutor. Para ello, existen técnicas que eliminan esta variabilidad dentro de lo posible y algunas de ellas son las siguientes:

- VTLN (Vocal Tract Length Normalization), normalización de la longitud del tracto vocal. Transforma las características maximizando la verosimilitud bajo un modelo.
- fMLLR (feature-space Maximun Likelihood Linear Regression), transformación no lineal de la distribución de los datos que será alineada con una distribución normal de referencia.
- fMPE (feature-space mínimum phone error), al estimar las características discriminativas se usan técnicas que permiten obtener offsets que dependen del tiempo, a partir de una proyección lineal de un espacio de gaussianas.

2.4.1.1. Filtro de pre-énfasis

Como la señal de voz se atenúa al incrementar la frecuencia, es necesario dar mayor importancia a las frecuencias más elevadas. Para ello, mediante un filtro de pre-énfasis, se analiza la señal de voz. Como la señal es digital se suele aplicar un filtro FIR con función de transferencia $\mathrm{H(z)} = 1 - \mathrm{a} * \mathrm{z}^{-1}$ donde a suele valer 0,97. El cero de transmisión, de este filtro, varía según el valor de a, dando lugar a un filtro plano cuando a = 0 o a un cero de transmisión en la frecuencia 0 cuando a = 1.

2.4.1.2. Enventanado

A continuación, se realiza el enventanado de la señal, que consiste en multiplicar las muestras de la señal de entrada por una ventana de longitud finita. Este enventanado se va a realizar con una ventana de tipo Hamming, ya que suele ser la más usada por sus cualidades espectrales.

Las muestras en los extremos sufrirán una ponderación, debido a la forma irregular de la ventana de Hamming. Para compensar este efecto, se solaparán unas ventanas con otras, de forma que se anule. De ese modo, el tamaño de la ventana será de 25 ms con un desplazamiento de 10 ms, que corresponde con un 60 % de solape entre ventanas.

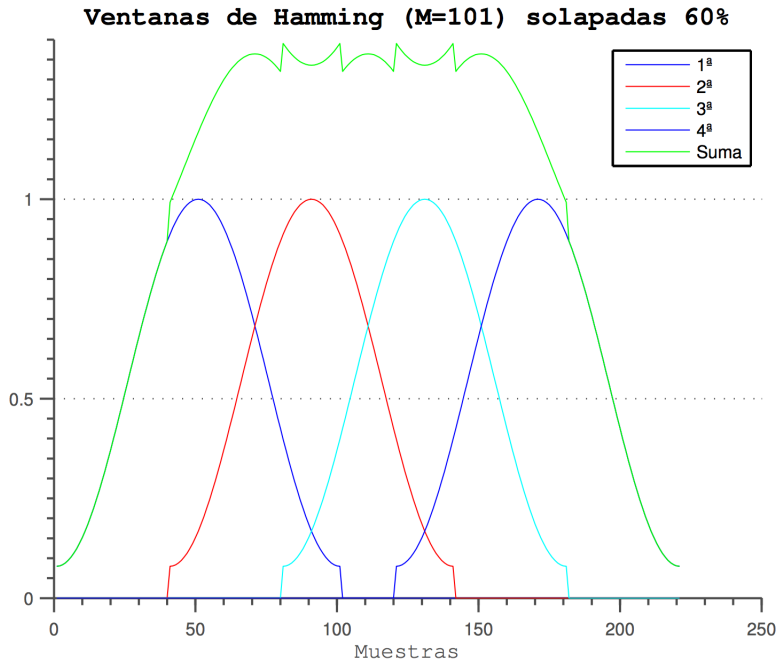

Figura 2.6: Ejemplo de ventanas de Hamming solapadas un 60 %.

2.4.1.3. Transformada discreta de Fourier

Si consideramos un sistema lineal e invariante en el tiempo, algunas de sus propiedades se pueden representar en el dominio de la frecuencia. Una de ellas es que la respuesta a señales

sinusoidales es otra señal sinusoidal o una combinación de exponenciales complejas, lo que las hacen muy útiles cuando se trata con señales de voz. Están basadas en secuencias base de exponenciales complejas.

En señales discretas de duración finita se usa la Transformada Discreta de Fourier (DFT).

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi kn}{N}}$$

Aunque en la práctica se utiliza el algoritmo de la FFT, para realizar la transformada discreta de Fourier y su inversa con un coste computacional mucho menor.

La salida de la transformada de Fourier se filtra con un banco de filtros en la escala Mel (Figura 2.3), estando sus características inspiradas en el proceso auditivo humano.

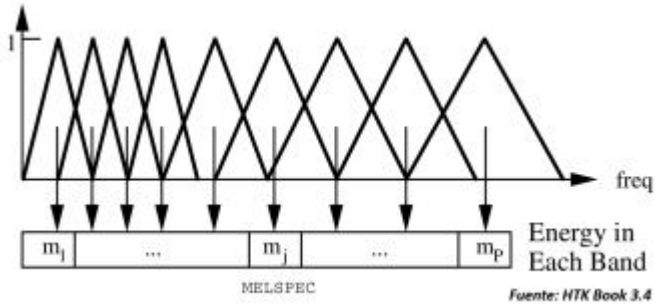

Figura 2.7: Mel-Scale Filter Bank.

2.4.1.4. Transformada discreta del coseno

A la salida de los filtros se aplica un logaritmo en el cual los coeficientes son decorrelados a partir de la transformada discreta del coseno. La transformada discreta del coseno (DCT) y la transformada discreta de Fourier son muy similares entre sí, excepto en que las secuencias base de la DFT son exponenciales complejas y en la DCT son funciones cosenoidales.

$$X(k) = \sum_{n=0}^{N-1} x(n) cos\left[\frac{k\pi}{N}\left(n + \frac{1}{2}\right)\right]$$

Como la mayor parte de la energía resultante de este tipo de transformada se aglutina en los coeficientes de baja frecuencia, este tipo de transformada es muy útil en aplicaciones de compresión de datos.

2.4.1.5. El dominio cepstral

En este caso, se calculan 13 coeficientes cepstrales de frecuencia Mel (Mel Frequency Cepstral Coeficiente - MFCC). Para ello, se utiliza un filtro de decorrelación homomórfica

(Cepstrum) que, mediante la transformada inversa de Fourier, lleva los coeficientes espectrales al dominio de la cuefrencia obteniendo así coeficientes cepstrales [8].

Debido a este filtrado se pueden decorrelar los espectros de los filtros en bandas adyacentes, evitando la limitación de las técnicas de análisis espectral que operan en el dominio de la potencia espectral logarítmica.

De esta forma, los coeficientes cepstrales representan la señal temporal, que corresponde con el espectro logarítmico de la potencia.

Como el dominio de la cuefrencia es un dominio homomórfico del dominio temporal, las convoluciones en el dominio temporal se convierten en sumas en el dominio de la cuefrencia, esto permite separar las señales de voz de los ruidos convolucionales con los que están mezcladas.

Así, las componentes de excitación y la envolvente espectral del tracto vocal estarán en zonas separadas del dominio de la cuefrencia, permitiendo separarlo mediante ventanas, lo que se llama liftering.

En el dominio de la frecuencia hay otros métodos de extracción de características, a parte de los MFCC (Mel Frecuency Cepstral Coeficients), como son LPC (Linear Predictive Coding), y Cepstrum PLP entre otros [10].

2.4.2. Reconocimiento de patrones

Es la comparación de los modelos acústicos con los parámetros de la señal de entrada que se quiere clasificar y que se encuentra representada en el mismo dominio que la muestra. Estos modelos representan el comportamiento de los parámetros para cada uno de los sonidos del idioma que se está reconociendo.

Esta técnica sirve de referencia para realizar comparaciones con alto grado de confianza; para ello, se necesita un conjunto de muestras etiquetadas y una metodología de entrenamiento [11].

2.4.2.1. Etapas

Al representar los patrones, estos puede ser una plantilla (template) o un modelo estadístico (HMM, DNN), y se podrá aplicar a un sonido, una palabra o una frase. Esta técnica puede dividirse en dos etapas: entrenamiento y comparación.

- **Entrenamiento:** en esta etapa se construye un patrón de referencia asociado a cada palabra o sub-unidad de palabra que se quiere reconocer, basándose en los vectores de características de las unidades empleadas en el proceso de entrenamiento. Hay varias formas de llevar a cabo este proceso:
    - Entrenamiento casual: se asigna un único patrón de sonido para generar el patrón de referencia o un modelo estadístico aproximado.
    - Entrenamiento robusto: con varias versiones de un mismo locutor, se genera un patrón de referencia promedio o modelo estadístico promedio.

    * Entrenamiento por clustering: con mucho volumen de datos y varias versiones de muchos locutores, se construyen patrones de referencia o modelos estadísticos con alto grado de confianza.

- **Comparación:** en esta etapa se realiza una comparación del vector característico asociado a la señal de voz (a reconocer) y todos los posibles patrones entrenados, para determinar el mejor ajuste respecto a un criterio establecido. Se definirá una distancia entre vectores característicos a partir del que se obtendrá el patrón de referencia mejor ajustado a la señal a reconocer.

2.4.2.2. Modelo Acústico

Es uno de los elementos más importantes en el reconocimiento de patrones. Se trata de un conjunto de representaciones estadísticas de los diferentes sonidos del espacio acústico con el que se está trabajando. Se elabora a partir de un volumen de datos de entrenamiento, formados por datos de voz con su correspondiente etiquetado (transcripciones), permitiendo una asignación de cada sonido a su representación o carácter gráfico.

A continuación, se van a mencionar las técnicas más importantes empleadas para generar estos modelos:

Dynamic Time Warping (DTW)

Es una técnica que ha sido muy usada a lo largo de los años en reconocimiento de voz. Se basa en la realización del alineamiento temporal de una locución de test y los parámetros del patrón. Como resultado de la alineación de ambas locuciones, se consigue la función de menor coste. Existen una gran cantidad de caminos de alineamiento, que son reducidos por un conjunto de límites locales y una serie de limitaciones [12].

IMAGEN

Vectorial Quantization (VQ)

Consiste en representar las características de los fonemas como un espacio vectorial, que está formado por un conjunto infinito de vectores posibles (espacio de características). En este espacio, se asigna un conjunto de patrones desconocidos (test) a un conjunto finito de patrones de referencia; de forma que al vector a reconocer, se le asigna un vector patrón cuya distancia a él sea mínima.

El espacio queda dividido en zonas o regiones, y al vector representativo de esa región se le denominará “codeword” (centroide). Los vectores que caigan en esa región, se asignarán a dicho centroide. El conjunto de todos los centroides se denomina “codebook” (muestrario).

Cada región se puede corresponder con cada uno de los fonemas, los centroides se asignan de forma aleatoria y los vectores de test son asignados al centroide más cercano.

Hidden Markov Model (HMM)

Son modelos estadísticos que sirven para representar secuencias de datos espaciales o temporales como la señal de voz. Estos modelos sustituyeron desde los años 80

a las técnicas de comparación de patrones como los DTW, que modelaban la voz de forma determinista y son la base tecnológica de los sistemas de reconocimiento de voz.

El sistema a modelar es un proceso de Markov de parámetros desconocidos, los cuales se calculan a partir de los parámetros observables. Este procedimiento ha sido uno de los empleados en este proyecto, por ello, se dará una explicación más amplia en la sección 2.5.

Deep Neural Networks (DNN)

Las redes neuronales profundas [13] son una forma alternativa de aprendizaje y de procesamiento automático, que intenta simular el funcionamiento del cerebro. Para ello utiliza una red neuronal basada en feed-forward, la cual toma varias tramas de coeficientes como entrada y como salida genera probabilidades sobre los estados de HMM. Estas redes tienen un gran número de capas ocultas y son entrenadas usando nuevos métodos que mejoran los mencionados anteriormente.

La conexión entre capas adyacentes se consigue con la asignación de unos pesos iniciales de baja magnitud y aleatorios. Así se evita que todas las unidades ocultas en una capa tengan exactamente los mismos valores en cuanto a los pesos.

Los modelos más flexibles, son aquellos con un gran número de capas y de elementos en cada capa, que son capaces de modelar relaciones altamente no lineales entre entradas y salidas.

Este procedimiento ha sido uno de los empleados en este proyecto, por ello, se dará una explicación más amplia en la sección 2.6.

2.4.2.3. Diccionario

El diccionario, también llamado lexicon, es el nexo entre el nivel acústico y la secuencia de palabras que sale del reconocedor.

Está formado tanto por las palabras conocidas por el sistema, como por sus transcripciones fonéticas, que sirven para la construcción de los modelos acústicos para cada entrada. Para LVCSR, generalmente el vocabulario se elige de tal forma que se maximice la cobertura para un tamaño de diccionario dado, pudiendo contener palabras iguales con más de una pronunciación.

La generación del diccionario se puede ver influida, también, por aspectos como el tipo de habla, leída o espontánea [14], y es recomendable tratar esta diferencia en las pronunciaciones, para obtener el mayor rendimiento posible del sistema.

2.4.2.4. Modelo de lenguaje

Permite definir una estructura del lenguaje, es decir, restringir correctamente las secuencias de las unidades lingüísticas que sean más probables. Se usan en sistemas con sintaxis y semántica compleja. Su función consiste en aceptar (con alta probabilidad) frases correctas y rechazar (o asignar baja probabilidad) secuencias

de palabras incorrectas. Se pueden tener dos tipos de modelos: gramática cerrada de estados finitos y modelo de N-gramas.

- **Gramática cerrada de estados finitos:** presenta restricciones del lenguaje, modelando dependencias tan largas como uno quiera. Su aplicación tiene una gran dificultad si se usan lenguajes próximos a lenguajes naturales.
- **Modelo de N-gramas:** sirven para ayudar a discernir cuál es la palabra correcta cuando el modelo acústico no es suficiente. Se basan en cómo se combinan entre sí las palabras estadísticamente. Si se emplea la información contextual, sobre todo con palabras con la misma pronunciación u homónimas, se mejora la precisión del sistema.

2.4.3. Decisión

Este bloque es uno de los que más relevantes para el diseñador de la arquitectura del sistema de reconocimiento, ya que es la única salida observable por el usuario. Consiste en asignar un patrón de los que han sido generados en la fase de entrenamiento. Para ello, se comparan la realización acústica de entrada y el conjunto de los modelos conocidos por el sistema, y con los valores de similitud que se obtengan, el reconocedor tomará la decisión sobre los sonidos que ha generado la señal de audio de entrada.

El teorema de decisión de Bayes expresa la probabilidad de que los sonidos de entrada o combinaciones de ellos (trifonemas) puedan ser generados por alguno de los estados que modelan todas las posibles unidades del espacio acústico. La decisión se basará en la que tenga una mayor similitud tanto del modelo acústico como del modelo del lenguaje.

3
Base de Datos

El primer paso necesario para la etapa de entrenamiento consiste en la adaptación de la base de datos, que es uno de los elementos más importantes a la hora de generar un sistema de reconocimiento de habla. De ella se extraerá la información necesaria para las siguientes fases del reconocedor. La calidad de dicha base de datos influirá en la consecución de un buen entrenamiento de los modelos acústicos. En este caso se han usado diversas bases de datos cedidas por telefónica I+D con las que se ha formado la base de datos final.

3.1. Preparación de los datos

La base de datos está formada por una lista de archivos de audio (.wav) y otra de archivos de etiquetas que tienen el mismo nombre, pero distinta extensión (.info). Estos archivos de etiqueta contienen dos partes:

- La transcripción del archivo de audio con extensión.
- Información adicional, en ficheros de texto, correspondiente a información obtenida de los audios como puede ser: nombre de la base de datos, identificador del hablante, sexo, idioma, dialecto, SNR del archivo, pitch del hablante, fecha y lugar de grabación, eventos fonéticos (como saturación, mala pronunciación...), tipo de ruido (si lo hubiera).

A partir de estos archivos de etiquetas generales, se puede extraer el campo de transcripción y adaptarlo a lo que necesita cada programa de entrenamiento. Para el software Kaldi, el archivo que contiene las etiquetas debe tener todas las palabras de la transcripción en una misma línea y sin añadir símbolo alguno de final de archivo (.trn).

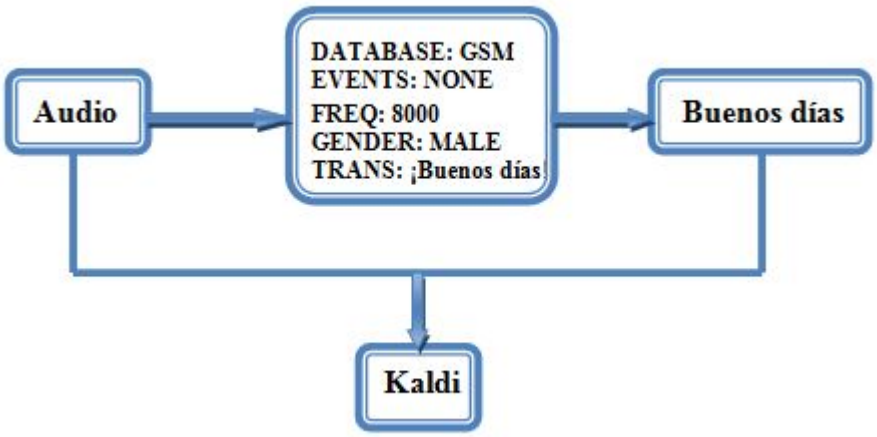

Figura 3.1: Archivos necesarios para HTK.

Para evitar entrenar los fonemas de forma errónea, los archivos deben estar transcritos de manera correcta. Los archivos de audio deben ser fieles a las transcripciones, asegurándose que no están vacíos ni intercambiados.

Como en cualquier proyecto de tratamiento de voz, es necesario usar diferentes partes de la base de datos que realizarán diferentes funciones en la construcción del sistema final. Para ello, los audios se dividen en distintas partes:

- Train: serán usados para el entrenamiento inicial del sistema.
- Test: serán utilizados para evaluar el sistema que fue entrenado previamente.

A continuación se muestra un esquema de esta fase:

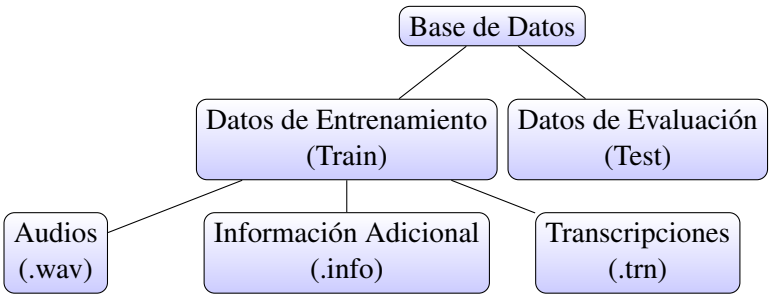

Figura 3.2: Diagrama de entrenamiento

La riqueza de la base de datos se puede medir por diversos factores:

- La cantidad de archivos que la forman.
- El número de frases y palabras de entrenamiento que son distintas.
- El número total de palabras y caracteres.
- El promedio de caracteres por palabra.
- El número total de horas de audio y cuántas de ellas son de silencio y voz.
- La cantidad de frases de hombres y mujeres.
- El número de hablantes distintos.
- La SNR promedio.
- El pitch promedio.

La base de datos de entrenamiento usada para este proyecto, fue generada, anteriormente a este proyecto, a partir de bases de datos más pequeñas que se limpiaron y unieron, eliminando archivos que tenían demasiado ruido, estaban incompletos, con SNR muy baja, etc. Además, se redujo la tasa binaria de los archivos de audio con velocidad superior, para ajustarlos a la calidad telefónica.

Para evaluar esta base de datos, se extrajeron 1000 archivos del total para hacer pruebas de reconocimiento con archivos no entrenados que tuvieran una SNR promedio de la base de datos. También se extrajeron otros 1000 archivos con ruido, para hacer pruebas con archivos ruidosos no entrenados, teniendo cada conjunto un total de 70 minutos de audio.

Las bases de datos usadas tienen diferentes características, siendo aproximadamente el 60 % voz telefónica (tanto GSM como línea fija) grabada en diferentes ambientes (calle, bares, coches, hogares...) y un 40 % voz limpia grabada en estudio.

La base de datos se encuentra en idioma español de España y contiene muestras de casi todas las comunidades autónomas con los diferentes acentos. Además está en formato WAV, a 8 kHz, 16 bits y 1 canal.

En la siguiente tabla se muestran el resto de parámetros de la base de datos usada para el entrenamiento:

3.2. Conjuntos de archivos de test

En esta sección se muestran las características de los diferentes conjuntos de test utilizados para los experimentos que se muestran en las siguientes secciones.

3.2.1. Ficheros de test y auto-test

Para evaluar el sistema base de reconocimiento, se han generado 6 grupos distintos de archivos. Los 4 primeros son de Auto-test, es decir, se prueba a reconocer archivos que se han empleado para entrenar los modelos acústicos.

Características	Valor
Nº Archivos / Frases Totales	244.132
Nº Frases distintas	19.997
Nº Palabras Totales	1.063.866
Nº Letras sin espacios	5.746.114
Promedio letras por palabra	5,40
Nº Palabras distintas	11.182
Nº Horas de audio	275,64
Nº Horas sin silencios	124,95
Silencio (%)	54,67
Voz (%)	45,33
Frases hombres	141.391
Frases mujeres	102.700
Hombres (%)	57,92
Mujeres (%)	42,08
Hablantes distintos	32.309
Velocidad de muestreo (Hz)	8.000
Bits por muestra	16
Canales de audio	1
SNR promedio	33,7
Pitch Promedio Hombres (Hz)	123
Pitch Promedio Mujeres (Hz)	202

Cuadro 3.1: Características de la base de datos de español

Los otros dos grupos son de archivos que no forman parte de la base de datos de entrenamiento. La segmentación se ha hecho entorno a la SNR de los archivos para ver de paso cómo afecta esta a la calidad del reconocimiento. Además, todas las listas de test están balanceadas, es decir, que contienen un número proporcional de los archivos de cada base de datos de origen. A continuación se muestra una tabla con los valores:

Grupo	Lista de Archivos	Nº Archivos	Duración (min)	Rango SNR (dB)
Auto-test	Autotest-10-20	1000	70	10 - 20
	Autotest-20-30	1000	70	20 - 30
	Autotest-30-40	1000	70	30 - 40
	Autotest-40-99	1000	70	40 - 99
Test	Test-30-40	1000	70	30 - 40
	Test-ruidoso	1000	70	<10

Cuadro 3.2: Grupos de archivos de pruebas

3.2.2. Ficheros de pitch

Se trata de una base de datos cuyos ficheros se encuentran segmentados por la frecuencia fundamental de la voz. Se han dividido en 5 grupos, cuyas frecuencias centrales son: 50, 120, 160, 200,

Frecuencia (Hz)	0–100 Hz	100–140 Hz	140–180 Hz	180–220 Hz	> 220Hz
Nº Archivos	250	250	250	250	250

Cuadro 3.3: Detalle de la base de datos de pitch

3.2.3. Ficheros de vídeos

Esta base de datos está compuesta por diferentes contenidos audiovisuales sacados de internet. Los 3 primeros tienen contenidos concretos de tres tópicos actuales y diferentes: economía, moda y deporte. El último engloba contenidos de informativos de televisión, en los cuales se pueden tratar los tres temas anteriores o incluso alguno más, ya que no tienen una temática cerrada.

Tema	Locutores	Voz	Dist.micro	Música	Nº Vídeos	Duración
Economía	Uno principal	Masc.	Cerca	No	3	70 min.
Moda	Uno/vídeo	Fem.	Lejos	Si	5	70 min.
Deporte	Múltiples	Ambos	Cerca	Si	8	70 min.
Telediarios	Múltiples	Ambos	Cerca	Si	3	180 min.

Cuadro 3.4: Detalle de la base de datos de vídeos

Otros factores relevantes son:

- La distribución de SNRs, para poder caracterizar mejor la calidad de los archivos de la base de datos.
- La distribución del pitch de los hablantes, que da información sobre las frecuencias fundamentales de la voz. Es un factor más relevante que solo diferenciar entre hombre y mujeres.

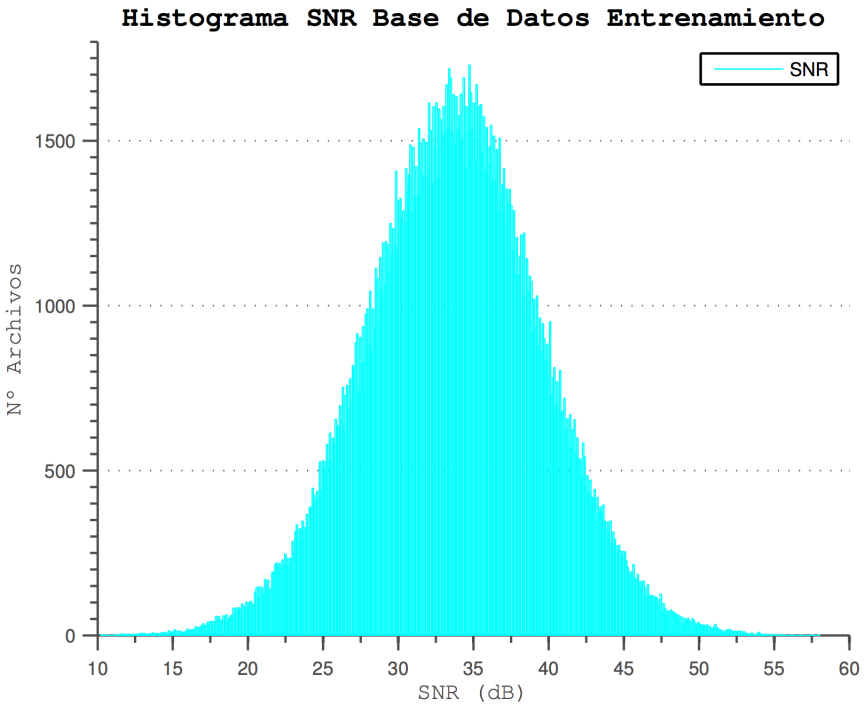

Figura 3.3: Histograma de la distribución de la SNR de la base de datos.

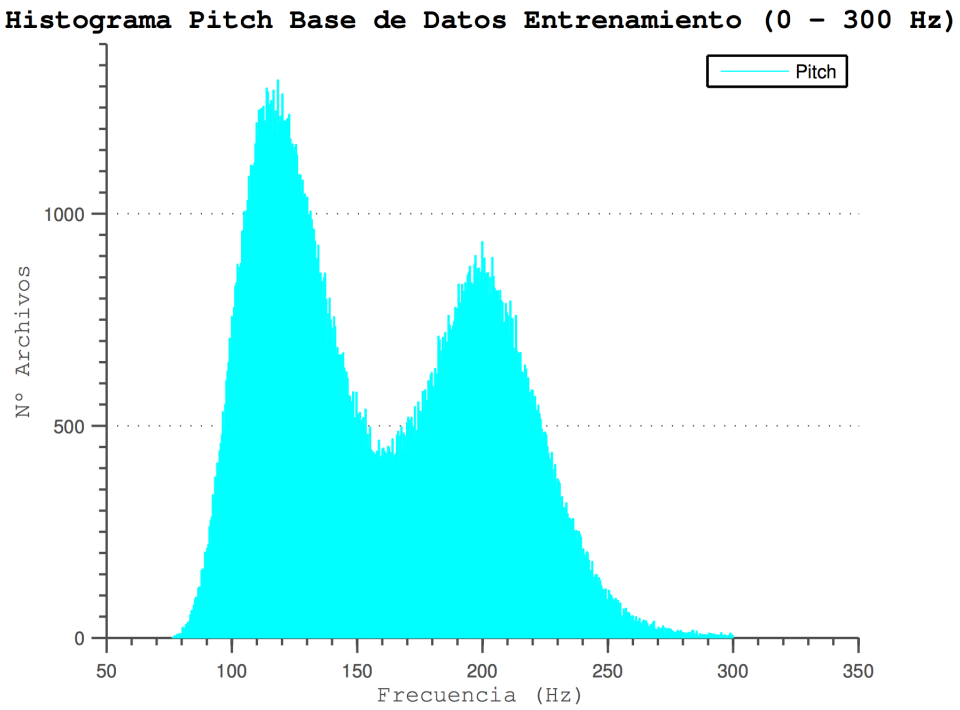

Figura 3.4: Histograma de la distribución del pitch promedio de la base de datos.

4

Generación del Sistema Base Kaldi y Comparativa con HTK

En este capítulo se detallarán los pasos para el desarrollo del sistema base que se utilizará y en el que se aplicarán una serie de mejoras en capítulos posteriores.

4.1. Introducción

El proyecto está formado principalmente por tres partes:

1. La generación de un sistema base de reconocimiento de habla continua basado en el software libre Kaldi con Modelos Ocultos de Markov y la comparativa del mismo con el sistema generado en HTK-Julius en proyectos finales de carrera anteriores de este mismo grupo [2] [3].

   Con el fin de hacer una comparativa lo más justa posible, se han tomado los mismos parámetros de configuración de HTK y se han introducido en Kaldi.

2. La segunda consiste en la mejora del sistema base anterior, utilizando diferentes métodos de entrenamiento mediante la aplicación de diferentes transformadas (LDA+MLLT, MMI, bMMI, MPE).

3. Por último, se implementa este sistema utilizando redes neuronales profundas. En concreto, se han utilizado Redes Neuronales Recurrentes (RNN) del tipo Long Short-Term Memory (LSTM), dado que tras un estudio inicial de la bibliografía existente, se observó que estas eran las que mejores resultados proporcionaban, junto a las BLSTM, que es una variante bidireccional de las mismas, aunque con mayor coste computacional [15].

4.1.1. Software utilizado

- **Linux:** se ha decidido usar un sistema operativo Ubuntu-Linux, debido a que el software principal de este proyecto, Kaldi, ha sido desarrollado principalmente para él. Además, goza de una mayor facilidad a la hora de compilar y ejecutar el mismo.

  También se han utilizado los lenguajes "Shell Script (Bash)" y Python para realizar varias de las tareas, como las diferentes llamadas a herramientas de Kaldi y tratamiento de archivos.

- **Kaldi:** es un conjunto de herramientas para reconocimiento de voz, escrita en C++ y bajo la licencia de APACHE v2.0. Su uso está dirigido para la investigación en el ámbito del reconocimiento de voz [1]. Es un software libre y de código abierto, que está continuamente en desarrollo.

  Esta herramienta cuenta con una librería externa OpenFST, referente a transductores de estados finitos los cuales sirven para modelar las transiciones entre los diferentes estados.

  LOGOTIPO

  Comenzó su desarrollo en el año 2009 en la Universidad Johns Hopkins situada en Baltimore, Maryland. Este software, cuando empezó a desarrollarse, era dependiente de otro software llamado HTK (Hidden Markov Model ToolKit). Algunas de los miembros que lo estaban desarrollando, decidieron reunirse en 2010 en la Universidad Tecnológica de Brno en República Checa para crear una receta basada en lo realizado en 2009 y crear un conjunto de herramientas de voz de propósito general. El problema que la receta anterior estaba basada en ambos códigos HTK y Kaldi y no era fácil de encapsular y se decidió hacer una receta general de licencia abierta. Durante ese año se continuó con la implementación del código, pero no fue hasta mayo de 2011 cuanto el código fue libre.

  Desde su lanzamiento, Kaldi ha sido mantenido y desarrollado, en gran parte, por Daniel Povey y Karel Vesely entre muchos otros. Hasta la fecha unas 70 personas han contribuido con código o scripts.

- **SRI Languaje Model:** Stanford Research Institute Language Modelling Toolkit [http://www.speech.sri.com/projects/srilm/] es un conjunto de herramientas para la construcción y aplicación de modelos estadísticos del lenguaje, principalmente para su uso en el reconocimiento de voz y la traducción automática. Está en desarrollo en el laboratorio de Investigación y Tecnología del Habla del SRI desde 1995. SRILM consta de los siguientes componentes:

  - Un conjunto de librerías en C++ para la aplicación de modelos de lenguaje, estructura de datos y diversas funciones.
  - Un conjunto de programas ejecutables creados por encima de las librerías anteriores para la realización de diferentes tareas como etiquetado, segmentación de texto etc.
  - Una colección de scripts que realizan otras tareas menores.

- **HResults:** es una herramienta de HTK y sirve para comparar archivos de transcripciones con las obtenidas a la salida de un reconocedor de voz. Devuelve la tasa de acierto de palabras, de frases así como la precisión (WORD, SENT, ACC).

  Siendo "WORD" la tasa de palabra correctas:

$$\%\text{Correctas} = \frac{H}{N} \times 100\,\%$$

  "SENT" es la tasa de frases correctas totales:

$$\%\text{Correctas} = \frac{H}{N} \times 100\,\%$$

  "ACC" es la precisión teniendo en cuanta también el número de inserciones:

$$\%\text{Correctas} = \frac{H - I}{N} \times 100\,\%$$

4.1.2. Hardware utilizado

Para realizar este proyecto, se ha utilizado un servidor HP ProLiant D380 G5 con dos Intel Xeon de 4 núcleos cada uno a 3.2GHz. Contaba con 8GB de RAM en un comienzo, que se ampliaron a 20GB. También dispone de 8 discos duros SAS en RAID 0, sumando 750GB.

Además, para la parte de las redes neuronales, se adquirieron dos tarjetas gráficas.

4.1.3. Recetas principales de Kaldi

La receta inicial utilizada en este proyecto ha sido la correspondiente a "vystadial-en". Se decidió hacer uso de la misma, debido a que era una de las pocas cuya base de datos estaba disponible para su descarga de una forma sencilla y gratuita, ya que el resto de recetas empleaban datos de LDC, todos ellos de pago.

Gracias a ella, se pudo comprender cuál era la forma en la que Kaldi espera recibir los datos de entrada, además de la forma en la que se lleva a cabo el entrenamiento de HMM, y así poder adaptar nuestra base de datos a lo que utiliza este software. También se actualizaron los scripts de "vystadial" destinados a ello para que puedan aceptar, no solo esta base de datos, sino cualquier base de datos que tenga una carpeta con los ficheros de audio .wav y otra con los ficheros de transcripción .trn.

Dado que "vystadial" es una receta que ya no se sigue desarrollando, y que se abandonó antes de la introducción de las redes neuronales, la preparación de modelos con redes neuronales ha sido basada en la receta de "swbd" (SwitchBoard). Se ha adaptado la receta de "vystadial" en la generación de las listas y modelos base necesarios para el desarrollo de las redes neuronales a partir de la parte final de la receta de "swbd". Así de esta forma, se han tomado los scripts destinados

al entrenamiento de redes neuronales de tipo LSTM para realizar los diferentes experimentos.

4.2. Proceso de entrenamiento

En este apartado se van a detallar todos los pasos necesarios para la generación de los modelos acústicos y de lenguaje que se usan en los experimentos.

4.2.1. Generación de listas

Como parte del proceso de adaptación de la receta de "vystadial" a las bases de datos, se modificó parte del código para adaptarlo a bases de datos más genéricas, como la descrita en el capítulo 3, y así poder generar las listas precisadas por Kaldi:

- **wav.scp:** contiene el nombre de todos los archivos de entrenamiento y la ubicación que tiene cada uno en el sistema.
- **spk2utt y utt2spk:** contienen tanto el nombre del fichero de audio como el del locutor involucrado en cada una de las locuciones de la base de datos. En este caso no se han querido utilizar, por lo que se ha mapeado cada archivo a sí mismo. Esta es la forma que ofrece Kaldi para omitir estas listas.
- **text:** se genera un fichero de texto (text), que contiene el nombre del archivo de audio con su transcripción que se utilizan en el entrenamiento del sistema así como la propia transcripción.

4.2.2. Generación de los archivos de transcripciones

Ya se disponía de los archivos de transcripción en el formato usado por HTK (.lab) gracias a los anteriores proyectos desarrollados, por lo que para generar los archivos de las transcripciones en formato Kaldi, se adaptaron estos archivos ".lab" al formato usado por Kaldi ".trn".

Los archivos de transcripciones en HTK están formados por las palabras de la transcripción colocadas una por cada línea y cada archivo finaliza con un punto. Para adaptarlos al formato necesario para Kaldi, se pusieron todas las palabras en una línea y se quitó el punto y con ello se generaron los archivos ".trn".

4.2.3. Diccionario

En este paso se trata de elaborar un diccionario con las transcripciones fonéticas de las palabras, a partir de las transcripciones de los archivos, que servirá para todo el entrenamiento. Para ello, se ha usado un diccionario con las pronunciaciones, que posteriormente fue revisado manualmente, con aproximadamente 150.000 palabras.

A partir de este diccionario, Kaldi se encarga de tomar aquellas palabras que se encuentran en las transcripciones de las frases de entrenamiento y las envía a un fichero de lexicón (lexicon.txt).

Además, se generan la lista de fonemas y se añaden los fonemas de ruido y silencio al lexicón (lexicon), así como los fonemas de inicio y final de palabra.

4.2.4. Generación del modelo de lenguaje

Una vez que está preparada la base de datos, el siguiente paso es la construcción de un modelo de lenguaje que asignará una probabilidad a las secuencias de palabras. El modelo de lenguaje se construye a partir de los datos de entrenamiento (text). Se trata de un modelado estadístico calculado a partir de la frecuencia de las diferentes secuencias de palabras.

Se mantuvieron algunas de las etiquetas heredadas de la base de datos de "vystadial" como pueden ser "laugh", "noise", "inhale" y "ehm" por retrocompatibilidad con la base de datos de vystadial, pero en este entrenamiento no se utilizan.

4.2.5. Conversión a formato FST y generación de grafo HCLG

Una vez que el modelo de lenguaje ha sido entrenado ya sería funcional. Sin embargo, Kaldi aprovecha los Transductores de Estados Finitos (FST, Finite State Transducers) para poder optimizar la decodificación mediante la construcción de un grafo.

Para ello, se usan las librerías de OpenFST para convertir el modelo de lenguaje del formato ARPA al formato necesario para Kaldi, FST y el archivo generado se denomina "G.fst". Para cada LM se genera su correspondiente FST.

Para el proceso de reconocimiento se emplea un grafo HCLG, el cual se construye a través de grafos simples FST:

$$\text{HCLG} = \text{H} \circ \text{C} \circ \text{L} \circ \text{G}$$

donde el símbolo '$\circ$' representa una operación binaria asociativa de composición de FSTs, mediante la cual se toman dos elementos de dos conjuntos dados y los asigna a otro elemento, perteneciente a uno de los dos conjuntos. A continuación, se explica la funcionalidad de cada transductor:

1. H contiene las definiciones de los HMM, que toman como número de identificación de entrada las funciones de densidad de probabilidad y dependen del contexto de retorno de los fonemas.
2. C representa la relación entre los contextos de entrada y salida de los fonemas.
3. L representa el léxico. Sus entradas son fonemas y sus salidas son palabras.
4. G codifica el modelo de lenguaje.

4.2.6. Extracción de características

Se extraen las siguientes características acústicas de los ficheros de audio:

- **MFCC:** (Mel Frenquency Cepstral Coefficients) los Coeficientes Cepstrales en la Escala de Mel representan la amplitud del espectro del habla de manera sencilla.

  Primero, se aplica un filtro a la señal y esta se divide en tramas, en Kaldi la ventana de las tramas tiene una duración de 25 ms con 10 ms de solape entre tramas consecutivas. Esto sirve para eliminar los bordes de la señal y dar más importancia a la parte central de la trama para su análisis [16].

  Después se hace la Transformada Discreta de Fourier (DFT) de cada trama y la amplitud obtenida es pasada al dominio de Mel. La escala Mel se basa en mapear la frecuencia actual al pitch que percibe, es como simular una escucha humana lineal por debajo de 1 KHz y logarítmica por arriba.

  A continuación, se realiza el algoritmo de la señal y se aplica la Transformada Discreta del Coseno (DCT).

  Finalmente, se toman los coeficientes deseados por cada trama (MFCC), que son las amplitudes del espectro de la señal resultante.

- **CMVN:** (Cepstral Mean and Variance Normalization) es una técnica de normalización muy robusta para reconocimiento de voz con ruido. Esta normalización consiste en forzar a todas las iteraciones a que tengan media 0 y varianza 1. Esto sirve para unificar, hacerlo más robusto y reducir el coste computacional [17].

  Además, sirve para eliminar los errores introducidos por las diferencias que existen entre los distintos locutores y las calidades de las grabaciones.

4.2.7. Generación HMM

A continuación se van a detallar cada uno de los pasos a seguir para obtener los HMM finales del sistema de referencia del que partiremos para realizar mejoras que serán detalladas en los siguientes capítulos. Este sistema, en el que se generan HMM mediante Kaldi, es el que será comparado con los proyectos anteriores, en los que se usó HTK-Julius para la generación de HMM.

El primer paso es la generación de los monofonemas (independiente del contexto), para los cuales se usan coeficientes MFCC con primera y segunda derivada (delta 'Δ' y delta-delta '$\Delta\Delta$').

A continuación, se realiza un alineamiento forzado de los vectores de características, que consiste en asignar a cada fonema sus marcas de tiempo inicial y final, con el fin de hacer más precisos los entrenamientos posteriores.

En segundo lugar se entrena un modelo de trifonemas, donde el contexto ya es importante, tomando el fonema anterior y el posterior. Se genera al modelo tri1a, y se generan nuevos alineamientos forzados para los datos de entrenamiento, tri1a-ali.

A partir de estos alineamientos parten otros dos modelos: en este apartado nos vamos a centrar en el primer entrenamiento, que nos da el modelo tri2a, y en el siguiente capítulo se explicará el modelo tri2b.

Tri2a solo utiliza MFCC + Δ + $\Delta\Delta$, que es lo correspondiente a lo usado en los proyectos anteriores con los que vamos a comparar los resultados de este proyecto.

4.3. Experimentos y resultados

Hay que tener en cuenta unos factores de diseño y diferencias entre ambos diseños antes de ser comparados:

Se usaron todos los archivos disponibles de la base de datos de entrenamiento, definida en el capítulo 3.1, ya que en los experimentos con HTK-Julius dados se hicieron pruebas variando la cantidad de archivos y se demostró que cuantos más archivos se usaran mejores eran los resultados. Por lo tanto, para lo experimentos con Kaldi se tomó el mayor número de archivos disponibles.

Experimento 1: se hicieron pruebas variando el número de gaussianas y estados. Primero se probó con el valor que traía Kaldi por defecto, 19.200 gaussianas con 1.200 estados (16 gaussianas por estado en promedio). Después con un número más parecido a las usadas en el sistema de HTK-Julius de los proyectos precedentes, que fueron aproximadamente 192.000 gaussianas con 12.000 estados (también 16 gaussianas por estado en promedio).

Este experimento se ha realizado con las diferentes bases de datos disponibles para este proyecto, sin embargo aquí se muestra la más relevante. A modo de ejemplo podemos ver las tasas referentes a la base de datos de entrenamiento para hacer auto-test. Las demás pruebas se pueden ver en el apéndice A. A continuación se muestra uno a modo de ejemplo:

Sistema / Test set	Autotest-30-40
HTK-Julius	97,60
Kaldi (19.200)	**97,42**
Kaldi (192.000)	97,12

Cuadro 4.1: Tasa de acierto frente a número de gaussianas

Se puede observar en la tabla, que los resultados obtenidos para las diferentes pruebas son bastante similares. Se pueden sacar varias conclusiones:

1. Como al comparar los resultados de HTK-Julius frente a Kaldi el resultado es parecido, significa que el sistema desarrollado para este proyecto con Kaldi, por lo menos hasta este punto, funciona correctamente.

2. Si se comparan los dos resultados obtenidos con Kaldi, usando diferente número de gaussianas, se observa que al aumentar el número de gaussianas disminuye

la tasa de acierto. Aunque es muy poca la diferencia, esto se puede deber a que la cantidad de datos de entrenamiento es muy pequeño para el elevado números de gaussianas. Quizás si se tuviera una base de datos de entrenamiento mayor, sería interesante aumentar el número de gaussianas.

Por lo tanto, no merece la pena aumentar el número de gaussianas por estado, debido a que el coste computacional es mucho mayor y no hay mejora significativa.

3. Hay que añadir que, en el sistema de referencia empleado en los proyectos fin de carrera anteriores, HTK usa HMM sin mezclar y Kaldi usa GMM (Gaussian Mixture Model), es decir, realiza una mezcla de gaussianas y agrupa aquellas que son parecidas en una, por lo que se reduce el tiempo de computación, el almacenamiento necesario y por eso es probable que no sean necesarias tantas gaussianas como con HTK.

   A partir de ahora, solo se mostrarán los resultados obtenidos con 19.200 gaussianas.

Experimento 2: se realiza la comparación del reconocedor de HTK-Julius con el reconocedor de Kaldi. Para ello se han hecho varias pruebas con los diferentes conjuntos de test, explicados en la sección 3.2, usando para todos ellos el modelo acústico tri2a (sección 4.2.7), y un modelo de lenguaje cerrado de trigramas.

Un modelo de lenguaje cerrado o acotado, es aquel en el que todas las palabras a reconocer están en el modelo y no hay ninguna palabra del modelo fuera de las frases a reconocer, es decir, han sido generados con los corpus exactos extraídos de las transcripciones de los mismos.

Sistema Test set	Auto-test 10-20 dB	Auto-test 20-30 dB	Auto-test 30-40 dB	Auto-test 40-99 dB	Test 30-40 dB	Test ruidoso
HTK-Julius	95,74	96,18	96,60	95,94	95,46	90,88
Kaldi	92,77	97,33	97,42	97,53	96,73	94,13

Cuadro 4.2: Comparativa HTK-Julius con Kaldi para Auto-test y test

En este primer set de pruebas, que compara archivos con distinta calidad de audio, podemos observar cómo Kaldi rivaliza en calidad con HTK. Las tasas de acierto son muy parecidas en promedio.

Sistema Test set	Pitch 0-100 Hz	Pitch 100-140 Hz	Pitch 140-180 Hz	Pitch 180-220 Hz	Pitch 220-máx Hz
HTK-Julius	94,39	92,58	90,21	91,20	89,73
Kaldi	92,08	93,81	92,72	96,10	78,65

Cuadro 4.3: Comparativa HTK-Julius con Kaldi para pitch

En el segundo set de pruebas, que compara archivos en función de la frecuencia fundamental de su locutor, podemos observar cómo los resultados de Kaldi siguen pareciéndose a los de HTK. Se observan variaciones significativas en las frecuencias altas, siendo en este caso mejor Kaldi para las frecuencias en torno a 200 Hz, y siendo peor en las mayores de 220 Hz.

Sistema / Test set	Telediarios	Economía	Deporte	Moda
HTK-Julius	——	54,54	58,36	33,98
Kaldi	**78,37**	60,74	69,26	41,5

Cuadro 4.4: Comparativa HTK-Julius con Kaldi para vídeos de internet

El último conjunto de archivos de test, consta de archivos de audio sacados de videos de distintos contextos, como pueden ser un telediario completo, noticias deportivas, ponencias económicas y video blogs de moda.

En este caso, podemos observar cómo Kaldi lleva una gran ventaja respecto a HTK. En las pruebas anteriores hemos visto que HTK-Julius y Kaldi daban tasas similares. Por tanto, en estas pruebas la mejora puede venir dada debido a que Kaldi utiliza un sistema distinto a HTK-Julius para tratar los modelos de lenguaje, los FST (Finite State Transducer).

En los bancos anteriores, las frases eran cortas e inconexas, mientras que en estas pruebas, al tratarse de videos reales que siguen un mismo hilo argumental, el modelo de lenguaje cobra más relevancia, aunque se trate de un modelo cerrado.

Experimento 3: se quiere ver la diferencia que hay en las tasas, si en vez de utilizar un modelo de lenguaje cerrado como en el apartado anterior, se usa un modelo de lenguaje abierto.

Un modelo de lenguaje abierto es aquel que no está necesariamente ajustado a los contenidos que se quieren reconocer, lo que supone una reducción en la tasa de acierto respecto al uso de un modelo de lenguaje cerrado, debido a la confusión originada entre las palabras similares que están en el diccionario.

Viendo los resultados del experimento anterior, se puede observar que dentro de los ficheros de vídeos, el que mejor tasa tiene son los telediarios. Por lo tanto se ha realizado esa misma prueba usando esta vez un modelo de lenguaje abierto.

Sistema / Test set	Telediarios (LM abierto)	Telediarios (LM cerrado)
HTK-Julius	61,17	——-
Kaldi	**63,16**	**78,37**

Cuadro 4.5: Comparativa HTK-Julius con Kaldi para telediarios con modelos abierto y cerrado

Como se esperaba, la tasa disminuye al utilizar un modelo de lenguaje abierto, en vez de cerrado. Esto se debe, a que en este caso, el modelo no está ajustado a

los contenidos que se van a reconocer y puede haber palabras del modelo que se encuentren fuera de las frases a reconocer (OOVs – Out Of Vocabulary). Además, al tener un vocabulario mucho más amplio, hay mayor confusión entre palabras que se pronuncian de forma parecida.

5

Optimización del Sistema Base y Resultados

Una vez terminado el sistema de referencia, el siguiente paso es mejorarlo. Para ello las pruebas se centrarán en utilizar distintos métodos de entrenamiento usando las diferentes transformadas, y así conseguir generar un modelo más robusto que abarque todo tipo de voces y distintas condiciones de entorno.

5.1. Motivación específica y problema a resolver

Para conseguir un sistema más universal, para todos los tipos de voz, se ha decidido dejar fijo el tamaño de la ventana, a diferencia del proyecto realizado anteriormente [2] en el cual se adaptaba la ventana al pitch medio de la señal de entrada. Normalmente en reconocimiento de habla continua se utiliza una ventana de 25 ms [18] con solapamiento de 10 ms.

En este proyecto ya se parte de la premisa de que adaptar el tamaño de la ventana de desplazamiento a la frecuencia fundamental de la voz que se va a reconocer podría suponer grandes mejoras en las tasas de acierto, gracias a los resultados aportados en el proyecto fin de carrera [2]. Se demostró que era verdad, pero la mejora no era muy significativa para todos los tamaños de ventana.

Usar una ventana de 25 ms, nos permite procesar como mínimo dos periodos de una señal de voz de 80 Hz, tres periodos de una señal de 120 Hz y cuatro periodos de una de 160 Hz. Esto es importante para modelar tanto las zonas estables de la señal como las zonas de transición entre sonidos al cubrir una zona suficientemente larga de la señal. El análisis de los resultados del proyecto fin de carrera mencionado anteriormente, indica que la ventana de 25 ms es la que se comporta mejor para todas las señales de voz, proporcionando los mejores resultados para voces cuya frecuencia fundamental promedio es de 120 Hz.

El problema de esta ventana surge a partir de 160 Hz de pitch promedio, momento en el que la ventana de 25 ms abarca más de cuatro periodos de la señal y por debajo de 80 Hz, donde la ventana abarca menos de dos periodos de la señal.

En el estudio realizado en el proyecto de referencia, se muestra que la frecuencia fundamental media de la voz masculina, está centrada entre 110-120 Hz y la de la voz femenina entre 180-200 Hz. Esto explica por qué la voz femenina tiene peores tasas de reconocimiento, en general, que la masculina y que las tasas empeoren según el pitch medio aumenta.

Como se ha comentado anteriormente, Kaldi usa por defecto una venta de 25 ms con un solapamiento entre ventanas de 10 ms, aún así se ha considerado dejar fijo el valor de la ventana y para poder evaluar la mejora por otros medios.

Por consiguiente, el objetivo de este capítulo será evaluar la mejora que se consigue al utilizar las diferentes transformadas que se describirán en la siguiente sección.

5.2. Condiciones Experimentales

En esta sección se definen las diferentes condiciones para los experimentos y se describen los diferentes métodos a utilizar mediante el uso de transformadas.

5.2.1. Transformadas

En la sección 4.2.7 definió y explicó el procedimiento para llegar a tri2a ($\Delta + \Delta\Delta$). A continuación se describen las transformadas que se han aplicado en los siguientes pasos que se han tomado:

- **LDA + MLLT:** en este paso se aplican dos técnicas; Linear Discriminant Analysis (LDA) y Maximum Likelihood Linear Transform (MLLT) sobre el modelo anterior de trifonemas.

  LDA es un método de transformación lineal que trata de encontrar las combinaciones lineales de las variables originales para proporcionar a nuestro conjunto de datos la mayor separación posible.

  MLLT es un método de estimación basada en LDA que trata de mejorar la toma de decisiones y añade una mejora en la velocidad en la decodificación [19].

  Esta fase consta de un entrenamiento LDA + MLLT y posteriormente un alineamiento del modelo, dando lugar a los modelos tri2b y tri2b-ali.

- **MMI:** de sus siglas Maximun Mutual Information. Se reconocen todos los archivos de entrenamiento con el fin de obtener un lattice del numerador que al compararlo con el del denominador, generado a partir de las transcripciones correctas, permite hacer una aproximación más eficaz y recalibrar los modelos acústicos en función de los errores encontrados al decodificar [20].

  Se realiza un entrenamiento MMI sobre LDA + MLLT y obtenemos el modelo tri2b-mmi.

- **bMMI:** es un entrenamiento discriminativo MMI con boosting. La modificación, respecto a MMI, consiste en aumentar el factor de 'b' que amplifica las probabilidades de las frases que tienen un mayor error relativo. Esto permite ajustar mejor las frases que más errores tienen, mejorando la calidad del sistema [21].

  En este caso, el valor usado para b = 0,05 y se genera el modelo tri2b-bmmi.

- **MPE:** Minimum Phone Error, son aproximaciones que suavizan la tasa de error de fonemas en un contexto de reconocimiento de palabras, pero se pueden usar los lattices desarrollados para MMI. Con esta técnica se obtiene el modelo tri2b-mpe [22].

  A continuación se muestra un esquema de lo anterior descrito:

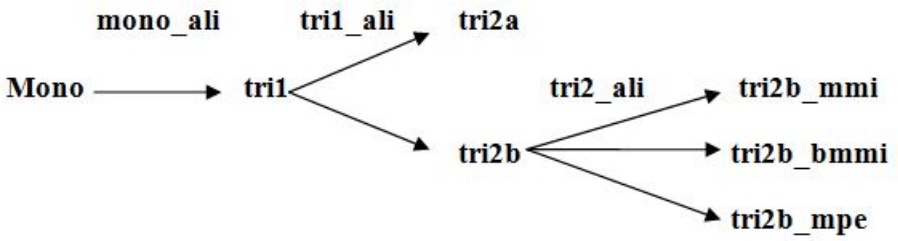

Figura 5.1: Esquema de los diferentes modelos

Método de entrenamiento	Nombre script
Monofonema	mono
Trifonema	tri1
$\Delta + \Delta\Delta$	tri2a
LDA + MLLT	tri2b
LDA + MLLT + MMI	tri2b-mmi
LDA + MLLT + bMMI	tri2b-bmmi
MPE	tri2b-mpe

Cuadro 5.1: Pasos a seguir durante el entrenamiento

5.2.2. Experimentos y resultados

En este apartado se van a llevar a cabo una serie de experimentos, los cuales son continuación del experimento 2 realizado en la sección 4.4.

Experimento 4: esta vez se van a comparar los resultados obtenidos con el sistema de reconocimiento desarrollado en Kaldi en la sección 4.4. Las diferencias van a estar en las mejoras que producen, en las tasas de acierto, las diferentes transformadas.

Para este apartado se ha usado el mismo modelo de lenguaje cerrado de trigramas, empleado en las pruebas de la sección 4.4, para cada conjunto de test.

La primera columna de valores corresponde a los resultados obtenidos en el experimento 4.4 con modelos de tri2a.

Como se puede observar, la primera tabla que se muestra a continuación presenta los resultados en autotest para distintas relaciones señal a ruido:

Kaldi	ΔΔ tri2a - REF	MLLT+LDA tri2b	MMI tri2b-mmi	MMI+Boost tri2b-bmmi	MMI+MPE tri2b-mpe
Auto-test 10-20 dB	92,77	92,42	97,66	97,41	93,76
Auto-test 20-30 dB	97,33	97,38	98,73	98,59	97,89
Auto-test 30-40 dB	97,42	97,88	98,94	98,90	98,28
Auto-test 40-99 dB	97,53	97,58	98,75	98,65	98,08
Test 30-40 dB	96,73	97,54	98,54	98,54	98,03
Test ruidoso	94,13	94,60	97,26	97,18	96,42

Cuadro 5.2: Resultados Auto-test y test para las distintas transformadas

De la tabla anterior cabe destacar:

- Los mejores resultados los aporta el modelo MMI respecto al modelo de referencia tri2a. También se observa que bMMI da tasas similares con variaciones muy pequeñas entre ambos y para este conjunto de datos no se puede sacar una conclusión de ello.
- Para los conjuntos de autotest 10-20 dB y para test ruidoso, el paso de tri2a a MMI supone una subida de 5 y 3 puntos respectivamente. Esto es relevante debido a que, estos conjuntos de test, están formados por frases de peor calidad que las demás y las tasas de acierto suben hasta casi igualar los conjuntos con frases con mejor calidad.
- Se observa que MMI es mejor que MPE, para este conjunto de datos. Para este caso, la aproximación por frases de MMI da mejores resultados que la aproximación de MPE por fonemas.

En la segunda tabla, se pueden ver los resultados obtenidos al agrupar a los usuarios por el pitch medio:

En la tabla anterior, se puede apreciar que el reconocedor basado en MMI da muy buenos resultados para la voz aguda, respecto al modelo de referencia tri2a. Se aprecia una subida de 5 puntos y es importante porque es la voz peor entrenada. Por lo que se demuestra que está bien usar transformadas, que no han sido usadas en los proyectos anteriores, para mejorar las tasas.

Kaldi	ΔΔ tri2a - REF	MLLT+LDA tri2b	MMI tri2b-mmi	MMI+Boost tri2b-bmmi	MMI+MPE tri2b-mpe
Pitch 0-100 Hz	92,08	93,77	94,55	94,51	94,12
Pitch 100-140 Hz	93,81	98,28	99,10	99,07	98,68
Pitch 140-180 Hz	92,72	94,76	95,25	95,28	94,95
Pitch 180-220 Hz	96,10	98,15	98,97	98,91	98,66
Pitch 220-máx Hz	78,65	82,13	83,75	83,97	83,37

Cuadro 5.3: Resultados de pitch para las distintas transformadas

Finalmente, en la tercera tabla se muestran los resultados obtenidos en la base de datos de noticias:

Kaldi	ΔΔ tri2a - REF	MLLT+LDA tri2b	MMI tri2b-mmi	MMI+Boost tri2b-bmmi	MMI+MPE tri2b-mpe
Telediarios	78,37	76,75	82,39	82,15	82,17
Economía	60,74	59,71	63,77	63,36	64,57
Deporte	69,26	65,33	70,36	69,67	73,05
Moda	41,50	36,76	39,16	37,48	37,21

Cuadro 5.4: Resultados pruebas vídeos internet para las distintas transformadas

En este caso queda demostrado, que para situaciones reales como vídeos, las tasas vuelven a mejorar mediante el uso de transformadas, así que también es válido para este objetivo.

Cabe destacar, que para todos los tópicos mejora, excepto para moda. Esto se debe a que este conjunto de datos está formado por archivos con muy mala calidad y se comportan de forma diferente a los distintos modelos.

Experimento 5: viendo los resultados del experimento anterior, se puede observar que dentro de los ficheros de vídeos, el conjunto que mejor tasa tiene es el correspondiente a los contenidos de informativos de televisión. Además, la técnica de entrenamiento que proporciona mejor tasa de acierto de palabras, se corresponde con la aplicación de la transformada MMI. Por lo tanto se ha realizado esa misma prueba usando esta vez un modelo de lenguaje abierto:

Conclusiones:

- Se puede observar, que hay una diferencia más notable con modelo abierto, ya que se produce una subida de 7 puntos respecto al modelo de referencia.

Sistema / Test set	Telediarios
Kaldi (tri2a)	63,16
Kaldi (tri2b-mmi)	**70,11**

Cuadro 5.5: Comparación sistema base y MMI para LM abierto

- Como podemos observar, al aplicar transformaciones mejora notablemente la tasa de acierto. La que mejores resultados ha dado en casi todos los conjuntos de test ha sido LDA+MLLT+MMI (tri2b-mmi).

6

Optimización del Sistema Mediante Redes Neuronales

Una vez optimizado el sistema de referencia, el siguiente paso es mejorarlo usando redes neuronales profundas. Para ello, las pruebas se centrarán en variar los distintos parámetros disponibles y comparar los resultados para obtener un sistema más robusto.

6.1. Introducción

Una vez que se ha conseguido mejorar el sistema base aplicando las transformadas explicadas en el capítulo anterior, el siguiente paso, y objetivo de este capítulo, es el uso de redes neuronales. Para ello, las pruebas se centran en variar los diferentes parámetros disponibles para ajustar una red neuronal dedicada al reconocimiento de habla continua. Estos se compararán con los parámetros por defecto del sistema y serán adaptados para conseguir los mejores resultados posibles.

El entrenamiento con redes neuronales necesita los modelos entrenados en tri2a (delta-delta) y los alineamientos obtenidos del modelo tri2b (LDA+MLLT), que en la máquina disponible dura unas 72 horas.

Se comienza con una variación aleatoria del volumen de los archivos de entrada, dado que a diferencia de los HMMs, los diferentes volúmenes en la entrada les afectan muy negativamente si no se han entrenado previamente. Después, se calculan los MFCCs de alta resolución, con 40 coeficientes en lugar de los 13 habituales. A continuación, se genera un modelo ligero con LDA+MLLT basado en alineamientos realizados con los modelos de tri2a sobre los 100.000 primeros archivos de la base de datos, que se usará como semilla del entrenamiento de redes neuronales.

Posteriormente, se entrena la diagonal UBM (Universal Background Model),

con una parte pequeña de los datos (30.000 ficheros). Seguidamente, se entrena el extractor de los iVector, y se extraen los mismos de los archivos de audio, quedándose en principio 2 por cada hablante. En nuestro caso nos quedamos todos porque no hacemos distinción de locutor.

Más tarde, se generan los archivos de configuración de las capas de las redes neuronales, y por último se comienza el entrenamiento como tal. De esta última parte, cabe destacar la generación de los "egs" (examples), de entrenamiento y su posterior aleatorización. Son archivos dispuestos de tal forma que el entrenamiento de redes neuronales pueda aprovecharlos al máximo. Hasta este punto, todos estos pasos llevan aproximadamente unas 36 horas.

Por último, llega el entrenamiento en sí, realizado mediante Stochastic Gradient Descent (SGD) en las tarjetas gráficas. Y una vez ha terminado, se calcula la combinación final de los modelos y se realiza el reconocimiento de prueba. En las pruebas realizadas, esta parte lleva unas 16 horas en promedio.

6.2. Redes neuronales

Como se vio en el apartado 2.3.5, en este proyecto se ha decidido usar las redes neuronales recurrentes profundas de tipo LSTM debido a que en la bibliografía existente se ha podido observar que aportan mejores resultados [15].

6.2.1. LSTM

Long Short-Term Memory (LSTM) es un tipo de de red neuronal recurrente, Recurrent Neural Network (RNN), las cuales fueron diseñadas para modelar secuencias temporales con mayor precisión que las redes recurrentes convencionales.

Las redes neuronales recurrentes (RNN) tienen conexiones cíclicas que las hacen más robustas a la hora de modelar secuencias de datos continuos que las redes neuronales de tipo feed-forward. Las RNN se suelen usar para tareas como reconocimiento de escritura o modelado del lenguaje. En el reconocimiento de voz, las predominantes eran las DNN, pero hace unos años empezaron a cobrar importancia las redes neuronales recurrentes.

Estas redes contienen ciclos que permiten la activación de la red un paso de tiempo anterior, actúan como entradas a la red e influyen en las predicciones de ese mismo instante. Estas activaciones se almacenan en los estados internos de la red, que pueden mantener la información contextual temporal a largo plazo.

De estas redes surgen otras que son las LSTM bidireccionales (BLSTM), las cuales operan en la secuencia de entrada pero en ambas direcciones para tomar una decisión para la entrada actual.

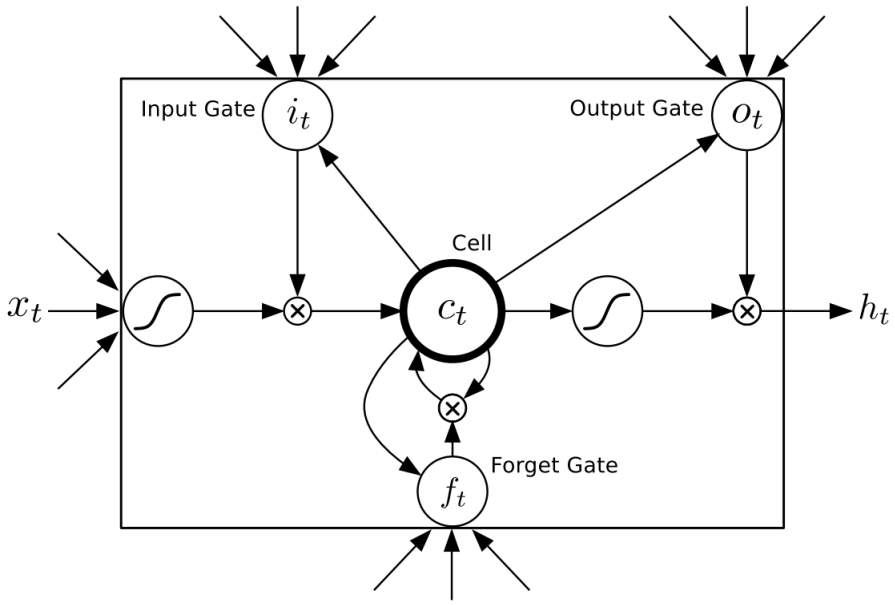

Figura 6.1: Esquema de una LSTM

6.2.2. Deep LSTM

Al igual que con las demás arquitecturas de redes neuronales profundas, las DLSTM se utilizan para el reconocimiento de voz debido a los buenos resultados que aportan al realizar esta tarea [23].

Estas han sido las utilizadas en este proyecto y se construyen mediante el apilamiento de múltiples capas LSTM. A continuación se muestra la arquitectura de las redes LSTM, tanto el sistema profundo como el simple:

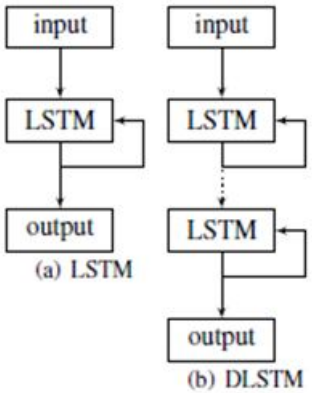

Figura 6.2: Diferencia entre LSTM y DLSTM

Se puede ver que las variables atraviesan múltiples capas no lineales, igual que ocurre en las DNN. Pero en este caso, la característica de un instante de tiempo dado solo se procesa por una única capa no lineal antes de producir la salida en ese instante de tiempo, por lo tanto, la profundidad de esta red tiene una importancia adicional.

La entrada a la red en un intervalo de tiempo dado, pasa a través de múltiples capas LSTM, además de propagarse a través del tiempo y las capas LSTM.

Por ser redes neuronales recurrentes, ya tienen el beneficio de poder aprender en distintos instantes de tiempo sobre la entrada. Pero por usar además las DLSTM, pueden hacer un mejor uso de interconexiones entre las neuronas debido a su distribución en el espacio a través de múltiples capas.

6.3. Pruebas y resultados

En este apartado, se van a mostrar las diferentes pruebas realizadas, así como las conclusiones que se obtienen de ellas. Para ello, se van a hacer diferentes pruebas usando los conjuntos de datos expuestos en el capítulo 3, mediante el uso de redes neuronales profundas recurrentes LSTM.

6.3.1. Parámetros a modificar

Para realizar las diferentes pruebas propuestas, se han modificado una serie de parámetros que aparecen en las distintas recetas de Kaldi. Así se consigue ajustar estos valores para obtener los mejores resultados posibles.

- Número de GPUs, realizando pruebas con una y con dos GPUs.
- Número total de capas, que incluye la capa de entrada, la de salida y todas las capas ocultas.
- Número de capas ocultas, este parámetro solo varía el número de las capas ocultas que se quieran añadir entre la de entrada y la de salida.
- Número de neuronas de la capa de entrada de la red.
- Número de neuronas de las diferentes capas ocultas posibles.
- Número de neuronas de la capa de salida.
- Número de “epoch” de entrenamiento: número de veces que se entrenan todos los datos disponibles.
- Learning rate inicial y final: indican el valor inicial y final de las tasas de aprendizaje que sirven para tratar de igualar los cambios relativos en los parámetros para cada capa. Por lo general se realiza un decaimiento exponencial.

6.3.2. Experimentos

En este apartado se describen los experimentos realizados, así como las diferentes conclusiones que se obtienen de ellos.

Se han realizado sobre el conjunto de archivos de telediarios, ya que al ser más naturales y más complicados de reconocer que las frases de entrenamiento son un conjunto de datos muy adecuado para ver la mejora obtenida con las redes neuronales profundas.

Para estos experimentos, se parte de una serie de parámetros por defecto que vienen dados en las recetas de Kaldi, los cuales se han modificado para obtener mejores tasas. A continuación se muestran los parámetros por defecto:

Nº capas	**5**
Nº capas ocultas	**3**
Neuronas capa entrada	**1024**
Neuronas capas ocultas	**1024**
Neuronas capa salida	**256**
Nº entrenamientos	**8**
Learning Rate inicial	**0,001**
Learning Rate final	**0,0005**

Cuadro 6.1: Parámetros por defecto de Kaldi

Experimento 6: variación del número de GPUs

Este experimento se intentó realizar en primera instancia con CPUs, ya que no se disponía de tarjetas graficas dedicadas. La primera prueba tardó más de un mes, y arrojó unas tasas de acierto muy inferiores a las obtenidas previamente con MMI. Visto esto se decidieron adquirir dos tarjetas graficas, ya que en el propio script de Kaldi se decía que no estaba optimizado para su uso en CPU.

De esta forma, para este primer experimento de redes neuronales, se han utilizado los parámetros por defecto de la tabla anterior y se han hecho pruebas con una y con dos GPUs. En la tabla se muestran las tasas:

Nº GPUs	Tasa de acierto
1	85,57
2	**85,31**

Cuadro 6.2: Resultados frente a número de GPUs

Como se puede observar, la tasa de acierto no varía prácticamente. La ventaja de usar una GPU a dos se encuentra en el tiempo de computación, ya que aumentando a dos el número de GPUs se consigue reducir a la mitad el tiempo de entrenamiento del sistema [24]. Por lo que los experimentos siguientes, se han realizado siempre utilizando las dos tarjetas gráficas.

Experimento 7: variación de número de epoch o tandas de entrenamiento

El experimento anterior fue realizado con los parámetros por defecto, en el cual se usaban como número epochs 8. Se han realizado experimentos para ajustar ese

número y ver cómo afecta a la tasa de acierto, ya que al disminuirlo también decrece el tiempo empleado para el entrenamiento.

Nº epoch	Fracción de tiempo	Tiempo entrenamiento	Tasa de acierto
1	$\frac{1}{8}$	8 h	83,10
2	$\frac{1}{4}$	16 h	**85,33**
8 (referencia)	1	96 h	85,31

Cuadro 6.3: Resultados frente a número de epoch

Como se puede ver, al reducir a una unidad, la tasa de acierto decae 2 puntos respecto a la de referencia. Sin embargo, si se realizan al menos dos epochs, la tasa se mantiene y además se consigue un tiempo de entrenamiento cuatro veces menor. Por lo que para los experimentos posteriores, este parámetro será de dos.

Experimento 8: variación del learning rate

En este experimento, se han variado el learning rate tanto inicial como final, como se puede ver en la siguiente tabla:

Lrate inicial	Lrate final	Tasa de acierto
0,005 (x5)	0,0025	73,49
0,0015 (x1,5)	0,00075	74,41
0,00075 ($\frac{1}{3}$)	0,000375	84,55
0,001 (x1) (Ref.)	0,0005	85,33
0,001 (x1)	**0,00055 (x1,1)**	**85,61**

Cuadro 6.4: Resultados frente a variación del learning rate inicial y final

Tras realizar múltiples variaciones de las tasas de aprendizaje, con la única que se ha observado mejora es aumentando el learning rate final y manteniendo el inicial. Tasas de aprendizaje demasiado altas pueden llevar a entrenamientos demasiado rápidos que no son capaces de ajustar correctamente los parámetros, y aprendizaje demasiado lentos necesitan más pasadas de entrenamiento aumentando el número de epoch, y por tanto el tiempo de procesado. El objetivo es conseguir un compromiso entre tiempo de entrenamiento y tasa de acierto de palabras.

A partir de este punto, se tomará como referencia del valor de learning rate final el 0,00055.

Experimento 9: variación del número de neuronas

Para las siguientes pruebas, se han modificado el número de neuronas tanto de las capas de entrada y salida, como las de las capas ocultas.

Para la primera prueba, se ha realizado una disminución de un 25 % de neuronas respecto a las utilizadas por defecto y para la segunda prueba se ha aumentado un 25 %.

Nº Neuronas			Tasas
Capa de entrada	Capas ocultas	Capa de salida	Tasa de acierto
1024	1024	256	85,61 (Ref.)
768	768	192	85,19
1280	1280	320	85,27
768	1024	192	85,41

Cuadro 6.5: Resultados frente a números de neuronas de las diferentes capas

Como se puede ver el valor es prácticamente el mismo, aunque baja aproximadamente un punto la tasa del experimento 8.

Por todo ello, se han dejado la capa de entrada y de salida como los mismos valores de la primera prueba, ya que al dar resultados parecidos con estos valores el tiempo de entrenamiento es menor, y se ha probado a dejar las mismas que por defecto en las capas ocultas. Y con ello, se ha conseguido un valor parecido al del experimento anterior.

Como no ha mejorado, para nuestro conjunto de datos, se ha decidido continuar con el número de neuronas de las diferentes capas igual a los de por defecto. Aunque en algún experimento posterior se volverán a variar junto al número de capas.

También debe hacerse notar, que cuantas menos neuronas se utilicen, más rápido es el entrenamiento y el reconocimiento.

Experimento 10: variación del número de capas ocultas

Según los parámetros por defecto, el número de capas ocultas eran 3 y por lo tanto el total de capas era de 5 (sumando la de salida más la de entrada).

En este caso se ha probado a aumentar y a disminuir una capa oculta y estos son los resultados:

Capas totales	**4**	**5**	**6**
Capas ocultas	**2**	**3**	**4**
Neuronas capa entrada	1024	1024	1024
Neuronas capas ocultas	1024	1024	1024
Neuronas capa salida	256	256	256
Tasa de acierto	83,73	**85,61**	72,42

Cuadro 6.6: Resultados frente a número de capas ocultas

Como se observa, tanto si aumentamos como si disminuimos una capa, perdemos tasa de acierto. Hay que matizar que el número de neuronas no se ha modificado y se corresponden con los mismos valores dados al principio.

Surge la hipótesis de que al añadir una capa haya demasiadas neuronas para este conjunto de datos, por lo que se va a reducir el número de neuronas para observar el

efecto provocado:

Capas totales	6	6
Capas ocultas	4	4
Neuronas capa entrada	1024	**768**
Neuronas capas ocultas	1024	**768**
Neuronas capa salida	256	**192**
Tasa de acierto	72,42	**83,85**

Cuadro 6.7: Resultados frente a variación de capas y neuronas

Efectivamente, se puede comprobar que al aumentar una capa oculta era necesario disminuir el número de neuronas, aunque las tasas siguen siendo peores que con los parámetros por defecto.

Experimento 11: comparativa con HMM usando modelos de lenguaje cerrados y abiertos

Para finalizar la comparativa con HMMs, se ha añadido una última prueba en la que se ha reconocido utilizando las DLSTM con un modelo de lenguaje cerrado y con uno abierto que consta de 120.000 palabras, el mismo que en el experimento de HMMs.

Tipo LM	Cerrado	Abierto
HMM-MMI (%WORD)	82,39	70,11
DLSTM (%WORD)	85,61	76,43
Diferencia	3,22	**6,32**

Cuadro 6.8: Comparación HMM-MMI con Kaldi para LMs abierto y cerrado

Como podemos observar, las redes neuronales suponen una clara mejora en la tasa de acierto. Cabe destacar, que al utilizar un modelo de lenguaje abierto, con más palabras y que no está adaptado a las frases que se van a reconocer, la mejora en la tasa de acierto es el doble que en el caso anterior.

Se nota una mejora de más de 6 puntos al utilizar LMs abiertos, y de 3 puntos al utilizar LMs cerrados. Esto nos enseña que en momentos en los que hay más probabilidad de confusión entre palabras, como al haber muchas más palabras en el diccionario, las redes neuronales ayudan mucho mejor a discriminar los sonidos y por tanto acertar las palabras correctas.

7

Aplicación del Sistema

Después de haber conseguido un sistema base de reconocimientos de habla natural, haberlo evaluado y optimizado, se ha introducido en una solución comercial de la empresa.

Esta se encarga de indexar contenidos audiovisuales de Internet para poder realizar búsquedas con ellos. El sistema de reconocimiento transcribe el audio de los contenidos, en un segundo plano, y genera las marcas de tiempo de cada palabra. Esto dará una mayor robustez y fiabilidad a la indexación, consiguiendo mejores resultados.

En el siguiente apartado, se muestra el funcionamiento de la aplicación y la importancia de la transcripción de los contenidos audiovisuales en ella.

7.1. Demostración

Para la demostración se van a utilizar varias noticias extraídas de telediarios. Cuando lo vídeos hayan sido reconocidos e indexados, se puede realizar la búsqueda en distintas partes de la noticia: en el título, descripción del vídeo e incluso en el mismo audio con su marca de tiempo asociada.

La interfaz tiene un diseño práctico e intuitivo, lo que hace un fácil manejo para el usuario (figuras). En la parte superior, cuenta con un campo de texto donde se introduce la cadena a buscar. Más abajo, se encuentra el visualizador de los vídeos, además de una barra que marca cada una de las apariciones de la búsqueda en el instante de tiempo correspondiente, en color azul. En la parte inferior, se muestra la información más importante de la noticia como: título, fecha, tema y descripción. Por último, a la derecha se muestran los resultados de la búsqueda ordenados por orden de fiabilidad de aparición dentro del vídeo.

Ejemplo

A modo de ejemplo se introduce en la barra de búsqueda la cadena "$redneuronal$" y "$pokemon$" respectivamente. En ese momento aparecerá en primer plano el vídeo que tiene mayor probabilidad de que aparezca lo que se ha pedido.

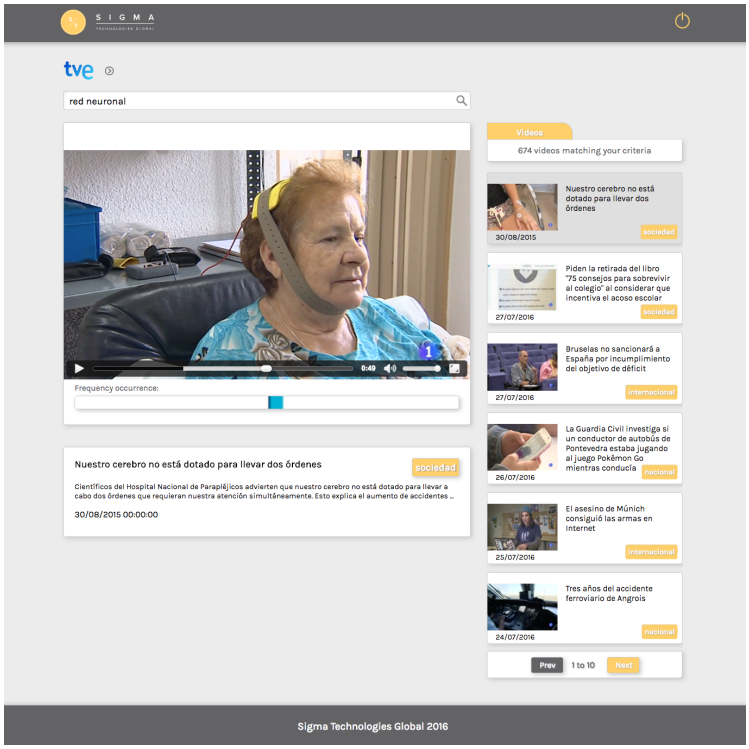

Figura 7.1: Imagen de la aplicación. Búsqueda concreta.

Se puede ver como a la derecha aparece marcado el vídeo que se está reproduciendo, y en la parte inferior derecha se informa de que la búsqueda aparece tanto en el audio como en la descripción de la noticia, apareciendo la búsqueda en negrita.

Además, en la segunda figura se ve que la marca de tiempo pasa a un color azul más oscuro. Eso indica el cambio de locutor que menciona esa misma cadena que fue buscada anteriormente.

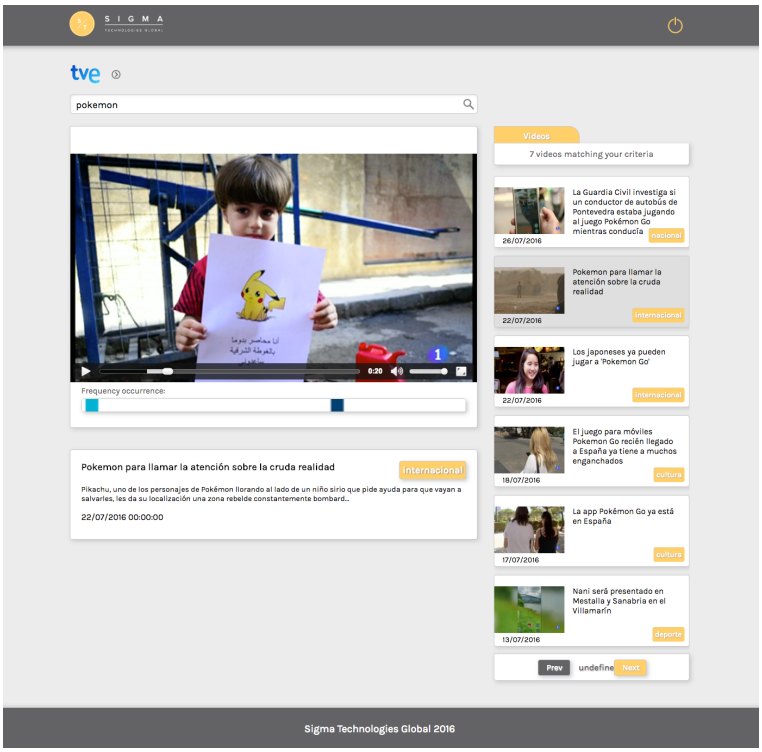

Figura 7.2: Imagen de la aplicación. Diferentes apariciones en el vídeo

8

Conclusiones y Trabajo Futuro

8.1. Conclusiones

Para conseguir el objetivo de este proyecto, realizar un sistema de reconocimiento de habla natural orientado a la transcripción de contenidos audiovisuales de Internet, se ha realizado un estudio, con sus posteriores experimentos, sobre cómo mejorar el sistema para hacerlo más robusto y conseguir una mayor tasa de acierto. Para ello, se han seguido una serie de pasos.

Al comienzo, se realizó una búsqueda y estudio de la información necesaria del estado de arte, para analizar los sistemas de reconocimiento de voz, sobre todo aquellos que utilizan modelos ocultos de Markov y redes neuronales profundas.

Una vez realizado el estudio, se comenzó a desarrollar un sistema de entrenamiento base, utilizando la herramienta Kaldi, y se midió su calidad comparando los resultados de las pruebas realizadas con el reconocedor con los datos que se tenían de otros sistemas de proyectos anteriores.

Por último, se hicieron varias optimizaciones del sistema utilizando diferentes transformadas. A continuación, se realizaron pruebas usando redes neuronales con el fin de mejorar más la tasa de acierto, confirmando que estas últimas muestran mejores resultados.

Así se ha conseguido el objetivo de este proyecto y mediante los experimentos realizados, podemos sacar una serie de conclusiones:

- Aplicar transformadas como MMI, MPE o métodos de entrenamiento como las redes neuronales, parecen compensar con creces el incremento conseguido con entrenamientos específicos realizados para optimizar el tamaño de la ventana. Estos últimos conseguidos en proyectos de fin de carrera anteriores.

- Si se realiza una comparación de los resultados del sistema base, en el cual se usaron modelos "$\Delta\Delta$"(Capítulo 4), con el sistema mejorado utilizando diferentes transformadas (Capítulo 5), se observa que a la hora del reconocimiento de audio, mejora ampliamente la calidad del mismo, si se aplican estas transformadas. Además que de todas las probadas, las que mejores resultados muestran son los realizados con la transformada MMI.

Tipo LM	Cerrado	Abierto
HMM-$\Delta\Delta$ (REF.) (%WORD)	78,37	63,16
HMM-MMI (%WORD)	82,39	70,11
Diferencia	4,02	**6,95**

- A continuación, al realizar la comparación de los experimentos en los cuales se usan HMM con distintas transformadas, en este caso MMI que son las que mejores resultados arrojan (Capítulo 5), y el sistema optimizado mediante redes neuronales (Capítulo 6), se muestra que efectivamente, las redes neuronales dan los mejores resultados. En este caso se han usado las redes neuronales recurrentes DLSTM.

Tipo LM	Cerrado	Abierto
HMM-MMI (%WORD)	82,39	70,11
DLSTM (%WORD)	85,61	76,43
Diferencia	3,22	**6,32**

- Si ahora comparamos los resultados de los experimentos del sistema de referencia (Capítulo 4), con el sistema final optimizado (Capítulo 6), se observa una diferencia, para modelo cerrado, de casi 7 puntos y para el modelo abierto de casi 13 puntos absolutos.

Tipo LM	Cerrado	Abierto
HMM-$\Delta\Delta$ (REF.) (%WORD)	78,37	63,16
DLSTM (%WORD)	85,61	76,43
Diferencia	7,24	**13,27**

  Como se puede ver, las redes neuronales suponen una gran mejora en la tasas de acierto respecto a los modelos ocultos de Markov. Además se produce mayor mejora con un modelo de lenguaje abierto, que nos muestra que las redes neuronales, en momentos en los que hay más probabilidad de confusión entre palabras, discriminan mejor los sonidos y por tanto aciertan las palabras correctas.

- Además, se ha conseguido probar el correcto funcionamiento de este sistema de reconocimiento al ser incluido en una solución comercial, el cual ayuda a mejorar la búsqueda dentro de los contenidos audiovisuales.

8.2. Trabajo Futuro

Una vez finalizado el proyecto, y con el fin de mejorar aún más el sistema, se proponen diferentes líneas de trabajo:

- Sería bueno encontrar nuevos métodos de supresión de ruido de los vídeos que se van a reconocer, debido a que provienen directamente de Internet y no se puede garantizar su calidad. Esto mejoraría la precisión del reconocimiento.
- Dado que se ha demostrado en este proyecto que las redes neuronales proporcionan mejores resultados, se propone realizar pruebas con distintos tipos de ellas. Esto sería importante debido a que existen diferentes tipos de redes neuronales y en función del tipo de datos y de la cantidad de ellos disponibles, pueden dar mejores resultados unas u otras.
- En este proyecto se han variado una serie de parámetros, correspondientes a las redes neuronales, para conseguir mejores resultados. Se han variado aquellos que se pensaba podían influir más en los resultados finales y otros se han dejado fijos. Como trabajo futuro se propone variar los parámetros restantes que no han sido variados en este proyecto.

Bibliografía

[1] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The kaldi speech recognition toolkit. In IEEE 2011 Workshop on Automatic Speech Recognition and Understanding. IEEE Signal Processing Society, December 2011. IEEE Catalog No.: CFP11SRW-USB.

[2] Javier Antón Martín et al. Desarrollo de un sistema de reconimiento de habla natural independiente del locutor. 2015.

[3] Perero Codosero, Juan Manuel, et al. Desarrollo de un sistema de reconimiento de habla natural para transcribir contenidos de audio en internet. 2015.

[4] Paul Lamere, Philip Kwok, William Walker, Evandro B Gouvêa, Rita Singh, Bhiksha Raj, and Peter Wolf. Design of the cmu sphinx-4 decoder. In INTERSPEECH. Citeseer, 2003.

[5] Damián Jorge Matich. Redes neuronales: Conceptos básicos y aplicaciones. Cátedra de Informática Aplicada a la Ingeniería de Procesos–Orientación I, 2001.

[6] Emre Cakir, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen. Polyphonic sound event detection using multi label deep neural networks. In 2015 international joint conference on neural networks (IJCNN), pages 1–7. IEEE, 2015.

[7] Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks, 3361(10):1995, 1995.

[8] George Saon and Jen-Tzung Chien. Large-vocabulary continuous speech recognition systems: A look at some recent advances. IEEE Signal Processing Magazine, 29(6):18–33, 2012.

[9] Li Deng, Alex Acero, Mike Plumpe, and Xuedong Huang. Large-vocabulary speech recognition under adverse acoustic environments. In INTERSPEECH, pages 806–809, 2000.

[10] Hynek Hermansky. Perceptual linear predictive (plp) analysis of speech. the Journal of the Acoustical Society of America, 87(4):1738–1752, 1990.

[11] Lawrence Rabiner and Biing-Hwang Juang. Fundamentals of speech recognition. 1993.

[12] Chotirat Ann Ratanamahatana and Eamonn Keogh. Everything you know about dynamic time warping is wrong. In Third Workshop on Mining Temporal and Sequential Data. Citeseer, 2004.

[13] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Magazine, 29(6):82–97, 2012.

[14] Martine Adda-Decker and Lori Lamel. The use of lexica in automatic speech recognition. In Lexicon Development for Speech and Language Processing, pages 235–266. Springer, 2000.

[15] Hasim Sak, Andrew W Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In INTERSPEECH, pages 338–342, 2014.

[16] Guillermo Arturo Martínez Mascorro and Gualberto Aguilar Torres. Reconocimiento de voz basado en mfcc, sbc y espectrogramas. Ingenius, (10), 2013.

[17] N Vishnu Prasad and Srinivasan Umesh. Improved cepstral mean and variance normalization using bayesian framework. In Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on, pages 156–161. IEEE, 2013.

[18] Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, et al. The htk book, vol. 2. Entropic Cambridge Research Laboratory Cambridge, 4, 1997.

[19] Mark JF Gales. Maximum likelihood linear transformations for hmm-based speech recognition. Computer speech & language, 12(2):75–98, 1998.

[20] LR Bahl, Peter F Brown, Peter V De Souza, and Robert L Mercer. Maximum mutual information estimation of hidden markov model parameters for speech recognition. In proc. icassp, volume 86, pages 49–52, 1986.

[21] Daniel Povey, Dimitri Kanevsky, Brian Kingsbury, Bhuvana Ramabhadran, George Saon, and Karthik Visweswariah. Boosted mmi for model and feature-space discriminative training. In 2008 IEEE International Conference on Acoustics, Speech and Signal Processing, pages 4057–4060. IEEE, 2008.

[22] Daniel Povey and Philip C Woodland. Minimum phone error and i-smoothing for improved discriminative training. In Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on, volume 1, pages I–105. IEEE, 2002.

[23] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In 2013 IEEE international conference on acoustics, speech and signal processing, pages 6645–6649. IEEE, 2013.

[24] Hang Su and Haoyu Chen. Experiments on parallel training of deep neural network using model averaging. arXiv preprint arXiv:1507.01239, 2015.

A

Experimentos y Resultados

Los siguientes experimentos se han realizado con modelos de lenguaje cerrados:

Sistema Test set	Auto-test 10-20 dB	Auto-test 20-30 dB	Auto-test 30-40 dB	Auto-test 40-99 dB	Test 30-40 dB	Test ruidoso
Kaldi (19.200)	92,77	97,33	97,42	97,53	96,73	94,13
Kaldi (192.000)	92,60	96,69	97,12	97,61	96,09	90,76

Cuadro A.1: Comparativa Kaldi frente a número de gaussianas para Auto-test y test

Sistema Test set	Pitch 0-100 Hz	Pitch 100-140 Hz	Pitch 140-180 Hz	Pitch 180-220 Hz	Pitch 220-máx Hz
Kaldi (19.200)	92,08	93,81	92,72	96,10	78,65
Kaldi (192.000)	93,14	94,34	92,01	94,09	74,75

Cuadro A.2: Comparativa Kaldi frente a número de gaussianas para pitch

Sistema / Test set	Telediarios	Economía	Deporte	Moda
Kaldi (19.200)	**78,37**	60,74	69,26	41,5
Kaldi (192.000)	79,84	66,76	63,01	40,99

Cuadro A.3: Comparativa Kaldi frente a número de gaussianas para vídeos de internet

Kaldi	ΔΔ tri2a - REF	MLLT+LDA tri2b	MMI tri2b-mmi	MMI+Boost tri2b-bmmi	MMI+MPE tri2b-mpe
Auto-test 10-20 dB	92,77	92,42	97,66	97,41	93,76
Auto-test 20-30 dB	97,33	97,38	98,73	98,59	97,89
Auto-test 30-40 dB	97,42	97,88	98,94	98,90	98,28
Auto-test 40-99 dB	97,53	97,58	98,75	98,65	98,08
Test 30-40 dB	96,73	97,54	98,54	98,54	98,03
Test ruidoso	94,13	94,60	97,26	97,18	96,42

Cuadro A.4: Resultados Auto-test y test para las distintas transformadas con 19.200 gaussianas

Kaldi	ΔΔ tri2a - REF	MLLT+LDA tri2b	MMI tri2b-mmi	MMI+Boost tri2b-bmmi	MMI+MPE tri2b-mpe
Auto-test 10-20 dB	92,53	91,00	98,41	98,44	94,01
Auto-test 20-30 dB	96,69	97,06	99,44	99,41	98,89
Auto-test 30-40 dB	97,12	97,58	99,62	99,64	99,06
Auto-test 40-99 dB	97,61	97,22	99,23	99,24	98,88
Test 30-40 dB	96,09	96,16	98,52	98,52	98,06
Test ruidoso	90,76	91,26	96,37	96,45	95,87

Cuadro A.5: Resultados Auto-test y test para las distintas transformadas con 192.00 gaussianas

Kaldi	ΔΔ tri2a - REF	MLLT+LDA tri2b	MMI tri2b-mmi	MMI+Boost tri2b-bmmi	MMI+MPE tri2b-mpe
Pitch 0-100 Hz	92,08	93,77	94,55	94,51	94,12
Pitch 100-140 Hz	93,81	98,28	99,10	99,07	98,68
Pitch 140-180 Hz	92,72	94,76	95,25	95,28	94,95
Pitch 180-220 Hz	96,10	98,15	98,97	98,91	98,66
Pitch 220-máx Hz	78,65	82,13	83,75	83,97	83,37

Cuadro A.6: Resultados de pitch para las distintas transformadas con 19.200 gaussianas

Kaldi	ΔΔ tri2a - REF	MLLT+LDA tri2b	MMI tri2b-mmi	MMI+Boost tri2b-bmmi	MMI+MPE tri2b-mpe
Pitch 0-100 Hz	93,14	94,16	95,10	95,01	94,67
Pitch 100-140 Hz	94,34	96,74	99,57	99,57	98,85
Pitch 140-180 Hz	92,01	93,09	95,06	95,09	93,87
Pitch 180-220 Hz	94,09	95,18	99,32	99,32	97,64
Pitch 220-máx Hz	74,75	78,83	80,71	80,63	80,93

Cuadro A.7: Resultados de pitch para las distintas transformadas con 192.000 gaussianas

Kaldi	ΔΔ tri2a - REF	MLLT+LDA tri2b	MMI tri2b-mmi	MMI+Boost tri2b-bmmi	MMI+MPE tri2b-mpe
Telediarios	78,37	76,75	82,39	82,15	82,17
Economía	60,74	59,71	63,77	63,36	64,57
Deporte	69,26	65,33	70,36	69,67	73,05
Moda	41,50	36,76	39,16	37,48	37,21

Cuadro A.8: Resultados pruebas vídeos internet para las distintas transformadas con 19.200 gaussianas

Kaldi	ΔΔ tri2a - REF	MLLT+LDA tri2b	MMI tri2b-mmi	MMI+Boost tri2b-bmmi	MMI+MPE tri2b-mpe
Telediarios	79,84	79,83	79,23	79,21	79,61
Economía	66,76	62,88	57,40	57,51	60,14
Deporte	63,01	62,94	59,90	59,25	64,06
Moda	40,99	37,08	29,72	29,35	32,96

Cuadro A.9: Resultados pruebas vídeos internet para las distintas transformadas con 192.000 gaussianas

Sistema / Test set	Telediarios
Kaldi (19.200)	61,17
Kaldi (192.000)	63,16

Cuadro A.10: Resultados pruebas telediarios con sistema base para LM abierto

B

Presupuesto

1. Ejecución Material
   - Servidor ..... 2.000 €
   - Tarjetas gráficas y fuente de alimentación ..... 2.000 €
   - Material de oficina ..... 150 €
   - Total de ejecución material ..... 4.150 €
2. Gastos generales
   - 16 % sobre Ejecución Material ..... 664 €
3. Beneficio Industrial
   - 6 % sobre Ejecución Material ..... 249 €
4. Honorarios Proyecto
   - 1.800 horas a 15 €/hora ..... 27.000 €
5. Material fungible
   - Gastos de impresión ..... 80 €
   - Encuadernación ..... 30 €
6. Subtotal del presupuesto
   - Subtotal del presupuesto ..... 32.173 €
7. I.V.A aplicable
   - 21 % Subtotal Presupuesto ..... 6.756,33 €

8. Total presupuesto

   - Subtotal Presupuesto . . . . . 38.929,33 €

Madrid, Septiembre de 2016

El Ingeniero Jefe de Proyecto

Fdo.: Carolina Camacho Costumero

Ingeniero de Telecomunicación

C

Pliego de condiciones

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, del DESARROLLO DE UN SISTEMA DE RECONOCIMIENTO DE HABLA NATURAL BASADO EN REDES NEURONALES PROFUNDAS. En lo que sigue, se supondrá que el proyecto ha sido encargado por una empresa cliente a una empresa consultora con la finalidad de realizar dicho sistema. Dicha empresa ha debido desarrollar una línea de investigación con objeto de elaborar el proyecto. Esta línea de investigación, junto con el posterior desarrollo de los programas está amparada por las condiciones particulares del siguiente pliego.

Supuesto que la utilización industrial de los métodos recogidos en el presente proyecto ha sido decidida por parte de la empresa cliente o de otras, la obra a realizar se regulará por las siguientes:

Condiciones generales.

1. La modalidad de contratación será el concurso. La adjudicación se hará, por tanto, a la proposición más favorable sin atender exclusivamente al valor económico, dependiendo de las mayores garantías ofrecidas. La empresa que somete el proyecto a concurso se reserva el derecho a declararlo desierto.

2. El montaje y mecanización completa de los equipos que intervengan será realizado totalmente por la empresa licitadora.

3. En la oferta, se hará constar el precio total por el que se compromete a realizar la obra y el tanto por ciento de baja que supone este precio en relación con un importe límite si este se hubiera fijado.

4. La obra se realizará bajo la dirección técnica de un Ingeniero Superior de Telecomunicación, auxiliado por el número de Ingenieros Técnicos y Programadores que se estime preciso para el desarrollo de la misma.

5. Aparte del Ingeniero Director, el contratista tendrá derecho a contratar al resto del personal, pudiendo ceder esta prerrogativa a favor del Ingeniero Director, quien no estará obligado a aceptarla.

6. El contratista tiene derecho a sacar copias a su costa de los planos, pliego de condiciones y presupuestos. El Ingeniero autor del proyecto autorizará con su firma las copias solicitadas por el contratista después de confrontarlas.

7. Se abonará al contratista la obra que realmente ejecute con sujeción al proyecto que sirvió de base para la contratación, a las modificaciones autorizadas por la superioridad o a las órdenes que con arreglo a sus facultades le hayan comunicado por escrito al Ingeniero Director de obras siempre que dicha obra se haya ajustado a los preceptos de los pliegos de condiciones, con arreglo a los cuales, se harán las modificaciones y la valoración de las diversas unidades sin que el importe total pueda exceder de los presupuestos aprobados. Por consiguiente, el número de unidades que se consignan en el proyecto o en el presupuesto, no podrá servirle de fundamento para entablar reclamaciones de ninguna clase, salvo en los casos de rescisión.

8. Tanto en las certificaciones de obras como en la liquidación final, se abonarán los trabajos realizados por el contratista a los precios de ejecución material que figuran en el presupuesto para cada unidad de la obra.

9. Si excepcionalmente se hubiera ejecutado algún trabajo que no se ajustase a las condiciones de la contrata pero que sin embargo es admisible a juicio del Ingeniero Director de obras, se dará conocimiento a la Dirección, proponiendo a la vez la rebaja de precios que el Ingeniero estime justa y si la Dirección resolviera aceptar la obra, quedará el contratista obligado a conformarse con la rebaja acordada.

10. Cuando se juzgue necesario emplear materiales o ejecutar obras que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras obras o materiales análogos si los hubiere y cuando no, se discutirán entre el Ingeniero Director y el contratista, sometiéndolos a la aprobación de la Dirección. Los nuevos precios convenidos por uno u otro procedimiento, se sujetarán siempre al establecido en el punto anterior.

11. Cuando el contratista, con autorización del Ingeniero Director de obras, emplee materiales de calidad más elevada o de mayores dimensiones de lo estipulado en el proyecto, o sustituya una clase de fabricación por otra que tenga asignado mayor precio o ejecute con mayores dimensiones cualquier otra parte de las obras, o en general, introduzca en ellas cualquier modificación que sea beneficiosa a juicio del Ingeniero Director de obras, no tendrá derecho sin embargo, sino a lo que le correspondería si hubiera realizado la obra con estricta sujeción a lo proyectado y contratado.

12. Las cantidades calculadas para obras accesorias, aunque figuren por partida alzada en el presupuesto final (general), no serán abonadas sino a los precios de la contrata, según las condiciones de la misma y los proyectos particulares que para ellas se formen, o en su defecto, por lo que resulte de su medición final.

13. El contratista queda obligado a abonar al Ingeniero autor del proyecto y director de obras así como a los Ingenieros Técnicos, el importe de sus respectivos honorarios facultativos por formación del proyecto, dirección técnica y administración en su caso, con arreglo a las tarifas y honorarios vigentes.

14. Concluida la ejecución de la obra, será reconocida por el Ingeniero Director que a tal efecto designe la empresa.

15. La garantía definitiva será del 4 % del presupuesto y la provisional del 2 %.

16. La forma de pago será por certificaciones mensuales de la obra ejecutada, de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.

17. La fecha de comienzo de las obras será a partir de los 15 días naturales del replanteo oficial de las mismas y la definitiva, al año de haber ejecutado la provisional, procediéndose si no existe reclamación alguna, a la reclamación de la fianza.

18. Si el contratista al efectuar el replanteo, observase algún error en el proyecto, deberá comunicarlo en el plazo de quince días al Ingeniero Director de obras, pues transcurrido ese plazo será responsable de la exactitud del proyecto.

19. El contratista está obligado a designar una persona responsable que se entenderá con el Ingeniero Director de obras, o con el delegado que éste designe, para todo relacionado con ella. Al ser el Ingeniero Director de obras el que interpreta el proyecto, el contratista deberá consultarle cualquier duda que surja en su realización.

20. Durante la realización de la obra, se girarán visitas de inspección por personal facultativo de la empresa cliente, para hacer las comprobaciones que se crean oportunas. Es obligación del contratista, la conservación de la obra ya ejecutada hasta la recepción de la misma, por lo que el deterioro parcial o total de ella, aunque sea por agentes atmosféricos u otras causas, deberá ser reparado o reconstruido por su cuenta.

21. El contratista, deberá realizar la obra en el plazo mencionado a partir de la fecha del contrato, incurriendo en multa, por retraso de la ejecución siempre que éste no sea debido a causas de fuerza mayor. A la terminación de la obra, se hará una recepción provisional previo reconocimiento y examen por la dirección técnica, el depositario de efectos, el interventor y el jefe de servicio o un representante, estampando su conformidad el contratista.

22. Hecha la recepción provisional, se certificará al contratista el resto de la obra, reservándose la administración el importe de los gastos de conservación de la misma hasta su recepción definitiva y la fianza durante el tiempo señalado como plazo de garantía. La recepción definitiva se hará en las mismas condiciones que la provisional, extendiéndose el acta correspondiente. El Director Técnico propondrá a la Junta Económica la devolución de la fianza al contratista de acuerdo con las condiciones económicas legales establecidas.

23. Las tarifas para la determinación de honorarios, reguladas por orden de la Presidencia del Gobierno el 19 de Octubre de 1961, se aplicarán sobre el denominado en la actualidad "Presupuesto de Ejecución de Contratаz anteriormente llamado "Presupuesto de Ejecución Material"que hoy designa otro concepto.

Condiciones particulares.

La empresa consultora, que ha desarrollado el presente proyecto, lo entregará a la empresa cliente bajo las condiciones generales ya formuladas, debiendo añadirse las siguientes condiciones particulares:

1. La propiedad intelectual de los procesos descritos y analizados en el presente trabajo, pertenece por entero a la empresa consultora representada por el Ingeniero Director del Proyecto.

2. La empresa consultora se reserva el derecho a la utilización total o parcial de los resultados de la investigación realizada para desarrollar el siguiente proyecto, bien para su publicación o bien para su uso en trabajos o proyectos posteriores, para la misma empresa cliente o para otra.

3. Cualquier tipo de reproducción aparte de las reseñadas en las condiciones generales, bien sea para uso particular de la empresa cliente, o para cualquier otra aplicación, contará con autorización expresa y por escrito del Ingeniero Director del Proyecto, que actuará en representación de la empresa consultora.

4. En la autorización se ha de hacer constar la aplicación a que se destinan sus reproducciones así como su cantidad.

5. En todas las reproducciones se indicará su procedencia, explicitando el nombre del proyecto, nombre del Ingeniero Director y de la empresa consultora.

6. Si el proyecto pasa la etapa de desarrollo, cualquier modificación que se realice sobre él, deberá ser notificada al Ingeniero Director del Proyecto y a criterio de éste, la empresa consultora decidirá aceptar o no la modificación propuesta.

7. Si la modificación se acepta, la empresa consultora se hará responsable al mismo nivel que el proyecto inicial del que resulta el añadirla.

8. Si la modificación no es aceptada, por el contrario, la empresa consultora declinará toda responsabilidad que se derive de la aplicación o influencia de la misma.

9. Si la empresa cliente decide desarrollar industrialmente uno o varios productos en los que resulte parcial o totalmente aplicable el estudio de este proyecto, deberá comunicarlo a la empresa consultora.

10. La empresa consultora no se responsabiliza de los efectos laterales que se puedan producir en el momento en que se utilice la herramienta objeto del presente proyecto para la realización de otras aplicaciones.

11. La empresa consultora tendrá prioridad respecto a otras en la elaboración de los proyectos auxiliares que fuese necesario desarrollar para dicha aplicación industrial, siempre que no haga explícita renuncia a este hecho. En este caso, deberá autorizar expresamente los proyectos presentados por otros.

12. El Ingeniero Director del presente proyecto, será el responsable de la dirección de la aplicación industrial siempre que la empresa consultora lo estime oportuno. En caso contrario, la persona designada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.