Facultat d'Informàtica de Barcelona (FIB)
Universitat Politècnica de Catalunya (UPC) –
BarcelonaTech

Sistema de transcripción y análisis de voz empleando técnicas de Procesado de Lenguaje Natural y Machine Learning

Alumno:
Javier López Calderón

Estudio:
Grado en Ingeniería Informática especialidad en Computación

Director:
Javier Béjar Alonso
Dept. de Ciencias de la Computación

Enero 2019

Resum

Aquest projecte és l'etapa final del Grau en Enginyeria Informàtica de la Facultat d'Informàtica de Barcelona en la Universitat Politècnica de Barcelona (Barcelonatech). Tenint això en compte, el treball deu reflectir i posar en pràctica tots els coneixements adquirits durant la carrera i, en especial, els coneixements de l'especialitat de Computació.
Aquest treball tracta sobre la investigació, desenvolupament i implementació d'un sistema que permeti deixar constància i gestionar, de forma àgil i automàtica, les reunions en empreses.
Per aconseguir aquest objectiu, s'ha desenvolupat una aplicació mòbil que permet gravar una reunió i aplicar-li diversos algoritmes de deep learning, amb la finalitat d'obtenir la transcripció de la reunió i senyalar, en la transcripció, qui ha dit cada frase.

Resumen

Este proyecto es la etapa final del Grado en Ingeniería Informática de la Facultad de Informática de Barcelona en la Universidad Politécnica de Barcelona (Barcelonatech). Teniendo esto en cuenta, el trabajo debe reflejar y poner en práctica todos los conocimientos adquiridos durante la carrera y, en especial, los conocimientos de la especialidad de Computación.
Este proyecto trata sobre la investigación, desarrollo e implementación de un sistema que permite dejar constancia y gestionar, de forma ágil y automática, las reuniones en empresas.
Para lograr dicho objetivo se ha desarrollado una aplicación móvil que permite grabar una reunión y aplicarle diversos algoritmos de deep learning, con la finalidad de obtener la transcripción de la reunión y señalar, en la transcripción, quién ha dicho cada frase.

Abstract

This project is the final stage of the Degree in Computer Engineering of the Barcelona School of Informatics at the Technical University of Catalonia (Barcelonatech). Taking this into account, the work should reflect and put into practice all the knowledge acquired during the career and, especially, knowledge of the specialty of Computing.
This project deals with the research, development and implementation of a system that allows to register and manage, in an agile and automatic way, meetings in companies.
To achieve this goal, a mobile application has been developed that allows recording a meeting and applying several deep learning algorithms, to obtain the transcription of the meeting and mark, in the transcript, who said each phrase.

Agradecimientos

En primer lugar, quiero agradecer a mi tutor de proyecto Javier Béjar Alonso por todo el tiempo y dedicación que semanalmente me ha dedicado, gracias a sus consejos me ha permitido completar este proyecto profundizando en temas que nunca había tocado como el tratamiento de sonidos con inteligencia artificial.

En segundo lugar, quiero agradecer a mis padres Enrique de la paz López y Ana Calderón, simplemente les debo todo, ellos son el motivo por el cual puedo romper cualquier límite, afrontar cualquier problema y siempre han estado cuando más lo he necesitado; también quiero agradecer a mi familia, en especial a mi hermana, por las largas horas arreglando el mundo y por todas las ideas que hemos tenido juntos y que tendremos.

En tercer lugar, quiero agradecer, por todo el apoyo incondicional, a mi actual pareja Laura Roldán, una bellísima persona cargada de paciencia, que me ayuda en todas mis ideas locas y que me permite levantarme cada vez que me caigo, sin su apoyo no hubiera sido posible enfrentarme a retos cada vez más difíciles.

En cuarto lugar, quiero agradecer a mis dos mejores amigos, Pau Gallardo y Laura González, ellos son un pilar sólido que me ha aportado tranquilidad y seguridad, porque estando con ellos tengo la sensación de que no existe ninguna dificultad que no pueda superarse.

Finalmente, quiero agradecer a todos los compañeros del grado que han trabajado conmigo, por todas las dificultades superadas y las largas noches de estudio.

Simplemente, gracias.

Índice general

Índice de figuras

Índice de cuadros

1. Preámbulo

1.1. Conceptos Previos

En esta sección se van a definir los conceptos generales mínimos para poder comprender mejor los temas que se tratan a lo largo del trabajo.

1.1.1. Algoritmo

Un algoritmo es, en esencia, toda secuencia de pasos que permiten resolver un problema en tiempo finito, es decir, siempre termina y ofrece una solución.

1.1.2. Application programming interface (API)

Es un conjunto de funciones o métodos que ofrece y expone una librería o sistema externo para comunicar, delegar o abstraer una determinada funcionalidad.

1.1.3. Red neuronal (RN)

Las redes neuronales son un modelo matemático y computacional, inspirado en el funcionamiento de las neuronas del cerebro del ser humano. Se caracterizan por tener una capa de entrada, cero o más capas ocultas, una capa de salida y necesitar un proceso de entrenamiento para que aprendan a generalizar el concepto o patrón a identificar.

1.1.4. Red neuronal recurrente (RNN)

Es un tipo específico de red neuronal artificial donde los nodos de la red forman un grafo dirigido y la salida de la red puede ser entrada de otro nodo de la misma red. Esto suele ser utilizado para detectar patrones y modelar secuencias temporales, siendo ideales para tareas como de visión por computador y reconocimiento de voz.

1.1.5. Red neuronal convolucional (CNN)

Es otro tipo específico de red neuronal artificial, se caracteriza por tener diversas capas de mapas de características(filtros) junto a capas de muestreo en ventana que permiten tratar una entrada grande generalizando las partes más relevantes y, posteriormente, emplear estas abstracciones en una red neuronal que, finalmente, proporciona el resultado de la red.

1.1.6. Red neuronal artificial convolucional siamesa

Es un subtipo de las redes convolucionales que se caracteriza por tener duplicada parte de la red neuronal desde la entrada hasta cierto punto, donde se convergen las dos subredes neuronales en una sola empleando una función. Se suele emplear para comparar dos muestras y se suelen unir con funciones de distancia como la distancia euclídea o la distancia coseno.

1.1.7. Sincronización forzada

La sincronización forzada es un procedimiento que realiza un programa automáticamente para unir una voz con su correspondiente texto o subtítulo.

1.2. Repaso breve de la historia del reconocimiento de voz

El reconocimiento de voz tal y como lo conocemos empezó en 1952, ese año Bell Labs desarrolló un sistema capaz de identificar los dígitos de forma automática, a este sistema lo denominaron "Audrey"[1].

Más adelante, los años 60, aparecieron numerosos algoritmos y técnicas. Por un lado, los soviéticos inventaron el algoritmo de "Dynamic Time Warping"(DTW) capaz de identificar 200 palabras segmentando el audio en fragmentos de 10ms. En paralelo, Leonard Baun desarrolló las matemáticas de las cadenas de markov, siendo la base de los modelos probabilísticos "Hidden Markov Models"(HMM)[2] desarrollados por primera vez por James & Janet Baker[3] poco tiempo después.

A finales de los 70, principios de los 80, IBM mejoró las técnicas existentes presentando "Tangora", un sistema de reconocimiento de voz basado en HMM que permitía reconocer 20.000 palabras, desplazando a las técnicas existentes que empleaban DTW; este sistema contaba con un elaborado modelo de lenguaje que incrementó notablemente la precisión de los algoritmos y técnicas de la época.

A finales de los años 80, Slava M. Katz desarrolló los modelos de lenguaje N-gram, que modelan la probabilidad condicionada de una palabra basándose en N-1 palabras de la secuencia; estos modelos de lenguaje mejoraron en gran medida los sistemas de reconocimiento de voz existentes hasta la fecha.

A principios de los años 90, apareció en el mercado Dragon Dictation[3], siendo una de las primeras empresas en comercializar a nivel general un sistema de reconocimiento de voz, a mediados de los 90, Dragon Dictation fue comprada por Lernout & Hauspie, dando pié al desarrollo posterior de uno de los algoritmos de reconocimientos más extendidos de la historia, el algoritmo de reconocimiento de voz integrado con Windows XP que, debido a su masificación internacional, convirtió a Lernout & Hauspie en los líderes de la industria en tecnologías de reconocimiento de voz.

Más adelante, Lernout & Hauspie desapareció debido a un escándalo contable en 2001, vendiendo su tecnología de reconocimiento de voz a ScanSoft, quienes pasaron a desarrollar y mantener

el algoritmo de reconocimiento de voz hasta su cambio de nombre en 2005 por Nuance.

En 2005, ScanSoft, ahora Nuance, se alió con Apple para ofrecer las tecnologías de reconocimiento de voz en los productos de la empresa de la manzana, fruto de esta unión comercial, Apple obtuvo la licencia del software para proporcionar la capacidad de reconocimiento de voz a su asistente de voz virtual Siri[4].

En 2007, Google empezó a contratar investigadores de Nuance[5], la finalidad de ello fue desarrollar un servicio de directorio telefónico que permitió a Google recopilar cientos de horas de voz que posteriormente emplearía para desarrollar "Google Voice Search", con su propio sistema de reconocimiento de voz; posteriormente, Google integró su tecnología en la mayoría de sus productos: Maps, Earth, Google Translate app y muchos más.

1.3. Estado del arte

A continuación se mencionan los dos ámbitos del estado del arte, por un lado, la parte enfocada a hablar sobre el estado actual en investigación y, por otro, el uso que se está dando a tecnologías y algoritmos por parte de las empresas en general.

1.3.1. Investigación

La transcripción de voz a texto es un problema clásico[6] del procesado de lenguaje natural como se ha mencionado anteriormente, sin embargo, hoy en día la mayoría de empresas e instituciones están abordando el problema desde la perspectiva del aprendizaje profundo. Este enfoque permite a los investigadores centrarse en resolver el problema, empleando potentes conjuntos de datos para dejar que el sistema infiera automáticamente todos los patrones necesarios que anteriormente se realizaban a mano.

Actualmente, a fecha de la redacción de este documento, el algoritmo que más se está utilizando y el que mejor rendimiento está ofreciendo se denomina Deep Speech[7] (o variaciones del mismo), este algoritmo de aprendizaje profundo permite entrenar un modelo para realizar la tarea de transcripción de voz a texto únicamente empleando muestras de voz con su correspondiente transcripción.

Existen diversas versiones del algoritmo Deep Speech[7], la mayoría de empresas están implementando la segunda versión de este algoritmo o adaptaciones del mismo, a la espera de que Baidu research haga público el paper de la tercera iteración de este algoritmo, actualmente en investigación.

Por otro lado, el reconocimiento de interlocutor o de hablante también es un problema clásico del procesado de lenguaje natural y, como en el caso de la transcripción de voz a texto, también se están desarrollando algoritmos de aprendizaje profundo que resuelven el problema[8][9].

Uno de los enfoques con aprendizaje profundo más utilizados es la elaboración de un modelo que consta de una red neuronal artificial convolucional siamesa[10][11], que permite la entrada de dos muestras de voz, extraer sus características, aplicar una función de distancia y, finalmente, codificar la salida de la red en función de las necesidades del proyecto.

Como se puede observar, para la mayoría de tareas relacionadas con la voz, hoy en día, el estado del arte es aplicar un enfoque que permita hacer uso de redes neuronales artificiales.[10][11][12][13]

1.3.2. Uso

En la actualidad es común encontrar numerosos dispositivos que cuentan con alguna característica de reconocimiento de voz. La mayoría se centran en mejorar la experiencia de usuario simplificando la forma de interactuar con el dispositivo.

Existen diferentes áreas de aplicación para el reconocimiento de voz a nivel comercial, los más usuales son:

- Asistentes (ej. Siri[14] y Cortana[15])
- Aplicaciones de dictado y transcripción (ej. Dragon Mobile Assistant[16] y Evernote[17])
- Identificación biométrica de voz (ej. GradiantVoice[18])
- Uso y control mediante comandos de voz (ej. Cyberon voice commander[19])
- Traducción (ej. Google Translate[20])

Los asistentes permiten que un usuario pueda realizar acciones complejas con su dispositivo únicamente empleando el reconocimiento de voz.

Las aplicaciones de dictado y transcripción son las típicas aplicaciones en las que históricamente se ha aplicado el reconocimiento de voz, permiten redactar todo tipo de documentos empleando la voz.

La identificación biométrica consta de una serie de muestras de voz contra las que se compara la voz proveniente de un micrófono, si coincide en gran medida, se da por identificado al usuario.

Los comandos de voz son otro de los elementos clásicos de aplicación de las tecnologías de voz, permiten realizar acciones en un equipo o sistema únicamente empleando ordenes de voz.

Finalmente, la traducción es uno de los campos más interesantes, existen muchas formas de resolverlo, pero es usual hoy en día traducir en dos fases, utilizando una red neuronal, se codifica una entrada en un idioma y, posteriormente, otra red se encarga de descodificar en la otra lengua, además, únicamente cambiando el descodificador, se logra obtener un sistema que permite traducir un lenguaje a cualquier otro siempre y cuando existan suficientes ejemplos.

2. Contextualización del proyecto

2.1. Por qué se realiza esta aplicación

Se ha decidido realizar este proyecto esencialmente por dos motivos, el primero de ellos, para tener la oportunidad de conocer los algoritmos que se están utilizando para resolver las diferentes tareas de reconocimiento de voz.

El segundo motivo, más práctico, es la ausencia, en el mercado, de una solución asistencial completa para reuniones, que cuente con dos características útiles para este fin, la transcripción de voz a texto y la detección de las personas que participan en la reunión.

2.2. Objetivos

Se pretende desarrollar una aplicación para smartphone que permita utilizar el terminal como asistente de reuniones, con capacidad para elaborar la transcripción de la reunión e identificar a las personas que han participado en la misma, reflejando esta información en un documento PDF que puede ser integrado y compartido con otras personas o sistemas.

2.3. Alcance del trabajo

Para poder resolver con éxito este trabajo, se deben dar solución, en un mismo sistema, a tres problemas conocidos del procesado del lenguaje natural, por un lado el problema de reconocimiento de voz ("Qué" se ha dicho) para poder transcribirlo a texto, el problema de identificación de personas mediante la voz ("Quién" lo ha dicho) para poder identificarlo en la reunión y, finalmente, identificar el género de una persona mediante su voz.

Los problemas que se abordan en este proyecto pertenecen a la clase de problemas NP-HARD, dicho de otra forma, son problemas computacionales duros y para los cuales no existe un algoritmo que los resuelva de forma completa en tiempo razonable y eficientemente, por ello, encontrar la solución óptima es simplemente inabordable y estos problemas históricamente se han resuelto empleando técnicas que aproximan soluciones razonables usando probabilidad.

Teniendo en cuenta lo anterior, se va a desarrollar una aplicación para smartphones Android con las siguientes características:

- Capturar audio procedente del micrófono.

- Almacenar las reuniones en la memoria externa del dispositivo.
- Envío y procesado de las reuniones en un servidor.
- Obtener documentos PDF con la información obtenida de las reuniones.
- Compartir las reuniones y los documentos PDF de forma sencilla.

Para poder maximizar la cantidad de usuarios que podrían utilizar la aplicación de este trabajo, se ha decidido desarrollar la misma empleando compatibilidad con Android 6.1 Lollipop en adelante, permitiendo que más del 80 % de los usuarios de Android puedan instalar y utilizar sin impedimentos la tecnología desarrollada en este trabajo.

Debido a la alta compatibilidad que se ofrece, hay que tener en cuenta las reducidas capacidades de los terminales de gama baja y media, por tanto, se ha decidido trasladar las partes que requieren más capacidad de cómputo a un servidor externo que se encarga de todos estos procesos y expone una sencilla API.

Dicho servidor, únicamente ejercerá las siguientes tareas:

- Login y registro de empresas.
- Recepción de reuniones y muestras de voz.
- Detección y corte de sonido usando silencios.
- Transcripción de voz a texto.
- Detección de interlocutores.
- Comparación de interlocutores con muestras.
- Generación de fichero PDF.

Respecto a los modelos que se van a emplear para resolver los problemas del proyecto, simplemente se implementará el mejor algoritmo/modelo obtenido teniendo en cuenta las restricciones temporales y de procesamiento existentes.

2.4. Metodología

En esta parte se van a tratar todos los aspectos del trabajo relacionados con la gestión y consecución de los objetivos.

2.4.1. Gestión

La gestión de las tareas durante el desarrollo se ha realizado con ciclos cortos de Scrum[21] con una duración de una semana entre ellos, el objetivo es realizar entregas parciales de funcionalidad, cada semana, al director del trabajo.

Esta forma de afrontar el proyecto permite mantener un seguimiento con cierta tolerancia a cambios imprevistos y añadir o corregir partes del trabajo que el director considere oportunos.

2.4.2. Seguimiento

El seguimiento del trabajo principalmente se ha realizado mediante una reunión semanal, en la que tanto el alumno como el director del trabajo revisan el estado del proyecto hasta ese punto, acuerdan los objetivos para la siguiente semana y se debate, si fuera necesario, sobre posibles problemas, incidencias y soluciones.

No obstante, se han dispuesto de otros medios de seguimiento del proyecto como un panel Kanban en Trello con todas las tareas y su estado, para que visualmente, tanto el alumno como el director puedan ver en que tarea se está trabajando actualmente. Además, también se han realizado tareas de seguimiento empleando emails.

2.4.3. Herramientas de desarrollo

Para este proyecto se han utilizado las siguientes herramientas de desarrollo:

- Git y Github
- Atom
- PyCharm
- Anaconda Navigator
- Keras
- Tensorflow
- Android Studio
- Android SDK
- Cuda Toolkit

Se ha empleado Git como sistema de control de versiones y Github como repositorio externo, como editores de texto se han empleado Atom para realizar pequeños y rápidos cambios, PyCharm para el desarrollo de la parte servidor y Android Studio para todo el código relacionado con la app de Android.

Por otro lado, se han empleado las herramientas de Anaconda Navigator para gestionar todo el entorno de desarrollo de aprendizaje profundo, contando con las herramientas Jupyter Notebook para trabajar, keras con tensorflow como herramientas para agilizar el desarrollo de modelos de aprendizaje profundo y Cuda Toolkit para poder realizar el entrenamiento de modelos de forma ágil, haciendo uso de las tecnologías de paralelización y aceleración sobre tarjetas gráficas.

2.5. A quién va dirigido

Esta aplicación está dirigida a empresas que buscan agilizar sus procesos internos, facilitar la obtención de la transcripción de sus reuniones y automatizar el proceso de incorporar este conocimiento e información a sus sistemas.

2.6. Actores implicados

El proyecto está concebido para que sea utilizado por la persona encargada de dejar constancia de las reuniones o por un responsable encargado de dicha tarea en la empresa.

La empresa se beneficia gracias al ahorro que suponen las horas de dedicación de un trabajador (anualmente) y, a su vez, este mismo trabajador no tiene que realizar la tediosa tarea de volver a escuchar la reunión desde el principio para realizar la transcripción, pudiendo reasignar este recurso a otras tareas más productivas.

3. Abordar el proyecto

3.1. Enfoque inicial

Al inicio del proyecto, la idea era realizar una aplicación de Android, con la capacidad de grabar una reunión, transcribirla e identificar a los empleados; todo ello implementado dentro de la propia aplicación, además, para cada empresa sería necesario desarrollar un modelo entrenado específicamente con las muestras del dispositivo; esto presentaba problemas diversos que tendrían que resolverse primero para poder completar el proyecto con éxito.

La primera dificultad es la capacidad de procesamiento reducido de un terminal Android medio, si se desea abarcar a la máxima cantidad de dispositivos, implica tener en consideración dispositivos con características técnicas sencillas.

La segunda dificultad encontrada fue el consumo energético que tendría que realizar el dispositivo para soportar horas de grabación, entrenamiento de un modelo de aprendizaje profundo y un proceso de cálculo para obtener los resultados.

La tercera dificultad encontrada es el entrenamiento del modelo en sí, para conseguirlo, es necesario disponer de todos los recursos del terminal, esta característica es difícil de asumir, el motivo son las librerías que permiten realizar esto de forma eficiente, la mayoría de las mismas aún no ha asumido una versión estable como para ser utilizada en entornos de producción.

Teniendo presentes estas dificultades, en las primeras decisiones que fueron tomadas, se definió que la traducción podría ser realizada por un servicio externo y que la aplicación únicamente se centre en resolver la identificación de las personas de la reunión; realizando el entrenamiento del modelo en un equipo externo e incorporándolo a la aplicación ya entrenado.

Con esta simplificación, el problema a resolver es más reducido, pero sigue siendo un problema la capacidad de procesamiento para realizar las predicciones necesarias. Además, aparece un problema añadido, entrenar el modelo en un ordenador, utilizando tensorflow por ejemplo y, una vez entrenado, incorporarlo a la aplicación desarrollada; los modelos no pueden utilizarse directamente en terminales móviles, deben ser adaptados y simplificados con la finalidad de mejorar el desempeño en estos dispositivos.

La tarea de computar un modelo con tensorflow u otra librería, actualmente es más sencilla, puesto que la gama alta de terminales Android cuenta con una característica reciente denominada Neural Networks API o NNAPI, que facilita la ejecución de modelos de aprendizaje

profundo en dispositivos que cuenten con esta característica.

3.2. Enfoque final

El enfoque inicial presentaba ciertas limitaciones que obligaban a resolver el problema de forma distinta, finalmente se decidió apostar por una arquitectura cliente-servidor que permite trasladar la dificultad computacional quitándola de la aplicación Android hacia un entorno que sí cuenta con las características y tecnologías necesarias para implementarlo en producción.

La principal idea de este enfoque es que el servidor realice todas las tareas pesadas que podrían ser un problema para los dispositivos móviles, gracias a ello, los terminales requerirán menos energía y recursos para poder utilizar todas las características de la aplicación, permitiendo que un mayor número de dispositivos sean compatibles y se pueda abarcar más mercado.

Por otro lado, debido a interés personal, la transcripción que se requiere para obtener el texto de la reunión no ha sido realizado por una fuente externa, se ha desarrollado especialmente para el proyecto con la finalidad de aprender sobre los algoritmos y técnicas actuales en materia de tratamiento de sonido.

Además, se ha decidido que, para mejorar las capacidades de integración con otros dispositivos de las empresas, la generación del documento PDF ocurra en el servidor y se distribuya un enlace que permita abrir el documento desde un navegador web, permitiendo que, de forma autorizada, puedan ser consultados y compartidos de forma ágil y simple.

Otro cambio, respecto a la versión inicial, es la estrategia de identificar a las personas en la reunión, se ha definido una estrategia genérica que permite resolver el mismo problema sin la necesidad de implementar y entrenar un modelo para cada empresa; la idea se basa en obtener un modelo que resuelva la siguiente pregunta: "¿En qué porcentaje son parecidas estas voces?", con un modelo centrado en esto, se puede utilizar para detectar todos los cambios de voz que hay en una conversación, como se detalla en la sección de Investigación, más adelante.

3.3. Tareas

Una vez tomadas la mayoría de las decisiones sobre el proyecto, es necesario dividir y planear el trabajo de forma ordenada.

En líneas generales, el proyecto puede englobarse en tres grandes tareas a resolver:

- Tareas relacionadas con deep learning
- Tareas relacionadas con la aplicación Android
- Tareas relacionadas con el servidor.

A continuación, se detallan todas las subtareas requeridas para completar el proyecto y una breve descripción de las mismas.

3.3.1. Subtareas

Tareas relacionadas con deep learning:

- Elaboración de dataset en castellano
  - Preparar el conjunto de datos para poder entrenar los algoritmos.
- Entrenar modelo para transcripción
  - Realizar el entrenamiento del modelo de transcripción de voz a texto.
- Entrenar modelo para comparación de voz
  - Realizar el entrenamiento del modelo de comparación de dos voces.
- Entrenar modelo para identificación de género
  - Realizar el entrenamiento del modelo de identificación de género.
- Volver a entrenar modelos
  - Permite asignar tiempo extra para entrenar modelos que aún no han sido suficientemente entrenados.

Tareas relacionadas con la aplicación de Android:

- Diseño de interfaz y pantallas
  - Diseñar la interfaz e interacción del usuario con la aplicación.
- Añadir sistema de fragments y permisos
  - Implementar el sistema de pantallas y solicitar los permisos necesarios.
- Implementación de todas las pantallas
  - Desarrollar visualmente todas las pantallas, preparando la funcionalidad.
- Implementar grabación del micrófono
  - Dotar a la aplicación de la capacidad de grabar del micrófono y almacenar el audio en memoria.
- Implementar comunicación con servidor
  - Desarrollar el método de comunicación con el servidor,
- Implementar envío de fichero a servidor
  - desarrollar el sistema de subir ficheros al servidor
- Implementar obtención de consentimiento
  - Añadir la característica para obtener el consentimiento de tratamiento de información.

Tareas relacionadas con el servidor:

- Implementar comunicación con Android
  - Desarrollar el método de respuesta a la comunicación para la aplicación móvil.
- Implementar subida de fichero de sonido
  - Desarrollar el sistema que permite recibir un fichero y almacenarlo en disco duro
- Implementar marcaje de silencios en un fichero
  - Desarrollar el sistema para identificar los silencios en un fichero de audio
- Implementar transcripción
  - Aplicar y usar el modelo de deep speech
- Implementar comparador de voz
  - Aplicar y usar el modelo de comparación de voces.
- Implementar identificador de voces
  - implementar la lógica que permite identificar a los empleados de la reunión.
- Implementar generador de documentos PDF
  - Generar el documento PDF con toda la información e implementar el endpoint que permite acceder al mismo de forma autorizada.
- Implementar sistema de colas
  - Distribuir el trabajo del servidor en tareas que pueden ser paralelizadas añadiendo más nodos al sistema.
- Implementar identificador de género
  - Usar el modelo de identificación de género y aplicarlo en la subida de muestras de empleados.
- Implementar derecho al olvido
  - Añadir la funcionalidad para cumplir con la legalidad vigente, permitiendo descargar y borrar toda la información del servidor.

4. Planificación

4.1. Plazos y fechas

La duración del trabajo es aproximadamente 16 semanas con dedicación de 6h diarias, dando como resultado cerca de 3 meses y medio, sin contar con el tiempo dedicado para la asignatura de gestión de proyectos.
Está previsto empezar con el desarrollo del mismo el 8 de octubre de 2018 hasta su finalización el 22 de enero de 2019, aproximadamente una semana antes del turno de lectura de TFG.

4.2. Planificación de tareas

A continuación, se van a concretar las tareas globales a resolver y las consiguientes subtareas que se van a realizar. Todas las subtareas incluyen la previsión de tiempo en horas o en días (jornada de 6h) para poder ser completadas.

4.2.1. Aprendizaje profundo

Esta tarea global contiene todas aquellas tareas relacionadas con la obtención de datasets, entrenamiento de modelos y algoritmos.

Estas tareas son tratadas de forma especial porque, debido a su naturaleza, la mayoría pueden realizarse en paralelo con otras tareas del trabajo, por ejemplo, mientras se desarrolla la aplicación de Android se pueden ir entrenando las redes neuronales; permitiendo reducir el tiempo total necesario para completar el trabajo.

Las subtareas que contiene son (ordenadas según resolución):

- Elaboración de dataset en castellano (12d)
- Entrenar modelo para transcripción (16d)
- Entrenar modelo para comparación de voz (7d)
- Entrenar modelo para identificación de género (7d)
- Volver a entrenar modelos (13d)

Recursos para completar estas tareas:
·Hardware:

- CPU AMD Ryzen 2600 3.9GHZ 6 Cores 12 Threads
- RAM 32 GB DDR4 2400
- GPU 2x Nvidia Geforce GTX 1080 Ti
- SSD 480 GB

·Software: Windows 10 Pro 64bits, Keras, Tensorflow y Nvidia Cuda SDK
·Humanos:

- Científico de Datos - Realización
- Director del proyecto - Revisión

4.2.2. Android

En esta tarea global se especifican todas las tareas necesarias para poder obtener la aplicación de Android con las características de análisis y transcripción de reuniones.

La aplicación de Android es el punto de entrada del usuario con el sistema y permite interactuar de forma sencilla con la funcionalidad.

Las subtareas que contiene son(ordenadas según resolución):

- Diseño de interfaz y pantallas (2d)
- Añadir sistema de fragments y permisos (2d)
- Implementación de todas las pantallas (3d)
- Implementar grabación del micrófono (3d)
- Implementar comunicación con servidor (4d)
- Implementar envío de fichero a servidor (3d)
- Implementar obtención de consentimiento (1d)

Recursos para completar estas tareas:
·Hardware:

- Macbook Pro retina 2017
- Smartphone Android con S.O. Lollipop o superior.

·Software:

- MacOS X
- Android SDK

·Humanos:

- Desarrollador - Realización
- Director del proyecto - Revisión

4.2.3. Servidor

En esta tarea global se especifican todas las subtareas necesarias para poder obtener el servidor para la aplicación con la funcionalidad deseada.

El servidor se ha de encargar de recibir ficheros de audio y aplicar las diversas técnicas sobre los sonidos para responder de la manera adecuada.

Las subtareas que contiene son (ordenadas según resolución):

- Implementar comunicación con Android (3d)
- Implementar subida de fichero de sonido (3d)
- Implementar marcaje de silencios en un fichero (4d)
- Implementar transcripción (9d)
- Implementar comparador de voz (4d)
- Implementar identificador de voces (4d)
- Implementar generador de documentos PDF (4d)
- Implementar sistema de colas (2d)
- Implementar identificador de género (6d)
- Implementar derecho al olvido (1d)
- Implementar derecho a la portabilidad (1d)

Recursos para completar estas tareas:
·Hardware:

- Macbook Pro retina 2017

·Software:

- MacOS X
- Python
- Flask
- CUDA Toolkit

·Humanos:

- Desarrollador - Realización
- Director del proyecto - Revisión

4.2.4. Imprevistos

Esta es una tarea global pensada para funcionar como una bolsa de horas única y exclusivamente para resolver cualquier incidencia, problema o desviación en alguna de las tareas acometidas en el proyecto.
Horas destinadas :

- Bolsa de horas (8d)

Recursos para completar estas tareas:
-Dependen de las tareas a solventar y, en todo caso, son los mismos recursos que los listado en las tareas globales descritas anteriormente.

4.2.5. Documentación

Es una tarea pensada para controlar el progreso de la documentación, las diferentes presentaciones y otros elementos que no son puramente el desarrollo y la creación del proyecto.
Estas tareas no tienen asignación horaria pues pueden realizarse durante todo proceso del desarrollo del trabajo.

4.3. Estimaciones de tiempo

Las estimaciones de tiempo asignadas al desarrollo del proyecto son las siguientes:

Desarrollo	**Horas previstas**
Elaboración dataset	72h
Android	108h
Servidor	246h
Imprevistos	48h

Tiempo total estimado de proyecto: 474h (aprox. 16 semanas)
Fecha de inicio de desarrollo: 8 de Octubre de 2018
Fecha estimada de finalización: 22 de Enero de 2019

4.3.1. Estimación de tiempo dedicado al entrenamiento de modelos

Todas las tareas de entrenamiento de modelos se realizarán en paralelo a todas las otras tareas del trabajo y no afectan a la planificación horaria, el motivo es que dichos entrenamientos no requieren de supervisión o acciones directas mientras se están computando, debido a esto, en la planificación se han añadido los tiempos que se estarán procesando dichos modelos con la finalidad de delimitar el alcance y orden de dicho entrenamiento.

La dedicación de tiempo en paralelo que se va a realizar con los modelos es:

Tarea	**Horas previstas**
Entrenamiento modelo transcripción	16d
Entrenamiento modelo comparación	7d
Entrenamiento modelo género	7d

Tiempo total estimado de entrenamientos: 180h

4.4. Control de imprevistos

La planificación de tareas ha sido realizada aplicando un margen de error y una bolsa de horas posterior para dedicar a cualquier tipo de incidencia que pueda ocurrir.

La estrategia en caso de no poder completar una tarea concreta es:

1. Notificar la incidencia al encargado/director del proyecto
2. Utilizar el tiempo restante de la tarea
3. Si no ha sido completada, simularla (mock) y continuar con la siguiente tarea prevista.
4. Más adelante, completar la tarea con las horas disponibles de la bolsa de horas.

Utilizando esta forma de controlar imprevistos se garantiza que ninguna tarea sea bloqueante y se permite volver a la misma en caso de contar con tiempo sobrante de otras tareas o utilizando la bolsa de horas.

4.5. Diagrama de Gantt

A continuación, en la figura 4.1, se muestra el diagrama de gantt con la planificación que se ha seguido durante todo el desarrollo del proyecto.

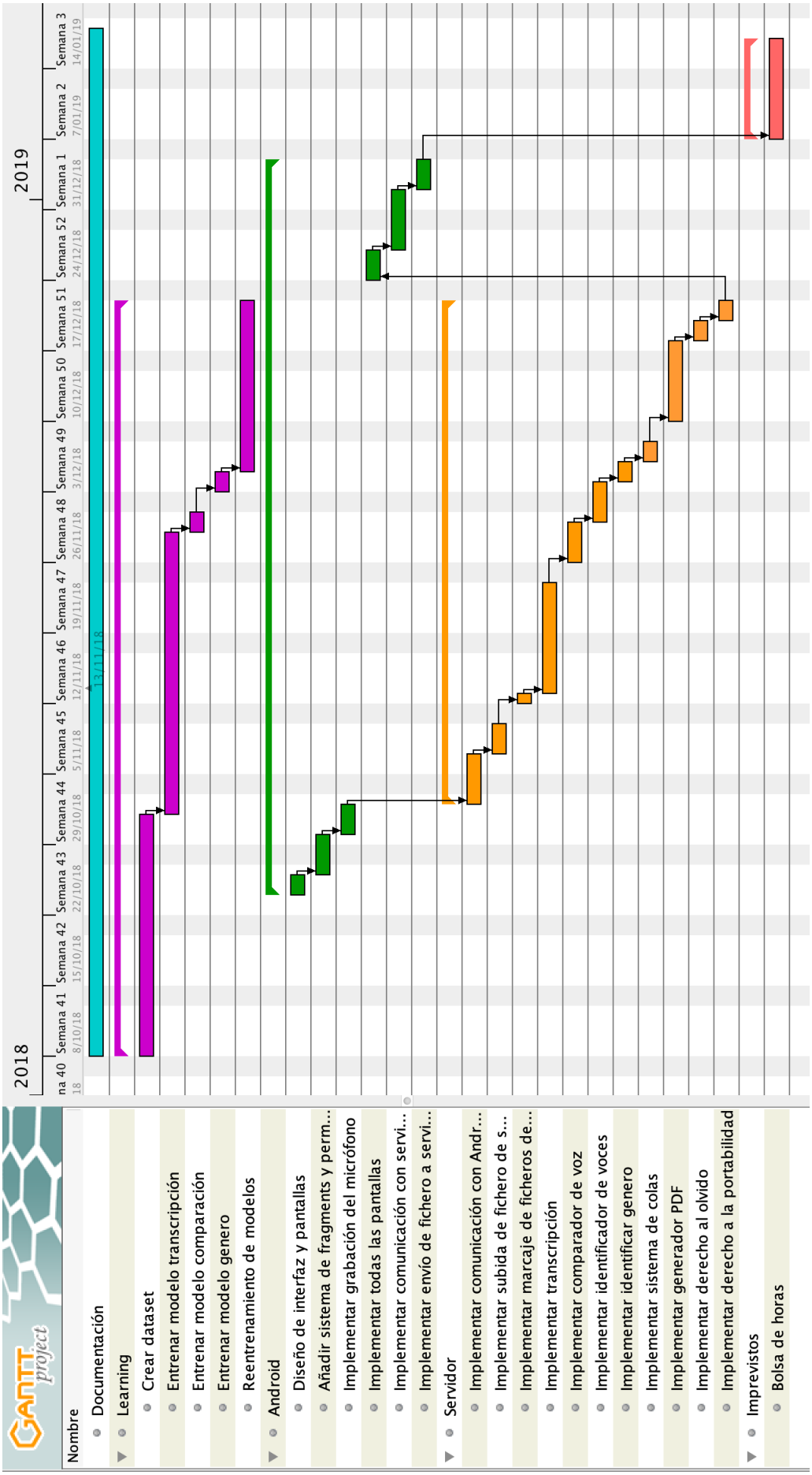

Figura 4.1: **Planificación del proyecto**

5. Gestión Económica

5.1. Presupuesto del proyecto

Para poder completar el proyecto con éxito, es necesario establecer un presupuesto que cuente con todos los gastos a los que se deben hacer frente.

En esta sección se van a detallar los gastos en hardware, software, recursos humanos, gastos indirectos e imprevistos para, finalmente, obtener un imagen completa del coste que tendría el proyecto.

5.1.1. Coste en Hardware

El coste en hardware implica todo aquel gasto en recursos físicos materiales para poder completar el proyecto, por ejemplo, ordenadores que se van a utilizar y otras herramientas necesarias.

En este proyecto se emplean 2 ordenadores, 2 tarjetas gráficas concretas y un smartphone; los costes y sus amortizaciones son los siguientes:

Producto	**Precio (euro)**	**Unidades**	**Vida Útil (años)**	**Amortización (euro)**
MacBook Pro 2017	2.030,48	1	5	62,58
PC	1.250	1	5	65,92
Nvidia Geforce GTX 1080 Ti	600	2	2	63,29
Android LG G3	160	1	2	1,73
TOTAL	4.640,48	-	-	193,52

Cuadro 5.1: Estimación del coste en hardware

El coste asociado de hardware aplicable a este proyecto es: 193,52€

5.1.2. Coste en Software

El coste en software implica todo aquel gasto en recursos inmateriales o intangibles empleados en el proyecto, por ejemplo, sistemas operativos y programas utilizados.

Este proyecto utiliza muchos elementos disponibles bajo licencia Open Source y, en consecuencia, no tienen un coste asociado directo en su uso ni requieren la compra de una licencia; se utilizan 3 sistemas operativos distintos, herramientas de gestión y control, herramientas de

desarrollo y frameworks.

Los costes y sus amortizaciones son los siguientes:

Producto	Precio (euro)	Unidades	Vida Útil (Años)	Amortización (euro)
Windows 10 Pro	250	1	-	13,18
MacOS X	20	1	-	0,76
Android	0	1	-	0
Android LG G3	0	1	-	0
Gantt Project	0	1	-	0
Trello	0	1	-	0
Google Docs	0	1	-	0
Github	0	1	-	0
Git	0	1	-	0
Atom	0	1	-	0
Flask	0	1	-	0
Cuda SDK	0	1	-	0
SocketIO	0	1	-	0
TensorFlow	0	1	-	0
Anaconda Navigator	0	1	-	0
TOTAL	270	-	-	13,95

Cuadro 5.2: Estimación del coste en software

El coste asociado de software aplicable a este proyecto es: 13,95€

5.1.3. Coste en recursos humanos

Para poder establecer correctamente el coste en recursos humanos, se han definido las tareas del proyecto y las correspondientes horas de cada rol para resolver dicha tarea.

En la siguiente tabla se encuentran las tareas, las horas de dedicación de cada perfil y la duración completa de las tareas.

Tarea	Director (h)	Desarrollador (h)	Tester (h)	Data Cientist (h)	Duración (h)
Elaboración de dataset	2	9	1	60	72
Diseño de interfaz y pantallas	1	9	2	0	12
Añadir sistema de fragments y permisos	1	9	2	0	12
Implementación de todas las pantallas	2	14	2	0	18
Implementar grabación del micrófono	1	15	2	0	18
Implementar comunicación con servidor	1	19	4	0	24
Implementar envío de fichero a servidor	1	15	2	0	18
Implementar generador de documentos PDF	2	18	4	0	24
Implementar comunicación con android	1	15	2	0	18
Implementar subida de fichero de sonido	1	16	1	0	18
Implementar marcaje de silencios en un fichero	1	19	4	0	24
Implementar transcripción	8	40	6	0	54
Implementar comparador de voz	1	21	2	0	24
Implementar identificador de voces	1	21	2	0	24
Implementar identificar género	1	31	4	0	36
TOTAL	25	271	40	60	396

Cuadro 5.3: Distribución de tareas y roles correspondientes

Utilizando la información obtenida de la tabla anterior, se han estimado los costes de cada

rol teniendo en cuenta las horas totales de cada perfil y su coste por hora, resultando en la siguiente estimación por rol:

Rol	Horas	Precio/Hora (euros)	Coste (euros)
Director de Proyecto	25	45	1.125
Desarrollador	271	35	9.485
Tester	40	30	1.000
Data Cientist	60	40	2.400
TOTAL	396	-	14.010

Cuadro 5.4: Estimación del coste en recursos humanos

El coste asociado en recursos humanos a este proyecto es: 14.010€

5.1.4. Coste indirecto

El coste indirecto comprende todos aquellos gastos que son dedicados a servicios y recursos que son necesarios para poder desarrollar el trabajo.

En este proyecto, los servicios más importantes son la electricidad para hacer funcionar todos los dispositivos y la fibra óptica para poder disponer de internet y obtener todo el software e información necesaria.

A continuación, se presentan los recursos, servicios y sus correspondientes gastos asociados:

Producto	Precio (euro)	Consumo	Coste (euro)
Electricidad	0,152€/kWh	1347kWh	204
Fibra Óptica	40	4 meses	160
Consumibles	50	-	50
TOTAL			414

Cuadro 5.5: Estimación del coste indirecto

El coste indirecto asociado a este proyecto durante todo el transcurso del mismo es: 414€

5.1.5. Coste en imprevistos

El proyecto cuenta con una bolsa de horas dedicada a hacer frente cualquier imprevisto, cada rol o perfil tiene horas estimadas basándose en las dificultades que puedan desarrollarse.

Los dos perfiles que pueden encontrarse más problemas son el de desarrollador y el de científico de datos, puesto que el primero debe crear el sistema y el segundo obtener los modelos de aprendizaje profundo adecuados, por ello, la distribución de las horas son:

Rol	Horas	Precio/Hora (euros)	Coste (euros)
Director de Proyecto	4	45	180
Desarrollador	19	35	665
Tester	5	30	150
Data Cientist	20	40	800
TOTAL	48	-	1795

Cuadro 5.6: Estimación del coste en imprevistos

El coste estimado para imprevistos asciende a: 1795€

5.1.6. Coste total estimado

A continuación se resumen todos los costes anteriores y se obtiene el gasto estimado de elaboración de todo el proyecto, teniendo en cuenta las amortizaciones, tareas, horas, imprevistos, gastos indirectos y margen de contingencia.

Se ha establecido un margen de contingencia del 7 % para hacer frente a posibles desvíos en la planificación económica que pudieran ocurrir.

Concepto	Estimación del Coste (euro)
Coste en Hardware	193,52
Coste en Software	13,95
Coste en Recursos Humanos	14.010
Costes Indirectos	414
Costes Imprevistos	1.795
Subtotal	**16.426,47**
Contingencia (7 %)	1149,85
Total	**17.576,32**

Cuadro 5.7: Estimación total del proyecto

El coste total estimado del proyecto es: 17.576,32€

5.2. Control de gasto

Durante el desarrollo del proyecto se mantendrá actualizada la planificación de gastos conforme al presupuesto para detectar posibles desviaciones o problemas que deberán ser resueltos inicialmente gracias al margen de contingencia añadido al coste total del proyecto.

6. Elaboración de los datasets

La elaboración de los datasets es uno de los elementos vitales para este proyecto, la capacidad de aprender e inferir de todos los algoritmos de aprendizaje profundo que se han utilizado dependen en gran medida de la calidad y cantidad de los mismos.

Se ha optado por obtener un dataset general que permitirá alimentar a todos los algoritmos de este proyecto, las características del mismo han sido definidas según las necesidades del algoritmo de transcripción, puesto que dicho algoritmo posee unos requisitos más amplios que los de otros modelos.

Características que debe cumplir:

- Debe incluir al menos 350.000 muestras para considerarlo viable.
- Aproximadamente 175.000 voces masculinas y 175.000 voces femeninas.
- Como mínimo debe contar con 6 voces distintas.
- Debe contener todas las palabras del diccionario.
- Solo se pueden emplear caracteres compatibles con el algoritmo de transcripción.

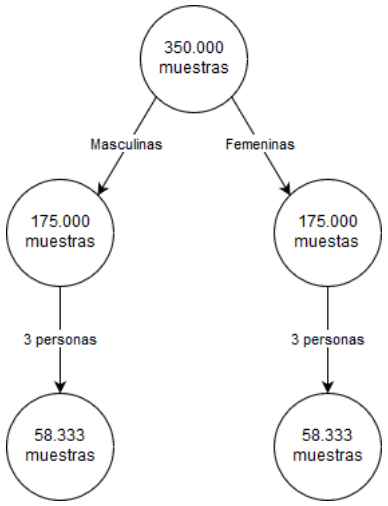

Figura 6.1: Distribución de las muestras del dataset de transcripción

Además, el tiempo asignado a obtener el dataset según la planificación son 12 días.

A continuación se va a detallar la construcción del dataset y los problemas que han surgido en la obtención del mismo.

6.1. Dataset para transcripción

En un primer momento se optó por generar manualmente cada muestra del dataset, para ello, se buscaron libros con copyright expirado o libre y se procedió a grabar frase a frase cada libro.

El primer día de la planificación fue dedicado en exclusiva a esta tarea, permitiendo, al acabar el día, hacer un pequeño balance y proyección de la capacidad de generación.

Extrapolando el número de muestras realizadas en 1 día proyectándolos sobre los 12 días restantes daban como resultado 22.000 muestras, una cifra claramente insuficiente para los objetivos y necesidades del proyecto.

Por tanto, se decidió cambiar la estrategia, hablando con el director de proyecto sobre el problema en cuestión, se propuso utilizar audio-libros como material para el dataset, a priori, la idea es razonable puesto que se puede conseguir el texto del libro y las voces.

Para poner la idea en funcionamiento, se dedicaron 4 días a buscar audio-libros con licencia libre de copyright y generar los scripts necesarios para transformar el formato de los libros y audios al formato que debe tener el dataset.

Los pasos necesarios para transformar un audio-libro al formato del dataset son:

- Obtener los capítulos del libro.
- Restringir los símbolos del libro a los caracteres admitidos por el algoritmo de transcripción.
- Aplicar sincronización forzada con el texto de un capítulo y su correspondiente voz.
- Segmentar por frases.
- Convertir el audio de cada frase al formato WAV.
- Registrar el audio y su transcripción en un fichero CSV.

El siguiente paso es evaluar cómo funciona este proceso y observar cuantas muestras pueden generarse hasta finalizar el tiempo dedicado en la planificación. La capacidad de producción de muestras con este método mejoró claramente el método anterior, se calculó que podrían obtenerse cerca de 80.000 muestras en el tiempo de planificación restante (7 días), pero esta cifra es claramente insuficiente para los requisitos mínimos del proyecto.

Además, aparecieron complicaciones que perjudicaban la obtención de muestras nuevas, en concreto, muchos de los audio-libros contenían elementos no deseados que deben ser corregidos antes de poder ser utilizados, un ejemplo de ello son las voces que leen con música de fondo para amenizar la lectura o las licencias creativas del narrador respecto al texto del libro, siendo estas un verdadero problema puesto que modificaban el texto o exageraban la pronunciación arruinando el documento.

Por lo anterior, se decidió buscar una alternativa que, en el tiempo restante de la planificación, pudiera resolver el volumen de muestras requerido; la solución fue, finalmente, auto-generar el dataset con el contenido de libros sin copyright o con licencia expirada.

Para ello, se diseñaron scripts para, dado un libro, segmentarlo en frases y hacer que un programa automático leyera la frase y se almacenase en el formato correcto.

Para conseguir esto, los pasos que realizaba el script son los siguientes:

- Seleccionar un libro y "cortarlo" por líneas.
- Limpiar cada línea para obtener una frase que contenga únicamente los símbolos del alfabeto a utilizar.
- Utilizar un programa de lectura automatica o TTS (Text to Speech)
- Redirigir la salida del programa a un fichero en lugar de a los altavoces.
- Convertir el fichero de audio al formato WAV.
- Registrar la muestra en un documento CSV.

Gracias a esta forma de obtener el dataset, en los días restantes se pudieron auto-generar las 350.000 muestras mínimas y con todos los requisitos establecidos mencionados anteriormente, produciendo un dataset con un peso total próximo a 28GB.

6.2. Dataset para comparar voces

Para poder entrenar el modelo de comparación de voces es necesario contar con un dataset específicamente pensado para esta tarea. La ventaja de este dataset es que se reutilizan las muestras obtenidas para entrenar el modelo de transcripción, de forma que únicamente deben replantearse que tipo de muestras seleccionar para entrenar.

Hay muestras de 6 personas distintas, por tanto, existen las siguientes 36 combinaciones para seleccionar muestras: [1-1, 1-2, ..., 2-1, 2-2, ..., 6-6] de las cuales, en el caso positivo nos quedaremos con las que pertenecen a la misma persona: [1-1, 2-2, 3-3, 4-4, 5-5, 6-6], mientras que, para el caso negativo, es necesario seleccionar todos los otros casos: [1-2, 1-3, ..., 2-1, 2-3, ..., 6-5].

Las características que debe tener este dataset son:

- Debe contar como mínimo con 160.000 muestras
- 80.000 deben ser afirmativas (misma persona)
- 80.000 deben ser negativas (distinta persona)
- Deben seleccionarse 13.334 muestras de los 6 casos posibles positivos.
- Deben seleccionarse 2667 muestras por cada caso de los 30 posibles negativos.
- 6.667 muestras positivas de las 13.334 de cada caso deben ser iguales.

Una vez definidas las características que requiere el dataset, únicamente hay se obtener un script que permita seleccionar las muestras y dejarlas registradas en un fichero CSV.

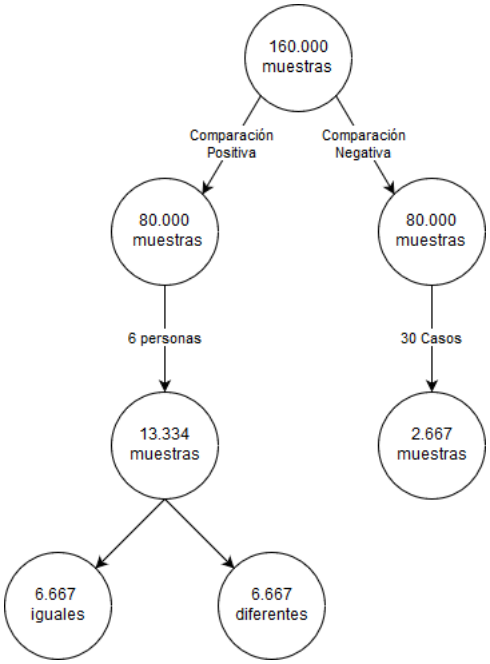

Figura 6.2: Distribución de las muestras del dataset de comparación de voz

6.3. Dataset para identificar el género

Para poder entrenar el modelo identificar el género es necesario contar con un dataset, nuevamente, específicamente pensado para esta tarea. Como ocurre con el dataset de comparación, en este dataset también se reutilizan las muestras obtenidas del entrenamiento del modelo de transcripción, de forma que únicamente deben replantearse que tipo de muestras seleccionar para entrenar.

Las características que debe tener este dataset son:

- Debe contar como mínimo con 160.000 muestras
- 80.000 deben ser masculinas
- 80.000 deben ser femeninas
- Deben seleccionarse 26.667 muestras de cada una de las 6 personas disponibles.

Una vez definidas las características que requiere el dataset, únicamente hay se obtener un script que permita seleccionar las muestras y dejarlas registradas en un fichero CSV.

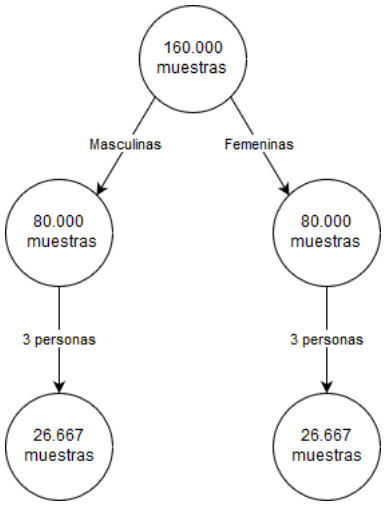

Figura 6.3: Distribución de las muestras del dataset de identificación de género

7. Investigación y pruebas de concepto

7.1. Automatizaciones y Scripts

A continuación se detallan todos los scripts utilizados para preparar los datos y formar los datasets.

7.1.1. Procesado automático de audio-libros con sincronización forzada

La idea de este script radica en la unión automática de la voz agrupada en capítulos con el texto del libro agrupado de la misma forma, el objetivo es lograr que un programa externo se encargue de generar un mapa con los tiempos donde empiezan y terminan todas las frases, proceder al corte del fichero de voz para, posteriormente, elaborar el fichero CSV con todas las muestras de voz recogidas del audio-libro.

```
import spacy
import re
import nltk
import unidecode
nltk.download('punkt')
import pandas as pd
import json
from nltk.tokenize import sent_tokenize
from pydub import AudioSegment
from aeneas.executetask import ExecuteTask
from aeneas.task import Task
from jupyter_core.paths import jupyter_data_dir

books_path = "books/"
audio_path = "audio/"
splits_path = "splits/"
outputs_path = "outputs/"
csv_path = "csv/"
book_id = "1412"
split_id = str(12)
audio_name = "audio_"+split_id

with open(books_path+book_id+'/'+book_id+'.txt', 'r') as f:
    data = f.read()
clean_data = unidecode.unidecode(data)
paragraphs = clean_data.split("\n\n")
paragraph_sentence_list = []
for paragraph in paragraphs:
    paragraph = paragraph.replace("\n", " ")
    paragraph = paragraph.replace("--", "")
    paragraph = paragraph.replace("_", "")
    paragraph = re.sub(r'[^a-zA-Z0-9_*.,?!áäöèÅÄÖÈÉçëË]', ' ', paragraph)
    paragraph_sentence_list.append(sent_tokenize(paragraph))
```

```
text = ""
count = 0
for paragraph in paragraph_sentence_list:
    if " ".join(paragraph).isupper():
        with open(books_path+"book_"+book_id+"_aeneas_data_"+str(count)+".txt", "w") as fw:
            fw.write(text)
        text = ""
        count += 1
        text += "\n".join(paragraph)
        text += "\n\n"
    elif "End of the Project Gutenberg EBook" in " ".join(paragraph):
        break
    else:
        text += "\n".join(paragraph)
        text += "\n\n"

config_string = u"task_language=spa|is_text_type=plain|os_task_file_format=json"
task = Task(config_string=config_string)
task.audio_file_path_absolute = audio_path+book_id+"/"+audio_name+".mp3"
task.text_file_path_absolute = books_path+book_id+"/book_"+book_id+"_aeneas_data_"+split_id+".txt"
task.sync_map_file_path_absolute = splits_path+book_id+"/syncmap_"+split_id+".json"

# process Task
ExecuteTask(task).execute()
# output sync map to file
task.output_sync_map_file()
book = AudioSegment.from_mp3(audio_path+book_id+"/"+audio_name+".mp3")
with open(splits_path+book_id+"/syncmap_"+split_id+".json") as f:
    syncmap = json.loads(f.read())

sentences = []
for fragment in syncmap['fragments']:
    if ((float(fragment['end'])*1000) - float(fragment['begin'])*1000) > 400:
        sentences.append({"audio":book[float(fragment['begin'])*1000:float(fragment['end'])*1000],
        "text":fragment['lines'][0]})

df = pd.DataFrame(columns=['filename','text','up_votes','down_votes','age','gender','accent','duration'])

# export audio segment
for idx, sentence in enumerate(sentences):
    text = sentence['text'].lower()
    sentence['audio'].export(outputs_path+book_id+"/audio-"+book_id+"-"+split_id+"-"+str(idx)+".mp3", format="mp3")
    temp_df = pd.DataFrame([
        {'filename':book_id+"/audio-"+book_id+"-"+split_id+"-"+str(idx)+".mp3",
         'text':text}],
        columns=['filename','text'])
    df = df.append(temp_df)

# export csv
df.to_csv(csv_path+book_id+"/"+book_id+"-"+split_id+".csv",index=False)
```

7.1.2. Lectura automática de libros

La lectura automática de libros se consigue de la siguiente forma, en primer lugar, hay que preprocesar el libro que se desea leer, la idea es leer línea a línea el libro y reemplazar todos aquellos elementos que no pertenezcan al alfabeto de deep speech (únicamente alfabeto inglés), en segundo lugar, una vez limpiada cada línea, se realiza una llamada al programa "say" de MacOSX que permite, dada una frase, obtener la misma leída por los altavoces del sistema; el script aquí mostrado redirige la salida de los altavoces a un fichero para, posteriormente, añadir esa línea al fichero CSV que contendrá todas las muestras de voz.

```
from subprocess import call
import pandas as pd
import re
import os

book_id = str(29640)
author = "soledad"
```

```
text_path = "books/"+book_id+".txt"
with open(text_path) as f:
    text_raw = f.readlines()

def clean(t):
    replaces = [('--',' '),('-',''),('_',''),('à', 'a'),('á', 'a'),('è', 'e'),('é', 'e'),('í', 'i'),('ó', 'o'),
                ('ú', 'u'),('ñ', 'n'),('ü', 'u'),('?', ''),('¿', ''),('!', ''),('¡', ''),('(', ''),(')', ''),('«', ''),
                  ('»', ''),('º', ''),('ª', ''),('%', ''),('$', ''),('#', ''),('@', ''),('[', ''),(']', ''),(';', ''),
                 (',', ''),('.', ''),('*', ''),('\n', ' '),('^', '')]

    t = t.lower()
    for (a, b) in replaces:
        t = t.replace(a, b)
    t = t.strip()
    return t

lines = []
for line in text_raw:
    if line.strip() and len(line) > 50:
        line = clean(line)
        lines.append(line)

df = pd.DataFrame(columns=['wav_filename', 'wav_filesize', 'transcript'])

for i, line in enumerate(lines):
    namefile = "audio_"+book_id+"_"+author+"_"+str(i)
    path_audio = "audio/"+book_id+"/"+namefile+".wav"
    #say "string" > file.wav
    call(["say", "-o", path_audio, "--data-format=LEF32@8000", line])
    temp_df = pd.DataFrame([
        {'wav_filename':path_audio,
         'wav_filesize':os.path.getsize(path_audio),
         'transcript':line
        }], columns=['wav_filename', 'wav_filesize', 'transcript'])
    df = df.append(temp_df)

df.to_csv("csv/"+book_id+"_"+author+".csv", index=False)
```

7.1.3. Leer de forma automática todas las palabras presentes en los libros

Este script se basa en el anterior, el procesado es muy parecido con una única diferencia, se construye un conjunto con todas las palabras presentes en el libro mientras se recorre el mismo línea a línea; cuando se termina de leer, el resultado es una bolsa de palabras sin repeticiones que puede hacerse leer de forma automática volviendo a utilizar la estructura del anterior script.

```
from subprocess import call
import pandas as pd
import re
import os
from os import listdir
from os.path import isfile, join

books_path = "books/"
onlyfiles = [f for f in listdir(books_path) if isfile(join(books_path, f)) and '.txt' in f]

def clean(t):
    replaces = [('--',' '),('-',''),('_',''),('à', 'a'),('á', 'a'),('è', 'e'),('é', 'e'),('í', 'i'),('ó', 'o'),
                ('ú', 'u'),('ñ', 'n'),('ü', 'u'),('?', ''),('¿', ''),('!', ''),('¡', ''),('(', ''),(')', ''),('«', ''),
                  ('»', ''),('º', ''),('ª', ''),('%', ''),('$', ''),('#', ''),('@', ''),('[', ''),(']', ''),(';', ''),
                 (',', ''),('.', ''),('*', ''),('\n', ' '),('^', '')]

    t = t.lower()
    for (a, b) in replaces:
        t = t.replace(a, b)
    t = t.strip()
    return t

words_set = set()
```

```
for filename in onlyfiles:
    with open(books_path + filename) as f:
        for line in f.readlines():
            if line.strip() and len(line) > 50:
                line = clean(line)
                words_set = words_set.union(set(line.split()))

df = pd.DataFrame(columns=['wav_filename', 'wav_filesize', 'transcript'])
author = "soledad"

for i, word in enumerate(words_set):
    #print(i, line)
    namefile = "audio_"+author+"_"+str(i)
    path_audio = "audio/words5/"+namefile+".wav"
    #say "string" > file.wav
    call(["say", "-o", path_audio, "--data-format=LEF32@8000", word])
    temp_df = pd.DataFrame([
        {'wav_filename':path_audio,
         'wav_filesize':os.path.getsize(path_audio),
         'transcript':word
        }], columns=['wav_filename', 'wav_filesize', 'transcript'])
    df = df.append(temp_df)
#Export to csv
df.to_csv("csv/words_"+author+".csv", index=False)
```

7.1.4. Generar los ficheros CSV por persona

Una vez que se han obtenido todos los ficheros CSV con los libros leídos, este script permite agrupar en un solo CSV todas las muestras de voz asociadas a la misma persona.

```
dataset_path = '../datasets/'
csv_path = dataset_path + 'csv/'
ready_path = csv_path + 'ready/'

def getAllfilesInFolder(path):
    listFiles = [f for f in listdir(path) if isfile(join(path, f))]
    return listFiles

def filterListByName(listFiles, name):
    return [f for f in listFiles if name in f]

def generateTrainTestFromDataframe(dataframe, train_size=0.8):
    dataframe = shuffle(dataframe)
    msk = np.random.rand(len(dataframe)) <= train_size
    return dataframe[msk], dataframe[~msk]

listFiles = getAllfilesInFolder(csv_path)
for name in ['juan', 'carlos', 'jorge', 'soledad', 'angelica', 'monica']:
    listFiltered = filterListByName(listFiles, name)
    df_by_name = pd.DataFrame()
    for file in listFiltered:
        df_file = pd.read_csv(csv_path + file, sep=',')
        df_by_name = df_by_name.append(df_file)
    df_by_name.to_csv(ready_path + name + '.csv', sep=',', index=False)
```

7.1.5. Generar el fichero CSV con todas las muestras

Este script permite agrupar los CSV de todos los ficheros de las 6 personas en un solo fichero, obteniendo el dataset que será utilizado para entrenar el algoritmo de Deep Speech para la transcripción de la voz a texto.

```
dataset_path = '../datasets/'
csv_path = dataset_path + 'csv/'
ready_path = csv_path + 'ready/'

def getAllfilesInFolder(path):
    listFiles = [f for f in listdir(path) if isfile(join(path, f))]
    return listFiles
```

```
listFiles = getAllfilesInFolder(ready_path)
df_all = pd.DataFrame()
for file in listFiles:
    df = pd.read_csv(ready_path + file, sep=',')
    df_all = df_all.append(df)
df_all.to_csv(ready_path + 'all.csv', sep=',', index=False)
```

7.2. Generar datasets

En esta sección se muestran los diversos scripts que han permitido generar los datasets para cada modelo a entrenar.

7.2.1. Transcripción

Para la transcripción no hay que generar ningún dataset especial, se utiliza la versión con todas las muestras obtenidas.

7.2.2. Comparación de voz

A continuación, se muestra el script que permite obtener el dataset para comparar voces con la estructura mostrada en el capítulo 6 figura 6.2; esencialmente permite controlar todos los casos y elaborar el fichero CSV que posteriormente se empleará para entrenar.

```
batch_pairs = [(0,1),(0,2),(0,3),(0,4),(0,5),(1,2),(1,3),(1,4),(1,5),(2,3),(2,4),(2,5),(3,4),(3,5),(4,5)]
true_elements_size = 6667
false_elements_size = 2667
csv_path = '../datasets/csv/ready/'
store_path = './files/datasets/'
dataset_path = '../datasets/'
preload_path = '../datasets/precomputed/voicecomparation/'
cols = ['wav_filename']

def read_csv_shuffle(path, cols, sep=','):
    df = pd.read_csv(path, sep=',', usecols=cols)
    df = shuffle(df)
    return df.iloc[0:true_elements_size]

def generate_true_equal(data):
    df = pd.DataFrame(columns=['file_a', 'file_b', 'result'])
    for i in range(len(data)):
        for file_name in data[i]['wav_filename']:
            df = df.append({'file_a': file_name, 'file_b': file_name, 'result':1}, ignore_index=True)
    return df

def generate_true_different(data):
    df = pd.DataFrame(columns=['file_a', 'file_b', 'result'])
    for i in range(len(data)):
        temp = pd.DataFrame(columns=['file_a', 'file_b', 'result'])
        l = data[i]['wav_filename'].tolist()
        l2 = l[::-1]
        for (a, b) in zip(l, l2):
            temp = temp.append([{'file_a': a, 'file_b': b, 'result':1}, {'file_a': b, 'file_b': a, 'result':1}],
            ignore_index=True)
        temp = shuffle(temp)
        df = df.append(temp.iloc[0:true_elements_size])
    return df

def generate_false(data, pairs):
    df = pd.DataFrame(columns=['file_a', 'file_b', 'result'])
    for (n1, n2) in pairs:
        l1 = data[n1]['wav_filename'].tolist()
        l2 = data[n2]['wav_filename'].tolist()
        temp1 = pd.DataFrame(columns=['file_a', 'file_b', 'result'])
```

```
        temp2 = pd.DataFrame(columns=['file_a', 'file_b', 'result'])
        for (a, b) in zip(l1, l2):
            temp1 = temp1.append({'file_a': a, 'file_b': b, 'result':0}, ignore_index=True)
            temp2 = temp2.append({'file_a': b, 'file_b': a, 'result':0}, ignore_index=True)
        df = df.append(temp1, ignore_index=True)
        df = df.append(temp2, ignore_index=True)
    return df

def generateTrainTestFromDataframe(dataframe, train_size=0.8):
    msk = np.random.rand(len(dataframe)) <= train_size
    return dataframe[msk], dataframe[~msk]

def make_mfcc_shape(filename, padlen=1079):
    fs, audio = wav.read(dataset_path +filename)
    r = p.mfcc(audio, samplerate=fs, numcep=26)
    t = np.transpose(r)
    X = pad_sequences(t, maxlen=padlen, dtype='float', padding='post', truncating='post').T
    return X

dataset = (read_csv_shuffle(csv_path + 'angelica.csv', cols),
           read_csv_shuffle(csv_path + 'carlos.csv', cols),
           read_csv_shuffle(csv_path + 'jorge.csv', cols),
           read_csv_shuffle(csv_path + 'juan.csv', cols),
           read_csv_shuffle(csv_path + 'monica.csv', cols),
           read_csv_shuffle(csv_path + 'soledad.csv', cols))

df = pd.DataFrame(columns=['file_a', 'file_b', 'result'])
df = df.append(generate_true_equal(dataset))
df = df.append(generate_true_different(dataset))
mini_set = (dataset[0].iloc[0:false_elements_size],
            dataset[1].iloc[0:false_elements_size],
            dataset[2].iloc[0:false_elements_size],
            dataset[3].iloc[0:false_elements_size],
            dataset[4].iloc[0:false_elements_size],
            dataset[5].iloc[0:false_elements_size])
df = df.append(generate_false(mini_set, batch_pairs))
df = shuffle(df)
train, test = generateTrainTestFromDataframe(df)
#train.to_csv(store_path + 'train.csv', sep=',', index=False)
test.to_csv(store_path + 'validation.csv', sep=',', index=False)
```

7.2.3. Identificar género

A continuación, se muestra el script que permite obtener el dataset para identificar el género con la estructura mostrada en la figura 6.3; esencialmente permite controlar todos los casos y elaborar el fichero CSV que posteriormente se empleará para entrenar.

```
male_voices = ['juan', 'carlos', 'jorge']
female_voices = ['angelica', 'monica', 'soledad']
elements_size = 26600
csv_path = '../datasets/csv/ready/'
store_path = './files/datasets/'
dataset_path = '../datasets/'
preload_path = '../datasets/precomputed/genderclassification/'
cols = ['wav_filename']

def read_csv_shuffle(path, cols, sep=','):
    df = pd.read_csv(path, sep=',', usecols=cols)
    df = shuffle(df)
    return df.iloc[0:elements_size]

def generate_elements(data, gender):
    df_ = pd.DataFrame(columns=['file', 'gender'])
    for file_name in data['wav_filename']:
        df_ = df_.append({'file': file_name, 'gender': gender}, ignore_index=True)
    return df_

def generateTrainTestFromDataframe(dataframe, train_size=0.8):
    msk = np.random.rand(len(dataframe)) <= train_size
    return dataframe[msk], dataframe[~msk]
```

```
def make_mfcc_shape(filename, padlen=1079):
    fs, audio = wav.read(dataset_path +filename)
    r = p.mfcc(audio, samplerate=fs, numcep=26)
    t = np.transpose(r)
    X = pad_sequences(t, maxlen=padlen, dtype='float', padding='post', truncating='post').T
    return X

dataset = (read_csv_shuffle(csv_path + 'angelica.csv', cols),
           read_csv_shuffle(csv_path + 'carlos.csv', cols),
           read_csv_shuffle(csv_path + 'jorge.csv', cols),
           read_csv_shuffle(csv_path + 'juan.csv', cols),
           read_csv_shuffle(csv_path + 'monica.csv', cols),
           read_csv_shuffle(csv_path + 'soledad.csv', cols))

progress = IntProgress(min=0, max=(10))
progress.description = str(progress.value)
display(progress)
df = pd.DataFrame(columns=['file', 'gender'])
df = df.append(generate_elements(dataset[0],2))
progress.value += 1
progress.description = str(progress.value)
df = df.append(generate_elements(dataset[1],1))
progress.value += 1
progress.description = str(progress.value)
df = df.append(generate_elements(dataset[2],1))
progress.value += 1
progress.description = str(progress.value)
df = df.append(generate_elements(dataset[3],1))
progress.value += 1
progress.description = str(progress.value)
df = df.append(generate_elements(dataset[4],2))
progress.value += 1
progress.description = str(progress.value)
df = df.append(generate_elements(dataset[5],2))
progress.value += 1
progress.description = str(progress.value)
df = shuffle(df)
progress.value += 1
progress.description = str(progress.value)
df.to_csv(store_path + 'all.csv', sep=',', index=False)
progress.value += 1
progress.description = str(progress.value)
train, test = generateTrainTestFromDataframe(df)
progress.value += 1
progress.description = str(progress.value)
train.to_csv(store_path + 'train.csv', sep=',', index=False)
progress.value += 1
progress.description = str(progress.value)
test.to_csv(store_path + 'test.csv', sep=',', index=False)
```

7.3. Modelos

En esta sección se detallan todos los modelo de aprendizaje profundo utilizados.

7.3.1. Modelo para transcripción

Para la transcripción se ha implementado la primera versión del algoritmo Deep Speech, la red cuenta con la siguiente estructura:

Como puede observarse en la figura 7.1, el algoritmo presenta 6 capas iniciales formadas alternativamente por una capa densa y una capa de dropout, acto seguido, el modelo cuenta con una capa LSTM y una capa densa, para terminar en una función lambda encargada de simular la matriz de probabilidad CTC.

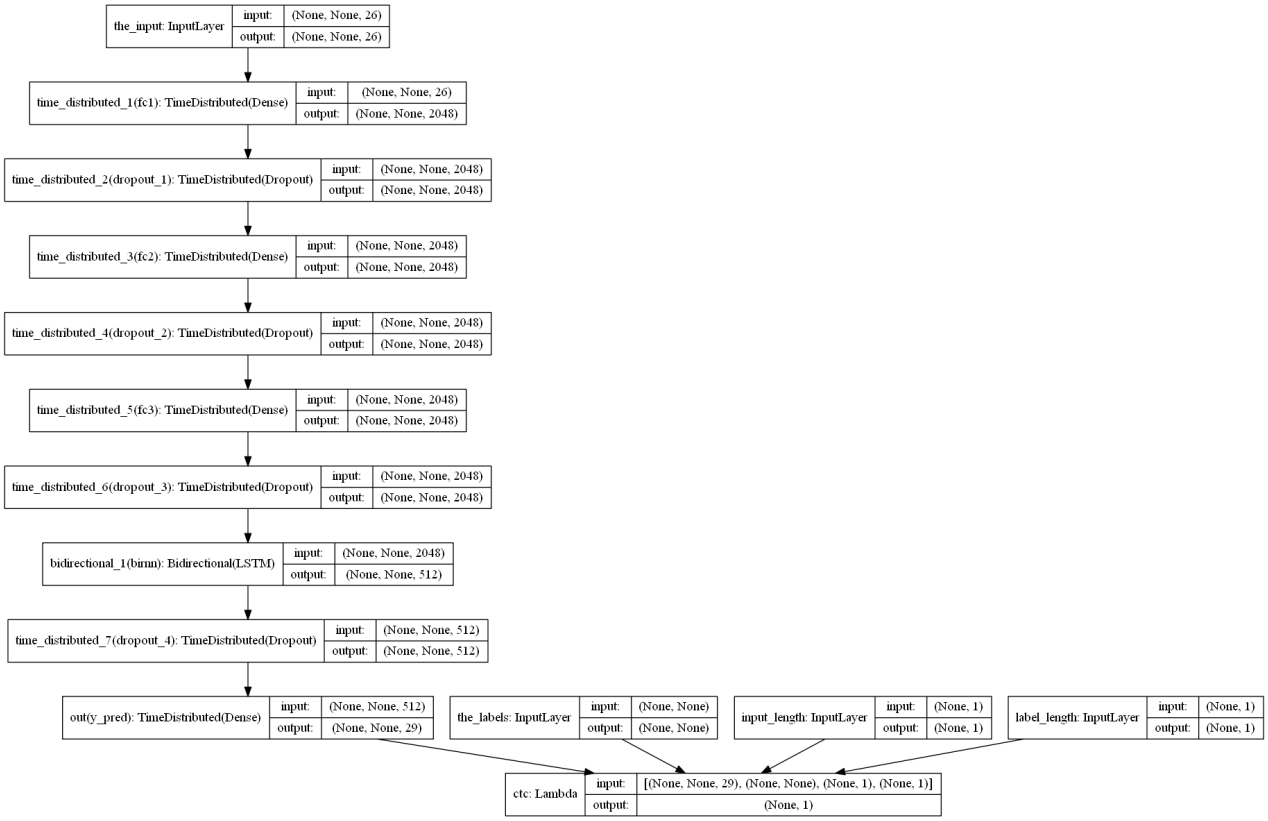

Figura 7.1: Estructura del modelo deep speech

Todas las capas implementan como función de activación ReLU, se ha utilizado como optimizador Adam y se han empleado capas con distribución debido a la naturaleza del problema que se está resolviendo.

El resultado del modelo es una matriz CTC con las probabilidades de cada carácter del alfabeto en cada instante, gracias a esto, es posible seleccionar el símbolo con mayor probabilidad y presentarlo como mejor opción en cada momento del tiempo, sin necesidad de comprobar la alineación del mismo; por tanto, la salida de la red es una cadena de caracteres que representa los símbolos más probables del alfabeto para la información proporcionada.

7.3.2. Modelo para comparar voces

Para la comparación de voces se ha implementado un algoritmo de aprendizaje profundo un poco especial, se trata de una red neuronal convolucional siamesa, acto seguido se puede apreciar su estructura:

Las redes siamesas tienen la peculiaridad de contar con parte de la red neuronal duplicada, como puede observarse en la figura 7.2, la red cuenta con 2 entradas que corresponden a las 2 muestras que se pretenden comparar; acto seguido, la red cuenta con una serie de capas que están duplicadas pero a nivel interno se representan como “sequential_1” y, no son más que 3 capas de convolución con sus correspondientes “max pollings”, esto permite extraer características del sonido como si se tratara de una imagen y realizar una comparación simple que, en este caso, es una resta de las características encontradas por la red; Para finalizar, la salida

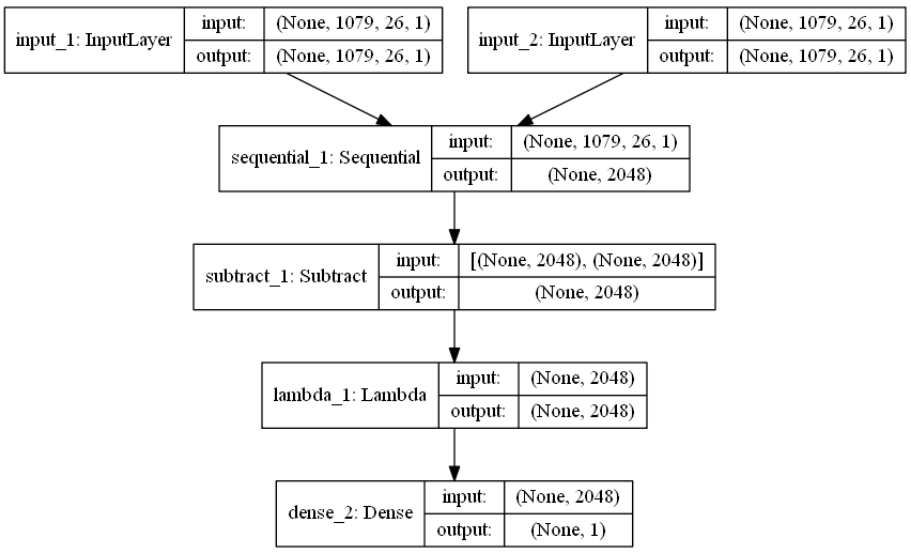

Figura 7.2: Estructura del modelo de comparación de voz

es una capa densa con una única neurona, encargada de dar como resultado un valor entre 0 y 1 que representa la probabilidad de que dos voces sean la misma.

La red convolucional siamesa cuenta con 69.454.529 parámetros que se pueden entrenar, aún y contar con casi 50 millones de parámetros más que el modelo de deep speech, este modelo se ha podido entrenar a gran velocidad debido a no contar con capas recurrentes.

7.3.3. Algoritmo para identificar género

La red para identificar el género de una muestra de voz es una versión modificada del modelo para comparar voces, alterado para distinguir entre 2 clases (masculino y femenino) y únicamente necesitar una muestra como input de la red.

Las modificaciones respecto a la red de comparación de voces son:

- Se ha eliminado la parte duplicada de la red siamesa.
- Se ha cambiado la salida de la red a 2 nodos, uno por cada clase a categorizar.
- Se ha empleado una función de activación softmax al final de la red.

La estructura de la red es:

Como puede apreciarse en la figura 7.3, solamente existe una capa de entrada, se ha suprimido la capa encargada de hacer la operación de restar, se ha modificado la función de activación de la última capa por una softmax con una neurona adicional en la salida de la red para obtener 2 clases e implementar un clasificador. El resto de la estructura permanece igual a como se encontraba en el modelo de comparación de voz.

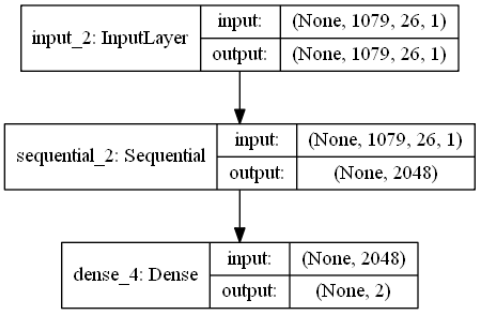

Figura 7.3: Estructura del modelo de identificación de género

La red de identificación de género cuenta con 69.456.578 parámetros que se pueden entrenar, una cantidad ligeramente superior al modelo de comparación de voces, esto se debe al nodo nuevo que aparece por la aparición de una clase más para clasificar.

7.3.4. Algoritmo para obtener MFCC

Todos los algoritmos de aprendizaje profundo del trabajo utilizan como entrada un vector de características que representa la voz con la que se debe trabajar, a continuación se describen los pasos para obtener el vector de características MFCC[22] de una fuente de audio.

Pasos para obtener los coeficientes MFCC[22]:

- Separar la señal de audio en pequeños fragmentos
- Aplicar la transformada de fourier discreta
- Aplicar filtros correspondientes a la escala de Mel (escala musical)
- Realizar el logaritmo de todas las energías
- Aplicar transformada de coseno discreta a los logaritmos anteriores.

Hoy en día, la mayoría de librerías que permiten el tratamiento del sonido incluyen la extracción de coeficientes MFCC de forma sencilla, para el proyecto se ha utilizado la version de la librería “python_speech_features”.

7.3.5. Algoritmo Word Error Rate

El algoritmo Word Error Rate es, en esencia, el algoritmo para calcular la distancia levenshtein pero en lugar de aplicarse sobre caracteres se aplica sobre palabras.

```
import numpy
def word_error_rate(s, t):
    a = len(s) +1
    b = len(t) +1
    m = numpy.zeros((a)*(b), dtype=numpy.uint8)
    m = m.reshape((a, b))
    for i in range(a):
        for j in range(b):
            if i == 0:
                m[0][j] = j
            elif j == 0:
                m[i][0] = i

    for i in range(1, a):
        for j in range(1, b):
            if s[i-1] == t[j-1]:
                m[i][j] = m[i-1][j-1]
            else:
                subs = m[i-1][j-1] + 1
                ins = m[i][j-1] + 1
                del = m[i-1][j] + 1
                m[i][j] = min(subs, ins, del)

    return m[len(r)][len(h)]
```

7.4. Pruebas de concepto

A continuación se muestran todas las diferentes pruebas realizadas durante el desarrollo del trabajo para ir comprobando las diferentes implementaciones, simplificaciones y modificaciones de los algoritmos.

7.4.1. Tratamiento del sonido

Las pruebas realizadas para el tratamiento del sonido han ido enfocadas a como realizar el proceso de cortar los ficheros de audio y como realizar la sincronización forzada con los audio-libros.

Sincronización forzada

La prueba de concepto de la sincronización forzada fue la implementación del script mostrado en la sección 7.1.1, dicho script permitía aplicar los pasos necesarios para resolver el problema pero Aeneas, el programa encargado de la sincronización forzada, ofrecía unos resultados caóticos necesitando un gran ajuste manual que impedía la viabilidad del método.

Cortar sonido empleando silencios

Para poder cortar el sonido de un fichero empleando los silencios, se elaboró un código de python que permite probar diversas configuraciones y ver visualmente el efecto sobre la onda. El código es el siguiente:

```
import pydub
import python_speech_features as p
import scipy.io.wavfile as wav
import numpy as np
```

```
import matplotlib.pyplot as plt
from keras.preprocessing.sequence import pad_sequences

def generate_mfcc_shape(audio, fs, padlen=1079):
    r = p.mfcc(audio, samplerate=fs, numcep=26)
    t = np.transpose(r)
    X = pad_sequences(t, maxlen=padlen, dtype='float', padding='post', truncating='post').T
    return X

def generate_mfcc_without_silences(path):
    #get audio and change frame rate to 16KHz
    audio_file = pydub.AudioSegment.from_wav(path)
    audio_file = audio_file.set_frame_rate(16000)
    _dbfs = audio_file.dBFS-30
    g = pydub.silence.detect_nonsilent(audio_file, silence_thresh=_dbfs, min_silence_len=200)

    plt.figure(1)
    plt.title('Signal Wave...')
    plt.plot(audio_file.get_array_of_samples(), color='g')

    for a, b in g:
        plt.axvline(x=((a*700000)/21887), color='r')
        plt.axvline(x=((b*700000)/21887), color='m')
    plt.show()
    chunks = pydub.silence.split_on_silence(audio_file, silence_thresh=_dbfs, min_silence_len=200)
    mfccs = []
    for chunk in chunks:
        #compute mfcc from chunk array
        np_chunk = np.frombuffer(chunk.get_array_of_samples(), dtype=np.int16)
        mfccs.append(generate_mfcc_shape(np_chunk, audio_file.frame_rate))
    return mfccs
```

La clave de todo el sistema radica en sobre la variable "_dbfs" cuyo valor se estableció a base de ensayo y error hasta conseguir cortar el fichero de audio de la mejor forma posible.

Las primeras pruebas se orientaban en producir cortes por frases aplicando un valor para el silencio más elevado, pero las pruebas con deep speech arrojaban un mejor comportamiento si, como input, se introducían palabras en lugar de frases completas, por ello, se decidió buscar un parámetro que ajuste de forma correcta para cortar el fichero de audio por palabras.

Una muestra del ajuste del parámetro sobre una frase completa puede observarse a continuación, como puede observarse en la figura 7.4, las zonas que se van a suprimir del fichero

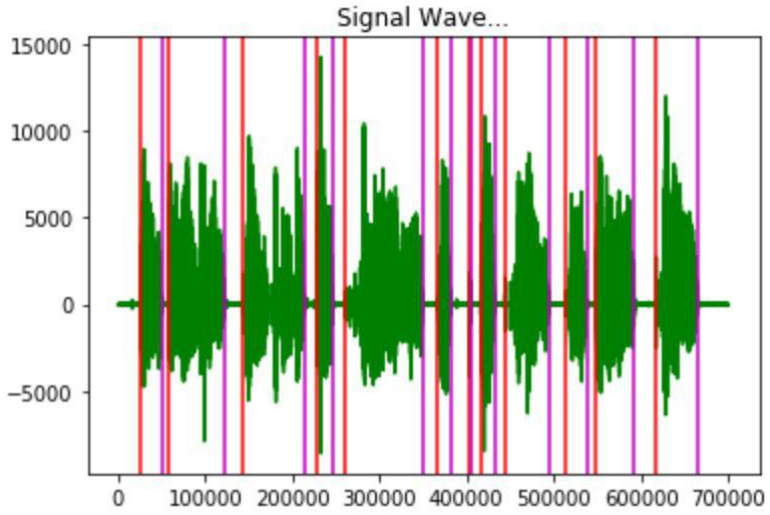

Figura 7.4: Señal con zonas de corte marcadas

de audio son todas aquellas comprendidas entre una línea lila hasta la siguiente línea roja, produciendo un fichero por cada "zona verde" de la onda sobre la cual se aplicará el algoritmo MFCC y serán la entrada de los algoritmos de aprendizaje profundo.

7.4.2. Transcripción

Todas las pruebas realizadas con el modelo de transcripción se han orientado a reducir el tamaño y dificultad del problema con la finalidad de obtener un modelo más sencillo que permita solucionar el problema de la transcripción en el tiempo establecido. Este punto se encuentra más detallado en el capítulo dedicado a las optimizaciones.

7.4.3. Comparación de voces

Las pruebas realizadas para resolver la comparación de voces han sido la implementación de modelos rápidos y sencillos con la finalidad de iterar de forma rápida hasta encontrar un modelo lo suficientemente bueno como para dar por resuelto el problema de comparación.

Red densa como comparador

En una primera aproximación al problema, se estudió la posibilidad de juntar en un solo input de tamaño 2N siendo N el tamaño de un vector de características MFCC, la entrada de una red neuronal para comparar las dos muestras de voz y producir una salida probabilística. Para ello se implementó una red neuronal profunda con la forma mostrada en la figura 7.5

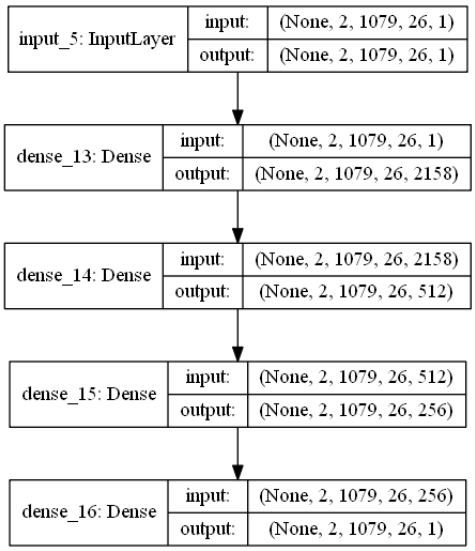

Figura 7.5: Red neuronal densa 2N para comparación

Red siamesa densa como comparador

Una segunda prueba para resolver la comparación de voces fue la implementación de una red siamesa densa, muy similar a la implementada finalmente en el trabajo. Se diferenciaba de la implementada finalmente únicamente en la sustitución de todas las capas de convolución y max pooling por capas densas.

7.5. Resultados

7.5.1. Tratamiento del audio

Finalmente, se encontró un valor para la constante que regula el silencio que ajusta de forma razonable la onda, permitiendo eliminar todas aquellas zonas con silencio y únicamente utilizar las zonas donde es más probable que haya información.

La constante para cada sonido es: _dbfs = audio_file.dBFS-30, como puede observarse, la constante para cada sonido tiene una parte variable y una parte fija, la parte fija es aquella que ha sido ajustada manualmente mediante ensayo y error, mientras que la parte variable se basa en una de las características del sonido, el dbFS del mismo.

7.5.2. Comparación de voces

Las redes implementadas como prueba de comparación de voz, han permitido encontrar una red que podía resolver el problema, la red siamesa densa, con más imprecisión que la implementada finalmente pero, en caso de no poder implementar la red final, se hubiera implementado esta red como sustitución.

7.6. Implementación

En todos los algoritmos se ha decidido utilizar el mismo tipo de dato, esto facilita el entrenamiento; el formato elegido ha sido el vector de características MFCC, a continuación, en la figura 7.6 se puede ver una imagen que representa visualmente una muestra MFCC utilizada.

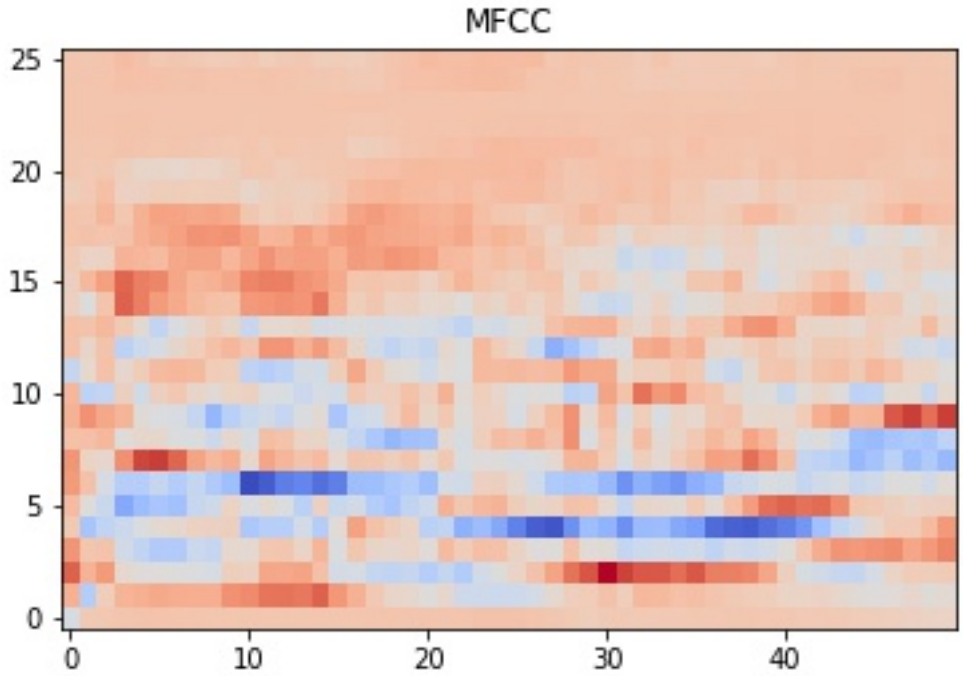

Figura 7.6: Representación MFCC de una muestra

Los colores presentes en la imagen representan el valor numérico del coeficiente MFCC en un instante de tiempo concreto t, para las necesidades de este trabajo se han empleado 26 coeficientes.

Sin embargo, de haber sido necesario optimizar más el problema para mejorar el entrenamiento, se podrían haber reducido el número de características a 18-19, puesto que en la mayoría de muestras, se puede observar que en la franja comprendida entre los coeficientes 20-26, tal y como se observa en la imagen anterior 7.6, no aporta información relevante para resolver el problema.

7.6.1. Transcripción

El procedimiento para poder transcribir ficheros de voz a texto sigue los pasos como se muestran en la figura 7.7, en primer lugar, a todo fichero se le extrae el vector de características MFCC, acto seguido, se introduce el vector de características como entrada para el modelo; el modelo produce una cadena de caracteres en función del entrenamiento y la entrada de la red. Acto seguido debe aplicarse un modelo de lenguaje, este paso permite extraer como resultado el fichero de voz transcrito.

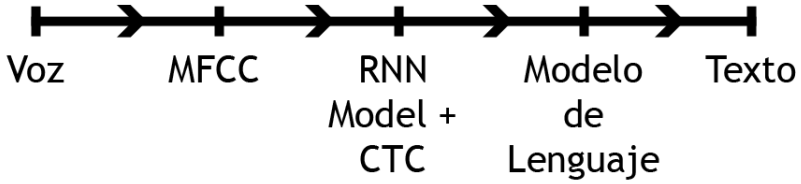

Figura 7.7: Pasos para obtener una transcripción

Consideraciones a tener en cuenta, el vector de características debe ser de tamaño fijo, puesto que es la entrada de los modelos de aprendizaje profundo, por ello, se aplica un relleno con "0" a todos los elementos del vector necesarios para poder tener el tamaño N adecuado como entrada de cada algoritmo; en caso de ser más grande que el valor establecido, simplemente se corta los elementos sobrantes.

En el caso práctico, como se están cortando los ficheros de audio empleando el silencio para obtener palabras, nunca se llena el vector de características y se recurre a rellenar los elementos que faltan con "0".

7.6.2. Comparación de voz

El procedimiento para poder comparar voz sigue los pasos como se muestran en la figura 7.7 hasta la aplicación del "Modelo de lenguaje", en primer lugar, a todo fichero se le extrae el vector de características MFCC, acto seguido, se introduce el vector de características como entrada para el modelo; el modelo produce una salida real entre 0 y 1 que permite decidir si se consideran muestras parecidas o no, todo en función de la predicción y su cercanía a 1, siento este resultado interpretado como "son iguales" y 0 como "Son distintos".

Por tanto, es necesario establecer un valor threshold a partir del cual se asume que la comparación ha resultado un éxito o no.

7.6.3. Identificar género

Identificar el género es similar a cómo se realiza la comparación de voz, se empieza obteniendo las características MFCC de una muestra de sonido, posteriormente se aplica el modelo para identificar el género y, de esta forma, el modelo clasifica con 1 si el resultado ha sido "masculino" y con un 2 si el resultado ha sido "Femenino"

8. Desarrollo

8.1. Android

Para el desarrollo de la aplicación de Android, primero se ha pensado en la interacción con el usuario y, posteriormente, en la funcionalidad que debe resolver.

8.1.1. Diseño de la aplicación

Antes de implementar cualquier pantalla, se ha pensado en la forma más cómoda de resolver el problema para el usuario de la aplicación, por ello, se han desarrollado los siguientes bocetos de como deberían ser las pantallas y sus implementaciones. En la figura 8.1 se puede apreciar

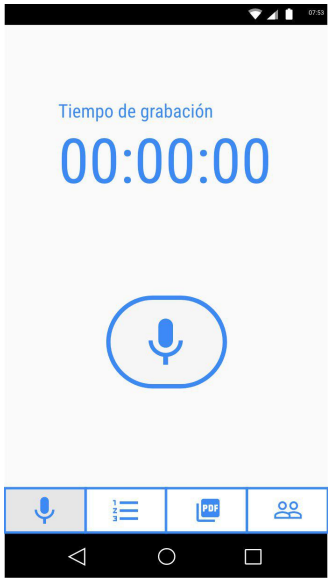

Figura 8.1: Diseño preliminar de la pantalla de grabación de reuniones

la pantalla de grabación de reuniones, en la misma se puede ver un botón que permitirá iniciar y parar la grabación y un cronómetro con el tiempo que ha transcurrido desde el inicio de la grabación.

Además, en la pantalla inferior se ha dispuesto de un conjunto de pestañas que permiten acceder al resto de funcionalidades que ofrece la aplicación.

Por otro lado, también se ha diseñado la pantalla encargada de mostrar la lista de las reuniones,

tal y como se puede observar en la figura 8.2, este diseño contempla una lista ordenada por fecha de la reunión (más recientes primero), donde cada elemento que se muestra contiene un nombre para identificar la reunión, la fecha de la misma y un icono en el lado derecho que permite borrar de la memoria del dispositivo una grabación.

Figura 8.2: Diseño preliminar de la pantalla de lista de reuniones

También se ha realizado el boceto de la pantalla de muestras, como se puede apreciar en la figura 8.3, la pantalla es muy similar a la lista de las reuniones, esto es intencional, el objetivo es reducir el tiempo de desarrollo rehusando todos los elementos que sean posibles.

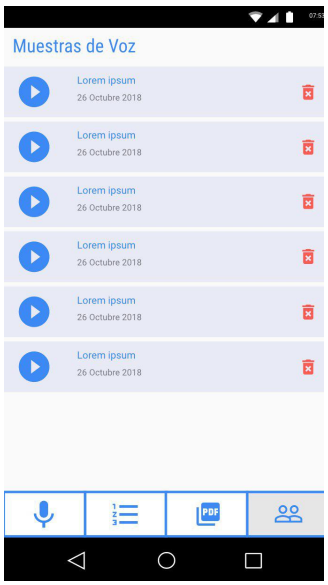

Figura 8.3: Diseño preliminar de la pantalla de lista de muestras

Las muestras de voz serán ordenadas por fecha (recientes primero), permitiendo que con un toque en la muestra se pueda seleccionar si se desea integrar esta muestra o no al sistema.

El siguiente diseño realizado ha sido la pantalla de recogida de muestras, esta pantalla, como se puede observar en la figura 8.4, es intencionalmente parecida a la pantalla de grabación de reunión, esto permite reutilizar funcionalidades como la del cronometro para ajustar el tiempo de desarrollo al tiempo de desarrollo planificado.

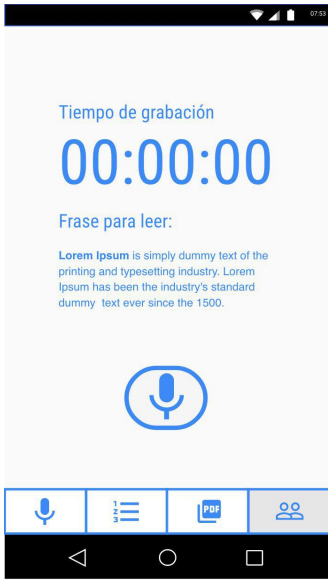

Figura 8.4: Diseño preliminar de la pantalla de obtención de muestras

El cambio más notorio respecto a la pantalla de grabación es la inclusión de una sección con una frase aleatoria que el empleado debe leer para obtener una muestra de voz válida.

Con todo lo anterior, las pantallas con la funcionalidad más importante ya se encuentran diseñadas y, a continuación, se va a proceder a la implementación realizada finalmente.

8.1.2. Implementación de pantallas y funcionalidad

Basando el diseño de la aplicación en los bocetos anteriores, se ha procedido a implementar las pantallas de la siguiente forma, empezando por la pantalla de grabar una reunión.

Como se puede apreciar, el diseño ha variado un poco desde el boceto inicial a la implementación final, el cambio más notorio es la ausencia de la barra de pestañas inferior con un botón a cada funcionalidad, en la versión final se ha sustituido por un menú lateral oculto en el lado izquierdo de la pantalla y que únicamente se muestra presionando el botón superior izquierdo de la pantalla.

El resto de los elementos en pantalla se han mantenido como en el diseño inicial, tal y como se puede observar en la figura 8.5, resultando en una pantalla sencilla y con la funcionalidad claramente definida.

A continuación se va a hablar sobre el menú lateral anteriormente mencionado, como se puede ver en la figura 8.6, cuando se presiona el botón superior izquierdo de la pantalla, se oscurece la pantalla y se muestra, desde el lado izquierdo, todas las funcionalidades disponibles de la

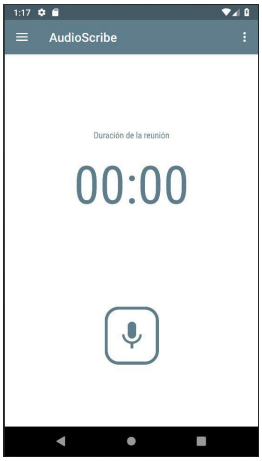

Figura 8.5: Implementación de la pantalla de grabación de reunión

aplicación.

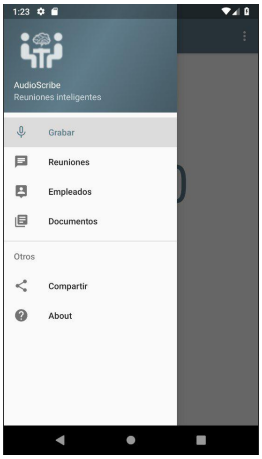

Figura 8.6: Implementación del menú de la aplicación

En este menú lateral, se muestra el icono de la aplicación en la parte superior, una sección con las 4 pantallas principales y una sección en la parte baja del menú para la funcionalidad de compartir y un botón para acceder a la información y ayuda.

Por otro lado, se ha implementado la pantalla para ver todas las reuniones que se han grabado, en la figura 8.7 se puede observar el resultado final de esta pantalla.

La pantalla muestra una lista con todas las reuniones ordenadas (recientes primero), donde cada reunión muestra el nombre de la misma, el tiempo de duración de la reunión y la fecha y hora exactas en la que se almacenó la reunión en el dispositivo.

Finalmente se ha decidido eliminar el botón lateral que permite eliminar la reunión, en su lugar se ha sustituido por una acción de menú desplegable únicamente cuando presionas sobre una reunión, este menú desplegable contiene también la funcionalidad de enviar reunión al servidor.

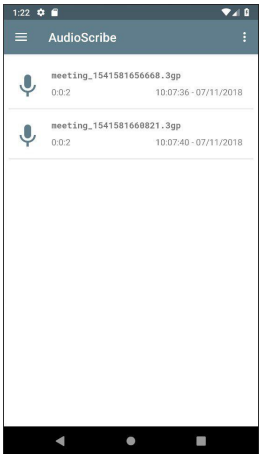

Figura 8.7: Implementación del la pantalla de lista de reuniones

La siguiente pantalla es la correspondiente a la lista de muestras, como se observa en 8.8, esta pantalla ha sufrido diversos cambios respecto a la versión original.

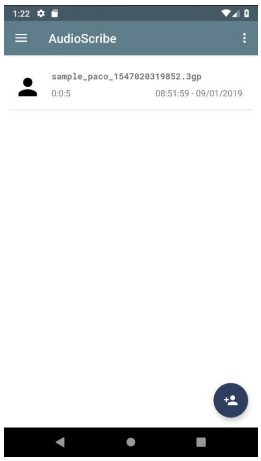

Figura 8.8: Implementación del la pantalla de lista de muestras

Ahora, presenta un diseño más simple, la pantalla muestra una lista con todas las muestras ordenadas que se han obtenido (más recientes primero) con un botón en la parte inferior derecha que permite añadir nuevas muestras de empleados. La pantalla dedicada a ver la lista de las reuniones transcritas presenta el mismo diseño, la única diferencia radica en la funcionalidad del botón inferior, que permite la compartición sencilla de la información generada por el sistema.

Finalmente, se ha desarrollado una pantalla para registrar una empresa nueva, esta pantalla tiene una doble funcionalidad pues permite registrar una empresa en el sistema y acceder, con usuario y contraseña, a la cuenta de una empresa ya existente. Además, se han incorporado menús secundarios para cumplir con la funcionalidad del derecho al olvido, solicitar el consentimiento del usuario y compartir los documentos y grabaciones con otras aplicaciones.

8.1.3. Conectividad

La conexión con el servidor se ha realizado mediante una API REST que permite subir ficheros al servidor y expone la información de forma controlada para que la aplicación funcione correctamente. Este punto se encuentra más detallado en la sección dedicada al servidor.

8.2. Servidor

8.2.1. Problema a resolver

El servidor se debe encargar de los siguientes elementos:

- Exponer una API para conectar con la aplicación Android.
- Permitir subir ficheros de audio.
- Integrar los modelos de aprendizaje profundo desarrollados.
- Procesar reuniones y muestras.

Todos estos puntos son tratados a continuación.

8.2.2. API REST y Conectividad con la aplicación

Para que la aplicación pueda funcionar correctamente, se han implementado los siguientes endpoints para exponer una API REST que permita la comunicación cliente-servidor.

- /registration
  - Permite dar de alta una empresa.
- /login
  - Permite obtener los tokens de acceso.
- /logout/access
  - Permite desconectar la sesión y invalidar los tokens de acceso.
- /token/refresh
  - Permite obtener un token de acceso nuevo cuando ha caducado el anterior.
- /upload/meeting
  - Permite subir una reunión y procesarla.
- /upload/sample
  - Permite subir muestras de la empresa.
- /samples
  - Permite gestionar las muestras del servidor.

- /status
  - Permite conocer el estado del cálculo de una reunión.
- /forbidden/data
  - Permite ejercer el derecho al olvido obteniendo todos los datos presentes en el servidor.
- /forbidden/delete
  - Permite ejercer el derecho al olvido permitiendo borrar de forma definitiva todos los archivos del servidor.

Con los endpoints anteriores, se define un esquema de comunicación REST que permite a la aplicación Android acceder a todos los recursos del servidor de forma controlada y ordenada. Para ello, únicamente debe solicitar un token (haciendo uso del usuario y contraseña) y emplear ese token con todas las peticiones que realice y podrá utilizar toda la funcionalidad disponible.

8.2.3. Integración con los modelos de aprendizaje profundo

En el servidor se han incorporado los modelos ya entrenados que resuelven cada uno de los problemas del trabajo, la forma de hacerlo ha sido aislando cada modelo en funciones completas que permiten, con una sola llamada, realizar el proceso de predicción con la información suministrada a la función.

Una vez implementados todos los modelos siguiendo el esquema anteriormente descrito, se ha desarrollado todo el camino desde que una reunión se recibe en el servidor hasta que finalmente se obtiene el documento PDF.

Los pasos que se han seguido han sido los siguientes:

- Romper el fichero en palabras utilizando los silencios.
- Se han obtenido todos los MFCC de cada una de las partes .
- Se ha realizado la transcripción de cada parte.
- Se han comparado todas las partes para identificar a todas las personas distintas presentes en la reunión.
- Se han comparado las personas detectadas contra las muestras presentes.
- Se ha generado un documento PDF con las transcripciones y la información de cada persona.

Para la recogida de las muestras de voz de los empleados, se ha realizado una implementación similar, descrita a continuación:

- Obtener el MFCC de la muestra de voz
- Predecir el género de la muestra utilizando el MFCC del paso anterior
- Almacenar la información MFCC y el género en el servidor

Con lo anterior, se ha descrito el funcionamiento del proceso de obtención de información para las reuniones y el funcionamiento de la obtención de muestras.

8.2.4. Sistema de Tareas

Debido a que el procesado para obtener el documento PDF de una reunión es laborioso, el servidor se queda bloqueado hasta que no ha procesado completamente la reunión en curso, por ello, ha sido necesario incorporar un sistema que permita ir procesando sobre la marcha y de forma paralela todas las reuniones que se soliciten desde la aplicación.

La forma de resolver el problema ha sido añadiendo un sistema de tareas y colas, antes de nada, se ha definido una tarea como todo el proceso desde que se recibe una reunión en el servidor hasta que se genera un documento PDF con la transcripción; este sistema permite ir encolando tareas que posteriormente una serie de workers irán recogiendo y procesando, de forma que el servidor no se bloquea y permite procesar multitud de peticiones de forma concurrente.

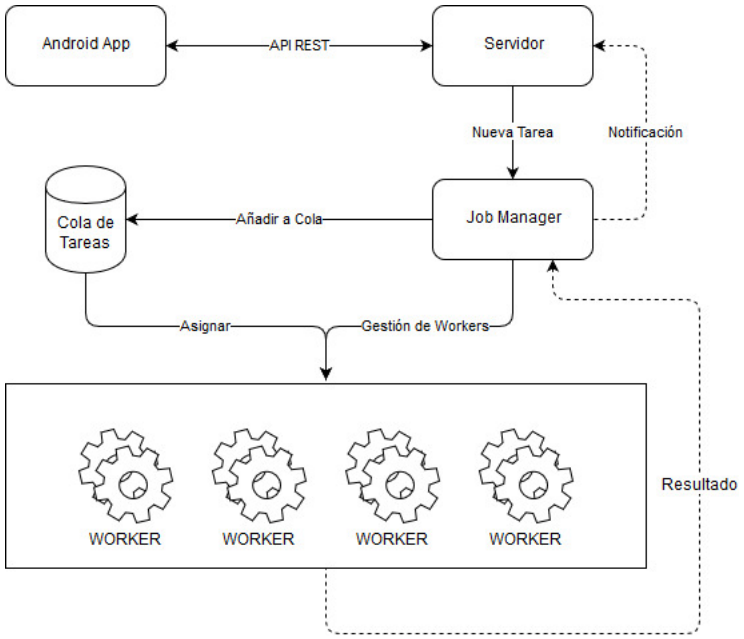

Figura 8.9: Arquitectura del servidor con el sistema de tareas

La estructura desplegada en el servidor está representada en la figura 8.9, en ella se muestra al servidor creando una nueva tarea que gestiona un job manager, el job manager es el encargado de recibir tareas, gestionar los workers activos y encolar las tareas nuevas.

Por otro lado, cuando un worker ha terminado una tarea, el job manager recibe el resultado y notifica al servidor el resultado para poder comunicarlo a la aplicación Android.

8.2.5. Límites

El servidor presenta, en función de los recursos disponibles, ciertas limitaciones, la principal limitación es el número de workers y la memoria que tienen disponible. Es recomendable como máximo un worker por cada hilo presente en la máquina donde se ejecute, además, la duración en minutos que puede asumir viene dado por el tamaño de la memoria RAM del sistema, siendo con 32GB de RAM capaz de gestionar reuniones de como máximo 1h y media aproximadamente.

8.3. Dificultades durante el desarrollo

Uno de los grandes problemas durante el desarrollo del proyecto ha sido el entrenamiento del modelo deep speech, la primera implementación realizada simplemente fue inviable, al principio no se podía ejecutar por problemas de espacio en RAM, simplemente el ordenador se quedaba congelado y era necesario reiniciar para solventarlo, más adelante, cuando se pudo optimizar lo suficiente como para que el ordenador funcionara sin problemas, el equipo alcanzaba unas temperaturas cercanas a los 98 grados en la CPU y cerca de 95 grados en las tarjetas gráficas.

Por las temperaturas anteriores, fue necesario implementar una refrigeración líquida para poder dejar el ordenador encendido sin temor a sobre-calentar sus componentes.

Otro de los problemas durante el desarrollo ha sido durante la implementación del sistema de colas en el servidor, inicialmente, aparecían errores que eran debidos a problemas de configuración de paquetes del entorno virtual donde se ejecutaba, siendo un auténtico problema detectar las causas del mismo y solventarlos.

9. Optimizaciones

En este capítulo se van a tratar todas las optimizaciones que han sido aplicadas al trabajo.

9.1. Optimizaciones sobre el dataset

Uno de los problemas que surgió fue el tiempo necesario para realizar una época de entrenamiento del modelo de transcripción, al realizar métricas de rendimiento, se observó que parte del problema era el flujo de lectura que se hacía de los ficheros de sonido. Cada vez que el algoritmo de entrenamiento solicitaba una muestra ocurría lo siguiente:

- Se cargaba el fichero en memoria
- Se aplicaba el algoritmo MFCC al fichero
- Se pasaba el fichero a formato Numpy Array
- Se utilizaba por el algoritmo para realizar el entrenamiento.

Esta serie de pasos era muy costosa de realizar para cada una de las muestras que componían el dataset, cada época tenía que repetir estos pasos para cada una de las 350.000 muestras.

Finalmente, se optó por dejar preparado y procesado todo el dataset completo y generar los datasets necesarios para cada modelo a entrenar; esto produjo una mejora de velocidad notable y un aumento considerable del dataset en disco duro, pasando de los 28GB a los 153GB.

Acto seguido se muestran los códigos que permitieron optimizar todo el dataset y generar las versiones preparadas y procesadas de cada uno de ellos.

9.1.1. Preparar y procesar MFCC para las muestras del modelo de transcripción

Permite preparar todas las muestras para no realizar estas operaciones y transformaciones a MFCC en tiempo de entrenamiento.

```
csv_path = '../datasets/csv/ready/'
dataset_path = '../datasets/'
csv_to_precompute = csv_path + 'train.csv'
num_chunks = 25

df = pd.read_csv(csv_to_precompute, usecols=['wav_filename', 'transcript'])
list_wav_files = df['wav_filename'].tolist()
```

```
list_transcript = df['transcript'].tolist()
block_size = int(len(list_wav_files)/num_chunks)

def storeData(path, name, idx):
    batch_x = list_wav_files[idx * block_size : (idx + 1) * block_size]
    batch_y = list_transcript[idx * block_size : (idx + 1) * block_size]
    temp = []
    temp2 = []
    df = pd.DataFrame(columns=['transcription'])
    for i in range(len(batch_x)):
        try:
            temp.append(make_mfcc_shape(dataset_path + batch_x[i], padlen=1079))
            temp2.append(get_max_time(dataset_path + batch_x[i]))
            df = df.append({'transcription': batch_y[i]}, ignore_index=True)
        except:
            print('error: ' + batch_x[i])
    np.save(path + name + '_'+ str(idx) +'.npy', np.array(temp))
    np.save(path + name + '_time_'+ str(idx) +'.npy', np.array(temp2))
    df.to_csv(path + name + '_'+ str(idx) +'.csv', sep=',', index = False)

progress = IntProgress(min=0, max=(num_chunks-1))
progress.description = str(progress.value)
display(progress)
for i in range(num_chunks):
    storeData(csv_path + 'preload/deepspeech/', 'train', i)
    progress.value += 1
    progress.description = str(progress.value)
```

9.1.2. Preparar y procesar MFCC para las muestras del comparador de voces

Permite preparar todas las muestras para no realizar estas operaciones y transformaciones a MFCC en tiempo de entrenamiento.

```
csv_to_precompute = store_path + 'test.csv'
#csv_to_precompute = store_path + 'train.csv'

df = pd.read_csv(csv_to_precompute, usecols=['file_a', 'file_b', 'result'])
list_file_a = df['file_a'].tolist()
list_file_b = df['file_b'].tolist()
list_result = df['result'].tolist()
num_chunks = 5
block_size = int(len(list_file_a)/num_chunks)

def storeData(path, name, idx):
    batch_xa = list_file_a[idx * block_size : (idx + 1) * block_size]
    batch_xb = list_file_b[idx * block_size : (idx + 1) * block_size]
    batch_y = list_result[idx * block_size : (idx + 1) * block_size]
    temp = []
    temp2 = []
    df = pd.DataFrame(columns=['result'])
    for i in range(len(batch_xa)):
        try:
            temp.append(make_mfcc_shape(dataset_path + batch_xa[i], padlen=1079))
            temp2.append(make_mfcc_shape(dataset_path + batch_xb[i], padlen=1079))
            df = df.append({'result': batch_y[i]}, ignore_index=True)
        except:
            print('error: ' + batch_x[i])
    np.save(path + name + '_a_' + str(idx) +'.npy', np.array(temp))
    np.save(path + name + '_b_' + str(idx) +'.npy', np.array(temp2))
    df.to_csv(path + name + '_'+ str(idx) +'.csv', sep=',', index = False)

progress = IntProgress(min=0, max=(num_chunks-1))
progress.description = str(progress.value)
display(progress)
for i in range(num_chunks):
    storeData(preload_path, 'test', i)
    progress.value += 1
    progress.description = str(progress.value)
```

9.1.3. Preparar y procesar MFCC para las muestras del identificador de género

Permite preparar todas las muestras para no realizar estas operaciones y transformaciones a MFCC en tiempo de entrenamiento.

```
#csv_to_precompute = store_path + 'test.csv'
csv_to_precompute = store_path + 'train.csv'

df = pd.read_csv(csv_to_precompute)
list_file = df['file'].tolist()
list_gender = df['gender'].tolist()
num_chunks = 5
block_size = int(len(list_file)//num_chunks)

def storeData(path, name, idx):
    batch_xa = list_file[idx * block_size : (idx + 1) * block_size]
    batch_y = list_gender[idx * block_size : (idx + 1) * block_size]
    temp = []
    df = pd.DataFrame(columns=['result'])
    for i in range(len(batch_xa)):
        try:
            temp.append(make_mfcc_shape(dataset_path + batch_xa[i], padlen=1079))
            df = df.append({'result': batch_y[i]}, ignore_index=True)
        except:
            print('error: ' + batch_xa[i])
    np.save(path + name + '_' + str(idx) +'.npy', np.array(temp))
    df.to_csv(path + name + '_'+ str(idx) +'.csv', sep=',', index = False)

progress = IntProgress(min=0, max=(num_chunks-1))
progress.description = str(progress.value)
display(progress)
for i in range(num_chunks):
    storeData(preload_path, 'train', i)
    progress.value += 1
    progress.description = str(progress.value)
```

9.2. Optimizaciones del modelo deep speech

Existen variaciones respecto al modelo original de deep speech, únicamente se han realizado estos cambios como optimizaciones para abordar el proyecto dentro de los plazos previstos según la planificación.

Las modificaciones han sido:

- Se ha empleado como función de activación únicamente ReLU o clipped_relu.
- Se ha reducido la densidad de nodos de las capas densas de 2048 a 1024 nodos.
- Se ha sustituido la capa recurrente por una CuDNNLSTM para acelerar el cómputo por hardware en GPUs.
- Se ha utilizado el alfabeto inglés para reducir los requisitos de procesamiento.
- Se han modificado los bits de precisión para reducirlos de FP32 a FP16

Con todas las modificaciones, el algoritmo deep speech simplificado ha contado con 18.952.733 parámetros que se pueden entrenar, siendo un modelo mucho más compacto frente a los cerca de 120.000.000 que contiene la versión original, esta versión permite realizar el entrenamiento dentro del margen de la planificación, realizando un compromiso entre la velocidad de entrenamiento y la capacidad de inferencia del modelo.

Otra optimización realizada para poder entrenar con éxito el algoritmo de deep speech ha sido la fragmentación de las muestras del dataset en bloques de 15GB para poder alojarlos directamente en memoria RAM y evitar un carga innecesaria sobre el disco duro, únicamente cuando hay un cambio de bloque necesario se deben descargar los 15GB

9.3. Optimizaciones generales

Se han realizado una serie de optimizaciones a nivel general que han permitido acelerar notablemente todas las necesidades de cómputo del proyecto, la primera optimización ha sido reducir los bits de precisión de FP32 a FP16 trabajando todos los modelos únicamente con valores flotantes de 16bits, mejorando el cálculo en paralelo de las GPUs que cuentan con unidades especializadas para realizar estas operaciones de manera más eficiente.

Otra optimización realizada ha sido utilizar el modo de entrenamiento multi-gpu que incorpora la librería Keras, este modo de entrenamiento permite copiar en la memoria de cada tarjeta gráfica el mismo modelo y, cada tarjeta, realiza una época con la mitad del dataset y posteriormente hay un proceso automático que se encarga de ajustar correctamente los pesos de la red.

De forma que, aplicando este modo multi-gpu se consigue una mejora en velocidad cercana a N veces, siendo N el número de tarjetas disponibles.

10. Validaciones y Comprobaciones

10.1. Deep Learning

La validación de los modelos de aprendizaje profundo son un caso especial, para la transcripción se ha optado por comparar la métrica WER o Word Error Rate siendo aquel con valor más bajo como el mejor modelo posible para incluirlo en el desarrollo del trabajo.

Por otro lado, tanto para el modelo de comparación como el de identificación de género, se han realizado pruebas basadas en sus correspondientes matrices de confusión, con el objetivo de seleccionar aquellos modelos con mayor precisión y recall, descartando todos aquellos que empeoraban o no mejoraban estos resultados.

A continuación se especifica el procedimiento para validar cada modelo.

10.1.1. Transcripción

Para validar el modelo de transcripción, se ha dispuesto de un subconjunto del dataset no utilizado para entrenar, que cuenta con 1000 frases sobre las que el modelo realiza las predicciones y, posteriormente se realiza el Word Error Rate de cada frase, utilizando como métrica del modelo el WERM o Word Error Rate Medio. Después de cada época de entrenamiento, se evalúa el rendimiento del modelo empleando este sistema, si el resultado mejora el WERM actual, el modelo se almacena en disco, en caso contrario, el proceso de entrenamiento continua con la siguiente época.

Las características del mejor modelo posible encontrado han sido:

- Entrenamiento: 81 % de acierto
- Testing: 72 % de acierto
- WERM (Word Error Rate Medio): 53,1 %

Cuanto más bajo es el valor WER significa que mejor precisión ha tenido el modelo, en este caso, un modelo con un WER tan elevado significa que las transcripciones no las termina de realizar correctamente, muy probablemente debido al conjunto de datos empleado para entrenar y de las múltiples simplificaciones realizadas sobre el modelo original de deep speech.

10.1.2. Comparación de voz

Por otro lado, las comprobaciones realizadas al modelo de comparación de voces se han centrado en el análisis de la matriz de confusión empleando un conjunto de datos no utilizado en el entrenamiento con 1000 muestras con 500 muestras positivas y 500 muestras negativas.

La matriz de confusión resultante ha sido:

	Si	No
Si	487	13
No	8	492

Como puede observarse de la matriz de confusión, el modelo tiene una precisión de 0,974 y un recall de 0,983, siendo un modelo razonablemente bueno.

10.1.3. Identificación de género

Finalmente, las comprobaciones realizadas sobre el modelo de identificación de género, se han centrado en el análisis de la matriz de confusión, de la misma forma que en el caso del modelo de comparación de voces, con 1000 muestras en total, distribuidas en 500 masculinas y 500 femeninas.

La matriz de confusión del modelo ha sido:

	Si	No
Si	494	6
No	3	497

Como puede observarse de la matriz de confusión, el modelo de identificación de género tiene una precisión de 0,988 y un recall de 0,9939, siendo un modelo también razonablemente bueno.

10.2. Android

Las comprobaciones que se han realizado sobre la app de android han sido:

- Comprobar que todas las pantallas son accesibles.
- Comprobar que todos los botones realizan las funciones correctamente.
- Se ha probado a cerrar y abrir la aplicación en medio de diferentes procesos para observar su comportamiento.
- Se han comprobado las llamadas al servidor para verificar que se envían los datos correctamente.

Con todo lo anterior, se ha garantizado que la aplicación funciona correctamente.

10.3. Servidor

Las validaciones para el lado del servidor han consistido en:

- Comprobar que todos los endpoints son accesibles.
- Comprobar que los endpoints protegidos evitan el acceso no autorizado.
- Comprobar que se pueden subir muestras de voz y reuniones.
- Comprobar que todos los modelos de aprendizaje profundo están bien implementados.
- Comprobar la generación de PDF y su publicación en carpeta
- Comprobar que el servidor puede dar de alta nuevas empresas.
- Comprobar que el servidor permite hacer login y provee los tokens de acceso correctos.
- Comprobar la funcionalidad del derecho al olvido.

Únicamente se ha dado por buena la implementación cuando toda la funcionalidad estaba terminada y comprobada.

10.4. Horas de planificación

El método de validación temporal ha sido la comparación entre las tareas asignadas contra las completadas en cada momento, teniendo en cuenta el tiempo designado a realizar la tarea y el tiempo consumido realmente en la realización de la misma; se ha restado el tiempo estimado de cada tarea con el tiempo real dedicado, de forma que puedan establecerse 2 criterios en función del resultado:

- positivo o cero: Tarea realizada en el tiempo previsto o inferior.
- negativo: Tarea realizada fuera de tiempo.

El seguimiento del tiempo se ha realizado en 2 niveles, en el ámbito de proyecto global y a escala de tarea individual.

11. Integración de conocimiento

11.1. Conocimientos de computación utilizados

A continuación se detallan los conocimientos de la especialidad de computación utilizados en este proyecto.
Inteligencia Artificial (IA):

- Aprendizaje Automático.
- Minería de Datos.
- Tratamiento Automático de Textos y del Habla.
- Inteligencia Artificial en entornos de Web Services.

Aprendizaje Automático (APA):

- Descripción y planteamiento de los problemas atacados por el aprendizaje automático.
- Aprendizaje automático supervisado.
- Redes neuronales para clasificación.

Tanto IA como APA, aportan una base teórica sobre la cual enfocar y resolver los diferentes problemas y retos que se plantean en este trabajo. Un ejemplo de ello es el conocimiento sobre técnicas de aprendizaje supervisado y redes neuronales, elementos clave en la resolución de este proyecto.

Búsqueda y Análisis de Información Masiva (CAIM):

- Preproceso y análisis léxico.
- Medidas de rendimiento (Recall, precision...).
- Arquitectura de sistemas para la gestión de información masiva.
- Análisis de información masiva (Minería de datos, aprendizaje...)

Los conocimientos de CAIM están orientados al pre-procesamiento de grandes volúmenes de información, en este caso, ficheros de audio y sus transcripciones.

Desde el principio del proyecto, se ha necesitado de diversos scripts y automatizaciones enfocados a procesar grandes volúmenes de datos como el conjunto de muestras de 28GB para prepararlo, procesarlo y almacenarlo en memoria ocupando 153GB, todas estas muestras se tienen que cargar en memoria RAM para que el proceso fuera eficiente, sin embargo, el sistema cuenta únicamente con 32GB de memoria RAM disponible, obligando a implementar un sistema que fuera cargando muestras en RAM y descargando aquellas que ya no fueran necesarias.

Para poder obtener esta capacidad de mantenimiento de la memoria, es necesario un proceso previo de estudio donde calcular de que tamaño pueden ser las particiones y, lo más importante, cuando cargar y descargar la memoria para que el proceso no se vea interrumpido en ningún momento.

11.2. Adecuación a la especialidad de computación

El proyecto se adecua a la especialidad de Computación por los siguientes motivos:
Los elementos a resolver en el trabajo son problemas clásicos conocidos en el ámbito del procesamiento del lenguaje natural, además, no se conocen algoritmos polinómicos que resuelvan estos problemas y se debe recurrir a métodos y técnicas con fundamentos matemáticos y estadísticos para poder aproximar soluciones que puedan ser útiles en determinados contextos.

Para el proyecto se deben preprocesar grandes cantidades de ficheros de audio y texto, para poder realizar este paso, se deben utilizar algoritmos y enfoques que permitan realizar esto de forma eficiente y en tiempo razonable.

12. Sostenibilidad y compromiso social

12.1. Ambiental

Durante el desarrollo del proyecto, se emplearán ordenadores personales normales y electrónica de consumo para poder ser realizado, por ello, el único impacto ambiental tendrá su origen en el consumo de energía para hacer funcionar estos equipos y dispositivos.

A nivel proyecto, únicamente puede velarse por un uso responsable y optimizado del gasto energético, no así como el origen y el cómo se ha generado la electricidad que depende, únicamente, de la responsabilidad de las empresas de electricidad contratadas.

El desarrollo del proyecto será realizado mayormente en un equipo portátil, siendo estos dispositivos mucho más eficientes, en materia de gasto energético, que cualquier ordenador personal fijo, estos dispositivos intentan maximizar la duración de la batería e incluyen elementos con bajo consumo eléctrico.

Actualmente, el problema que se pretende abordar se soluciona empleando recursos de las empresas para transcribir e informatizar las reuniones, por ejemplo, uno de los participantes de la reunión es el encargado de dedicar parte de su jornada laboral a transcribir la reunión en un equipo de la empresa, si el proyecto se finaliza con éxito, ya no será necesario emplear tiempo en el equipo antes mencionado siendo necesario únicamente un smartphone, mucho mejor optimizado a nivel consumo de energía, para realizar la misma tarea.

Finalmente, hay que tener en cuenta el consumo energético, en servidores, que la solución propuesta podría requerir, en todo caso, un único servidor puede atender miles de peticiones y, en caso de ser necesario, la solución puede adaptarse de forma responsable acorde a la demanda, haciendo un uso adecuado de los recursos, en todo momento.

12.2. Económico

Se ha presentado una elaborada planificación del coste estimado para realizar el proyecto (Ver Anexo Gestión Económica), contando con los recursos humanos necesarios y materiales.

La solución que plantea este proyecto está orientada a optimizar los recursos humanos y materiales de las empresas en el ámbito de las reuniones; actualmente, en todas las empresas se destinan recursos para dejar constancia de los puntos tratados en las reuniones y no existen muchos procesos para incorporar ese conocimiento de forma automática en la empresa.

Hoy en día, lo más común es encontrar a uno de los participantes de la reunión dedicando parte de su jornada laboral a realizar un documento con la información de la reunión y, posteriormente, incorporar este conocimiento a la empresa de distintas formas, por ejemplo, enviando el documento a todos los implicados.

Esta forma de resolver el problema, implica un gasto en tiempo (jornada laboral) de, como mínimo, un empleado y un ordenador con diversos aplicativos para elaborar el documento, con la solución propuesta, se genera una grabación de audio y un documento PDF que pueden ser incorporados directamente en los procesos de la empresa, con el único gasto de contar con un smartphone, la aplicación instalada y conexión a internet.

Si la solución se llega a implementar, para las empresas puede suponer un ahorro anual importante en función del volumen de reuniones y la duración de las mismas, asumiendo que la persona encargada de transcribir la reunión y elaborar la documentación es un mánager de proyecto o gestor de proyecto, dedicando 5h a la semana para elaborar documentación e introducirla en el sistema informático de la empresa, aproximadamente durante las 50 semanas laborales anuales, con un coste por hora de esta persona de 40€, se está hablando de cerca de 10.000€, dedicados en promedio a esta tarea; un coste que la empresa podría reducir notablemente.

12.3. Social

A nivel personal, este proyecto me permitirá profundizar en los diversos algoritmos y técnicas sobre deep learning, dándome una perspectiva más completa del sector de la inteligencia artificial aplicada y ayudándome a elegir mis futuros pasos profesionales.

La solución propuesta mejora otro aspecto importante para los empleados de la empresa donde se aplique, pues las horas dedicadas a elaborar documentación de forma rutinaria y pesada disminuyen, mejorando la calidad de vida y dedicando los esfuerzos en solucionar otras tareas que aporten valor.

Además, las empresas públicas también pueden utilizar la herramienta desarrollada para acercar la información y documentación oficial al ciudadano, por ejemplo, si se emplea para transcribir los plenos de los parlamentos o para transcribir ruedas de prensa, puede ser una interesante solución para automatizar todo el proceso y distribuir más eficazmente la información.

13. Leyes y regulaciones

Los aspectos relevantes del proyecto están regulados esencialmente por dos leyes:

·Ley de la Propiedad Intelectual o LPI

·Reglamento Europeo de Protección de Datos o RGPD

13.1. Ley de la Propiedad Intelectual o LPI

Según la vigente Ley de Propiedad Intelectual, este proyecto se encuentra sujeto a sus disposiciones estableciendo:

a)Todo el código de programación es propiedad del creador.

b)Toda obra derivada sigue perteneciendo al propietario original.

Entrando un poco más finamente en las partes del proyecto donde pueden existir derechos de autor, se distinguen tres partes susceptibles de derechos:

·Código Fuente del proyecto

·Libros empleados para elaborar el dataset

·Voces empleadas para leer los libros y generar los audios.

El código fuente utilizado en el proyecto tiene dos orígenes, el desarrollado por terceros y el propio, el desarrollado por terceros es software libre sin limitaciones ni restricciones de ningún tipo; por tanto, todo el software del proyecto puede utilizarse sin limitación ni restricción alguna.

Por otro lado, los libros utilizados para elaborar el dataset son obras con licencias de autor expiradas o de dominio público, por tanto, no existe ninguna limitación ni restricción en su uso.

Finalmente, las voces empleadas para leer los libros sí que poseen derechos de autor y solo pueden ser reproducidos i/o generar audios derivados en proyectos sin ánimo de lucro; en el proyecto, no se reproducen los sonidos de ninguna forma ni se generan audios derivados, por tanto, no se incurre en ninguna vulneración de derechos en los términos ni condiciones que están establecidos y, en cuyo caso, este proyecto es un trabajo sin ánimo de lucro y, por tanto, tampoco se incurriría en ninguna vulneración.

En conclusión, el proyecto no vulnera ningún derecho de propiedad intelectual.

13.2. Reglamento Europeo de Protección de Datos

El proyecto está regulado bajo la nueva ley de protección de datos, que establece los siguientes puntos de obligado cumplimiento:

EMPRESAS

·Rendición de cuentas

·Notificación de violaciones de seguridad

·Registro de las actividades de tratamiento

·Responsabilidad pro-activa

·Delegado de protección de datos

CIUDADANOS

·Derecho al olvido

·Derecho a la portabilidad

·Obtención del consentimiento

Este proyecto, al ser puramente académico, no aplican todas las normas, al no ser una empresa es únicamente de obligado cumplimiento todas aquellas normas que afectan a ciudadanos.

Por tanto, el proyecto tiene que gestionar correctamente el derecho al olvido, esta norma incluye la posibilidad de reunir toda la información existente del solicitante para poder descargarla/obtenerla y la capacidad de eliminar del sistema completa y definitivamente toda la información del solicitante.

Los supuestos en los que se puede aplicar este derecho son:

· Cuando los datos no son necesarios para el fin para el que fueron reunidos.

· Si se revoca el consentimiento.

· Si se han obtenido de forma ilegal.

Otra norma de obligado cumplimiento es el derecho a la portabilidad, esto implica que la persona que ha proporcionado sus datos, a un responsable que los está tratando de forma digitalizada, podrá requerir esos datos en un formato que le permita su traslado a otro responsable, en este caso, no aplica debido a que únicamente existe una persona encargada y, con el derecho al olvido, ya se obtiene la información en un formate que permite su traslado.

Finalmente, hay que contemplar la obtención del consentimiento, para ello la ley establece las siguientes directrices:

·El consentimiento debe ser libre, informado, específico, inequívoco e incuestionable.

·La aceptación no se puede deducir del silencio o de la inacción de los ciudadanos.

·El consentimiento debe ser 'manifiesto', por ejemplo, al autorizar el tratamiento de datos sensibles.

·El consentimiento debe ser verificable y quienes recopilen datos personales deben poder probar que el afectado les concedió su consentimiento.

Actualmente el proyecto cumple con todos los puntos aplicables a ciudadanos. En concreto, se solicita explícitamente el consentimiento en la aplicación móvil cada vez que se incorpora una muestra nueva y el servidor tiene incorporados dos endpoints que permiten ejercer el derecho al olvido de forma completa.

14. Mejoras aplicables al proyecto

Debido a las limitaciones de tiempo y de capacidad de cómputo, se han tomado una serie de decisiones que han tenido como resultado un compromiso entre lo abordable y lo deseado.

14.1. Dataset

El primer punto a mejorar del proyecto es, sin duda alguna, el dataset empleado para entrenar los diversos modelos de aprendizaje automático, la calidad y cantidad de muestras puede ser uno de los factores limitantes a la hora de obtener resultados, por ello, es conveniente dedicar el tiempo que sea preciso y los recursos que estén disponibles, cualquier mejora en este aspecto repercute directamente en la inferencia y aprendizaje de los modelos.

14.2. Alfabeto

El segundo punto a mejorar es el alfabeto empleado en el modelo de transcripción, se ha utilizado el alfabeto inglés por contener un menor número de símbolos para ser representado. Contar con un menor número de símbolos redujo la capacidad de procesamiento necesario pero el sistema pierde poder de representación, además, incorporó una tarea extra para adaptar los textos utilizados en el trabajo para que estuvieran representados con el alfabeto de símbolos reducido.

14.3. Retirar optimizaciones

El tercer punto a mejorar podría ser la retirada de alguna optimización implementada en los modelos, por ejemplo, en el modelo de traducción se está empleando una capa bidireccional recurrente acelerada directamente por la GPU, esto ha permitido un entrenamiento asumible dentro del plazo establecido, pero ha restringido la capacidad de ejecutar el modelo a, únicamente, equipos donde exista una GPU y se encuentren instaladas las librerías.

14.4. Añadir un modelo de lenguaje

El cuarto punto a mejorar es la calidad de la transcripción, para abordar este punto convendría añadir al proceso de transcripción un modelo de lenguaje como 2-gram o 3-gram. Con esto en el último paso del proceso, la calidad de la transcripción sería como mínimo igual o mejor a la actual.

15. Conclusiones y valoración

El propósito de este proyecto era obtener una aplicación móvil que automatice las reuniones de las empresas para, entre otras cosas, reducir el tiempo y dinero que anualmente se invierte.

Para cumplir con lo anterior, el sistema a desarrollar debía permitir grabar reuniones, tomar muestras de los empleados de la empresa, procesar las muestras y los audios, elaborar documentación de forma automática y permitir compartirla fácilmente.

Por tanto, los grandes problemas a resolver en este trabajo son:

- Resolver la transcripción de voz a texto
- Resolver la comparación de voces
- Resolver la identificación de género
- Desarrollar una aplicación móvil con las características antes mencionadas.

Para el caso de la transcripción, se ha alcanzado una solución razonable para el problema, dentro de los límites establecidos en el trabajo; la comparación de voz y la identificación del género han sido resueltos con éxito, además, se ha desarrollado una app de Android completa que permite acceder a toda la funcionalidad planteada en los objetivos.

Por tanto, se puede afirmar, que todos los objetivos que se perseguían han sido alcanzados satisfactoriamente, dadas la planificación temporal y los recursos computacionales disponibles.

16. Anexo

16.1. Manual de usuario para la aplicación Android

16.1.1. Dar de alta una empresa

Para dar de alta una empresa únicamente hay que iniciar la aplicación, acceder a la pantalla de acceso y registro, tal y como se muestra en la figura 16.1. Acto seguido, introducir el usuario y contraseña deseados y presionar el botón "Register", una vez realizado este paso, ya será posible acceder a la aplicación utilizando las credenciales generadas.

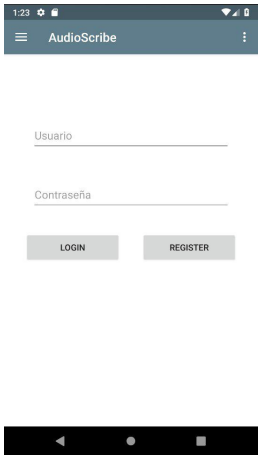

Figura 16.1: Pantalla de Acceso y Registro

16.1.2. Acceder a la aplicación

Para acceder a la aplicación con los datos de la empresa, únicamente hay que iniciar la aplicación y se mostrará la pantalla de acceso y registro (ver figura 16.1), únicamente hay que introducir las credenciales de la empresa y, automáticamente, se establecerá una sesión válida en el dispositivo con la que podremos empezar a operar.

16.1.3. Grabar una reunión

La grabación de una reunión es un proceso central para el uso de esta aplicación, por ello, es la primera pantalla que se muestra al usuario al acceder con credenciales correctas en la aplicación. Como se puede observar en la figura 16.2 es una pantalla muy sencilla donde únicamente

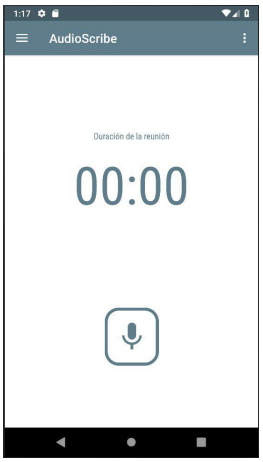

Figura 16.2: Pantalla de grabación de reunión

se muestra un único botón, este botón es el encargado de iniciar la grabación y detenerla, cada vez que se detiene la grabación, se genera un fichero de audio en la memoria externa con la grabación completa.

En esta ventana, también puede observarse un cronometro que será el encargado de marcar visualmente la duración de la reunión.

16.1.4. Menú de la aplicación

A continuación se muestra el elemento principal de navegación que posee la aplicación, el menú de acciones. Este menú es accesible desde todas las pantallas y despliega un menú lateral en el lado izquierdo de la pantalla.

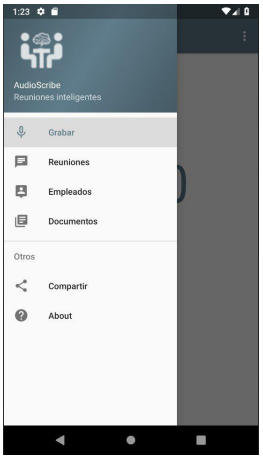

Figura 16.3: Menú de la aplicación

Este menú, contiene un botón hacia todas y cada una de las funcionalidades de la aplicación, tal y como se muestra en la figura 16.3.

16.1.5. Ver las reuniones grabadas

Para ver las reuniones que se han realizado en el dispositivo, es necesario acceder al menú lateral y presionar en el botón "Reuniones", tal y como se muestra en la figura 16.3.

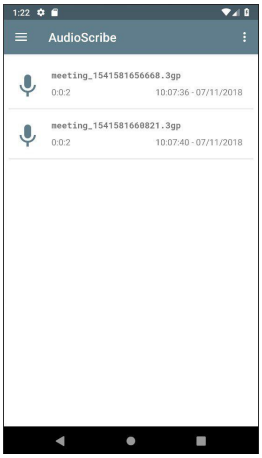

Figura 16.4: Pantalla de lista de reuniones grabadas

Una vez dentro de la pantalla de reuniones, figura 16.4, se podrá ver una lista con todas las reuniones que se encuentran en la memoria externa del dispositivo Android.

La información que se muestra de cada reunión es el nombre del fichero en la memoria, la duración de la reunión y la fecha en la que se realizó.

16.1.6. Ver las muestras de voz

Para ver las muestras de voz almacenadas en el sistema, hay que acceder al menú lateral izquierdo, acceder a la funcionalidad "Empleados" y, acto seguido, se mostrará la pantalla de la figura 16.5.

De la misma forma que en la pantalla de reuniones, aquí se mostrará una lista con todas las muestras recogidas presentes en el sistema. No se pueden reproducir, puesto que se procesa la muestra y se almacena como un fichero especial optimizado que no contiene la voz tal y como el empleado la registró.

Por lo anterior, únicamente es posible registrar nuevas muestras o eliminar alguna existente.

16.1.7. Añadir muestra de un empleado

Para añadir una muestra nueva al sistema, únicamente hay que acceder a la pantalla de muestras de voz(figura 16.5) y presionar sobre el botón azul ubicado en la parte inferior derecha de la misma.

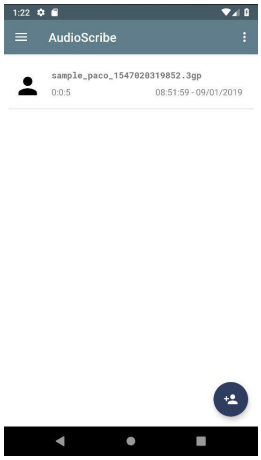

Figura 16.5: Pantalla de lista de muestras almacenadas

Esto redirige al usuario a la pantalla de grabación, donde únicamente debe hablar al micrófono del dispositivo durante al menos 10 segundos y un máximo de 20 segundos.

Con esta información, se generará un perfil de voz que permitirá comparar la muestra con las reuniones que se vayan procesando en un futuro.

16.1.8. Ver los documentos

Para ver los documentos que ha generado el sistema, únicamente hay que acceder a la funcionalidad "Documentos" del menú lateral izquierdo y se mostrará una pantalla con la lista de las reuniones transcritas.

16.1.9. Compartir información

Existen diversas formas de compartir la información presente en la aplicación, la primera de ellas es compartiendo los fichero directamente desde la memoria del dispositivo, los ficheros se encuentran en la memoria externa del terminal.

La segunda forma de compartir, es utilizar el sistema de compartición que ofrece la aplicación, para ello, únicamente hay que seleccionar la opción "Compartir" del menú lateral izquierdo y seleccionar un documento, a continuación, se ofrece añadir la grabación de voz a los elementos a compartir y, posteriormente, se ofrece una lista de acciones accesibles desde el sistema Android, como enviar la información por mensajería, email, MMS o compartir la información con otra aplicación del dispositivo.

16.2. Actas de Reunión

Se han añadido al trabajo todos los temas tratados en las diversas reuniones acontecidas durante la creación de este trabajo.

16.2.1. Día 3 de Octubre 2018

Temas tratados:

- Definir objetivos
- Definir alcance
- Definir a quien se dirige
- ¿Offline o Online?
- Como resolver cada problema
- Formas de resolver la transcripción
- Como detectar cambios de interlocutor
- Identificar personas
- Estado del arte
- Fuentes de información

16.2.2. Día 10 de Octubre 2018

Temas tratados:

- Planificar reuniones de control semanales
- Definir fechas y límites
- Definir hitos importantes
- Estimaciones de tiempo
- Establecer tareas
- Detección de cambios de interlocutor
- Resolver transcripción → ¿Deep Speech?
- redes siamesas

16.2.3. Día 17 de Octubre 2018

Temas tratados:

- Definir planificación
- Dataset
- Redes siamesas para comparar voces
- Comparar voces usando el espectrograma o el MFCC
- Economía
- Sostenibilidad

16.2.4. Día 24 de Octubre 2018

Temas tratados:

- Recopilar audiolibros
- Licencia audiolibros
- Objetivo → 200h de audio transcrito
- Competencias del proyecto
- Scripts para elaborar el dataset

16.2.5. Día 31 de Octubre 2018

Temas tratados:

- Entrega GEP 4
- Entrega GEP 5
- Presentación GEP
- Limpieza y conversión de libros
- Definir día de presentación GEP

16.2.6. Día 7 de Noviembre 2018

Temas tratados:

- Dataset y medidas de contingencia
- Desarrollo de la app
- Diseño de la app
- Funcionalidad de pantallas

16.2.7. Día 14 de Noviembre 2018

Temas tratados:

- Dataset completo
- Entrenamiento Deep Speech → 5,2h por época → 24h = 1 % → 70 días = 2,6 %
- Protocolo de comunicación REST
- Conezión App y servidor
- Seguridad en la aplicación → tokens JWT
- Envío de muestras al servidor

16.2.8. Día 21 de Noviembre 2018

Temas tratados:

- Entrenamiento comparador → 4h38min de media cada época
- Estado de API
- Modelos en producción
- Validar funcionalidad de envío de ficheros
- Objetivo de la semana: Optimizar entrenamientos

16.2.9. Día 28 de Noviembre 2018

Temas tratados:

- Mejoras con optimizaciones
- Deep Speech: 5,2h → 1,2h
- Comparación de voz: 4,4h → 11min
- Comparador entrenado y funcional
- Prueba de Comparador de voz
- Deep Speech en entrenamiento

16.2.10. Día 5 de Diciembre 2018

Temas tratados:

- Deep Speech: 70 % acierto en training
- Integración de modelos en servidor
- Comparador integrado
- Identificar género → hecho y integrado
- Definir flujo de trabajo en Servidor

16.2.11. Día 12 de Diciembre 2018

Temas tratados:

- Deep Speech: 80 % en training
- Alto indice de WER → Cambiar la forma de transcribir a palabras
- Sistema generador de PDF en servidor
- Añadir un sistema de colas
- Documento de Hito intermedio

16.2.12. Día 19 de Diciembre 2018

Temas tratados:

- Problemas generador PDF
- Cumplimiento leyes de datos
- Derecho al olvido
- Desarrollo de Android
- Definir puntos de la memoria

Bibliografía

[1] B.H. Juang, L.R. Rabiner, (2004), *Automatic Speech Recognition – A Brief History of the Technology Development*, Georgia Institute of Technology, Atlanta, Consultado desde https://web.ece.ucsb.edu/Faculty/Rabiner/ece259/Reprints/354_LALI-ASRHistory-final-10-8.pdf

[2] L.R. Rabiner, (Feb 1989),*A tutorial on hidden Markov models and selected applications in speech recognition*, AT&T Bell Lab., Murray Hill, NJ, USA, Consultado desde https://ieeexplore.ieee.org/abstract/document/18626/

[3] Sitio web: *History of Speech and Voice Recognition and Transcription Software*, Nuance, Consultado en Sep. 2018 https://web.archive.org/web/20150813223326/http://dragon-medical-transcription.com/history_speech_recognition.html

[4] Sitio web: *Nuance Exec on iPhone 4S, Siri, and the Future of Speech*, Steve Wildstrom on October 10, 2011, Consultado en Sep. 2018 https://techpinions.com/nuance-exec-on-iphone-4s-siri-and-the-future-of-speech/3307

[5] Sitio web: *The Power Of Voice: A Conversation With The Head Of Google's Speech Technology*, Jason Kincaid, 2011, Consultado en Sep. 2018 https://techcrunch.com/2011/02/13/the-power-of-voice-a-conversation-with-the-head-of-googles-speech-technology/?guccounter=1

[6] H. Beigi, (2011), *Fundamentals of Speaker Recognition*, Springer US Consultado desde https://www.springer.com/la/book/9780387775913

[7] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates and Andrew Y. Ng, (Dec 2014),*Deep Speech: Scaling up end-to-end speech recognition*, Consultado desde https://arxiv.org/abs/1412.5567

[8] E. Variani, X. Lei, E. McDermott, I.L. Moreno, J.Gonzalez-Dominguez, (2014), *Deep Neural Networks For Small Footprint Text-Dependent Speaker Verification*, ICASSP, IEEE International Conference on Acoustic Consultado desde https://static.googleusercontent.com/media/research.google.com/es//pubs/archive/41939.pdf

[9] Ariel Ephrat, Inbar Mosseri, Oran Lang, Tali Dekel, Kevin Wilson, Avinatan Hassidim, William T. Freeman and Michael Rubinstein, (Apr 2018),*Looking to Listen at the Cocktail Party: A Speaker-Independent Audio-Visual Model for Speech Separation*, Consultado desde https://arxiv.org/abs/1804.03619

[10] Hossein Salehghaffari, (Agosto 2018), *Speaker Verification using Convolutional Neural Networks*, Control/Robotics Research Laboratory (CRRL), Consultado desde https://arxiv.org/pdf/1803.05427.pdf

[11] K. Chen, A. Salman, (Marzo 2012), *Extracting Speaker-Specific Information with a Regularized Siamese Deep Network*, School of Computer Science, The University of Manchester, Consultado desde https://papers.nips.cc/paper/4314-extracting-speaker-specific-information-with-a-regularized-siamese-deep-network.pdf

[12] Dong Yu,Li Deng, (2015), *Automatic Speech Recognition*, Springer-Verlag London Consultado desde https://www.springer.com/us/book/9781447157786

[13] Joon Son Chung, Arsha Nagrani and Andrew Zisserman, (Jun 2018),*VoxCeleb2: Deep Speaker Recognition*, Consultado desde https://arxiv.org/abs/1806.05622

[14] Sitio web: *Siri*, Apple Inc., Consultado en Sep. 2018
https://www.apple.com/es/ios/siri/

[15] Sitio web: *Cortana, Tu asistente virtual y personal inteligente*, Microsoft Corp., Consultado en Sep. 2018
https://www.microsoft.com/es-es/windows/cortana

[16] Sitio web: *Dragon Mobile Assistant, Voice to text app for Android*, Nuance, Consultado en Sep. 2018
http://www.dragonmobileapps.com/android/

[17] Sitio web: *Evernote*, Evernote Corp., Consultado en Sep. 2018
https://evernote.com/intl/es

[18] Sitio web: *Gradiant Voice*, Gradiant, Consultado en Sep. 2018
http://www.biometricsbygradiant.com/

[19] Sitio web: *Cyberon Voice Commander*, App in Google Play, Consultado en Sep. 2018
https://play.google.com/store/apps/details?id=com.cyberon.cvc.ESPhl=es

[20] Sitio web: *Google Translate*, App in Google Play, Consultado en Sep. 2018
https://play.google.com/store/apps/details?id=com.google.android.apps.translatehl=es

[21] J. Sutherland, (Apr 2012),*The Scrum Papers: Nut, Bolts, and Origins of an Agile Framework*, Scrum, Inc., Consultado desde http://jeffsutherland.org/scrum/ScrumPapers.pdf

[22] S. Gupta, J. Jaafar, W.F.w. Ahmad and A. Bansal, (Agosto 2013), *Feature Extraction Using MFCC*, Indian institute of Information and Technology, Allahabad, India, Consultado desde http://aircconline.com/sipij/V4N4/4413sipij08.pdf