Universidad Zaragoza

TRABAJO DE FIN DE GRADO

Desarrollo de un banco de voces para la personalización de sistemas de síntesis de voz

Voice Database development to personalize speech synthesis systems

Autor

Borja Novo Huerta

Director

Eduardo Lleida Solano

Ingeniería de Tecnologías y Servicios de Telecomunicación

Escuela de Ingeniería y Arquitectura de Zaragoza

-

2021

Agradecimientos

En primer lugar, quiero dar las gracias a Eduardo por toda la ayuda que me ha brindado a lo largo del proyecto, por su paciencia, su comprensión y su entrega a la hora de apoyarme y resolver cualquier duda. He disfrutado y aprendido mucho más de lo que esperaba con este trabajo y tener esta oportunidad ha sido gracias a él.

De la misma manera, agradecer a Frank toda la ayuda que me ha prestado en la parte del desarrollo de la aplicación móvil de este trabajo. Siempre dispuesto a explicarme conceptos totalmente nuevos para mí y asistiéndome para conseguir el mejor resultado posible.

A Ana, que fue capaz de ver lo que podría lograr cuando ni yo mismo creía en mí, gracias por estar ahí desde que nos conocimos.

Gracias a todos mis amigos, los que tuve y los que tengo, desde Borja, el Juan de Lanuza, el hockey hierba, el Pedro de Luna, los juegos, Discord y la universidad. Vosotros le habéis dado sentido a mi vida y una recompensa por la que merecía la pena seguir luchando.

Por último, gracias a mi familia, y en especial a mis padres y mis hermanas. Gracias por cuidarme, alegrarme, quererme y confiar en mí.

Resumen

Los sistemas de síntesis de voz son una parte clave en las herramientas de conversión de texto a audio, donde éste último es reproducido por una voz artificial. Estos sistemas tienen aplicación en una gran variedad de campos diferenciados, entre los que se destaca especialmente las tecnologías de apoyo para las personas con dificultades en el habla.

Con el objetivo de crear modelos de voz para personas que en un futuro puedan perderla, se ha desarrollado un sistema completo compuesto por una aplicación móvil para la obtención de grabaciones, una base de datos que alberga un banco de voces y el software necesario para desarrollar la personalización de sistemas de síntesis de voz.

El resultado final de este trabajo es un sistema funcional capaz de crear un modelo de voz promedio masculino, femenino o mixto de la base de datos y adaptar éste a un usuario en concreto, consiguiendo así un modelo de voz final con la suficiente calidad y las distintivas características de esa persona.

Abstract

A speech synthesis system is a tool used for text to speech conversion where an artificially synthesized or a natural voice reproduces said text. These systems have many applications in different fields, especially excelling in supporting technologies for people with speech disorders.

With the aim of creating voice models for people that are going to lose their voices, a complete system composed of a mobile application to gather speech recordings, a voice storing database and the software needed to build the personalized voice models has been developed.

The final result of this project is a functional system capable of building an average voice model of male, female or mixed origin and adapting it to a particular user, thus achieving a high-quality personalized voice model with the distinctive characteristics of said user.

Tabla de contenidos

1. Introducción

1.1. Motivación

La síntesis de voz a día de hoy se encuentra en constante crecimiento, estando presente en la vida de la mayoría de las personas. Las voces artificiales de los asistentes de Google o Amazon, los mensajes informativos en los servicios de transporte o los contestadores automáticos de llamadas son algunos ejemplos del impacto de esta tecnología, que se usa tanto en entornos empresariales como públicos.

En el campo de la salud la síntesis de voz se utiliza dentro de las tecnologías de apoyo. Permite a sus usuarios realizar actividades que antes no eran posibles debido a sus condiciones médicas. Las aplicaciones de mayor uso son los lectores de pantalla, que facilitan la lectura a las personas con discapacidades visuales o dislexia, y las voces de ayuda para personas con dificultades del habla. Esta última aplicación es la que se desarrollará en este trabajo de fin de grado.

La motivación para la realización de este proyecto reside en la oportunidad de ayudar a personas con discapacidades del habla o que tengan riesgo de padecerlas a través de sistemas de síntesis de voz.

1.2. Objetivo

En este trabajo de fin de grado se busca implementar un sistema que cree un modelo de voz personalizado de calidad utilizando la base de datos de voces Albayzín y las grabaciones del propio usuario. Las grabaciones de los usuarios se han recogido desde una aplicación móvil desarrollada para tal uso, haciendo que el sistema sea más accesible para estos.

Los atributos de mayor importancia para un sistema de síntesis son la “inteligibilidad” y la “naturalidad":

- La inteligibilidad expresa la medida de entendimiento del mensaje que contiene el audio producido.
- La naturalidad, el grado de cercanía de la voz producida a la natural del usuario.

Para alcanzar los niveles deseados en estos atributos se puede optar por diferentes métodos de síntesis. En este proyecto se trabaja con el método basado en modelos ocultos de Markov, ya que es una tecnología consolidada en este campo y nos ofrece unos niveles de inteligibilidad y naturalidad altos, consiguiendo además un ahorro de memoria significativo a la hora de crear los modelos de voz si lo comparamos con otros métodos populares como el de síntesis concatenativa.

La aplicación móvil de este proyecto sirve como “front-end” entre el usuario y el sistema que creará su modelo de voz. Muestra por pantalla una serie de frases que el usuario debe grabar con el micrófono de su dispositivo. La aplicación sube estos audios a un servidor web, desde el que se ejecutará el proceso de creación del modelo de voz.

El objetivo de este proyecto es mostrar que a través de las últimas tecnologías de síntesis de voz, una aplicación móvil y una base de datos se puede desarrollar un sistema de creación de modelos

de voz que puede ser utilizado por entidades de salud públicas o privadas para ayudar a personas con dificultades del habla.

1.3. Visión general del proyecto

Este proyecto desarrolla un sistema de creación de voces artificiales que puede dividirse en una parte "front-end" formada por la aplicación móvil y otra "back-end" formada por el software de síntesis de voz.

El "front-end" del sistema es una aplicación móvil capaz de recoger las grabaciones que permitirán crear los modelos de voz para los usuarios. Esta aplicación se ha desarrollado en Android Studio, ya que esta herramienta es una de las más utilizadas en el mercado actualmente y ofrece una gran cantidad de librerías y funciones clave para la creación de esta aplicación. La aplicación conecta con una API a través de peticiones HTTP, permitiendo así crear una estructura completa de servicio al usuario con registros, inicios de sesión, elección de tratamientos y subida de grabaciones. Cabe destacar que la API con la que se realiza la comunicación no forma parte de la realización de este proyecto.

El "back-end" se ha desarrollado dentro de un servidor de la Universidad de Zaragoza, donde se han instalado varias herramientas de desarrollo de sistemas de síntesis como HTS y Festival.

Partiendo de estas herramientas se ha programado el software responsable de la síntesis de voz, el cual es capaz de crear varios tipos de modelos voz que se expondrán más adelante . Estos modelos han sido sometidos a pruebas de calidad utilizando el método llamado "Dynamic Time Warping" donde se comparan con grabaciones de voces naturales y se hace una medida de distorsión.

Por último, se ha desarrollado un script que, una vez se hayan tomado todas las grabaciones necesarias de un usuario desde la aplicación móvil, ejecute paso a paso todo el software desarrollado para crear el modelo de voz sin necesidad de intervención humana.

1.4. Estructura del proyecto

Tras haber expuesto la motivación, el objetivo y la visión general del proyecto, la estructura de la memoria continúa con los siguientes apartados:

- En el capítulo 2, "Metodología y herramientas", se explican los métodos utilizados y conceptos referentes a la síntesis de voz, el desarrollo de la aplicación móvil y las pruebas de calidad del producto final, así como las herramientas que los aplican.
- En el capítulo 3, "Implementación", se expone la puesta en uso de los métodos y las herramientas presentadas en el capítulo anterior. En concreto, se explican:
    - La instalación y obtención de todos los recursos y herramientas necesarios.
    - La construcción paso a paso de los tres tipos diferentes de modelos de voz.
    - La automatización del proceso completo a partir de la creación de un script ejecutable.
    - El desarrollo de la aplicación móvil.

- En el capítulo 4, “Resultados”, se presenta el producto final del proyecto, grabaciones artificiales creadas a partir de los modelos de voz generados y la interfaz de la aplicación móvil. También, se ofrece la evaluación de la calidad de las grabaciones obtenidas.
- Por último, en el capítulo 5, “Conclusiones”, se expone la conclusión del proyecto junto a varias propuestas de futuro que pudiesen dar continuidad a partir de éste.

2. Metodología y Herramientas

2.1. Metodologías

2.1.1. Modelado de una señal de voz

Hoy en día ya es ampliamente conocido que el proceso de producción del habla humana se aproxima a o puede representarse usando un filtro digital. Es importante destacar que el sistema natural es mucho más complejo que nuestra aproximación y por esto el filtro debe reproducir de la forma más simple y cercana posible la realidad.

El proceso de producción del habla comienza cuando desde el cerebro se envía a través del sistema nervioso la orden de generar voz. Esta orden hará que las cuerdas vocales reproduzcan una serie de vibraciones a la frecuencia indicada desde el cerebro si se desea producir un fonema sonoro o simplemente relajar las cuerdas vocales y dejar pasar el flujo de aire de los pulmones si el fonema debe ser sordo. Además, colocará los músculos del tracto vocal en la forma deseada, creando así un filtro acústico que modificará la onda de presión creada desde las cuerdas vocales generando voz por la boca. Este proceso se puede ver representado en la figura 1, junto con su aproximación digital en la figura 2.

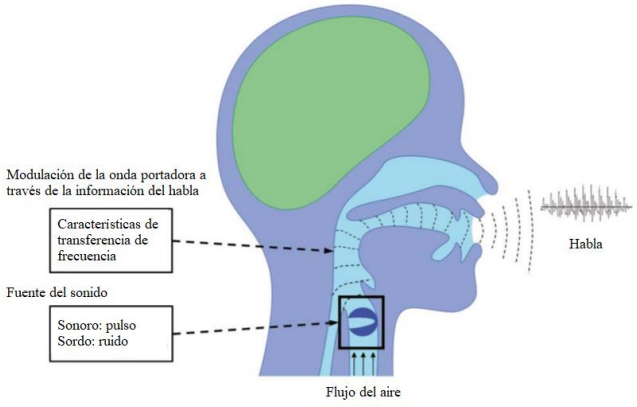


Figura 1. Proceso de producción del Habla.

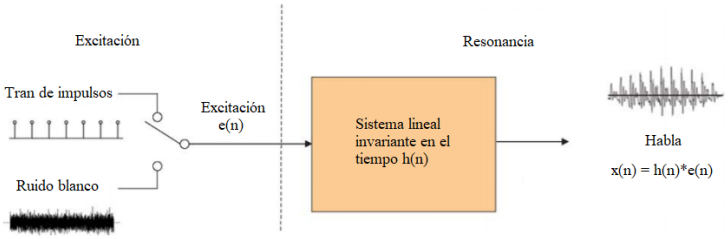


Figura 2. Aproximación digital del proceso de producción del habla.

El sistema digital trabaja con los conceptos del pitch y de los parámetros espectrales. El pitch se define como la frecuencia fundamental producida en la vibración de las cuerdas vocales, junto a los parámetros de la amplitud y la duración establece la entonación del fonema que se va a reproducir. Para los hombres esta frecuencia suele estar entre los 85 y 180 Hz mientras que para las mujeres y los niños varía desde los 165 a los 500 Hz. Esta variación proviene de las diferencias de longitud o nivel de tensión en las cuerdas vocales de cada persona.

Los parámetros espectrales contienen información de las frecuencias de resonancia generadas en el tracto vocal. Estas frecuencias se denominan formantes y se representan con el concepto cepstrum. El cepstrum es uno de los métodos de análisis de señal más utilizados para representar el tracto vocal. Es una transformación homomórfica que transforma convoluciones en sumas. El cepstrum se define como la transformada inversa del logaritmo del módulo de la transformada de Fourier de la señal. Esta nueva señal contendrá información de la envolvente del tracto vocal en los primeros coeficientes, por lo que si se tomasen únicamente éstos y se realizase el proceso inverso se podría obtener la respuesta impulsional del tracto vocal.

2.1.2. Síntesis de voz

A partir de los atributos del pitch y los parámetros espectrales se puede crear un sistema que simula de forma eficiente el de la figura 2 y permite hacer síntesis de voz. Este sistema funciona generando tramas de señal en las cuales se recogerá la información de una pequeña fracción de tiempo del audio. Dependiendo de si el fonema reproducido sea sonoro o sordo, se generará un tren de impulsos con distancia inversa a la frecuencia del pitch o ruido blanco. Esta señal a continuación pasará por el filtro que contiene los coeficientes espectrales y representa tanto el tracto vocal como las variaciones producidas por las cavidades de la nariz y los labios.

Es importante destacar que cuanto más se parezcan estos atributos (pitch y parámetros espectrales) a la generación de voz humana, más inteligibilidad y naturalidad tendrá la voz sintetizada, por lo que se debe perseguir maximizar la calidad de estos atributos.

2.1.3. Modelos ocultos de Markov

Tal y como se ha avanzado en la introducción, los modelos ocultos de Markov son una de las últimas y más utilizadas tecnologías para hacer síntesis de voz. Estos modelos de síntesis de voz se diferencian del resto por crear un sistema completamente estadístico que puede definir y reproducir la voz de un usuario. Al tratarse de un sistema estadístico y no determinista, las limitaciones son mucho menores, especialmente en la cantidad de memoria, ya que una vez se ha entrenado el sistema, se puede crear voz a partir solo de estadística, sin necesidad de almacenar grabaciones en una base de datos.

Observando la figura 3 [1] podemos comprender el funcionamiento y la estructura de un HMM (Hidden Markov Model). Un HMM, λ, está definido mediante un conjunto de tres parámetros [2]:

- π_i define la probabilidad de inicio de la cadena en el estado i.
- $\overrightarrow{a_i} = \{a_{ij}\}$ define el vector de probabilidades de transición del estado i. Es la probabilidad de que siendo i el estado de la cadena en el instante t, en el t+1 el estado sea j.
- $b_i(o_t)$ define la probabilidad de salida de un observable o_t perteneciente a un estado i.

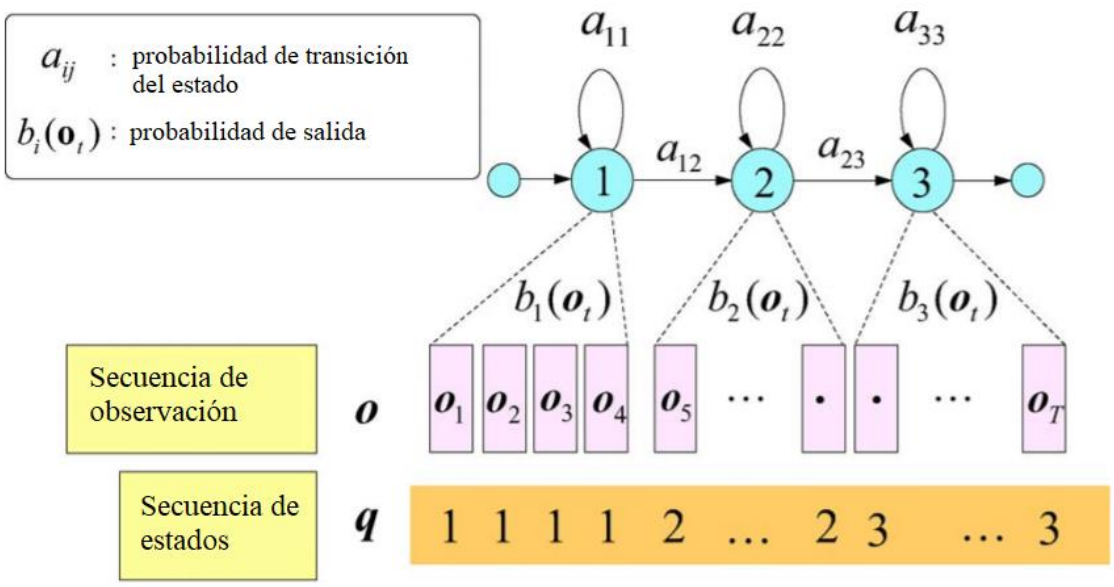


Figura 3. Ejemplo de una cadena triestado HMM.

Para poder obtener los valores de las probabilidades de este sistema (para entrenarlo) se debe partir de una base de datos de la voz a sintetizar, junto a los respectivos textos que está reproduciendo. Estos audios almacenados serán divididos en un número de tramas y clasificados por el tipo de fonema que guardan. Cada fonema almacenado tendrá asociado una cadena HMM con sus múltiples estados, siendo estos estados divididos en tramas también, a cada una de las cuales se les asocia un observable. Estos observables son vectores que contienen varios tipos de información, pero pueden dividirse en dos tipos simplificados: parámetros de excitación (pitch) y parámetros espectrales (sceptrum). Cada uno de estos dos tipos de información llevarán asociados a ellos parámetros dinámicos (velocidad y aceleración) que ayudarán a mantener la coherencia y la continuidad entre las tramas. La figura 4 [1] muestra un ejemplo de un vector de observación. La figura 5 [1] enseña un ejemplo de los parámetros de salida referentes a una serie de fonemas.

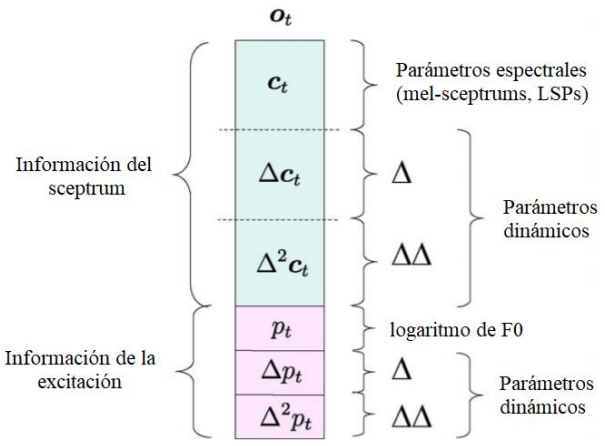


Figura 4. Ejemplo de un vector de observación de una trama.

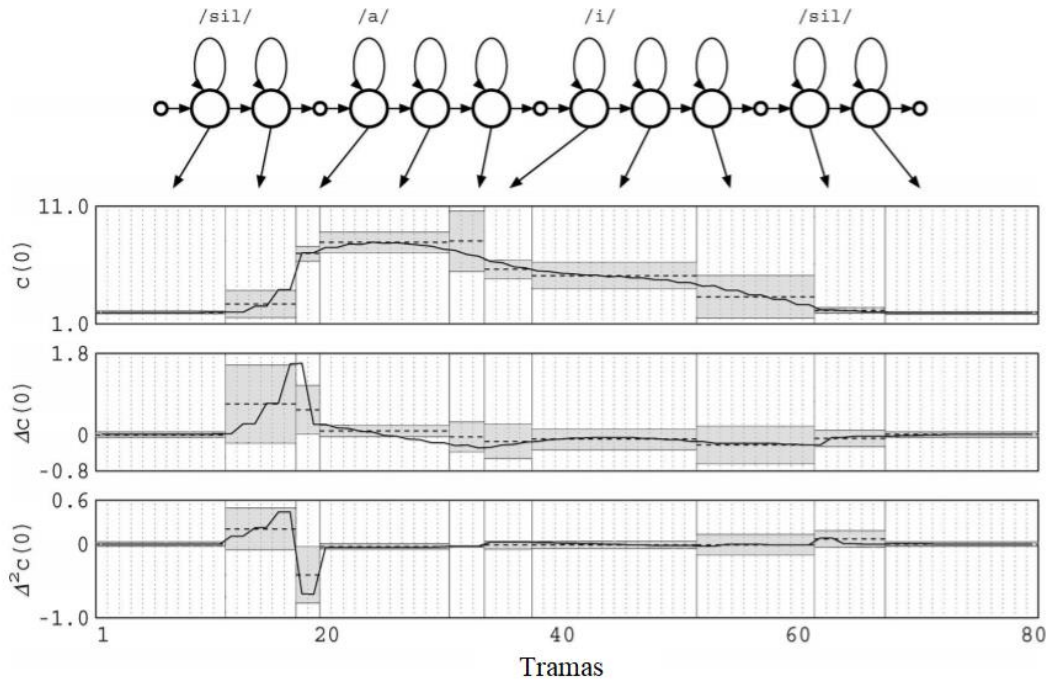


Figura 5. Ejemplo de los parámetros de salida referentes a una serie de fonemas.

Para este trabajo, la probabilidad de estos observables será continua, ya que trabajamos con señal de voz. Esto significa que se modelará la probabilidad de salida de los observables utilizando una combinación de contribuciones gaussianas diferente para cada estado.

El entrenamiento del sistema completo se realiza utilizando el algoritmo de Baum-Welch. Esto se debe a que el entrenamiento no cuenta con una solución analítica cerrada que permita garantizar la obtención del óptimo, pero si cuenta con un máximo al menos local de su verosimilitud. Este planteamiento brinda una gran ventaja: en la reestimación iterativa de los parámetros del sistema, la verosimilitud del modelo completo aumenta cada vez hasta alcanzar un máximo, como mínimo, local. En el apéndice A se puede ver un ejemplo más detallado del funcionamiento de los modelos ocultos de Markov.

Una vez el entrenamiento se ha completado, como se ha explicado antes, únicamente con las cadenas HMM de cada fonema y un texto a analizar se puede crear voz sintetizada. El texto que analizar será etiquetado y clasificado por fonemas, cada uno de estos fonemas tendrá asociado a sí un HMM, que devolverá los valores de pitch y parámetros espectrales asociados a éste. Una vez se han obtenido los valores de pitch y parámetros espectrales de todos los fonemas contenidos en el texto, con el modelo digital de producción de voz explicado anteriormente en los puntos 2.1.1 y 2.1.2 se puede crear la frase sintetizada. El sistema completo de entrenamiento y síntesis puede verse en la figura 6 [1].

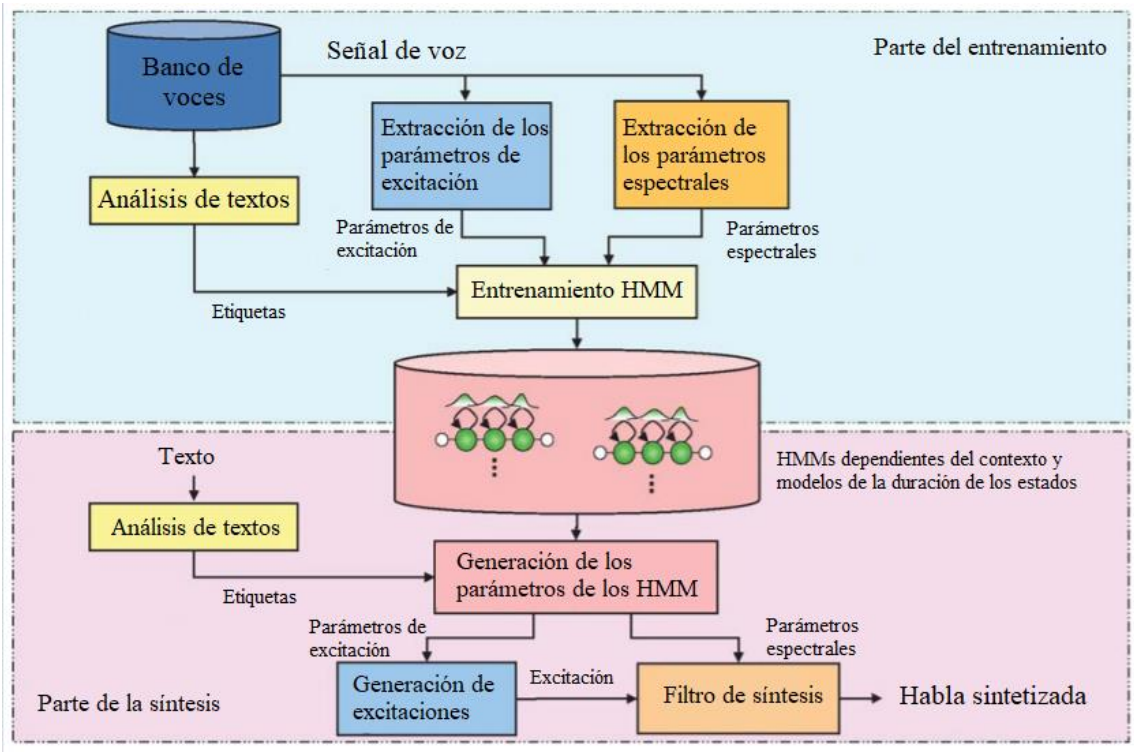


Figura 6. Sistema completo de entrenamiento y síntesis.

2.1.4. Estructuración de una aplicación móvil

En el desarrollo de aplicaciones móviles existen varios modelos de estructuración. En este trabajo de fin de grado en concreto se ha definido la estructura de ésta a partir de sus requisitos [3]. La clasificación de los distintos requisitos se define de la siguiente manera:

- Requisitos funcionales: agrupan las acciones principales que la aplicación ejecutará.
- Requisitos de datos: recogen el tipo, valor y la precisión de los datos que serán utilizados.
- Requisitos de usuario: describen las características de los potenciales usuarios de la aplicación.
- Requisitos de usabilidad: especifican los planes de usabilidad y las necesidades de uso.

Cada uno de los requisitos que entren dentro de esta clasificación estará definido además por su necesidad, su justificación y su descripción. Podemos ver un ejemplo de este método de estructuración en la figura 7.

Requisito funcional RF-001	
Nombre	Inicio de sesión
Necesidad	Esencial
Justificación	El usuario debe identificarse para usar la aplicación.
Descripción	Para poder acceder a la app el usuario deberá ingresar su correo y su contraseña

Una vez se han clasificado todos los requisitos de la aplicación móvil, se diseña un esquema del sistema que presente todas las funcionalidades de la aplicación y las interconecte de la misma forma que lo estarán dentro del dispositivo. Se ha optado por este método de estructuración ya que presenta de una forma ordenada y clara los requisitos y las funcionalidades de la aplicación a través de las tablas y define de una forma descriptiva y fácil de entender su funcionamiento con el esquema del sistema.

2.1.5. Dynamic Time Warping

Por último, una vez se haya realizado el entrenamiento del sistema y los modelos de voz ya sean funcionales, será necesario realizar pruebas de calidad de estos. Para ello se ha escogido el método llamado Dynamic Time Warping (DTW), debido a su gran popularidad en la comparación de patrones del habla.

El objetivo del DTW [4] es comparar dos secuencias similares e independientes de distintas longitudes que no llegan a estar perfectamente sincronizadas. Este método busca calcular la forma óptima de emparejar las muestras de estas dos secuencias, devolviendo una matriz de coste y una métrica llamada distancia, la cual será mayor cuanto menor parecido tengan las dos secuencias. En la figura 8 podemos ver una comparativa entre un emparejamiento euclídeo y otro realizado a través de DTW. Este ejemplo es muy representativo de la gran utilidad del DTW, ya que si optásemos por hacer una comparativa euclídea podríamos llegar a pensar que las secuencias no guardan ninguna relación, mientras que si utilizamos el Dynamic Time Warping podemos ver que ambas son muy similares. En el apéndice B se adjunta un ejemplo del cálculo del emparejamiento DTW.

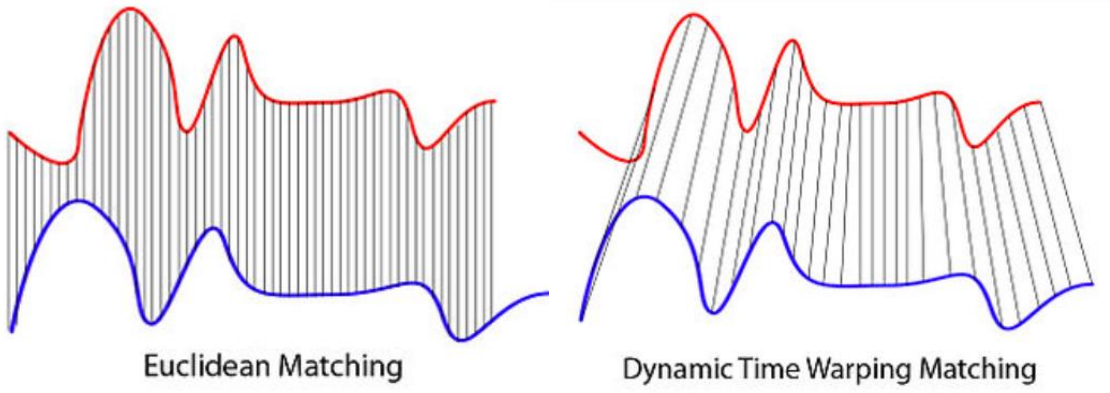


Figura 8. Comparativa entre el emparejamiento euclídeo y el DTW.

Enfocando la metodología en este trabajo, nuestro objetivo es comparar frases creadas con una voz natural respecto a las creadas por los modelos de voz artificiales. De esta forma podemos obtener datos clave como el número de frases mínimo necesario para crear un modelo de voz de calidad o ver los saltos de entre los diferentes tipos de modelos de voz, sin necesidad de usar únicamente nuestro oído como herramienta de medición.

2.2. Herramientas

2.2.1. HTS

HTS [5] es un software de código abierto de síntesis de voz basado en los Modelos Ocultos de Markov. Este sistema fue creado por el Instituto Tecnológico de Nagoya en Japón en el año 2002, a partir de otro sistema anterior llamado HTK (Hidden Markov Model Toolkit)

HTS permite configurar por completo el proceso de entrenamiento de la síntesis de voz, pudiendo escoger la longitud de la trama de los observables, la frecuencia de muestreo de los audios, el número de estados en las cadenas de Markov, etc. El sistema trabaja haciendo llamadas a diferentes funciones, que extraerán los atributos de importancia (pitch y parámetros espectrales) de la base de datos incorporada para después a través de un entrenamiento iterativo fijar las probabilidades de las cadenas de Markov para todos los fonemas observados, creando así un fichero final .htsvoice, el cual contendrá el modelo estadístico de la voz sintetizada.

Se ha optado por utilizar esta herramienta ya que además de estar basada en los HMMs, es de código abierto y se encuentra en constante desarrollo, por lo que evita cualquier disputa legal y facilita la posibilidad de mejorar el sistema en el futuro. También, las transiciones entre fonemas tienen una mayor inteligibilidad que los sistemas de métodos de concatenación y no se requiere una base de datos de las mismas dimensiones, además de que su uso es solo necesario para el entrenamiento.

2.2.2. Festival

Festival [6] es una herramienta de código abierto desarrollada por la Universidad de Edimburgo que realiza tareas de análisis y etiquetado de texto. En la figura 9 podemos ver un fragmento de un fichero .utt. Esta herramienta es necesaria ya que HTS no cuenta con funciones de este estilo.

Las dos funciones de este software que se utilizarán en el proyecto son las siguientes:

- *txt2utt*: Transforma el contenido de un fichero de texto a un formato utterance, que divide el fichero de texto en palabras y clasifica estas en tipos (palabras convencionales, números, fechas, etc.).
- *utt2lab*: Transforma el contenido de un fichero de utterance a un formato label, que etiqueta el texto por fonemas y es compatible con la herramienta HTS.

```
EST_File utterance
DataType ascii
version 2
EST_Header_End
Features max_id 157 ; type Text ; iform "\"Francia, Suiza y Hungr'ia ya hicieron causa com'un\"" ;
Stream_Items
1 id _1 ; name Francia ; punc , ; whitespace "" ; prepunctuation "" ;
2 id _2 ; name Suiza ; whitespace " " ; prepunctuation "" ;
3 id _3 ; name y ; whitespace " " ; prepunctuation "" ;
4 id _4 ; name Hungr'ia ; whitespace " " ; prepunctuation "" ;
5 id _5 ; name ya ; whitespace " " ; prepunctuation "" ;
6 id _6 ; name hicieron ; whitespace " " ; prepunctuation "" ;
7 id _7 ; name causa ; whitespace " " ; prepunctuation "" ;
8 id _8 ; name com'un ; whitespace " " ; prepunctuation "" ;
9 id _16 ; name com'un ; pbreak B ; pos nil ;
10 id _15 ; name causa ; pbreak NB ; pos nil ;
```

Figura 9. Fragmento de un fichero de etiquetado .utt.

2.2.3. Base de datos Albayzín

Para poder realizar un correcto proceso de entrenamiento de los Modelos ocultos de Markov es muy importante tener una buena base de datos, es decir, que ésta sea completa y esté equilibrada. Por este motivo se ha escogido la base de datos Albayzín [7], ya que consta de un total de 84 locutores, cada uno de ellos habiendo realizado 25, 50 o 200 grabaciones de su voz.

La característica de mayor importancia de esta base de datos es que sus grabaciones están fonéticamente balanceadas. Esto quiere decir que los contextos fonéticos de mayor relevancia en el castellano (aquellos que están presente en como mínimo el 10% de las veces que nos comunicamos a través del habla) se encuentran repetidos al menos en cuatro instancias. Gracias a esto, únicamente partiendo de la base de datos se pueden crear modelos de voz promedio muy robustos tanto para hombres como para mujeres, los cuales más adelante se pueden adaptar a un usuario en concreto para crear un modelo de voz personalizado sin necesidad de recoger una larga cantidad de grabaciones.

Cabe destacar que el formato de estos archivos de audio es .raw, con una frecuencia de muestreo de 16KHz, 16 bits por muestra y un único canal, por lo que tanto la configuración de HTS como las grabaciones del usuario al que se le vaya a crear el modelo de voz deberán tener también estas características.

2.2.4. Android Studio

Todo el desarrollo de la aplicación móvil se ha hecho desde el entorno de trabajo Android Studio. Se escogió éste sobre las otras posibles opciones como Eclipse o Visual Studio Code debido a sus características:

- Claridad: Android Studio renderiza los layouts en tiempo real, que permite desarrollar su diseño mucho más fácilmente.
- Eficiencia: Siempre que se tenga un equipo óptimamente configurado, los tiempos de compilación serán muy pequeños en comparación con las otras opciones.
- Documentación: Al estar reconocido como el framework oficial de desarrollo de Android [8], existe una gran plataforma de soporte y documentación para sus desarrolladores.
- Debug: Permite ejecutar en tiempo real la aplicación, incluso desde un dispositivo móvil, facilitando así enormemente la tarea de depuración y debug de la aplicación.

Por último, al usar este framework, el sistema operativo que dará soporte a la aplicación móvil será Android, permitiendo el mayor alcance posible de usuarios, ya que es el sistema operativo más extendido en entornos móviles en la actualidad.

2.2.5. Python

La última herramienta que se necesita para el desarrollo de este proyecto será un IDE de Python. Para ello se ha escogido PyCharm [9] ya que es uno de los entornos de desarrollo más completos para este lenguaje. PyCharm es un editor de código inteligente, que ofrece compatibilidad con Python, JavaScript, CSS y más. Su atributo de mayor importancia es la facilidad en la navegación de la ventana de herramientas, lo cual permite encontrar e instalar las funciones o librerías que se necesiten de forma rápida y segura.

Desde esta framework se desarrollará el script que aplique el emparejamiento DTW a las grabaciones obtenidas de los modelos de voz artificiales para realizar las pruebas de calidad, ya que las librerías que contienen las funciones de interés para la aplicación de este método usan Python.

3. Implementación

3.1. Instalación de los recursos necesarios

Todo el desarrollo de síntesis de voz de este proyecto se ha realizado desde el clúster GTCvoz voz01.unizar.es. Este clúster está compuesto por 4 nodos de almacenamiento y 12 de cálculo, donde los 12 nodos de cálculo tienen nombres que van desde voz01 a voz12. Para la realización de este trabajo se han utilizado los nodos voz02 a voz06. Dentro de este clúster se han instalado tanto las herramientas que componen HTS como Festival, al ser ambas de código abierto sus últimas versiones pueden ser descargadas desde las páginas web oficiales sin ningún tipo de restricción [10]. Para poder trabajar en el clúster se debe realizar una conexión remota ssh, por lo que se necesita una aplicación que pueda llevar a cabo este tipo de conexiones. Se ha escogido Visual Studio Code [11] ya que incorpora una extensión que permite realizar conexiones ssh, además, el manejo y la navegación con los archivos es mucho más intuitiva y clara que con otros clientes como Putty o MobaXTerm.

Dentro del clúster la estructura de los ficheros del trabajo está dividida en dos áreas. Por una parte se tiene un enlace simbólico a toda la base de datos de voces Albayzín y por otra se encuentra todo el software instalado de las herramientas descargadas, junto a los scripts que se desarrollan para el entrenamiento y la síntesis de los modelos de voz.

La aplicación móvil se ha desarrollado por completo desde el framework Android Studio [12], por lo que solamente es necesaria la instalación de este recurso. Cabe destacar que la tarea de depuración y debug de la aplicación se puede hacer también directamente desde un móvil, lo que requiere habilitar las opciones de depuración en ese dispositivo.

Por último, para poder llevar a cabo las pruebas de calidad se ha utilizado instalar Python junto al IDE PyCharm [13].

3.2. Construcción de un modelo de voz Mono

El primer modelo de voz que se ha desarrollado es el denominado modelo de voz Mono. Un modelo de voz Mono puede definirse como un modelo de un solo locutor, que se ha generado utilizando únicamente grabaciones de su propia voz. Este tipo de modelos tienen la ventaja de que solo necesitan grabaciones de un usuario, pero la calidad de la señal será peor que la de los modelos que utilicen bases de datos de múltiples locutores, ya que la cantidad de grabaciones será sustancialmente menor. El desarrollo de este modelo puede verse como un paso intermedio para llegar a desarrollar otro mejor, el Adaptado, que será expuesto más adelante.

El script de creación de este modelo (Albayzin-MONO) está basado en la demo HTS-demo_CMU-ARTIC-SLT [14], que se ha adaptado para poder generar modelos de voces en castellano.

3.2.1. Preparación de los datos

El primer paso en la preparación de los datos es evidentemente su obtención. Como se ha expuesto en la introducción, la idea final de este proyecto es que estas grabaciones se realicen con un

dispositivo móvil y que desde éste se suban a un servidor web. Al haberse desarrollado la parte de síntesis de voz en paralelo con la aplicación del móvil las grabaciones iniciales para la creación de modelos de voz se han recogido desde un ordenador, utilizando en concreto un micrófono incorporado en los auriculares Dragon USB Headset y la aplicación Audacity [15], la cual nos permite configurar fácilmente las características de las grabaciones a realizar (frecuencia de muestreo, un solo canal, 16 bits por muestra). La figura 10 muestra un ejemplo de una grabación recogida en Audacity.

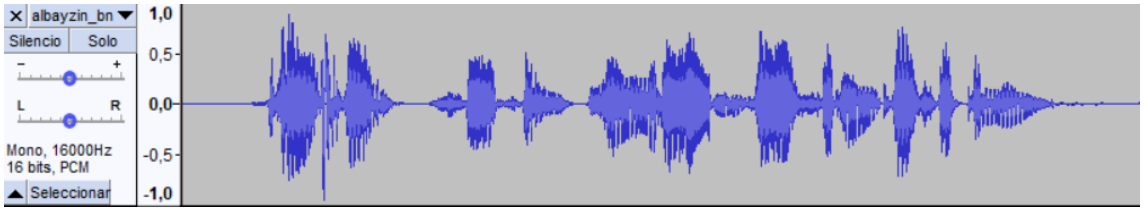


Figura 10. Grabación de voz recogida en Audacity.

Las frases reproducidas en estas grabaciones han sido extraídas de la base de datos Albayzín, ya que nos interesa mantener el balance fonético mencionado en el punto 2.2.3. En el apéndice C se muestran varias particiones de las frases de la base de datos Albayzín. Además, para mantener una nomenclatura regular, las grabaciones se guardarán como Albayzin_"código del locutor"_fp"número de grabación". Al tratarse de la primera grabación del modelo de voz Mono, ésta se llama Albayzin_bn_fp0001.

Las grabaciones se guardan en el formato .wav, e irán emparejadas con un fichero de texto .txt que contendrá la frase escrita. Para poder comenzar el proceso de modelado de la voz se necesita que el formato de los archivos de audio sea .raw (que se deshace de la cabecera .wav), por lo que se debe utilizar el script *wav2raw* de SPTK [16] que incorpora HTS. También se debe utilizar la función de Festival *txt2utt* para transformar el archivo de texto a uno de uterancias, como hemos visto en el punto 2.2.2. Este proceso divide la frase en palabras y realiza una clasificación de los tipos que encuentra (palabras convencionales, números, fechas, etc.) para de esta forma poder tratar adecuadamente la pronunciación y prosodia de cada palabra. El número total de grabaciones escogido para la creación de este modelo ha sido de 75.

3.2.2. Configuración del script

El último paso antes de comenzar el entrenamiento de los modelos es la configuración del script. Como se ha expuesto en el punto 2.2.1, HTS te permite configurar todos los parámetros clave del entrenamiento, como son la frecuencia de muestreo, el número de estados en las cadenas de Markov o el número de iteraciones del entrenamiento. La configuración se hará desde el fichero *Config.pm.in*. Al tratarse del modelo Mono, que como se ha explicado antes no será el modelo de elección a la hora de crear las voces artificiales de mayor calidad, no merece el esfuerzo necesario para la personalización, por lo que los valores de estos parámetros se mantendrán en los predeterminados.

3.2.3. Entrenamiento

El primer paso en el entrenamiento es la extracción de los datos en las señales de audio. Partiendo de los ficheros .raw se realiza una transformación para obtener de la secuencia tanto el pitch como los coeficientes cepstrum. Estos últimos se extraen empleando una ventana deslizante (la cual puede ser una ventana Blackman, Hamming o Hanning, siendo la de Hamming la predeterminada).

Un total de 25 coeficientes ceptsrum serán almacenados en ficheros .mgc para cada trama. A su vez, se crea un fichero .lf0 que contiene una transformación logarítmica, $\log(f_0)$, de los valores del pitch para cada trama. Finalmente, para tener los datos del audio listos para el entrenamiento, estos dos tipos de información se almacenarán conjuntamente en un mismo fichero de extensión .cmp, junto a unos coeficientes dinámicos, como podemos observar en la figura 11.

f_0	Δf_0	$\Delta^2 f_0$	$\underline{c}$	$\underline{\Delta c}$	$\underline{\Delta^2 c}$

Figura 11. Distribución de la información de una trama en un archivo .cmp

Para extraer los datos de los textos almacenados en cada archivo .utt, se elaborarán dos archivos de etiquetas con extensión .lab. El primer archivo, denominado *full*, contendrá toda la información relacionada con el contexto del fonema (duración, fonemas que le preceden o le suceden y posición dentro de la frase) mientras que el segundo, denominado *mono*, solo contendrá los fonemas con sus correspondientes duraciones.

Antes de comenzar el entrenamiento se generarán una serie de ficheros donde se guardarán las direcciones de los directorios donde todos los nuevos archivos de etiquetas se han almacenado.

El entrenamiento comienza con la obtención de los modelos iniciales. Éstos se obtienen a partir de la preparación de los datos que se acaban de exponer. En primer lugar, partiendo de la función *HCompV* se calcula la media y la varianza global de los parámetros de voz obtenidos en los archivos .cmp, generando un fichero init.mmf con los datos de las varianzas globales, que servirán como punto de partida a la hora de realizar las iteraciones.

Estas varianzas globales, también definidas como varianzas mínimas, serán las que permitirán elaborar los primeros modelos a través de la función *HInit* en dos etapas diferenciadas. Primero se realizará una segmentación uniforme de los fonemas, dividiendo éstos en partes iguales (siendo cada parte correspondiente a un estado en la cadena de Markov) y dándoles a cada uno una primera estimación de sus parámetros de salida. Una vez la segmentación uniforme se ha completado se pasará a aplicar el algoritmo de Viterbi. Este algoritmo obtiene de cada lectura que realice la

secuencia de estados más probable para ese fonema, calculando las probabilidades de transición entre estos estados a partir del número de veces que se haya visitado en el proceso de alineamiento.

El proceso de la función *HInit* finalizará una vez se haya alcanzado el número máximo de iteraciones o la mejora entre una iteración y la siguiente no sea lo suficientemente significativa. Podemos ver un esquema del proceso en la figura 12.

Figura 12. Esquema de la función *HInit*.

Para finalizar la obtención de los primeros modelos se llamará a la función *HRest*. Esta función se emplea para reestimar los modelos iniciales de los fonemas que no tienen información contextual. Su funcionamiento es similar al de la función *HInit*, diferenciándose en que ésta parte de los modelos generados por la primera y aplica el algoritmo de Baum-Welch en vez del de Viterbi.

El siguiente paso en el entrenamiento será la elaboración de los modelos para los fonemas de los cuales sí se tiene información contextual. Este paso es de vital importancia ya que permitirá que los fonemas que se pronuncian de diferente manera dependiendo de su contexto no sean tratados igualmente. A través de la función *HHEd* se manipulan los conjuntos de modelos y realizan nuevas agrupaciones. Inicialmente a distintos contextos del mismo fonema se le asignarán el mismo modelo, por lo que se deberá llamar a otra función, *HERest*¸ la cual aplicando de nuevo el algoritmo Baum-Welch de forma iterativa conseguirá mejorar estos modelos hasta el punto de que su pronunciación sea diferenciada.

Una vez completada la estimación inicial de los modelos es necesario realizar una re-estimación para que estos lleguen a ser funcionales. Esta re-stimación se divide en tres etapas, empezando por la aplicación del concepto llamado “Clustering”. El “Clustering” consiste en crear una relación entre modelos de fonemas que mantengan alguna similitud en sus estados. La función *HHEd* formará estas relaciones generando diferentes árboles de decisión de distinto tipo (pitch, parámetros espectrales y duración), ya que el contexto afecta a los atributos de la señal del audio de forma diferente. Una vez más se aplicará la función *HER*est de forma iterativa finalizando la primera etapa de la re-estimación.

La segunda etapa deshace las relaciones entre los modelos contextuales a través de la función *HHEd* y vuelve a aplicar la función *HERest*. Finalmente, la tercera etapa repetirá el proceso de “Clustering”, permitiendo así la generación de modelos de duración gracias a que el proceso de re-estimación ha mejorado significativamente los modelos de pitch y parámetros espectrales.

Finalizada la re-estimación de los modelos, el entrenamiento continúa con el cálculo de la varianza global. Este cálculo en necesario para solucionar el efecto de voz sorda producida por el “sobresuavizado” generado a lo largo del entrenamiento, ya que la varianza global está incorrelada con éste. Las funciones *HCompV, HERest* y *HHEd* llevarán a cabo este cálculo.

Por último, para finalizar el entrenamiento, se realiza la conversión de todos los datos generados a lo largo del proceso a un único fichero de formato .htsvoice. Usando la función *HHEd* se recogerán los modelos de los fonemas, los árboles de decisión y los cálculos de las varianzas globales. El fichero final resultante tendrá el nombre Albayzin_”código del locutor”.htsvoice.

3.2.4. Síntesis

Una vez se ha generado el fichero de voz .htsvoice el proceso de síntesis se realizará tal y como se ha expuesto en la figura 6 del punto 2.1.2. Primero se debe escribir la frase que se deseé reproducir en un archivo de texto .txt. A través del ejecutable de Shell Script *txt2lab.sh* el archivo de texto será transformado a un formato de etiquetas .lab. El ejecutable *lab2wav.sh* llamará a este archivo de etiquetas junto al fichero del modelo de voz .htsvoice y generará un .wav en el que la voz artificial reproducirá la frase contenida en el archivo de texto. En la figura 13 se muestra un esquema descriptivo del desarrollo de síntesis.

Figura 13. Esquema del proceso de síntesis.

Todo este proceso se ha automatizado dentro de un solo ejecutable, *tts.sh*, al cual se le dará como parámetro de entrada únicamente la frase a reproducir y el modelo de voz a utilizar.

Como podemos ver en la figura 14, las formas de onda generadas artificialmente (abajo) son muy similares a las del audio original (arriba). Si se reproducen ambas frases conjuntamente, podemos ver que este modelo ha logrado recoger la naturalidad de la voz original con éxito. No podemos decir lo mismo de la inteligibilidad, ya que como podemos ver más claramente en el espectrograma de la figura 15, este modelo ha generado un alto nivel de ruido, especialmente en las frecuencias más altas.

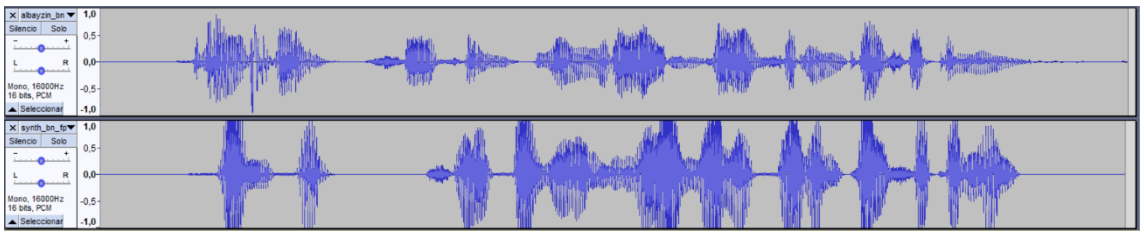


Figura 14. Comparativa de las formas de onda originales (arriba) con las sintetizadas (abajo).

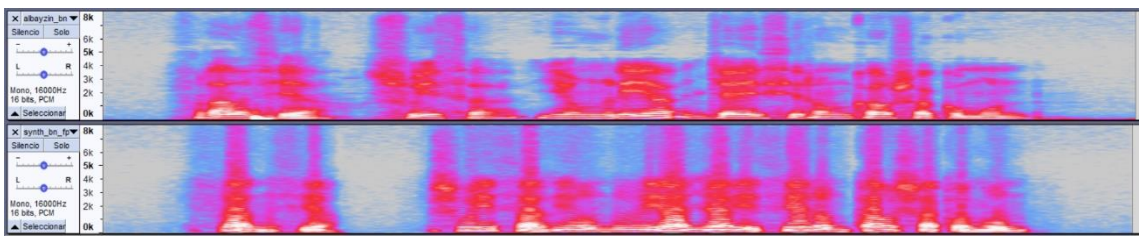


Figura 15. Comparativa de los espectrogramas de la frase original (arriba) y la sintetizada (abajo).

Se adjunta junto a las figuras un enlace directo a la carpeta Mono del repositorio de GitHub creado para este trabajo de fin de grado, donde se encuentran varios archivos de audio de ejemplo de este modelo de voz:

https://github.com/BorjaUnizar/HMM-Speech-Synthesized-Voices/tree/main/Mono

3.3. Construcción de un modelo de voz Promedio

Un modelo de voz Mono puede ser de utilidad siempre que no se tenga acceso a un banco de voces, ya que permite crear una voz artificial únicamente con una pequeña cantidad de grabaciones de un mismo usuario. Sin embargo, para poder alcanzar niveles de ruido menores y una mejor inteligibilidad manteniendo la naturalidad de la voz sí que es necesaria la ayuda de una base de datos, ya que nos otorga una cantidad significativamente mayor de señal con la que entrenar. Por esta razón el siguiente modelo de voz que se ha desarrollado es un modelo Promedio. Podemos definir un modelo de voz Promedio como un modelo neutro que se genera a partir de una gran cantidad de locutores conjuntos, consiguiendo así una voz artificial que será una media de todas con las que se ha realizado el entrenamiento.

El script de la obtención de los datos de este modelo (Albayzin-AVG) está basado en la parte inicial de la demo HTSdemo_CMU-ARCTIC-ADAPT [17]. Originalmente, este script únicamente realizaba la parte inicial del entrenamiento, sin llegar a generar un modelo de voz, además de estar diseñado para el inglés. Por ello se ha modificado el script (Albayzin-AVG-Voice) para entrenar voces en castellano y generar un fichero .htsvoice que contenga un modelo de voz promedio de todas las grabaciones entrenadas.

3.3.1. Preparación de los datos

Al tratarse del modelo de voz Promedio, los datos que se utilizarán para el entrenamiento serán los procedentes de la base de datos Albayzín. Como se ha explicado anteriormente, esta base de datos está compuesta por señales de voz y textos de un total de 84 locutores, que realizaron desde 25 hasta 200 grabaciones fonéticamente balanceadas en cada caso. Todos estos archivos de voz y texto ya se encuentran en sus debidos formatos (.raw para los audios y .utt para los textos) por lo que el único paso necesario en la preparación de los datos es crear un enlace simbólico con éstos y almacenarlos.

3.3.2. Configuración del script

El primer punto clave dentro de la configuración de este script es su modificación para que éste además de realizar la parte inicial del entrenamiento, también genere un fichero del modelo final .htsvoice que se pueda utilizar para hacer síntesis. Para esto se ha modificado el script original, añadiendo un último paso que de forma análoga al modelo de voz Mono, recoja todos los modelos, árboles y varianzas globales de los locutores de la base de datos Albayzín y cree un fichero de voz promedio .htsvoice con éstos. Otro punto de importancia es la separación de los locutores de la base de datos por su sexo. Como bien se ha expuesto anteriormente, las características del habla de las personas varían dependiendo de su sexo, por lo que crear dos modelos Promedio separados, uno compuesto por locutores femeninos y otro por locutores masculinos, puede ser de interés. Esto nos permitirá obtener unos modelos de voz que tengan una mayor naturalidad debido a sus

rangos de pitch y parámetros espectrales. Además, también será de gran ayuda para el desarrollo del modelo de voz final, el Adaptado, que se describirá más adelante.

Por último, para ambos scripts (Albayzin-AVG-Voice-Female y Albayzin-AVG-Voice-Male) la configuración de los parámetros de importancia en la generación de los modelos es la siguiente:

- Cinco estados de las cadenas de Markov de los fonemas.
- Cinco iteraciones de la aplicación de los algoritmos de entrenamiento.
- Frecuencia de muestreo de los datos en las señales de audio de 16000 Hz.
- Ventana deslizante de Hamming.

3.3.3. Entrenamiento

El proceso de entrenamiento de los modelos de voz Promedio utiliza los mismos pasos y funciones que el presentado en el punto 3.2.3 para el modelo de voz Mono, diferenciándose en que la señal con la que se entrena proviene de varios locutores, por lo que se hará una media de estas voces. En la figura 16 se muestra un esquema del proceso completo.

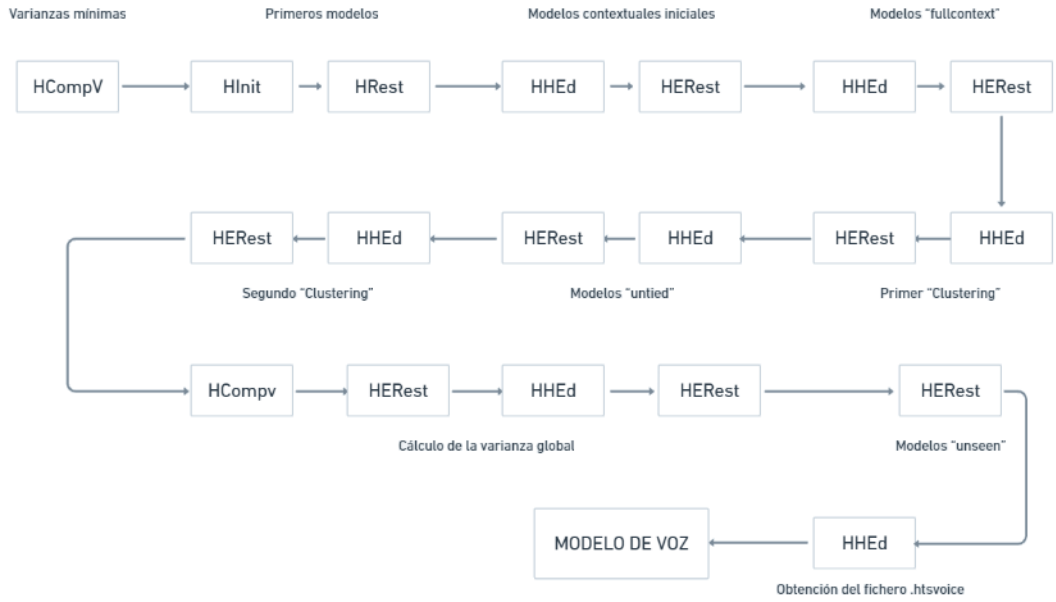


Figura 16. Esquema del entrenamiento de la voz Promedio.

3.3.4. Síntesis

Una vez obtenido el fichero .htsvoice, el proceso de síntesis sigue los mismos pasos expuestos en el punto 3.2.4, por lo que utilizando el ejecutable *tts.sh* con nuestro modelo de voz Promedio y el texto que se desee reproducir obtendremos los audios con la voz sintetizada.

Observando la figura 17 en la que se muestra la forma de onda de un audio sintetizado por la voz Promedio masculina podemos ver que el nivel de ruido existente es sustancialmente menor que el visto en los audios del modelo de voz Mono. Esto se debe a la diferencia en la cantidad de

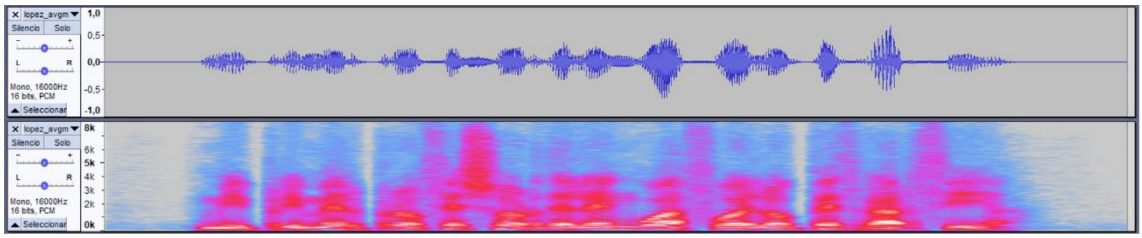


Figura 17. Formas de onda (arriba) junto al espectrograma (abajo) de una grabación del modelo de voz Promedio masculino.

señal. Cuando anteriormente la cantidad de grabaciones para el entrenamiento era de 75, para estos modelos se han usado más de 3000. Además, al haberse formado partiendo de una base de datos, siempre que ésta se amplíe, la calidad del modelo de voz que se obtenga mejorará.

Se adjunta junto a la figura un enlace directo a la carpeta Average del repositorio de GitHub creado para este trabajo de fin de grado, donde se encuentran varios archivos de audio de ejemplo de este modelo de voz:

https://github.com/BorjaUnizar/HMM-Speech-Synthesized-Voices/tree/main/Average

3.4. Construcción de un modelo de voz Adaptado

El último tipo de modelo de voz que se ha desarrollado en este proyecto ha sido el modelo de voz Adaptado. Este tipo parte de los datos del modelo de voz promedio y las grabaciones de un usuario. Mediante las herramientas de HTS se adapta la información de la voz promedio para que se asemeje a la extraída en las grabaciones del usuario. Utilizando una cantidad de grabaciones aproximada a la obtenida para la generación del modelo de voz Mono expuesto en el punto 3.2, podemos conseguir un modelo de voz robusto, de calidad y con la naturalidad característica del usuario objetivo, lo cual resalta la utilidad y el potencial de este modelo de voz.

El script de creación de este modelo (Albayzin-ADAPT) está basado en la parte final de la demo HTSdemo_CMU-ARCTIC-ADAPT [16]. Se ha adaptado para que trabaje con modelos de voz en castellano y, además, se han generado dos scripts separados (ADAPT-Female y ADAPT-Male) que realizarán la adaptación partiendo de los datos de las voces promedio de cada respectivo sexo.

3.4.1. Preparación de los datos

La preparación de los datos se divide en dos partes. La primera consiste en la obtención del modelo de voz Promedio, ya que sin él, no se podrá adaptar las grabaciones que se obtengan a ningún modelo de suficiente calidad. Al haber desarrollado ya los modelos de voz Promedio tanto para voces femeninas como masculinas, simplemente es necesario almacenar los datos obtenidos dentro del script de adaptación.

La segunda parte en la preparación de los datos será la obtención de las grabaciones del usuario al que se le creará el modelo de voz Adaptado, este proceso seguirá exactamente los mismos pasos que los expuestos en el punto 3.1.1.

3.4.2. Configuración del script

Dentro de la configuración del script se debe indicar qué voz Promedio se utilizará para realizar la adaptación. Evidentemente, para voces femeninas, es mejor utilizar la voz Promedio Female, ya que al haberse generado utilizando únicamente locutores femeninos, en la amplia mayoría de los casos, los datos de los parámetros espectrales y el pitch se acercarán más a los del usuario a adaptar que si se usase una mezcla o el masculino. La misma regla se puede aplicar para voces masculinas.

Por último, la configuración de los parámetros de importancia es la misma que la utilizada para el desarrollo del modelo de voz Promedio (ver punto 3.3.2), ya que con éstos se ha conseguido generar un modelo de voz Promedio de muy buena calidad y además, mantener la misma configuración para la adaptación permitirá conservar la coherencia entre los dos modelos.

3.4.3. Entrenamiento

Como se puede ver en la figura 18, el proceso de entrenamiento del modelo de voz Adaptado se divide en dos partes. En primer lugar se realiza la generación de los modelos "unseen" y de los árboles de clases de regresión. Después, se ejecutan los algoritmos MLLR y MAP. El proceso completo se desarrolla a través de la función *HHEd* de HTS.

Figura 18. Esquema del entrenamiento de la voz adaptada.

Los árboles de clases de regresión contienen componentes agrupadas en función de su proximidad en el espacio sonoro. A pesar de que este paso se ejecute en el proceso de adaptación, la generación de los árboles de clases de regresión parte de la información obtenida de la voz promedio. La generación de los árboles comienza con la creación de un árbol de clases de regresión binario y el etiquetado de cada componente que almacene. A partir de este paso inicial se continúa con el siguiente proceso:

- Selección de un nodo terminal a partir del cual ramificar.
- Cálculo de medias y varianzas de las mezclas en ese nodo.
- Creación de dos nodos hijo con la misma media que el padre y una fracción de la varianza.
- Asignación de una componente a cada hijo aplicando una media euclídea.
- Recálculo de la media del nodo hijo de acuerdo con la componente asignada.

En la figura 19 podemos ver un ejemplo de varias clases de regresión (izquierda), junto a un árbol de clases de regresión generado con este proceso (derecha).

```
~b ''baseclass_4.base''
<MMFIDMASK> CUED_WSJ*
<PARAMETERS> MIXBASE
<NUMCLASSES> 4
  <CLASS> 1  {(one,sil).state[2-4].mix[1-12]}
  <CLASS> 2  {two.state[2-4].mix[1-12]}
  <CLASS> 3  {three.state[2-4].mix[1-12]}
  <CLASS> 4  {four.state[2-4].mix[1-12]}
```

```
~r "regtree_4.tree"
<BASECLASS>~b "baseclass_4.base"
<NODE> 1 2 2 3
<NODE> 2 2 4 5
<NODE> 3 2 6 7
<TNODE> 4 1 1
<TNODE> 5 1 2
<TNODE> 6 1 3
<TNODE> 7 1 4
```

Figura 19. Ejemplos de un fichero de clases de regresión y un fichero de árbol de clases de regresión.

El siguiente paso en el proceso de adaptación comienza con el uso del algoritmo MLLR (Maximum Likelihood Linear Regression). Este algoritmo lee los árboles de clases de regresión de forma secuencial, comienza por el nodo raíz y recorre todo el árbol, generando transformaciones en los nodos que cumplan una de estas dos condiciones:

- Se sobrepasa un umbral de información mínima que contiene el nodo.
- El nodo es de tipo terminal.
- Los nodos hijos del nodo a evaluar no sobrepasan el umbral de información mínima.

Este algoritmo se ejecutará de forma iterativa con el objetivo de refinar los resultados. Para finalizar el proceso de adaptación, se aplica el algoritmo MAP (Maximum A Posteriori), con lo que se consigue una mejora en la estimación de los modelos obtenidos a través del uso del MLLR. Este algoritmo solo se utiliza una vez.

Por último, de la misma forma que para los anteriores modelos, se realiza la conversión de todos los archivos (incluyendo los nuevos árboles de clases de regresión) al fichero .htsvoice ejecutando la función *HHEd*, obteniendo así el modelo de voz Adaptado preparado para la síntesis.

3.4.4. Síntesis

Una vez obtenido el fichero .htsvoice, el proceso de síntesis sigue los mismos pasos expuestos en el punto 3.2.4, por lo que utilizando el ejecutable *tts.sh* con nuestro modelo de voz adaptado y el texto que se desee reproducir obtendremos los audios con la voz sintetizada.

Si nos fijamos en la figura 20, podemos ver que la forma de onda resultante de la síntesis (abajo) apenas contiene un nivel de ruido detectable, además de asemejarse más a la original (arriba) que con el modelo de voz Mono, esto sucede igualmente si nos fijamos en el espectrograma de la figura 21.

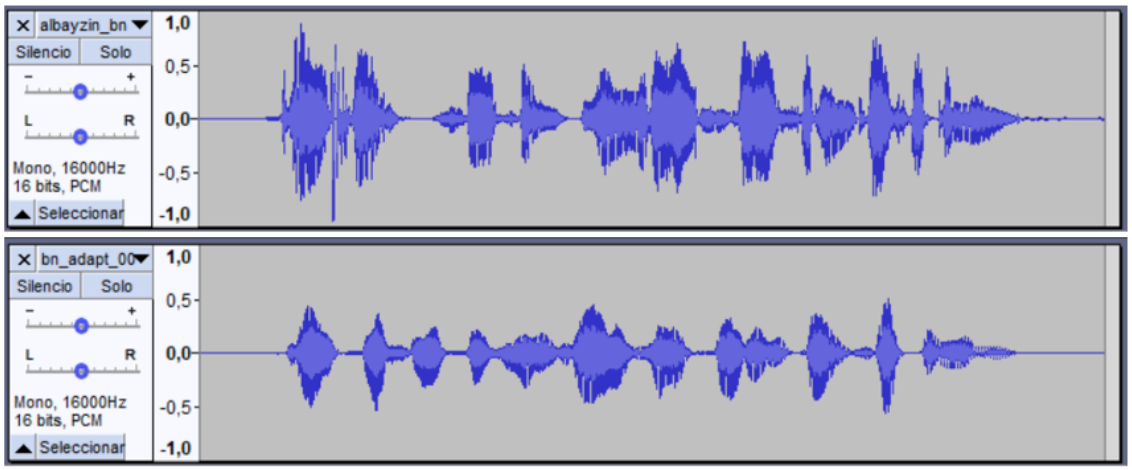


Figura 20. Comparativa de las formas de onda originales (arriba) con las sintetizadas (abajo).

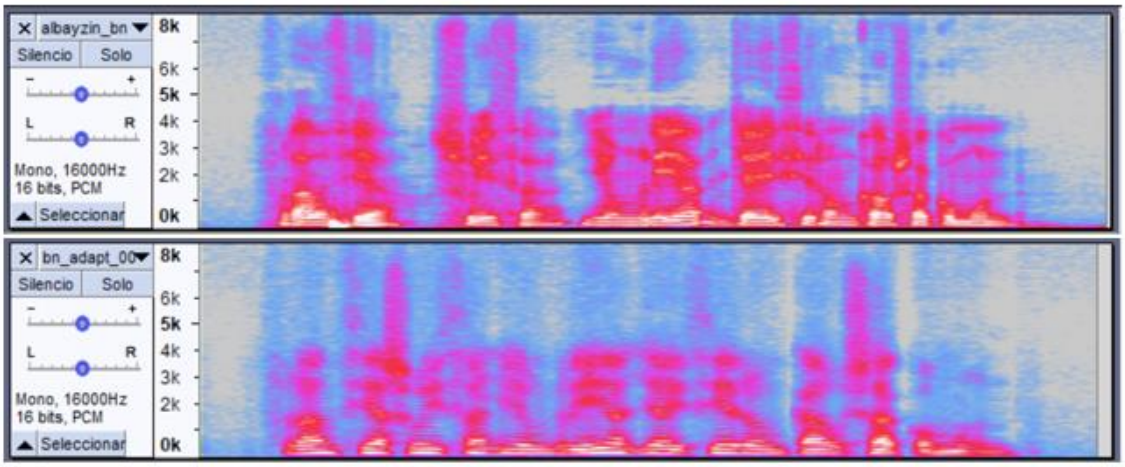


Figura 21. Comparativa del espectrograma original (arriba) con el sintetizado (abajo).

Se adjunta junto a las figuras un enlace directo a la carpeta Adapt del repositorio de GitHub creado para este trabajo de fin de grado, donde se encuentran varios archivos de audio de ejemplo de este modelo de voz:

https://github.com/BorjaUnizar/HMM-Speech-Synthesized-Voices/tree/main/Adapt

3.5. Automatización del proceso completo

Como bien se ha visto, el proceso de generación de un modelo de voz requiere de varios pasos (preparación de datos, configuración y entrenamiento), además de la ejecución de los múltiples scripts responsables de llevar éstos a cabo. Dependiendo del tipo de modelo que se desee generar, este proceso puede llegar a durar hasta 24 horas, necesitando intervención humana de forma intermitente durante éstas.

Por todas estas razones, se ha diseñado el ejecutable *genvoice.sh*, que solamente necesitará como parámetros de entrada el nombre de la carpeta que contiene los audios junto con los textos para el entrenamiento (que más adelante se utilizará como código de locutor, por lo que se recomienda

que sean dos caracteres) además del tipo de adaptación que se realizará (femenina o masculina). En la figura 22 se presenta un esquema con su funcionamiento.

Gracias a este script el proceso de generación de modelos se simplifica de gran manera. En el apéndice D se adjunta el script completo *genvoice.sh* junto a otros ejecutables de importancia en el desarrollo de los modelos de voz.

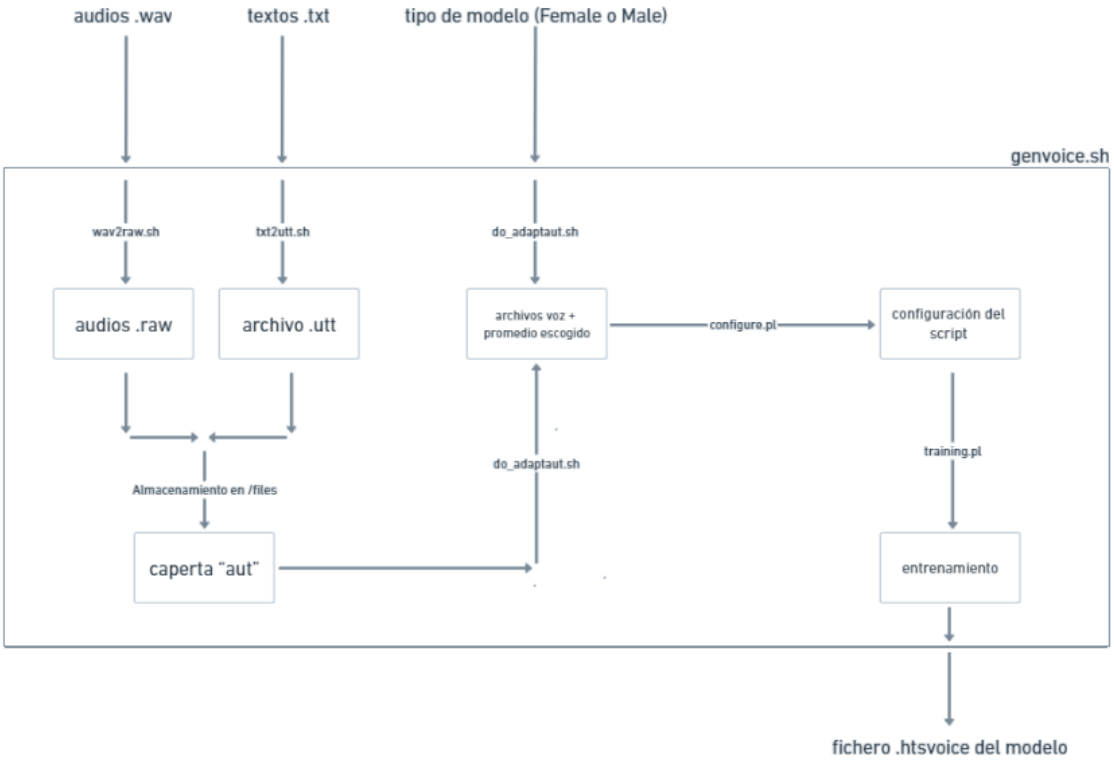


Figura 22. Esquema del funcionamiento del ejecutable genvoice.sh.

3.6. Desarrollo de la aplicación móvil

Como bien se ha expuesto en la introducción del proyecto, en este trabajo también se ha desarrollado un "front-end" cuya función será la obtención de los datos necesarios para la creación de modelos de voz. De todas las herramientas posibles para llevar a cabo la recogida de datos se ha optado por diseñar una aplicación móvil. Las razones de esta elección son las siguientes:

- Accesibilidad: En la actualidad prácticamente todo el mundo tiene su propio dispositivo móvil, o puede tener acceso a uno, lo que nos evita una potencial barrera de entrada al máximo posible de usuarios.
- Sencillez: Al tratarse de una App diseñada únicamente para este trabajo, se puede configurar para que la obtención de los datos se haga de la forma más sencilla posible, ofreciendo así al usuario una alternativa más simple y fácil de realizar que los métodos convencionales de obtención de datos.
- Adaptabilidad: Se pueden actualizar cambios o ajustar errores de forma prácticamente instantánea comparada con otros métodos de grabación. Además, a través de las mismas actualizaciones, la App puede encontrarse en un constante estado de mejora.

3.6.1. Definición de la estructura

El primer paso en el desarrollo de la aplicación móvil es la definición de su estructura. Como se muestra en la figura 23, la estructura de la App se divide en cuatro ventanas navegables:

- Ventana de registro: Esta será la primera ventana que aparecerá al ejecutar la App. Desde aquí se podrá registrar un nuevo usuario o navegar a la ventana de inicio de sesión.
- Ventana de inicio de sesión: De forma análoga a la ventana anterior, desde ésta se podrá o realizar el inicio de sesión, que nos mandará al menú principal del usuario, o navegar de vuelta a la ventana de registro.
- Menú principal del usuario: Desde esta ventana se le enseñará al usuario una lista de tratamientos disponibles. Cada uno de estos tratamientos contendrá una serie de frases. Para acceder a la estación de grabación y grabarlas, se deberá seleccionar primero un tratamiento de los que estén disponibles. Además, la ventana ofrece la posibilidad de obtener más información sobre el uso de la App, una lectura general del perfil del usuario que haya iniciado la sesión, o la posibilidad de cerrar ésta.
- Estación de grabación: En esta ventana se mostrarán las frases del tratamiento escogido. Junto a éstas, habrá la posibilidad de grabarlas, reproducirlas para comprobar su calidad, y subirlas a un servidor web. También se ofrece la opción de ver más información acerca del tratamiento seleccionado o volver al menú principal del usuario.

Evidentemente, para completar varias de las acciones claves de esta App (inicio de sesión, registro, peticiones de información, etc.), se necesita realizar conexiones con un servidor Web

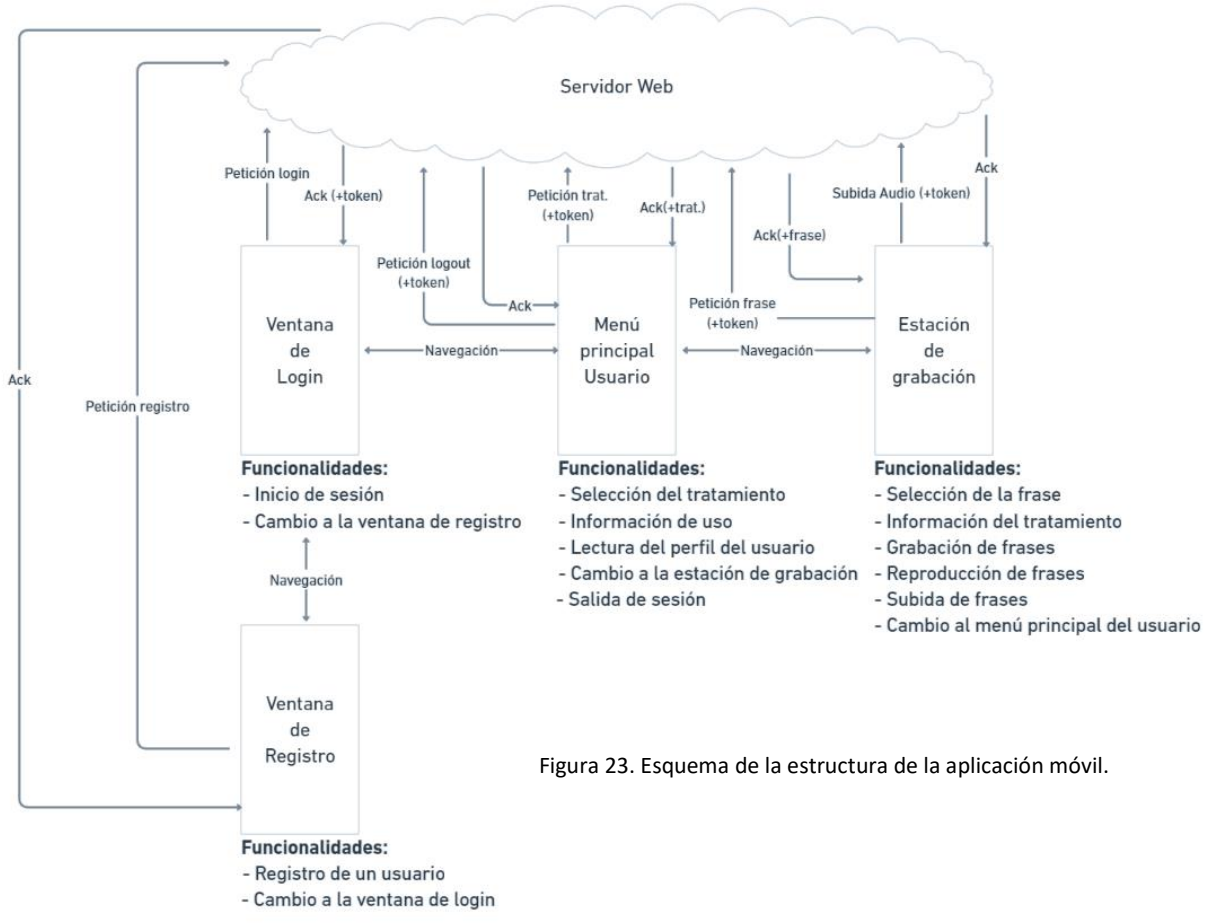


Figura 23. Esquema de la estructura de la aplicación móvil.

(en este caso, un servidor Web de la Universidad de Zaragoza). La comunicación se ha hecho a través de peticiones HTTP de diferentes tipos (POST y GET) y con diferentes parámetros o cabeceras extra. En la figura 24 se muestra un esquema generalizado del procedimiento.

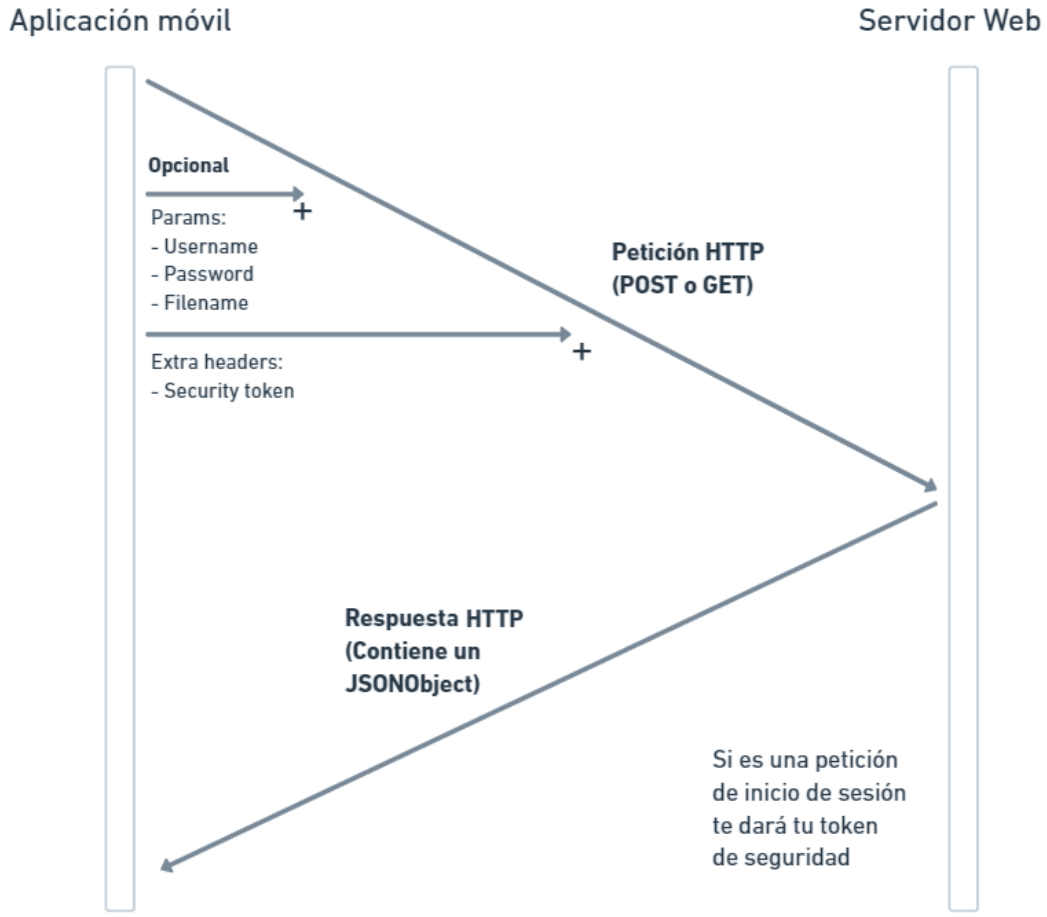


Figura 24. Generalización del proceso de comunicación con el servidor web.

3.6.2. Requisitos

Tal y como se ha explicado en el punto 2.1.4, para lograr presentar de una forma clara y ordenada la estructura de la aplicación móvil se ha realizado una clasificación por requisitos:

Requisitos funcionales:

Requisito funcional RF-001	
Nombre	Registro
Necesidad	Esencial
Justificación	Clave para el funcionamiento de la App identificar a los usuarios
Descripción	Registro de un nuevo usuario mediante un formulario

Requisito funcional RF-002	
Nombre	Inicio de sesión
Necesidad	Esencial
Justificación	Clave para la navegación y comunicación con el servidor web
Descripción	Inicio de sesión de un usuario ya registrado

Requisito funcional RF-003	
Nombre	Cierre de sesión
Necesidad	Esencial
Justificación	Desde un mismo dispositivo puede acceder más de un usuario
Descripción	Cierre de sesión de un usuario que haya abierto la sesión

Requisito funcional RF-004	
Nombre	Intercambio de información con el servidor web
Necesidad	Esencial
Justificación	Clave para poder obtener los tratamientos o subir los audios
Descripción	Mediante peticiones HTTP habrá intercambios de info con el servidor web

Requisito funcional RF-005	
Nombre	Navegación entre ventanas
Necesidad	Esencial
Justificación	Necesario para moverse entre las diferentes ventanas de la App
Descripción	Cambio de ventanas dependiendo de la acción que se realice

Requisito funcional RF-006	
Nombre	Grabación de voz
Necesidad	Esencial
Justificación	Necesario para poder recoger las grabaciones de los usuarios
Descripción	Habilitación de la grabadora para recoger las frases reproducidas

Requisito funcional RF-007	
Nombre	Reproducción de voz
Necesidad	Opcional
Justificación	Reproducir la voz ayuda a evaluar la calidad de la grabación que se haya hecho
Descripción	Reproducir la última grabación realizada

Requisito funcional RF-008	
Nombre	Información extra
Necesidad	Opcional
Justificación	Mostrar información extra puede ayudar a comprender el funcionamiento de la App
Descripción	Mostrar información extra sobre el funcionamiento de la App

Requisitos de datos:

Requisito de datos RD-001	
Nombre	Formato del audio
Necesidad	Esencial
Justificación	El audio recogido por la App debe estar en un formato que el servidor web entienda
Descripción	El formato del audio grabado es .3gp

Requisito de datos RD-002	
Nombre	Formulario del registro
Necesidad	Esencial
Justificación	Todos los datos pedidos en el registro deberán ser rellenados
Descripción	Para registrarse tendrás que indicar un nombre de usuario y un correo, además de la contraseña

Requisitos de usuario:

Requisito de usuario RU-001	
Nombre	Idioma castellano
Necesidad	Esencial
Justificación	La aplicación está diseñada únicamente en castellano
Descripción	Para poder usar la aplicación debes de ser hispanoparlante

Requisito de usuario RU-002	
Nombre	Manejo con los dispositivos móviles
Necesidad	Opcional
Justificación	Un buen manejo de los dispositivos móviles facilita la utilización de la App
Descripción	Tener un buen manejo con los dispositivos móviles

Requisitos de usabilidad:

Requisito de usabilidad RUS-001	
Nombre	Conexión a Internet
Necesidad	Esencial
Justifica-ción	Para lograr comunicarte con el servidor web se necesita conexión a Internet
Descripción	El dispositivo móvil debe tener conexión a Internet

Requisito de usabilidad RUS-002	
Nombre	Dispositivo con micrófono
Necesidad	Esencial
Justificación	Es necesario que el móvil tenga un micrófono para realizar las grabaciones
Descripción	El dispositivo móvil debe tener un micrófono

3.6.3. Programación

Una vez están bien definidos los requisitos y la estructura de la aplicación móvil se puede comenzar con la programación de ésta. Esta App se ha diseñado en Android Studio. Este framework divide las ventanas de la aplicación que diseñes en dos partes. Por un lado se encuentran las "Activities", scripts de Java que se encargarán de llevar a cabo toda la interactividad de la ventana y, por el otro, están los "Layouts", estos son ficheros .xml que definirán la estructura, el diseño y el estilo de la ventana.

Al crear un nuevo proyecto en Android Studio, además de generar una "Activity" y un "Layout" predeterminado, se crearán los siguientes ficheros de importancia:

- AndroidManifest.xml: Fichero .xml de configuración general del proyecto. El nombre de la aplicación, su icono, los permisos o los formatos de las actividades son varios de los posibles parámetros a configurar dentro del archivo.
- Gradle Scripts: Conjunto de ficheros encargados de descargar las librerías de interés para la aplicación y usar la versión de Android que se desee.
- Carpeta "res": Esta carpeta almacenará todos los recursos gráficos que se deseen utilizar para la aplicación.

En la figura 25 se muestra un esquema simplificado de la estructura de un proyecto de Android Studio.

El primer paso en el desarrollo de la App es el diseño de los "Layouts" correspondientes a cada una de las ventanas que habrá. Android Studio ofrece la posibilidad de desarrollar éstos en modo Diseño, el cual no necesita uso de programación, ya que muestra una paleta con una amplia variedad de componentes (textos, botones, imágenes, etc.) y da la posibilidad de colocar estos mismos en una vista predeterminada de la ventana. Es importante destacar que los "Layout" diseñados son de tipo "RelativeLayout". Escoger este tipo frente al predeterminado u otros es clave para que las distancias entre los componentes que aparezcan en la ventana no se alteren dependiendo del dispositivo en el que se utiliza la App (dos móviles de distintos tamaños deben ver la misma ventana, simplemente a una diferente escala).

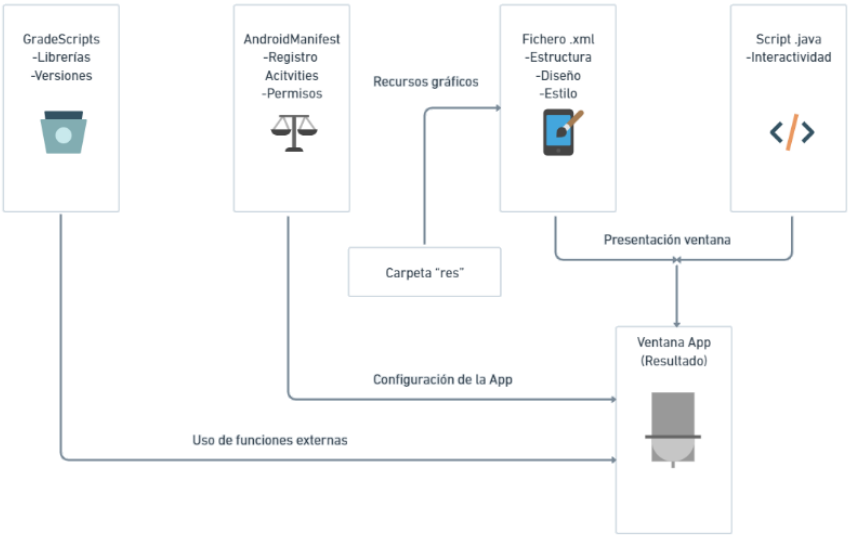


Figura 25. Esquema simplificado de la estructura de un proyecto de Android Studio.

Una vez completados los diseños de todas las ventanas de la App, se puede pasar a la programación de la parte interactiva de la App. En este capítulo se hará una descripción breve y concisa de los cuatro puntos claves en la interactividad de la App (navegación, comunicación con el servidor web, almacenamiento de memoria y grabación y reproducción de las frases), en el apéndice E se adjuntarán ejemplos de las secciones de código responsables del funcionamiento de éstos.

- Navegación: La navegación entre diferentes ventanas en Android funciona gracias a los objetos de tipo Intent. Un Intent es un objeto de mensajería que solicita una acción de otra componente de la App. De esta forma, si generamos un Intent con destino a la ventana que deseemos enseñar por pantalla y lo asociamos a una función startActivity(), podremos saltar de la ventana origen a la objetivo.
- Comunicación con el servidor: Como bien se ha expuesto anteriormente, la comunicación con el servidor web se realiza a través de peticiones HTTP. Para esto, es importante en primer lugar habilitar el permiso de uso de Internet en el AndroidManifest. Una vez hecho, Android ofrece una larga variedad de librerías que permiten realizar conexiones HTTP con servidores web. En este caso, se ha optado por Volley[18]. Volley es una biblioteca HTTP que facilita y agiliza el uso de redes en Apps para Android, ofrece una personalización de las peticiones clara y sencilla, lo cual nos permite modificar las cabeceras y los parámetros con facilidad. Un punto de importancia en la comunicación con el servidor web es el uso de tokens de seguridad (bearer token authetication). Al iniciar la sesión, el servidor web te devolverá dentro del JSONObject de la respuesta un token de seguridad, durante el resto de la sesión, deberás incluir este token en las cabeceras de tus peticiones, con el propósito de verificar tu identificación.
- Almacenamiento de memoria: Hay datos de importancia que la App debe guardar aunque ésta se cierre o se apague el dispositivo móvil. Un ejemplo de esta necesidad sería que al cerrar la App con la sesión iniciada, es inconveniente e incómodo para el usuario tener que iniciar sesión cada vez que abra la App. Para solucionar este problema se han utilizado objetos del tipo SharedPreferences[19], estos objetos permiten guardar pares de clave-valor, que se mantendrán a pesar de que se cierre la App. De esta forma se puede guardar el estado de la sesión o el token de seguridad de ésta.
- Grabación y reproducción de las frases: A través de la librería MediaRecorder[20] y su API, Android permite la captura y la codificación de una variedad de formatos comunes de audio. Cabe destacar que por razones legales para poder realizar grabaciones desde la App primero se debe pedir permisos al usuario. Para ello, se hace uso de la función interna requestPermissions().

4. Evaluación y resultados

4.1. Modelos de voz

Como se ha visto a lo largo del tercer capítulo, se ha logrado generar tres modelos de voz diferentes, pudiendo escuchar un par de ejemplos de cada uno en el repositorio de GitHub expuesto. Para poder calificar como exitosa la labor del proyecto, además de haber logrado la generación de estos modelos, es necesario estimar la carga de entrenamiento óptima y evaluar metódicamente la calidad de los modelos obtenidos (no usar únicamente nuestro oído como único sistema de medición).

Para realizar tanto la estimación de carga de entrenamiento como la evaluación de la calidad se utilizará el Dynamic Time Warping matching a través de un script de Python. Utilizando la librería de funciones Librosa[21], se obtendrán los coeficientes mfcc (Coeficientes Cepstrales en las Frecuencias de Mel) de varios audios, tanto naturales como artificiales, a los que se les aplicará el DTW matching. A partir de los resultados de la aplicación de este método, se podrá determinar un número de frases óptimo para entrenar un modelo de voz Adaptado y qué modelo entre los tres generados se asemeja más a la voz natural con la que se ha entrenado.

4.1.1. Estimación de la carga de entrenamiento óptima

El número óptimo de grabaciones a realizar para generar un modelo de voz de la suficiente calidad es un dato de gran valor, permite evitar crear modelos de voz con demasiadas grabaciones, que acaben teniendo una calidad semejante a otros con menores y resulten en una pérdida de recursos materiales y tiempo.

Para llevar a cabo esta estimación, se han generado a partir de una misma voz natural cuatro diferentes modelos Adaptados, con 25, 50, 75 y 100 grabaciones de entrenamiento. El DTW matching devuelve una matriz de coste acumulado junto a un valor de distancia, por lo que realizando una comparativa entre dos grabaciones originales (hechas con la voz natural) de la misma frase aplicando el DTW matching, podemos obtener una matriz de coste y un valor de distancia de referencia, que servirán como valor objetivo a alcanzar para acercarse a una voz natural. El resto de las comparaciones se harán con audios de la misma frase de los modelos generados y una de las frases originales. Se adjunta un enlace directo a la carpeta Adapt-Comparison del repositorio de GitHub creado para este trabajo de fin de grado, donde se encuentran los ficheros de audio utilizados durante la evaluación:

https://github.com/BorjaUnizar/HMM-Speech-Synthesized-Voices/tree/main/Adapt-Comparison

Aplicando lo explicado dentro del script *dtw.py* (el cual puede verse en el apéndice F), se obtienen los siguientes resultados para la comparación entre las dos grabaciones naturales en la figura 28.

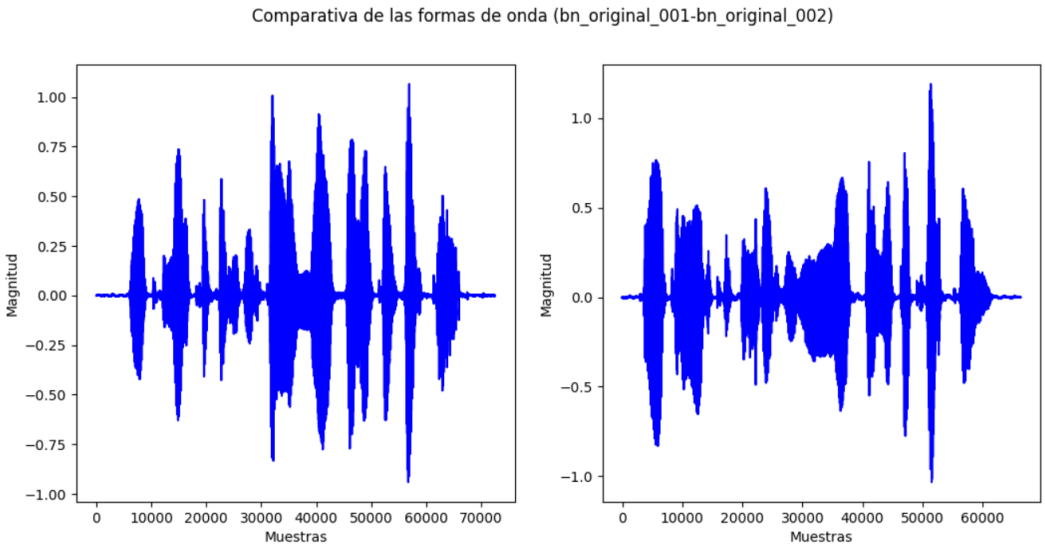


Figura 26. Comparativa de las formas de onda de los audios originales.

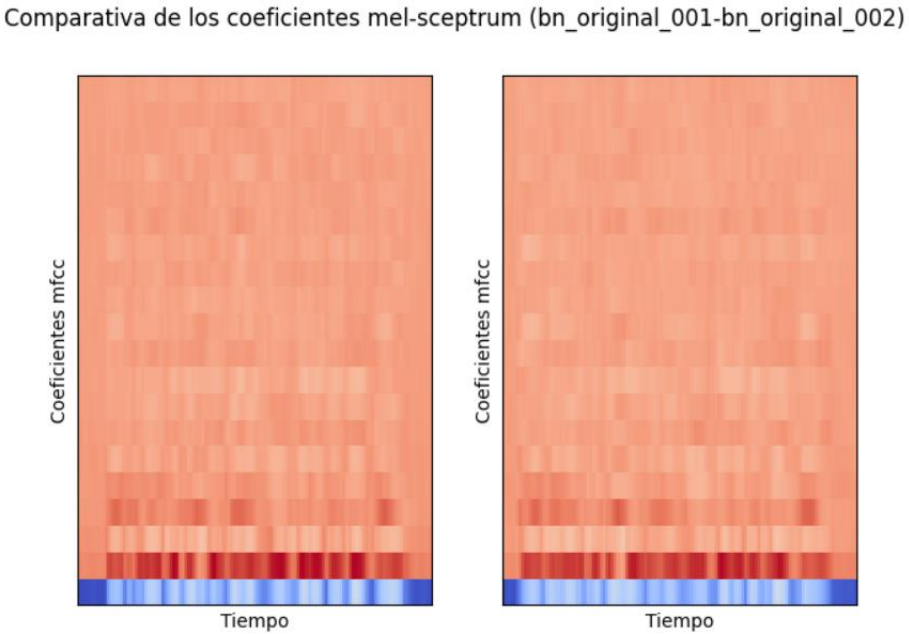


Figura 27. Comparativa de los coeficientes mel-sceptrum de los audios originales.

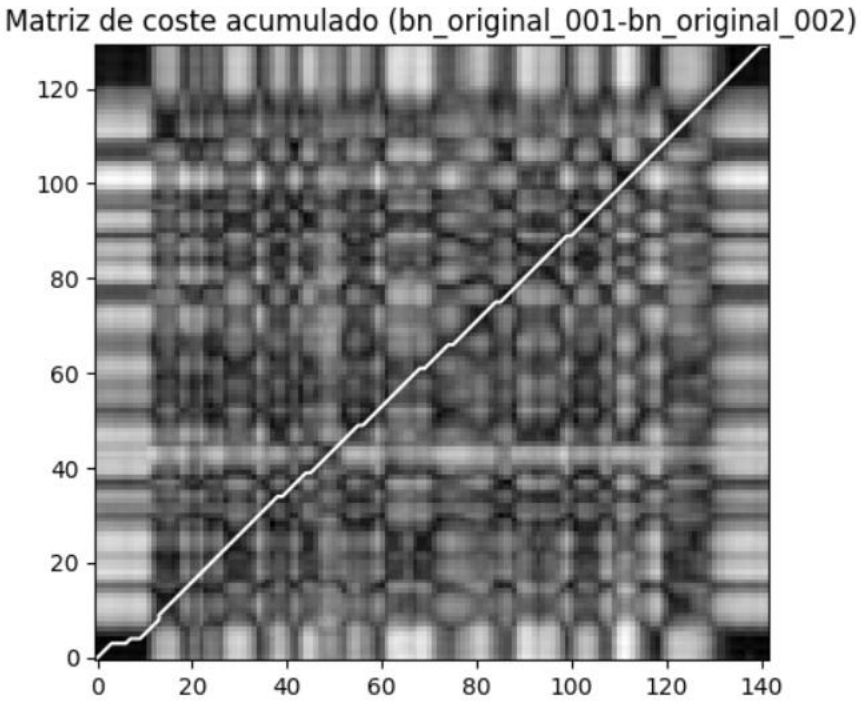
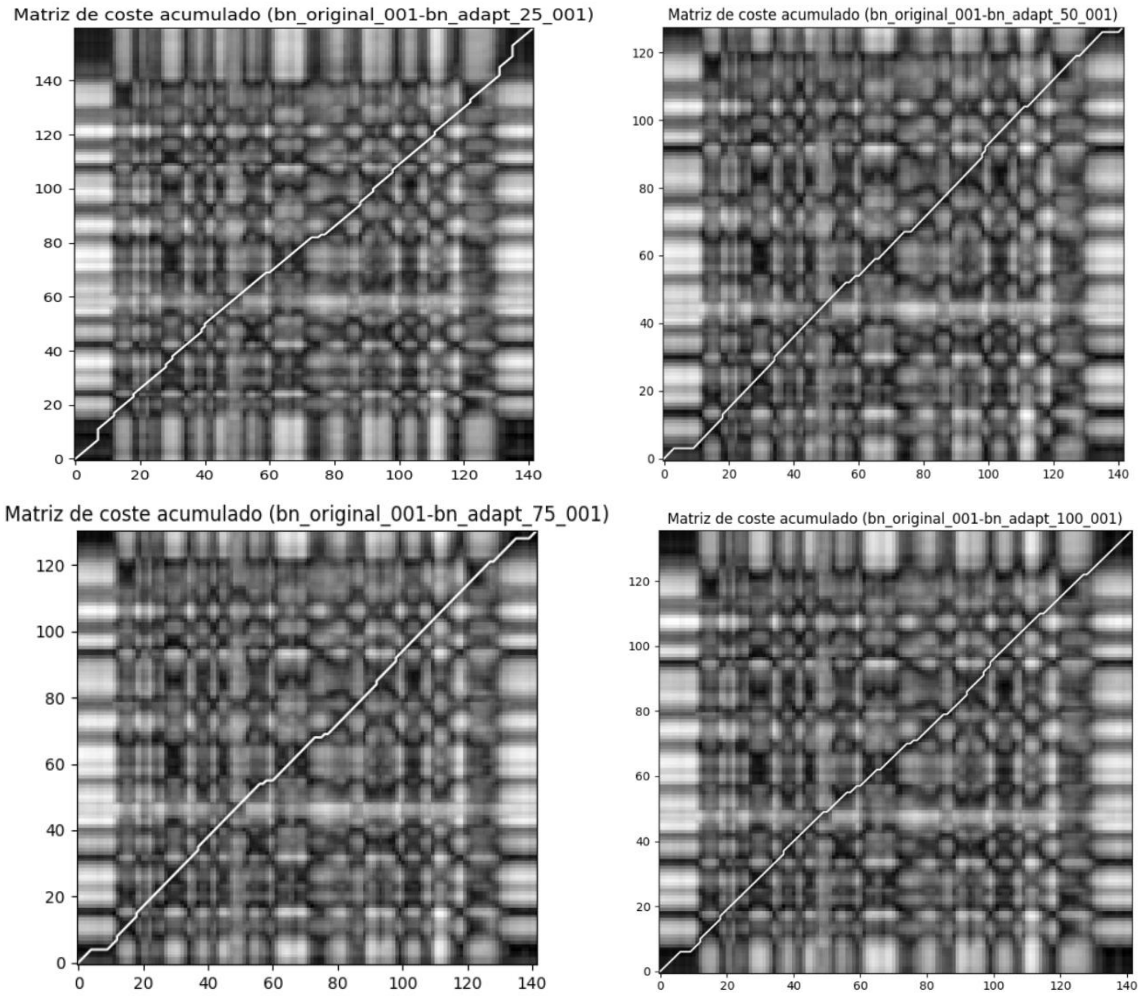


Figura 28. Matriz de coste acumulado resultante de la comparación de los audios originales.

Observando la figura 26, podemos ver que ambas grabaciones guardan una gran similitud, destacando la diferencia en las duraciones de los silencios iniciales y finales, además de la amplitud de la forma de onda. En cambio, al observar la figura 27, esta última discrepancia es mucho más leve, pudiendo observar únicamente la diferencia en la duración de los silencios. Este hecho recalca la preferencia de usar los coeficientes mfcc para el DTW matching antes que la forma de onda, ya que éstos extraen y representan las características del contenido de la señal del audio más relevantes. En la figura 28 se ve el resultado final de la aplicación del DTW. Esta matriz representa en el eje longitudinal los coeficientes mfcc del primer audio original y en el eje transversal los del segundo. La línea diagonal que surca desde la esquina inferior izquierda a la esquina superior derecha indica el emparejamiento de las muestras, como se ha elaborado en el apéndice B, esta línea determina el valor de la distancia. Para esta aplicación se ha obtenido el siguiente valor de distancia normalizada (la distancia normalizada se calcula dividiendo el valor de la distancia acumulada entre el producto de la longitud de las dos tramas que se comparan):

```
Distancia normalizada entre los dos audios (bn_original_001-bn_original_002.wav): 1.57
```

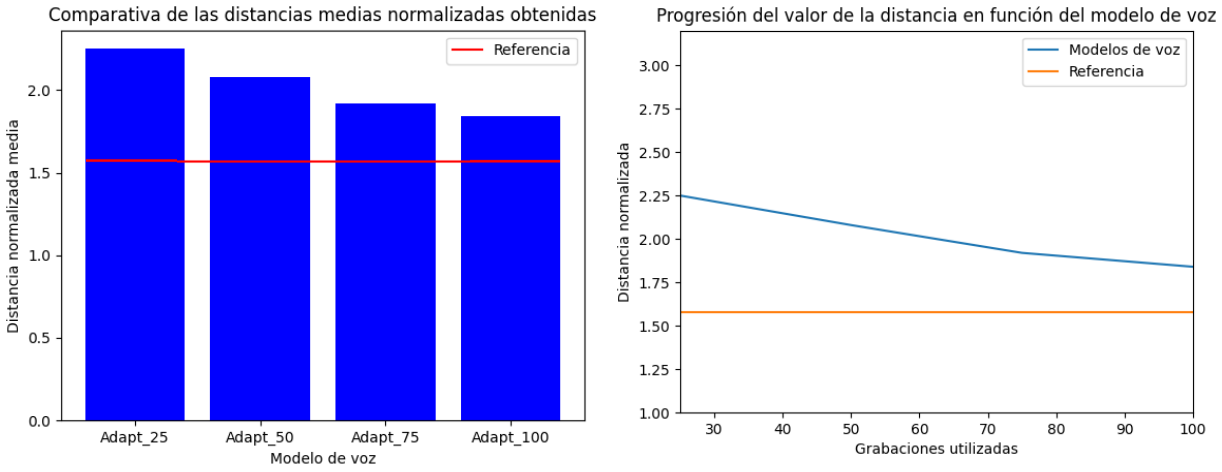
Teniendo ya el valor de referencia de la distancia se puede repetir este proceso, comparando esta vez el audio original frente a los generados con los modelos de voz artificiales, lo cual da los siguientes resultados:




```
Distancia normalizada entre los dos audios (bn_original_001-bn_adapt_25_001): 2.26
Distancia normalizada entre los dos audios (bn_original_001-bn_adapt_50_001): 2.05
Distancia normalizada entre los dos audios (bn_original_001-bn_adapt_75_001): 1.92
Distancia normalizada entre los dos audios (bn_original_001-bn_adapt_100_001): 1.82
```

Figura 29. Matrices de coste acumulado para las comparativas de audio original-sintetizado.

Como se puede ver gracias a las matrices de coste de la figura 29 y los valores de distancia obtenidos, conforme aumenta la cantidad de grabaciones en el proceso de entrenamiento del modelo de voz, la proximidad al valor de distancia normalizada de referencia es mayor. Esto indica que el uso de un mayor número de grabaciones para la generación de los modelos resulta en una mayor semejanza a la voz original. Para obtener una mayor rigurosidad de los datos se ha repetido este proceso 25 veces y se han obtenido unos valores de distancia medios de cada una de las 5 comparativas hechas (referencia, original-adapt_25, original-adapt_50, original-adapt_75 y original-adapt_100), que se pueden ver en la figura 30 a) y 30 b).




```
Distancia normalizada media (bn_original-bn_adapt_25): 2.25
Distancia normalizada media (bn_original-bn_adapt_50): 2.08
Distancia normalizada media (bn_original-bn_adapt_75): 1.92
Distancia normalizada media (bn_original-bn_adapt_100): 1.84
```

Figura 30 a) y b). Comparativa de las distancias medias normalizadas obtenidas (izquierda) junto a la progresión del valor de la distancia en función del modelo de voz (derecha).

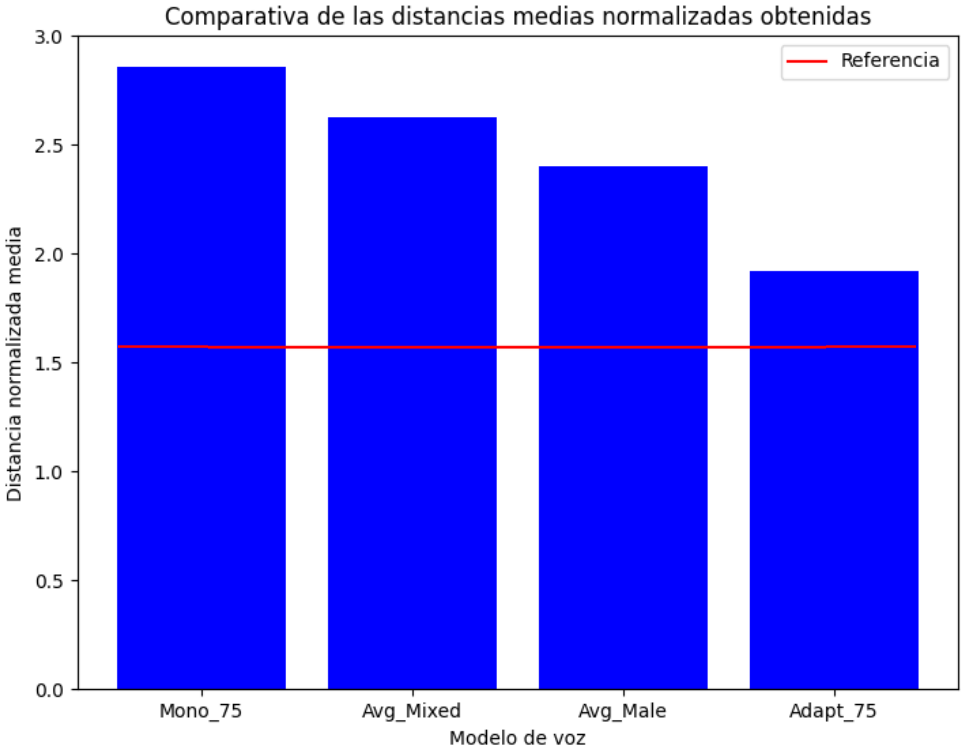
Si además de realizar una comparación auditiva se analizan los últimos resultados obtenidos, se puede razonar que la cantidad adecuada de grabaciones a realizar para el entrenamiento de un modelo de voz adaptado es 75. Auditivamente, no existe prácticamente diferencia entre este modelo y el generado con 100 grabaciones, y además, a través del método de comparación DTW se ha podido ver que ambos modelos obtienen un valor de distancia normalizada muy próximo, siendo la mejoría en la proximidad al valor de referencia mucho menor en el salto de 75 a 100 grabaciones que en los anteriores.

4.1.2. Evaluación de la calidad de los modelos

Utilizando el mismo procedimiento que en el punto 4.1.1 se evaluará la calidad de los diferentes modelos generados en el trabajo de fin de grado. Es decir, en función de la proximidad al valor de distancia de referencia calculado para la comparación entre las grabaciones originales en el punto anterior (apróx. 1.57) se clasificarán los siguientes modelos generados:

- Modelo de voz Mono, 75 grabaciones de entrenamiento.
- Modelo de voz Promedio mixto, base de datos Albayzín.
- Modelo de voz Promedio masculino, base de datos Albayzín.
- Modelo de voz Adaptado, 75 grabaciones de entrenamiento. Adaptado del modelo de voz Promedio masculino.

Ejecutando de nuevo el script *dtw_calc.py* para 25 grabaciones distintas de cada modelo obtenemos los siguientes resultados medios:




```
Distancia normalizada media (bn_original-bn_mono_75): 2.86
Distancia normalizada media (bn_original-albayzin_avg_mixed): 2.63
Distancia normalizada media (bn_original-albayzin_avg_male): 2.40
Distancia normalizada media (bn_original-bn_adapt_75): 1.92
```

Figura 31. Comparativa de las distancias medias normalizadas obtenidas.

Observando los valores de distancias normalizadas obtenidos en la figura 31, podemos destacar varios puntos clave de esta evaluación. En primer lugar, el modelo de voz Mono, a pesar de haberse generado únicamente con grabaciones de la voz original, es el modelo que más alejado se encuentra del valor de referencia, estando por detrás incluso de los modelos Promedio mixto y masculino, los cuales no han utilizado ninguna grabación procedente de la voz original para el entrenamiento. Esto se debe a la alta presencia de ruido en el modelo Mono, que desaparece de gran manera para los modelos Promedio, resaltando la importancia de tener una buena base de datos para poder generar modelos de voz de calidad. El otro punto clave en la evaluación es el valor de la distancia del modelo de voz Adaptado, ya que es el modelo que más se acerca al valor de referencia. Esto se debe a que el modelo de voz Adaptado combina los beneficios de los modelos anteriores (la naturalidad y voz característica del usuario en el modelo Mono junto a la inteligibilidad y calidad de señal del modelo Promedio) eliminando a su vez sus desventajas (alto nivel de ruido del modelo Mono y falta de parecido con el usuario original del modelo Promedio), estableciéndose como el mejor modelo de los tres estudiados en este trabajo.

4.2. Estado final de la aplicación móvil

La aplicación móvil resultante que se ha logrado desarrollar ha cumplido tanto con la estructura definida como con los requisitos establecidos. Como se puede ver en los ejemplos que se muestran en las figuras 32 y 33, se ha diseñado una interfaz visual clara y sencilla. Se ha realizado una demostración donde se observa cómo realizar el registro de un nuevo usuario, para después iniciar sesión y grabar, reproducir y subir un audio, mostrando a lo largo del transcurso de la demo todas las funcionalidades de la aplicación. Se adjunta un enlace directo con la demostración del funcionamiento completo de la App.

https://www.youtube.com/watch?v=yyI5Q9zo_no&ab_channel=BorjaUnizar

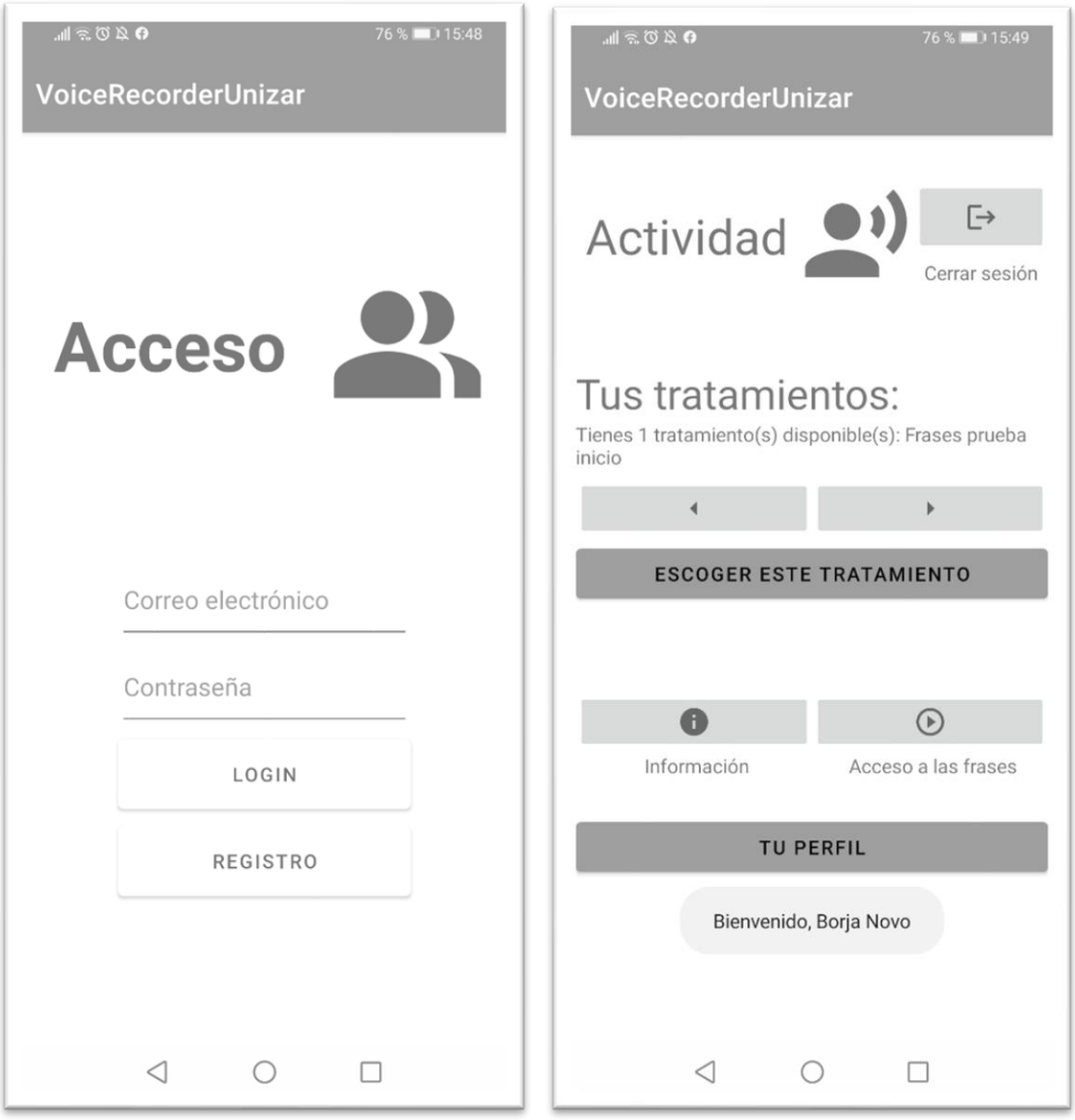


Figura 32. Interfaces visuales de las ventanas de inicio de sesión y el menú principal.

Figura 32. Interfaces visuales de las ventanas de registro y la estación de grabación.

5. Conclusiones y líneas futuras

5.1. Conclusiones

Se ha logrado cumplir con los objetivos de este trabajo de fin de grado, creando la infraestructura necesaria para construir un banco de voces para poder personalizar sistemas de síntesis de voz para pacientes con enfermedades degenerativas que en un futuro vayan a perder la capacidad del habla.

El entrenamiento de los tres modelos de voz planteados para el proyecto se ha llevado a cabo con éxito, logrando generar un modelo de voz Adaptado que consigue niveles altos de naturalidad e inteligibilidad en la voz sintetizada que produce. Además, gracias al script *genvoice.sh*, se ha automatizado el proceso completo de obtención de estos modelos, permitiendo así eliminar la necesidad de intervención humana a lo largo de su transcurso.

La aplicación móvil ha cumplido con todos los requisitos planteados y ha mantenido la estructura definida en el inicio del proyecto. Esta App permite a sus usuarios realizar un registro y un inicio de sesión a través de conexiones con un servidor web, navegar de forma fluida entre las diferentes ventanas que contiene y realizar grabaciones para reproducirlas o subirlas.

Por último, como conclusión final, este trabajo de fin de grado ha logrado reflejar el potencial de los modelos de voz artificiales entrenados con Modelos Ocultos de Markov. Gracias a éstos, se han podido crear modelos Adaptados de calidad con 75 grabaciones de un usuario y una buena base de datos, siendo esta última no necesaria para la síntesis una vez el modelo se ha entrenado con éxito, lo que permite un ahorro de memoria sustancial. Estos sistemas emparejados con la infraestructura necesaria para recoger grabaciones como lo aplicación móvil desarrollada permiten agrandar significativamente el alcance de posibles usuarios que puedan beneficiarse de un modelo artificial de su voz.

5.2. Líneas futuras

Partiendo desde lo realizado en este trabajo, se pueden llevar a cabo una serie de posibles proyectos futuros, en este punto se presentan las siguientes propuestas.

- Ampliar las funcionalidades de la App: La aplicación móvil desarrollada en este trabajo de fin de grado sirve principalmente como herramienta de obtención de las grabaciones de los usuarios. Una posibilidad de proyecto futuro sería ampliar la funcionalidad de la aplicación móvil, permitiendo a los usuarios que ya hayan obtenido su modelo realizar la síntesis de voz a través de ésta.
- Desarrollo de un sistema completo personal: Como se ha expuesto en el inicio del tercer capítulo, todo el desarrollo de síntesis de voz de este proyecto se ha realizado desde el clúster GTCvoz voz01.unizar.es. Diseñar un programa ejecutable para los sistemas operativos de mayor popularidad en la actualidad (como Windows o IOS) que llevase a cabo desde el proceso de entrenamiento hasta la síntesis de voz de forma clara, sencilla e intuitiva es una buena propuesta de proyecto futuro de ámbito comercial.

6. Bibliografía

[1] Las figuras 1 a 6 son modificaciones propias al castellano de figuras obtenidas de [2].

[2] K. Tokuda, Y. Nankaku, T. Toda, H. Zen, J. Yamagishi, and K. Oura, "Speech synthesis based on hidden markov models" Proceedings of the IEEE, vol. 101, pp. 1234–1252, May 2013.

[3] A. Loarte, M. Romano, "Diseño y desarrollo de aplicaciones móviles aplicando teorías y conceptos de gamificación", Universidad Carlos III de Madrid, Septiembre 2016.

[4] P. Senin, "Dynamic time warping algorithm review," Information and Computer Science Department, University of Hawaii at Manoa, Honolulu, USA, 2008.

[5] "HMM/DNN-based Speech Synthesis System (HTS)" (acceso en 2021). https://developer.android.com/?hl=es

[6] "The Festival Speech Synthesis System", University of Edinburgh (acceso en 2021). https://www.cstr.ed.ac.uk/projects/festival/

[7] R. Barra-Chicote, J. Yamagishi, J. M. Montero, S. King, S. Lufti, J. Macias-Guarasa. "Generación de una voz sintética en Castellano basada en HSMM para la Evaluación Albayzín 2008: conversión texto a voz". Artículo en actas de V Jornadas en Tecnología del Habla. Bilbao (España), Noviembre 2008.

[8] "Android Developers" (acceso en 2021) https://developer.android.com/?hl=es

[9] "PyCharm, IDE de Python para desarrolladores profesionales" (acceso en 2021). https://www.cstr.ed.ac.uk/projects/festival/

[10] "The Festival Speech Synthesis System", Edition 1.4 for Festival Version 1.4.0, June 1999 (acceso en 2021). https://www.cstr.ed.ac.uk/projects/festival/manual/festival_toc.html "The Hidden Markov Model Toolkit (HTK)" (acceso en 2021). https://htk.eng.cam.ac.uk/

[11] "Visual Studio Code" (acceso en 2021). https://code.visualstudio.com/

[12] "Android Studio – El IDE oficial para Android" (acceso en 2021). https://developer.android.com/studio#downloads

[13] "Active Python Releases" (acceso en 2021). https://www.python.org/downloads/

[14] "Demo de HTS Monolocutor" (acceso en 2021) http://hts.sp.nitech.ac.jp/archives/2.2/HTS-demo_CMU-ARCTICSLT.tar.bz2

[15] "Audacity, editor de audio libre" (acceso en 2021). https://audacity.es/

[16] "Speech Signal Processing Toolkit (SPTK)", Version 3.1, December 2017 (acceso en 2021). http://sp-tk.sourceforge.net/

[17] "Demo de HTS Multilocutor con adaptación final" (acceso en 2021). http://hts.sp.nitech.ac.jp/archives/2.2/HTS-demo_CMU-ARCTICADAPT.tar.bz2

[18] "Descripción general de Volley" (acceso en 2021). https://developer.android.com/training/volley?hl=es

[19] "Cómo guardar datos de pares claves-valor" (acceso en 2021). https://developer.android.com/training/data-storage/shared-preferences?hl=es

[20] "Descripción general de MediaRecorder" (acceso en 2021). https://developer.android.com/guide/topics/media/mediarecorder?hl=es

[21] "Librosa Gallery" (acceso en 2021). https://librosa.org/librosa_gallery/index.html

Apéndice A
Modelos Ocultos de Markov

En este apéndice se va a exponer un ejemplo de una cadena oculta de Markov (HMM) con el objetivo de entender mejor su funcionamiento.

En primer lugar, se define una secuencia con los siguientes parámetros:

- Un conjunto de estados $I = \{ i_1, i_2, \dots , i_n\}$
- Un vector de probabilidades de transición entre estos estados $\overrightarrow{a_i} = \{a_{ij}\}$
- Unos resultados observables producidos por los estados $X = \{ x_1, x_2, \dots , x_n\}$
- Una secuencia de observaciones que pueden tomar cualquiera de los valores que están en X, $O = \{ o_1, o_2, \dots , o_t\}$

Para que una secuencia de estas características pueda denominarse como una cadena de Markov de primer orden, debe cumplirse la propiedad de Markov:

$$P(I_{t+1} = i' \,|I_1 = i_1, \dots , I_t = i) = P(I_{t+1} = i' \mid I_t = i)$$

Esta propiedad quiere decir que, dado un estado actual, los estados pasados y futuros a éste son independientes. En la figura A1 puede verse un ejemplo de una cadena de Markov de 3 estados.

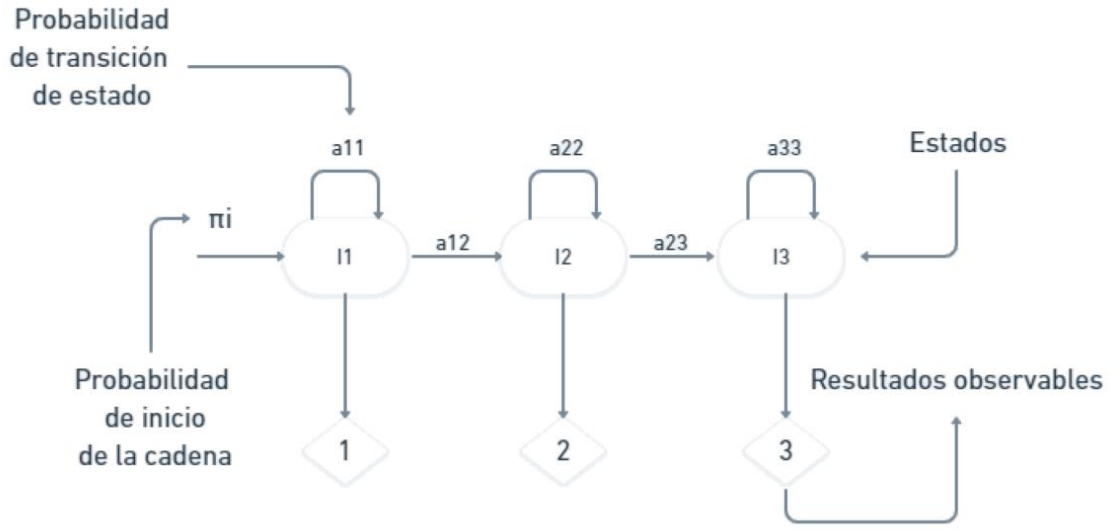


Figura A1. Ejemplo de una cadena de Markov.

Siguiendo esta estructura, si por ejemplo, se desease conocer la probabilidad de observar la secuencia O = {1,2,2,3} se calcularía de la siguiente forma:

$$P(O) = \pi_i a_{12} a_{22} a_{23}$$

Para que una cadena de Markov pueda definirse como una HMM (cadena oculta de Markov), sus estados no pueden ser directamente observables, sino que producen unos resultados observables dependientes de una cierta probabilidad. Esto significa que la secuencia que se observe no corresponderá directamente con una secuencia de estados, sino con sus probabilidades.

Como se ha expuesto en el punto 2.1.3, podemos definir una cadena oculta de Markov, λ, mediante un conjunto de tres parámetros:

- π_i define la probabilidad de inicio de la cadena en el estado i.
- $\overrightarrow{a_i} = \{a_{ij}\}$ define el vector de probabilidades de transición del estado i. Es la probabilidad de que siendo i el estado de la cadena en el instante t, en el t+1 el estado sea j.
- $b_i(o_t)$ define la probabilidad de salida de un observable o_t perteneciente a un estado i.

Esta última probabilidad de salida, $b_i(o_t)$, es:

$$b_i(o_t) = \mathrm{N}(o_t ; \mu_i, \Sigma_i)$$

Donde μ_i es la media y Σ_i la covarianza. En la figura A2 se puede ver una representación de una cadena oculta de Markov de 5 estados.

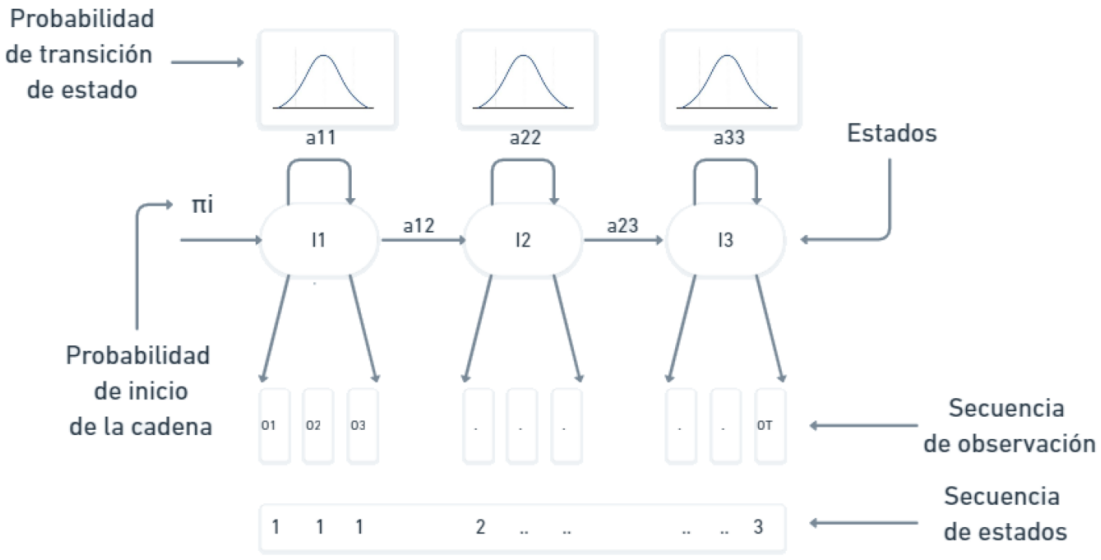


Figura A2. Ejemplo de una cadena oculta de Markov.

Para este ejemplo, la probabilidad de que una secuencia O se haya generado por una cadena de I estados es:

$$P(O|I)P(I) = \pi_1 b_1(o_1) \prod_{i=2}^{5} a_{i-1,i}\ b_i(o_i)$$

Apéndice B
Dynamic Time Warping

En este apéndice se va a exponer un ejemplo del funcionamiento del algoritmo DTW.

En primer lugar vamos a definir dos secuencias de enteros y de diferente longitud a comparar:

$$S_1 = \{1,2,3,2,1,0,-1,-2,-3,-2\}$$

$$S_2 = \{1,2,2,3,2,2,1,0,-3\}$$

En la figura B1 podemos ver una representación de estas secuencias.

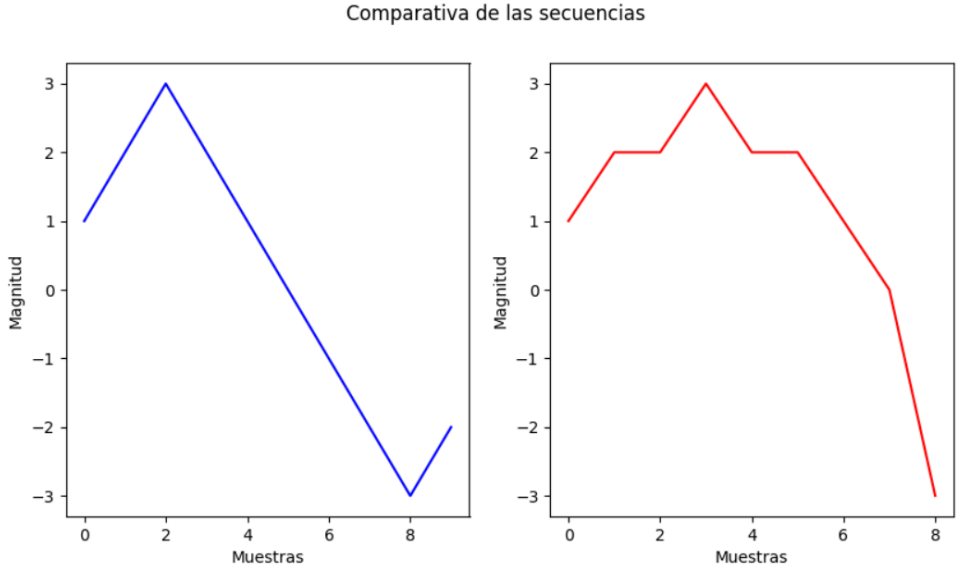
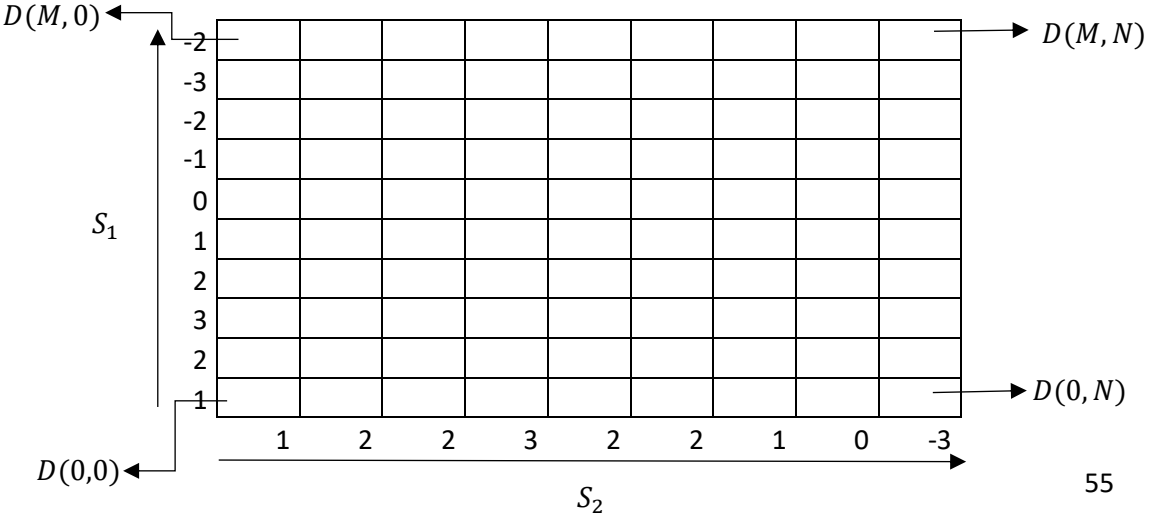


Figura B1. Representación de las dos secuencias a comparar.

Como se puede ver, las secuencias guardan bastante similitud. Para poder determinar cuanto se asemejan, aplicaremos el algoritmo DTW. En primer lugar, se formará una matriz D(i,j) de dimensiones MxN donde M será la longitud de la primera secuencia, y N la de la segunda:



En cada eje de la Matriz, la cual una vez se complete será nuestra matriz de coste, se escribirán las secuencias a comparar, como referencia a la hora de aplicar el algoritmo. Para calcular los valores de todos los componentes de la matriz, el algoritmo utiliza una serie de operaciones dependiendo de la posición de éste:

- Para el componente D(0,0) de la matriz, es decir, la esquina inferior izquierda de la matriz, se utiliza la siguiente ecuación:

$$D(0,0) = \left|S_{1_0} - S_{2_0}\right|$$

El resultado de aplicar este paso en nuestro ejemplo es el siguiente:

$$D(0,0) = \left|S_{1_0} - S_{2_0}\right| = |\,1 - 1| = 0$$

-2									
-3									
-2									
-1									
0									
1									
2									
3									
2									
1	0								
	1	2	2	3	2	2	1	0	-3

- Para calcular el valor del resto de los componentes D(0,j) de la matriz, es decir, la última fila de la matriz, se utiliza la siguiente ecuación:

$$D(0,j) = \left|S_{1_0} - S_{2_j}\right| + D(0,j-1)$$

Aplicando este paso hasta rellenar la fila en su totalidad obtenemos la siguiente matriz:

-2									
-3									
-2									
-1									
0									
1									
2									
3									
2									
1	0	1	2	4	5	6	6	7	11
	1	2	2	3	2	2	1	0	-3

- De forma análoga al caso anterior, para calcular el resto de los componentes de la columna D(i,0) se utiliza la siguiente ecuación:

$$D(i,0) = \left|S_{1_i} - S_{2_0}\right| + D(i-1,0)$$

Aplicando este paso hasta rellenar la columna en su totalidad obtenemos la siguiente matriz:

-2	17								
-3	14								
-2	10								
-1	7								
0	5								
1	4								
2	4								
3	3								
2	1								
1	0	1	2	4	5	6	6	7	9
	1	2	2	3	2	2	1	0	-3

- Por último, para el resto de las componentes de la matriz se utiliza la siguiente ecuación:

$$D(i,j) = \left|S_{1_i} - S_{2_j}\right| + \min\left(D(i-1,j-1), D(i-1,j), D(i,j-1)\right)$$

De esta forma, aplicando para el resto de las componentes completamos la matriz:

-2	17	18	19	21	19	19	13	8	0
-3	14	15	16	17	15	15	10	6	0
-2	10	11	11	12	10	10	6	3	0
-1	7	7	7	8	6	6	3	0	2
0	5	4	4	5	3	3	1	0	3
1	4	2	2	3	1	1	0	1	5
2	4	1	1	1	0	0	1	3	8
3	3	1	1	0	1	2	3	5	10
2	1	0	0	1	1	1	2	4	9
1	0	1	2	4	5	6	6	7	9
	1	2	2	3	2	2	1	0	-3

Al haber completado la matriz de coste D, el siguiente y último paso será calcular la distancia. Para esto se deberá recorrer desde el extremo superior derecho de la matriz hasta el extremo inferior izquierdo, pasando siempre por la componente adyacente de mínimo valor. El valor de la distancia será la suma acumulada de los valores de los componentes por los que se haya realizado el trayecto. Para nuestro ejemplo el resultado sería el siguiente:

-2	17	18	19	21	19	19	13	8	0
-3	14	15	16	17	15	15	10	6	0
-2	10	11	11	12	10	10	6	3	0
-1	7	7	7	8	6	6	3	0	2
0	5	4	4	5	3	3	1	0	3
1	4	2	2	3	1	1	0	1	5
2	4	1	1	1	0	0	1	3	8
3	3	1	1	0	1	2	3	5	10
2	1	0	0	1	1	1	2	4	9
1	0	1	2	4	5	6	6	7	9
	1	2	2	3	2	2	1	0	-3

En este ejemplo, al haber recorrido únicamente componentes con el valor 0, la distancia será 0 también. En la figura B2 se enseña el resultado final devuelto por el script *dtw.py* y en la figura B3 se muestra el emparejamiento final de las muestras de las secuencias de este ejemplo.

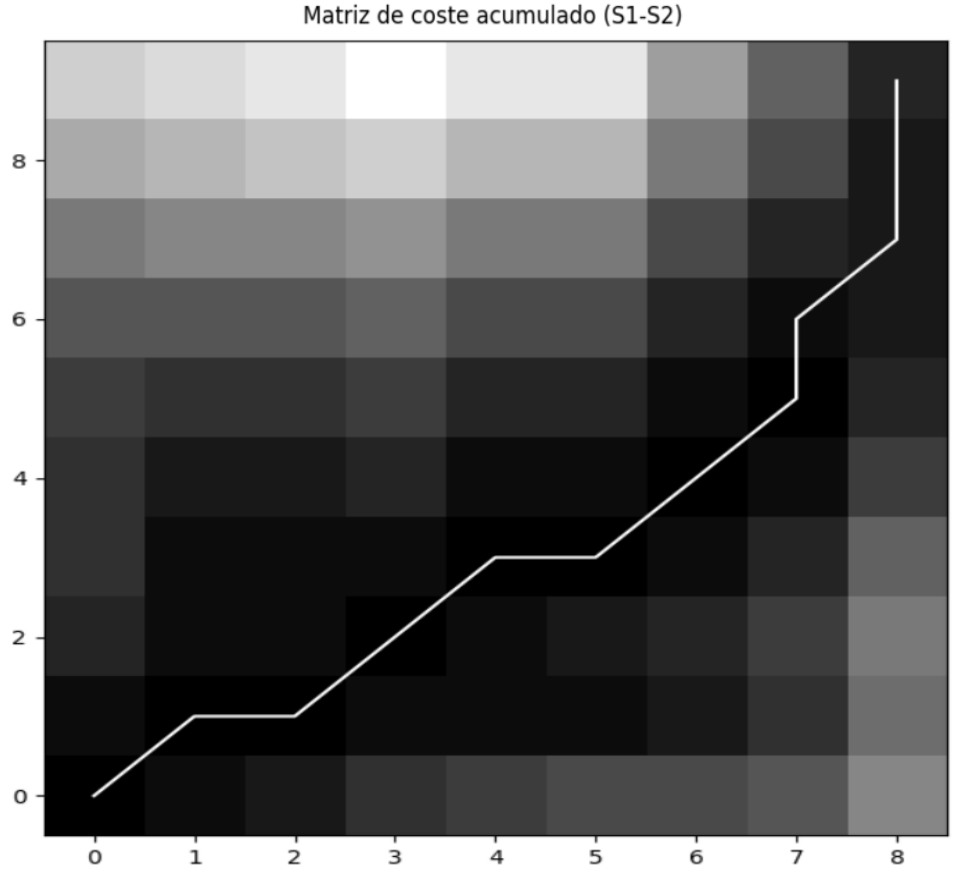


Figura B2. Matriz de coste construida por el script *dtw.py*.

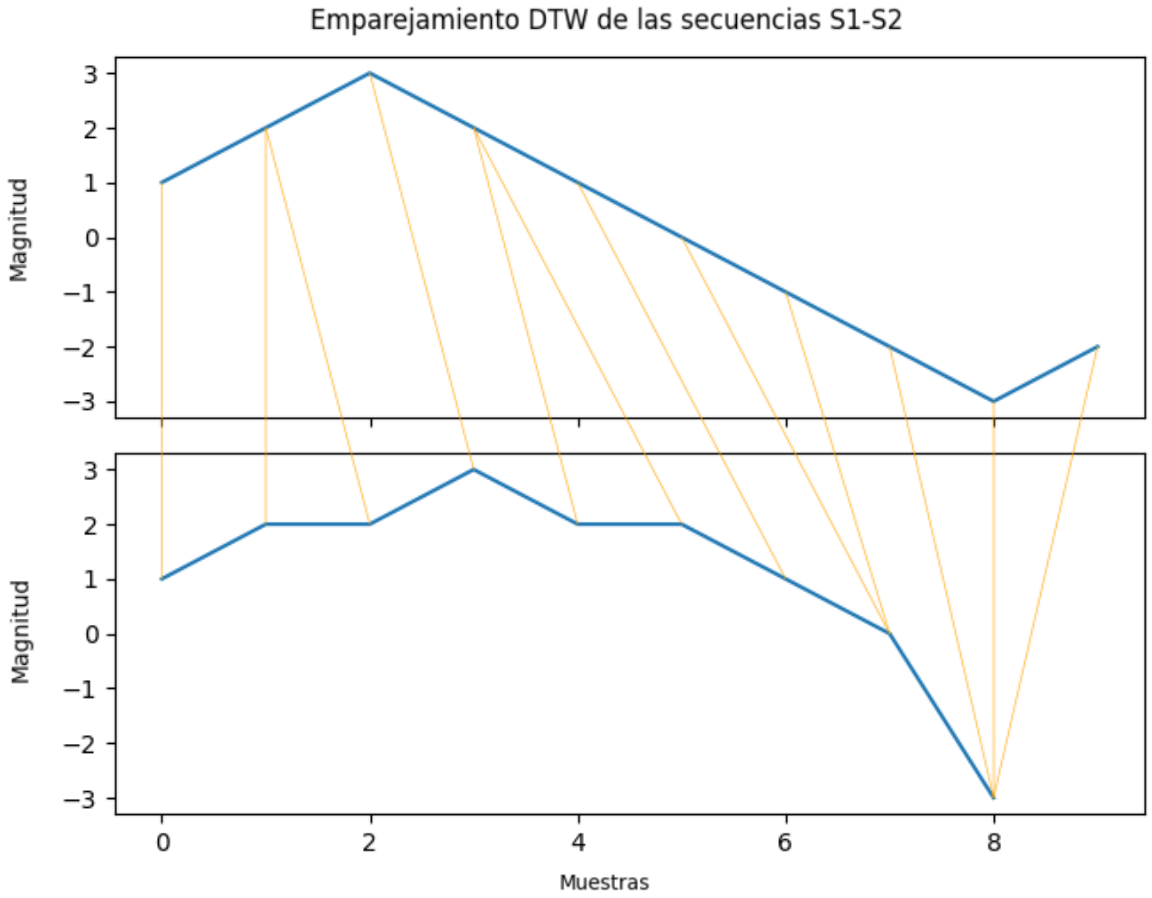


Figura B3. Emparejamiento resultante de aplicar el DTW a las secuencias S1 y S2.

Apéndice C

Particiones de frases de Albayzín

PARTICIÓN A 1

1 - Francia, Suiza y Hungr'ia ya hicieron causa com'un.
9 - Despu'es ya se hizo muy amiga nuestra.
17 - Los yernos de Ismael no engordar'an los pollos con hierba.
25 - Despu'es de la mili ya me vine a Catalu~na.
33 - Bajamos un d'ia al mercadillo de Palma.
41 - Existe un viento del norte que es un viento fr'io.
49 - Me he tomado un caf'e con leche en un bar.
57 - Yo he visto a gente expulsarla del colegio por fumar.
65 - Guadalajara no est'a colgada de las rocas. Cuenca s'i.
73 - Pas'e un a~no dando clase aqu'i, en Bellaterra.
81 - Pero t'u ahora elijes ya previamente.
89 - Les dijeron que eligieran una casa all'a, en las mismas condiciones.
97 - Cuando me gir'e ya no ten'ia la cartera.
105 - Tendr'a unas siete u ocho islas alrededor.
113 - Haciendo el primer campamento y el segundo campamento.
121 - Unas indemnizaciones no les iban del todo mal.
129 - Rezando porque ten'ia un miedo impresionante.
137 - Es un apellido muy abundante en la zona de Pamplona.
145 - No jug'abamos a b'asket, s'olo los mir'abamos a ellos.
153 - Aunque naturalmente hay un partido comunista.
161 - Dio la casualidad que a la una y media estaban all'i.
169 - Se alegraron mucho de vernos y ya nos quedamos a cenar.
177 - Ya empezamos a llorar bastante en el apartamento.
185 - En una ladera del monte se ubica la iglesia.
193 - Entonces lo 'unico que hac'iamos era ir a cenar.

PARTICIÓN A 2

2 - Mi primer profesor de lengua fue L'opez Garc'ia.
10 - Los achaques de Jes'us remitieron sin causar disgustos.
18 - Mi mujer es profesora de Lengua y Literatura.
26 - Estuve en Guernica dando clase de lengua y literatura.
34 - Por las noches organizaban bailes y fiestas.
42 - Nos dio el disgusto m'as grande de nuestra vida.
50 - Dentro de muy poco pues va a estar la mitad cubierto.
58 - A las ocho de la ma~nana ya estaba haciendo guardia.
66 - En diez minutos te puedes colocar en Pe~nacabarga.
74 - Ya empezar'an los malos modos o los morritos.
82 - Empezamos a reivindicar que se hiciese pueblo nuevo.
90 - Podr'iamos encasillarlo como hombre de derechas.
98 - Los servicios que pueden prestar son muy superiores.
106 - Son los cuadros m'as bonitos de las casas de mi pueblo.
114 - A aquella gente que le va mucho la vida militar.
122 - Las de dos pisos tienen un primer piso con una habitaci'on.
130 - Es uno de los momentos m'as bonitos del d'ia.
138 - Hab'ia estado en una tertulia en Logro~no.
146 - Es un s'otano que tendr'a diez metros de altitud.
154 - Casi haciendo frontera con Bilbao est'a el r'io As'on.
162 - Llegaba a Santander a las ocho de la ma~nana.
170 - Bueno, es otra vida completamente distinta.
178 - Le he dado un beso y me he vuelto a mi casa.
186 - Esto es lo que has hecho durante el a~no y durante el verano.
194 - No dormimos pr'acticamente en toda la noche.

PARTICIÓN A 3

3 - Guillermo y Yolanda practicaban ciclismo con Jaime.
11 - Su viuda se cas'o con un profesor de ingl'es de all'a.
19 - Sab'ia un poquito de ingl'es pero muy poco.
27 - Firmaban como c'antabros incluso en tumbas funerarias.
35 - Yo entr'e en filas a cumplir el servicio militar.
43 - Se hacen chorizos y salchichones de dos clases.
51 - Los autom'oviles circulan por el pueblo.
59 - Nos lo pasamos muy bien aunque no vimos nada.
67 - Iba vestida muy rara pero era superdivertida.
75 - El a~no pasado no pude porque cambi'e de trabajo.
83 - Nosotros esto lo tuvimos muy claro desde el principio.
91 - Los hombres normalmente chocarran el cochino.
99 - Esos a~nos estuvimos haciendo ocho asignaturas.
107 - Un partido socialista con representatividad.
115 - Por otra parte, yo estaba fuera de mi casa.
123 - Nos 'ibamos preparando con aquellos equipos tan gloriosos.
131 - La mitad va a estar cubierto por un pantano.
139 - Y mi padre es de aqu'i y no es tan bueno.
147 - Entonces se cre'o una cu~na natural que fue Santander.
155 - La mayor'ia de los guerreros ten'ian reuma.
163 - Se la conoce o la conocen como la peque~na Suiza.
171 - En el caso m'io, yo era delineante industrial.
179 - Los ni~nos son catalanes, han nacido en Blanes los tres.
187 - Bueno, pero vamos a la historia que te contaba.
195 - En todos los sitios hab'ia italianos.

PARTICIÓN A 4

4 - El primero en Guip'uzcoa y el segundo en Valladolid.
12 - Durante tus estudios has hecho alg'un viaje.
20 - En Julio hac'iamos en Burgos otra vez la preparaci'on.
28 - Aunque ellos ya engrasaron los ejes como yo les ense~n'e.
36 - Uno llevaba el chorizo, el otro llevaba el gazpacho.
44 - Antes de primero de bachiller ya te traumatizaban.
52 - Hay dos ni~nos o tres que le hab'ian dicho que les gustaba.
60 - Nos cogieron y nos llevaron otra vez al congreso.
68 - Como es l'ogico donde hay monta~nas hay valles.
76 - Yo iba hasta la ermita de San Gabriel con este cacharro.
84 - Lo han tirado abajo y han hecho un bloque nuevo.
92 - Yo me he dado cuenta cuando he vuelto aqu'i.
100 - Hab'ia un autob'us que nos llevaba otra vez para el cuartel.
108 - Produce dolor de cabeza a los monta~neses.
116 - Hoy no ten'ia que llevar bocadillo por la ma~nana.
124 - Hab'ia gente que quer'ia marcharse a vivir a Lumbreras.
132 - Los chorizos y los jamones est'an colgando de las paredes.
140 - Volvemos otra vez por encima de Santander.
148 - Quien mandaba era un alf'erez que era compa~nero.
156 - Al llegar a Barcelona tendremos que arreglarlo todo.
164 - El rey mand'o a su yerno a derrotar a los c'antabros.
172 - Esto es por todo el reglamento del mercado com'un europeo.
180 - No se carga a nadie, es bastante cari~nosa.
188 - La higiene que hab'ia all'i era bastante deprimente.
196 - Tiene espect'aculos bastante agradables en verano.

Apéndice D

Funciones y Scripts Síntesis

Script *genvoice.sh*:

```
#!/bin/sh

# The genvoice script automatically gener-
ates a voice model given the voice codename and its
# orientation (Male or Female)

echo 'Executing the automatic adaptation script'
echo ''

# Gathering resources

datafolder=$1
mkdir ../files/aut$1
mkdir ../files/aut$1/wav
mkdir ../files/aut$1/txt

cp -p ../files/$1/wav/*.wav ../files/aut$1/wav
cp -p ../files/$1/txt/*.txt ../files/aut$1/txt

chmod 777 ../files/aut$1/wav/*.wav
chmod 777 ../files/aut$1/txt/*.txt

# Transforming .wav to .raw

echo '----- Transforming .wav files to .raw'
echo ''

l=$(ls ../files/aut$1/wav/ | wc -l)

for i in $(seq 1 $l); do
    if [ $i -lt 10 ]
        then
            ../software/bin/wav2raw ../files/aut$1/wav/Al-
bayin_$1_fp000$i.wav
            echo 'Transforming ' Albayzin_$1_fp000$i.wav 'to .raw'
        else
            if [ $i -lt 100 ]
```

```
                then
                    ../software/bin/wav2raw ../files/aut$1/wav/Al-
bayzin_$1_fp00$i.wav
                    echo 'Transforming ' Albayzin_$1_fp00$i.wav 'to .raw'
                else
                    if [ $i -lt 1000 ]
                        then
                            ../soft-
ware/bin/wav2raw ../files/aut$1/wav/Albayzin_$1_fp0$i.wav
                            echo 'Transforming ' Al-
bayzin_$1_fp0$i.wav 'to .raw'
                    fi
            fi
        fi
done

# Saving .raw in new directory

mkdir ../files/aut$1/raw

echo ''
echo '----- Transforming completed, saving .raws in ' files/aut$1/raw
echo ''

cp -p ../files/aut$1/wav/*.raw ../files/aut$1/raw
rm -f ../files/aut$1/wav/*.raw

# Transforming .txt to .utt

echo '----- Transforming .txt files to .utt'
echo ''

l=$(ls ../files/aut$1/txt/ | wc -l)

for i in $(seq 1 $l); do
    if [ $i -lt 10 ]
        then
            ../software/bin/text2utt.sh ../files/aut$1/txt/Al-
bayzin_$1_fp000$i.txt
        else
            if [ $i -lt 100 ]
                then
                    ../software/bin/text2utt.sh ../files/aut$1/txt/Al-
bayzin_$1_fp00$i.txt
                else
                    if [ $i -lt 1000 ]
```

```
                        then
                            ../soft-
ware/bin/text2utt.sh ../files/aut$1/txt/Albayzin_$1_fp0$i.txt
                    fi
                fi
            fi
done

# Saving .utt in new directory

mkdir ../files/aut$1/utts

echo ''
echo '----- Transforming completed, saving .utts in ' files/aut$1/utts
echo ''

cp -p *.utt ../files/aut$1/utts
rm -f *.utt

# Start voice adaptation

echo '----- Starting '$2' voice adaptation'
echo ''

./do_adapt_aut.sh $1 $2

mkdir ../files/aut$1/voice

echo '----- Voice directory created'
echo ''
echo '----- '$2' voice adaptation is successfully in progress...'
echo ''

# Creating voice directory for the .htsvoice file
```

Script *do_adapt_aut.sh*:

```
#!/bin/sh

export LANG=es_ES.UTF-8
export HOMEWORK=/extra/scratch03/TTS
### Speaker code
SPKR=$1

### TTS system: 1-SPTK,  2-STRAIGHT,  3-AHOCODER
VER=1

### Files from average model

var1="$2"
var2="Male"
var3="Female"

if [ "$var1" = "$var2" ]
    then
        export AVGDIR=$HOMEWORK/hts_demo/demo/Albayzin-AVG-$2
        echo 'Exporting male database'
    else
        if [ "$var1" = "$var3" ]
            then
                export AVGDIR=$HOMEWORK/hts_demo/demo/Albayzin-AVG-$2
                echo 'Exporting female database'
            else
                export AVGDIR=$HOMEWORK/hts_demo/demo/Albayzin-AVG
                echo 'Exporting mixed database'
        fi
fi

### Obtaining the information files from the average directory of choice

rm -rf data/labels
rm -rf data/lists
rm -rf data/scp

mkdir -p data/labels; cp -p $AVGDIR/data/labels/*.mlf data/labels/
mkdir -p data/lists; ln -s $AVGDIR/data/lists/* data/lists/
mkdir -p data/scp; ln -s $AVGDIR/data/scp/train.scp data/scp/

MODELS=models/qst001/ver$VER
rm -rf $MODELS/cmp; rm -rf $MODELS/dur;
```

```
mkdir -p $MODELS/cmp; ln -s $AVGDIR/$MODELS/cmp/re_clustered.mmf $MOD-
ELS/cmp/
mkdir -p $MODELS/dur; ln -s $AVGDIR/$MODELS/dur/re_clustered.mmf $MOD-
ELS/dur/

GVDIR=gv/qst001/ver$VER
rm -f $GVDIR/*
mkdir -p $GVDIR;
ln -s $AVGDIR/$GVDIR/gv.list $GVDIR/
ln -s $AVGDIR/$GVDIR/*.inf $GVDIR/
ln -s $AVGDIR/$GVDIR/clustered.mmf $GVDIR/;

TREES=trees/qst001/ver$VER
rm trees/qst001/ver$VER/cmp/lf0.inf.un-
tied; rm trees/qst001/ver$VER/cmp/mgc.inf.untied
rm trees/qst001/ver$VER/dur/dur.inf.untied
mkdir -p $TREES/cmp; ln -s $AVGDIR/$TREES/cmp/*.untied $TREES/cmp/
mkdir -p $TREES/dur; ln -s $AVGDIR/$TREES/dur/*.untied $TREES/dur/

STATS=stats/qst001/ver$VER
rm -f $STATS/*
mkdir -p $STATS;
ln -s $AVGDIR/$STATS/dur.stats.untied $STATS/
ln -s $AVGDIR/$STATS/cmp.stats.re-clustered $STATS/

### Adapt-speaker files
rm -f data/raw/$SPKR/*
mkdir -p data/raw/$SPKR/
cp $HOMEWORK/hts_demo/files/aut$SPKR/raw/* data/raw/$SPKR/
rm -f data/utts/$SPKR/*
mkdir -p data/utts/$SPKR/
cp $HOMEWORK/hts_demo/files/aut$SPKR/utts/* data/utts/$SPKR/

rm -f data/labels/gen/*.lab;

### Go!
./configure --with-fest-search-path=$HOMEWORK/hts_demo/software/festi-
val/examples/  \
        --with-sptk-search-path=$HOMEWORK/hts_demo/soft-
ware/bin/            \
        --with-hts-search-path=$HOMEWORK/hts_demo/software/bin/         \
        --with-hts-engine-search-path=$HOMEWORK/hts_demo/soft-
ware/bin/      \
        --with-matlab-search-path=/usr/local/matlab2014a/bin/           \
        --with-straight-path=/$HOMEWORK/hts_demo/soft-
ware/STRAIGHTV40pcode      \
```

```
        --with-ahocoder-search-path=$HOMEWORK/hts_demo/software/aho-
coder        \
        FULLCONTEXT_FORMAT=HTS_TTS_ESP                          \
        DATASET=Albayzin                                \
        ADAPTSPKR=$SPKR                                     \
        ADAPTHEAD=fp0                                   \
        F0_RANGES=$SPKR' 70 210'                            \
        SAMPFREQ=16000                                  \
        FRAMELEN=400                                    \
        FRAMESHIFT=80                                   \
        FFTLEN=512                                  \
        FREQWARP=0.42                                   \
        MGCORDER=24                                 \
        MGCBANDWIDTH=25                                 \
        SAMPFREQ0=16000                                 \
        NUMADAPT=3                                  \
        ADDMAP=1                                    \
        USESTRAIGHT=`expr $VER == 2`                            \
        USEAHOCODER=`expr $VER == 3`                            \
        VER=$VER

make > make.v$VER.log 2>&1
mv log v$VER.log
```

Script *tts.sh*:

```
#!/bin/sh

# Inputs are the following:
# 1: .htsvoice file from the desired voice model
# 2: .txt file with the phrase to reproduce

rm gen/*.lab
./txt2lab.sh  $2

# Files *.lab inside gen/

rm wav/*.raw
rm wav/*.trace
rm wav/*.wav
find gen/ -name "*.lab" | \
    ./lab2wav.sh -v $1 -w # -w lets us use hts_engine_world

# .wav files will end up stored in the wav/ folder
```

Apéndice E

Ejemplos de Funciones y Scripts de la App de importancia

Ejemplo de una función de navegación de la Aplicación móvil, *navMainMenu()*:

```
// Clase que realizará la navegación hacia el menú principal del usuario
private void navMainMenu(){
    saveLog(true);
    // Guardamos en los "SharedPreferences" el inicio de la sesión
    Intent intent = new Intent(getApplicationContext(), UserActivity.class);
    startActivity(intent);
    // Generamos el objeto Intent, con el objetivo de ejecutar la clase Use-
rActivity.java
    finish();
}
```

Ejemplo de una función de comunicación con el servidor Web, *login()*:

```
private void login(){
    // Clase responsable de realizar la petición de inicio de sesion al ser-
vidor Web

    RequestQueue queue = Volley.newRequestQueue(LoginActivity.this);
    String url ="http://"+getIP()+"/api/login";
    // Se genera una RequestQueue por donde enviar las peticiones HTTP

    // Se solicita recibir toda la respuesta HTTP en un mismo String
    StringRequest stringRequest = new StringRequest(Request.Method.POST,url,
            new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {
                    // En el caso de que se reciba una respuesta con éxito
(No haya habido error)

                    try {
                        JSONObject respuesta = new JSONObject(response);
                        // Transformamos la respuesta a un JSONObject para
poder obtener la información
                        // con mayor facilidad

                        guardarToken(respuesta.get("access_token").toS-
tring());
                        guardarName(respuesta.getJSONOb-
ject("data").get("name").toString());
                        // Almacenaremos en los "SharePreferences" tanto el
token de seguridad que nos
                        // entregue el servidor web como el nombre del
```

```
usuario que acaba de iniciar
                        // la sesión

                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                    Toast.makeText(getApplicationContext(),"Bienvenido,
"+getName(),Toast.LENGTH_SHORT).show();
                    // Se muestra por pantalla un mensaje de bienvenida al
usuario

                    navMainMenu();
                    // Comienza la navegación al menú principal de la App
                }
            }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            Toast.makeText(getApplicationContext(),"Error en el re-
cibo"+error.toString(),Toast.LENGTH_SHORT).show();
            // En el caso de recibir un mensaje de error, se mostrará éste
por pantalla
        }
    }) {
        @Override
        protected Map<String, String> getParams() {
            Map<String, String> params = new HashMap<>();
            // the POST parameters:
            params.put("email",String.valueOf(correoIn.getText()));
            params.put("password",String.valueOf(passIn.getText()));
            return params;
            // Los parámetros de entrada de nuestra peticion HTTP serán el
email y la contraseña
            // que se haya ingresado
        }
    };
    // Se añade la petición a la RequestQueue.
    queue.add(stringRequest);

}
```

Ejemplos de unas funciones de manejo de datos “SharedPreferences”, *saveLog()* y *checkLog()*:

```
private void saveLog(boolean ini){
    editor.putBoolean("ini",true);
    editor.commit();
    // Generamos o modificamos el par clave-valor "ini" y lo difinimos como
true
}

private boolean checkLog(){
    return this.preferences.getBoolean("ini",false);
    // Devolvemos el valor almacenado emparejado con la clave "ini", si no
existe
    // ningun valor emparejado almacenado con esta clave, se devolverá por
defecto
```

```
    // false
}
```

Ejemplos de las funciones que realizar la grabación y la reproducción de los archivos de voz, *startRecording(), stopRecording(), startPlaying() y stopPlaying()*:

```
private void startRecording() throws IOException {

    recordPath = getExternalCacheDir().getAbsolutePath();
    SimpleDateFormat format = new SimpleDateFormat("yyyy_MM_dd_hh_mm_ss", Lo-
cale.GERMANY);
    Date now = new Date();
    recordFile = "Redcording_"+getName()+"_"+getTrat()+"_"+getPhrase()+"_" +
format.format(now) + ".3gp";
    // Se genera un nombre y una ruta de archivo con la información clave
para su almacenamiento

    mediaRecorder = new MediaRecorder();
    mediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
    mediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
    mediaRecorder.setOutputFile(recordPath + recordFile);
    mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
    // Se configura el mediaRecorder para que identifique el microfono a
usar, escoga el tipo de
    // codificación de audio, su formato y la ruta final de almacenamiento

    mediaRecorder.prepare();
    mediaRecorder.start();
    // Comienza la grabación
}
private void stopRecording(){

    mediaRecorder.stop();
    mediaRecorder.release();
    mediaRecorder = null;
    // La grabación finaliza, se liberan los datos almacenados y se vacia el
mediaRecorder
}

private void startPlaying() throws IOException {

    mediaPlayer = new MediaPlayer();

    mediaPlayer.setDataSource(recordPath + recordFile);
    mediaPlayer.prepare();
    mediaPlayer.start();
    // Partiendo de última ruta generada, se reproduce la última grabación
realizada
}

private void stopPlaying(){

    mediaPlayer.stop();
    // Finaliza la grabación que se esté reproduciendo
}
```

Ejemplo de una función de petición de permisos, *useMic()*:

```
private boolean useMic(){
    // Funcion que comprueba la posibilidad de utilizar el micrófono para
realizar las grabaciones
    // en caso de no tener permisos, los solicitará
    if(checkPermission(recordPermission,android.os.Process.myPid(), an-
droid.os.Process.myUid()) == PackageManager.PERMISSION_GRANTED) {
        // Comprobamos si se tienen los permisos para utilizar el micrófono
        return true;
    }
    else {
        requestPermissions(new String[]{recordPermission}, PERMISSION_CODE);
        // En caso de no tener los permisos, realizar una solicitud de estos
        return false;
    }
}
```

Ejemplo de la función de interactividad *OnClickListener()* del botón de inicio de sesión *log*:

```
Button log = findViewById(R.id.btn_log);
log.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        // En caso de que se presione el botón de inicio de sesión, se ejecu-
tarán las siguientes lineas
        String user = String.valueOf(correoIn.getText());
        String pass = String.valueOf(passIn.getText());
        // Almacenamos lo que haya escrito en los campos de Correo Electró-
nico y Constraseña

        if(!user.equals("")&&!pass.equals("")){
            // En el caso de que ninguno de los campos esté vacío, ejecutamos
la función que activará
            // la petición HTTP de inicio de sesión al servidor Web
            login();
        }
        else{
            Toast.makeText(getApplicationContext(),"Rellena todos los cam-
pos",Toast.LENGTH_SHORT).show();
            // En caso de que alguno de los campos esté vacío, se le notifi-
cará al usuario por pantalla
        }
    }
});
```

Ejemplo de un fichero .xml que define la estructura, el diseño y el estilo de la ventana del menú principal, *activity_homeuser.xml* :

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:layout_width="380dp"
        android:layout_height="80dp"
        android:layout_marginStart="10dp"
        android:layout_marginLeft="10dp"
        android:layout_marginTop="32dp"
        android:layout_marginEnd="10dp"
        android:layout_marginRight="10dp"
        android:orientation="horizontal"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <TextView
            android:id="@+id/textView2"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:gravity="center"
            android:text="Actividad"
            android:textSize="35sp" />

        <ImageView
            android:id="@+id/imageView"
            android:layout_width="70dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            app:srcCompat="@drawable/voice"
            tools:ignore="VectorDrawableCompat" />

        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:orientation="vertical">

            <ImageButton
                android:id="@+id/btn_out_ph"
                android:layout_width="match_parent"
                android:layout_height="53dp"
                app:srcCompat="@drawable/ic_logout"
                tools:ignore="VectorDrawableCompat" />

            <TextView
                android:id="@+id/textView3"
```

```
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:gravity="center"
            android:text="Cerrar sesión" />
    </LinearLayout>

</LinearLayout>

<LinearLayout
    android:id="@+id/linearLayout2"
    android:layout_width="380dp"
    android:layout_height="183dp"
    android:layout_marginStart="10dp"
    android:layout_marginLeft="10dp"
    android:layout_marginTop="165dp"
    android:layout_marginEnd="10dp"
    android:layout_marginRight="10dp"
    android:orientation="vertical"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">

    <TextView
        android:id="@+id/textView4"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Tus tratamientos:"
        android:textSize="30sp" />

    <TextView
        android:id="@+id/trat_ph"
        android:layout_width="match_parent"
        android:layout_height="41dp" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="45dp"
        android:orientation="horizontal">

        <ImageButton
            android:id="@+id/btn_left"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            app:srcCompat="@drawable/left"
            tools:ignore="VectorDrawableCompat" />

        <ImageButton
            android:id="@+id/btn_right"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            app:srcCompat="@drawable/right"
            tools:ignore="VectorDrawableCompat" />
    </LinearLayout>
```

```
    <Button
        android:id="@+id/btn_seltr"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Escoger este tratamiento"

        />

</LinearLayout>

<LinearLayout
    android:layout_width="380dp"
    android:layout_height="145dp"
    android:layout_marginStart="10dp"
    android:layout_marginLeft="10dp"
    android:layout_marginTop="400dp"
    android:layout_marginEnd="10dp"
    android:layout_marginRight="10dp"
    android:orientation="vertical"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <ImageButton
            android:id="@+id/btn_info"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            app:srcCompat="@drawable/info"
            tools:ignore="VectorDrawableCompat" />

        <ImageButton
            android:id="@+id/btn_frases"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            app:srcCompat="@drawable/playbutton"
            tools:ignore="VectorDrawableCompat" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="44dp"
        android:orientation="horizontal">

        <TextView
            android:id="@+id/textView6"
            android:layout_width="130dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="center"
```

```
                android:text="Información" />

            <TextView
                android:id="@+id/textView7"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:gravity="center"
                android:text="Acceso a las frases" />
        </LinearLayout>

        <Button
            android:id="@+id/btn_profile"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Tu perfil"

            />
    </LinearLayout>

</RelativeLayout>
```

Apéndice F

Script DTW

```
# Script de la aplicación del algoritmo DTW sobre dos ficheros de au-
dio
# a comparar, devolviendo como resultado su matriz de coste y la dis-
tancia
# resultante normalizada

import matplotlib.pyplot as plt
import librosa.display

from dtw import dtw
from numpy.linalg import norm

print(" ------------- COMIENZO DEL SCRIPT:")
print("")

# Audios a comparar

y1, sr1 = librosa.load('bn_original_001.wav')  # Frase Original 1
y2, sr2 = librosa.load('bn_original_002.wav')  # Frase Original 2

print(" ------------- AUDIOS CARGADOS CON ÉXITO")
print("")

# Representación de las formas de onda

plt.figure()

max_int16 = 2**15
y1n = (y1 / max_int16) * 1e5
y2n = (y2 / max_int16) * 1e5

plt.subplot(1, 2, 1)
plt.xlabel("Muestras")
plt.ylabel("Magnitud")
plt.plot(y1n, 'b')
plt.subplot(1, 2, 2)
plt.xlabel("Muestras")
plt.ylabel("Magnitud")
plt.plot(y2n, 'b')

plt.suptitle("Comparativa de las formas de onda (bn_original_001-
bn_original_002)")

# Representación de los coeficientes mfcc

plt.figure()

plt.subplot(1, 2, 1)
mfcc1 = librosa.feature.mfcc(y1, sr1)
librosa.display.specshow(mfcc1)
plt.ylabel('Coeficientes mfcc')
```

```
plt.xlabel('Tiempo')

plt.subplot(1, 2, 2)
mfcc2 = librosa.feature.mfcc(y2, sr2)
librosa.display.specshow(mfcc2)
plt.ylabel('Coeficientes mfcc')
plt.xlabel('Tiempo')

plt.suptitle("Comparativa de los coeficientes mel-sceptrum (bn_origi-
nal_001-bn_original_002)")

plt.figure()

# Comparativa usando DTW matching

print(" ------------- EJECUCIÓN DEL ALGORITMO DTW")
print("")

dist, cost, acc_cost, path = dtw(mfcc1.T, mfcc2.T, dist=lambda x, y:
norm(x - y, ord=1))
print('Distancia normalizada entre los dos audios (bn_original_001-
bn_original_002.wav):', dist/(cost.shape[0]*cost.shape[1]))
print('')

# Representación gráfica de la matriz de coste acumulado

plt.imshow(cost.T, origin='lower', cmap=plt.cm.gray, interpola-
tion='nearest')
plt.plot(path[0], path[1], 'w')
plt.xlim((-0.5, cost.shape[0]-0.5))
plt.ylim((-0.5, cost.shape[1]-0.5))
plt.title("Matriz de coste acumulado (bn_original_001-bn_origi-
nal_002.wav)")

plt.show()

print(" ------------- FINALIZACIÓN DEL SCRIPT:")
```