UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

AVC Asistente Virtual para la Comunicación

Presentado por José Miguel Ramírez Sanz
en Universidad de Burgos — 19 de junio de 2019
Tutor: Dr. José Francisco Díez Pastor y Dr. César Represa Pérez

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

Dr. D. José Francisco Díez Pastor, profesor del departamento de Ingeniería Civil, área de Lenguajes y Sistemas Informáticos y Dr. D. César Represa Pérez, profesor del departamento de Ingeniería Electromecánica, área de Tecnología Electrónica.

Exponen:

Que el alumno D. José Miguel Ramírez Sanz, con DNI 71303106R, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado *AVC Asistente Virtual para la Comunicación.*

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 19 de junio de 2019

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
Dr. D. José Francisco Díez Pastor	Dr. D. César Represa Pérez

Resumen

La parálisis cerebral es una discapacidad que afecta a la capacidad de movimiento y de la postura. Además, esta discapacidad suele ir acompañada de otras discapacidad en la percepción de los sentidos y en la capacidad cognitiva. Dentro de las personas afectas hay un grupo gravemente afectado con grandes discapacidades motoras, cognitivas y sobre todo comunicativas, ya que los únicos sonidos que emiten son ruidos.

Con la idea de facilitar la comunicación de estos pacientes gravemente afectados surgió la propuesta desde APACE Burgos para desarrollar un comunicador virtual con el que poder interpretar los sonidos que estas personas hacen. Es aquí cuando APACE Burgos se puso en contacto con la Universidad de Burgos para desarrollar AVC, el *Asistente Virtual para la Comunicación*.

Para la realización de este proyecto se han desarrollado varias aplicaciones. Una para la recogida de los sonidos procedentes de los pacientes con los que entrenar un clasificador, y otra aplicación para la interpretación de los sonidos emitidos por los pacientes en base a los datos registrados. También se ha desarrollado un servidor que permite el almacenamiento de las opciones adicionales necesarias para la clasificación, así como para poder interpretar los sonidos.

Las aplicaciones generadas durante este proyecto se han desarrollado en *Android* y se ha usado *Python* para desarrollar el servidor. El asistente pretende recnoocer emociones (hambre, tristeza, enfado y dolor) y respuestas (sí y no) a partir de métodos de clasificación. Para ello se han empleado algoritmos de extracción de características sobre audios, y para el clasificador final se ha usado un tipo de *ensemble* llamado *Bagging*, en concreto *Random Forest*.

Descriptores

Parálisis cerebral, comunicación, extracción de características, clasificación de audios, *Random Forest*, *Android*, servidor *Flask*.

Abstract

Cerebral palsy is a disability that affects the ability to move and posture. Besides, this disability is often accompanied by other diabilities in the senses and the cognitive ability. There is a group of people severely affected with large motor, cognitive and, above all, communicative disabilities, because they can only communicate by noises.

Thinking about facilating communication to severely affected patients, APACE Burgos suggested to develop a virtual communicator to interpret the sounds of these people. Here is when APACE Burgos contacted with University of Burgos to develop VCA, the *Virtual Assitant for Communication.*

For the realization of this project they have been developed some applications. One for collectiong data from patients, and other one to interpret the sounds emitted by patients. A server has also been developed, to allow storage of the necesaary additional options for the classification.

The generated applications during this project have been developed in *Android* and we have used *Python* for the development of the server. To design the algorithms of classification of emotions (hungry, sadness, anger and pain) and answers (yes and no) feature extraction algorithms on sounds has been used, and for the final classifier we have applied a type of ensemble called *Bagging*, specifically *Random Forest.*

Keywords

Cerebral palsy, communication, feauture extraction, sound classification, *Random Forest*, *Flask* server.

Índice general

Índice de figuras

Índice de tablas

Introducción

La **parálisis cerebral** es una discapacidad, normalmente originada en la fase de gestación del feto, que afecta al sistema motor de la persona, causando problemas en los movimientos y la postura. Esta discapacidad o conjunto de trastornos son **permanentes y no progresivos**. A menudo, estas personas padecen otro tipo de discapacidades, estas suelen afectar al sistema nervioso, causando limitaciones en la capacidad de usar los sentidos con los cuales percibir los estímulos exteriores y limitaciones en la capacidad intelectual, es por ello que a la parálisis cerebral se le puede llamar una pluridiscapacidad. [8, 34, 28]

Este conjunto de **discapacidades** producen una gran limitación en las tareas que las personas que lo sufren pueden realizar. Al ser parte de las discapacidades relacionadas con la postura y el sistema motor del cuerpo, las carencias en cuanto al movimiento son claras, además las discapacidades en el sistema nerviosos que pueden causar limitaciones sensoriales y cognitivas que pueden llevar a restricciones en la percepción del mundo exterior, en la conducta o en la **comunicación**.

La parálisis cerebral se origina principalmente en la gestación del feto y es causada por tres tipos de factores: **prenatales** causados por alteraciones en la coagulación o de la placenta, por traumatismos, por infecciones y por otras causas incluida la gestación múltiple, **perinatales** donde nos encontramos con diferentes causas entre las que destaca la prematuridad y por último, **postnatales**, a partir de más de 28 días después del nacimiento, donde encontramos causas como traumatismos craneales postnatales, paradas cardio-respiratorias, etc.[5, 19]

Como ya se ha comentado antes, la parálisis cerebral es una discapacidad que afecta a un gran número de personas, pero cada parálisis cerebral es

diferente. Según ASPACE, 1 de cada 500 personas nacen en España con parálisis cerebral, lo que nos lleva a un total de 120.000 personas en todo el país.[6]

Hay distintos tipos, e incluso de clasificaciones, de parálisis cerebral. En el extremo de las personas más afectadas nos encontramos con las mayores carencias que he comentado, estas personas sufren de una gran limitación de movimiento y postura, teniendo que estar todo el día en una cama y si se quieren mover, montados en sillas especiales hechas a medida para no producirles ningún tipo de molestia ni de lesión. Una de los mayores problemas que tienen las personas que tienen este grado de discapacidad es la limitación en la **comunicación**, ya que debido a sus carencias en la capacidad de moverse y de controlar estos movimientos solo pueden emitir **ruidos** o carcajadas, y en algunos casos ni siquiera se puede saber si tienen una intención comunicativa, es decir, emiten esos ruidos para intentar comunicarse, o simplemente son ruidos sin sentido.

Hoy en día se usan distintas aplicaciones y programas para que personas con parálisis cerebral se puedan comunicar [14], pero estas aplicaciones no pueden ser usadas por las personas gravemente afectadas debido a sus restricciones en el movimiento y control de su cuerpo, esto es un gran problema ya que les excluye de usar estas herramientas para poder comunicarse. Como he comentado anteriormente, las personas gravemente afectadas solo emiten ruidos, estos ruidos pueden o no tener una intención comunicativa, solo son los familiares y los cuidadores más cercanos a la persona los que pueden llegar a intuir si ese ruido tiene alguna intención, y, si la tiene, poder interpretarlo para saber si, por ejemplo, tiene hambre o le duele algo. El problema está en si esta persona con parálisis cerebral se tiene que quedar, por ejemplo, con un cuidador nuevo que no conoce ni puede interpretar estos sonidos.

Es por ello que comenzó el desarrollo de **AVC**, una aplicación Android accesible e intuitiva, que nos permite interpretar la emoción o la respuesta de una persona con parálisis cerebral gravemente afectada a partir de los sonidos que emite, gracias a métodos de clasificación de minería de datos.

1.1. Estructura de la Memoria

La presente memoria se compone de los siguiente apartados:

- **Introducción:** Descripción de la parálisis cerebral, planteamiento del problemas e introducción de la solución.

- **Objetivos del proyecto:** Conjunto de objetivos funcionales, técnicos y personales que se han tenido a lo largo de este proyecto.
- **Conceptos teóricos:** Explicación de nociones teóricas básicas para poder entender el proyecto.
- **Técnicas y herramientas:** Conjunto de técnicas y herramientas esenciales en el desarrollo del proyecto.
- **Aspectos relevantes del desarrollo:** Explicación temporal de los hechos más relevantes del proyecto.
- **Trabajos relacionados:** Herramientas y proyectos semejantes.
- **Conclusiones:** Conclusiones sobre el desarrollo del proyecto y posibles líneas futuras de trabajo.

1.2. Estructura de los Anexos

El documento con los anexos tiene los siguiente componentes:

- **Plan del proyecto:**
- **Requisitos:**
- **Diseño:**
- **Manual del programador:**
- **Manual de usuario:**

1.3. Materiales adjuntos

Junto con esta memoria se entrega un CD con el siguiente contenido:

- Memoria en formato pdf.
- Anexos en formato pdf.
- Aplicación para la recogida de datos (prototipo inicial).
- Aplicación para la recogida de datos, versión final.

- Aplicación AVC para poder interpretar los sonidos de las personas con las que se ha entrenado.
- Servidor en Flask para la administración de los datos y para el procesamiento y clasificación de los audios.
- Manuales de usuario para las dos aplicaciones.
- Presentaciones de las aplicaciones que se han usado para enseñar el proyecto en diversas situaciones.
- Vídeos e imágenes de las diversas presentaciones que he realizado en el proyecto.

Objetivos del proyecto

En este apartado voy a comentar los distintos objetivos funcionales, técnicos y personales del proyecto.

2.1. Objetivos funcionales

En este subapartado voy a contar cuales han sido los objetivos, en cuanto a qué queríamos que hiciesen las distintas aplicaciones del proyecto.

- Tener una aplicación Android que nos permita recoger los audios y las opciones adicionales de cada paciente, y poder subirlos para su posterior tratado y análisis, con finalidad de encontrar el mejor algoritmo de clasificación posible para la aplicación final.

- Realizar una aplicación Android para la interpretación de los audios. En la aplicación se tiene que poder modificar las opciones adicionales de los pacientes, almacenándose en el servidor para no tener que ponerlas en cada ejecución.

- Hacer que ambas aplicaciones sean simples de usar y con un ejecución muy lineal.

- Diseñar una aplicación que sea accesible para que pueda ser usada hasta por compañeros con parálisis cerebral u otro tipo de discapacidad.

- Dar difusión mediática del proyecto, grabando vídeos, haciendo presentaciones, etc.

2.2. Objetivos técnicos

Los objetivos técnicos son los que he tenido en cuenta a lo largo del desarrollo en cuanto a las herramientas y el uso de estas.

- Generar dos aplicaciones, la de obtención de datos y la de interpretación de sonidos, que tengan una versión de API mínima de 23 (a partir de Android 6.0), para poder abarcar un mayor número de dispositivos donde se puedan usar.
- Utilizar *Python* para la clasificación, por el conocimiento que ya tengo y por el uso de librerías avanzadas en métodos de clasificación como puede ser *Scikit-Learn*.
- Usar Flask como framework para el servidor.
- Usar un repositorio Git para tener un seguimiento del desarrollo y un control de versiones usando GitHub.
- Usar ZenHub para controlar las tareas y tiempos del proyecto.
- Realizar pruebas unitarias y de integración sobre la aplicación final.
- Realizar el desarrollo del proyecto siguiendo el modelo SCRUM, orienta a trabajo personal, sobre todo en la filosofía incremental de los productos.

2.3. Objetivos personales

En este apartado quiero comentar los objetivos propios que he tenido con este proyecto.

- Poder ayudar a personas con parálisis cerebral.
- Mejorar mi capacidad de comunicación y exposición, preparando diversas presentaciones.
- Usar el conocimiento obtenido a lo largo de la carrera.
- Aprender a programar en Android, ya que es uno de los sistemas operativos más importantes a día de hoy.
- Aprender a desarrollar y desplegar un servidor.

- Aprender a usar nuevas herramientas de documentación, como LaTeX.
- Aprender a usar nuevas herramientas de diseño y programación.

Conceptos teóricos

En este apartado voy a comentar los distintos conceptos teóricos del proyecto, que van desde la parálisis cerebral hasta conceptos como *Random Forest*.

3.1. Parálisis Cerebral

Como ya he comentado, la parálisis cerebral es un discapacidad en el sistema motor y en la postura de la persona. Además, esta discapacidad suele ir acompañada de otras discapacidades en el sistema nervioso que producen limitaciones en los sentidos y en la capacidad cognitiva de la persona [28, 34].

El origen de esta discapacidad se debe a varios factores, mayoritariamente relacionados con el desarrollo del feto. Además, el grado de afección de la parálisis cerebral es distinto en cada caso, teniendo personas levemente afectadas que solo sufren de una discapacidad motora, hasta personas gravemente afectadas con grandes discapacidades motoras, cognitiva, sensoriales...

Existen diversas asociaciones que ayudan a las personas que tienen parálisis cerebral y a las familias, amigos o cualquier persona que acompañe a la persona afectada. Por suerte, en España existen multitud de estas asociaciones, siendo Confederación ASPACE [7] la confederación a nivel nacional. En Castilla y León la asociación es ASPACE Castilla y León, y en Burgos tenemos a APACE Burgos 3.1 [9].

Figura 3.1: Logotipo de APACE Burgos a la izquierda y ASPACE CyL a la derecha.

3.2. Señal de audio

El asistente desarrollado toma como entrada un audio y un conjunto de opciones adicionales que hemos obtenido en colaboración con APACE.

Sonido

Una señal sonora es una perturbación mecánica (vibraciones) en la presión del aire [22], esta se convierte a una señal eléctrica en concreto a una señal analógica, que es una señal generada por un fenómeno electromagnético la cual se puede representar a través de una función continua cuyos parámetros son la amplitud y el periodo [10], mediante un proceso electromagnético y que posteriormente es digitalizada a través de un proceso de muestreo, determinado por la frecuencia de muestreo o *sampling rate*, y de un proceso de cuantificación, que junto con la frecuencia de muestreo determina el *bitrate* de la señal digital. A esta señal almacenada se le llama señal de audio, y esta puede ser obviamente almacenada, reproducida y modificada.

Espectrograma

Es una representación gráfica de las señales de audio, es decir, una representación de las frecuencias de la señal sonora almacenada en la señal analógica que compone la señal de audio [18]. En esta representación la distribución de energía en tres ejes, siendo el eje de las x el eje temporal, el eje de las y el de las frecuencias y el color (como eje z) muestra el dominio o la distribución de la señal de audio con respecto a los otros dos ejes, el tiempo y las frecuencias [11].

Como ya comenté, las señales de audio se caracterizan por su periodo y amplitud, como se puede ver en la representación de la onda en la figura 3.2. En cambio el espectrograma de este se muestra en la figura 3.3 en una escala lineal, y en una escala logarítmica en la figura 3.4.

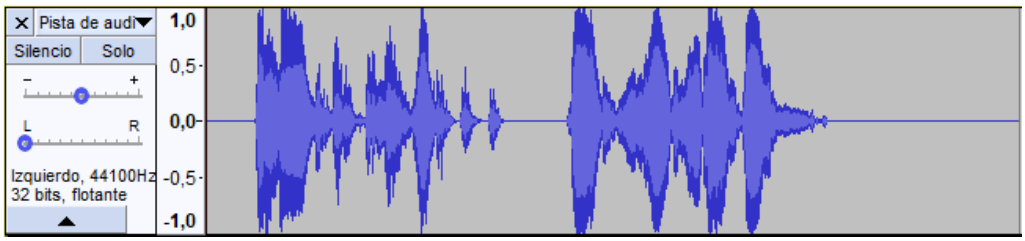

Figura 3.2: Representación de la onda. En el eje x se representa el tiempo y en el eje y la amplitud.

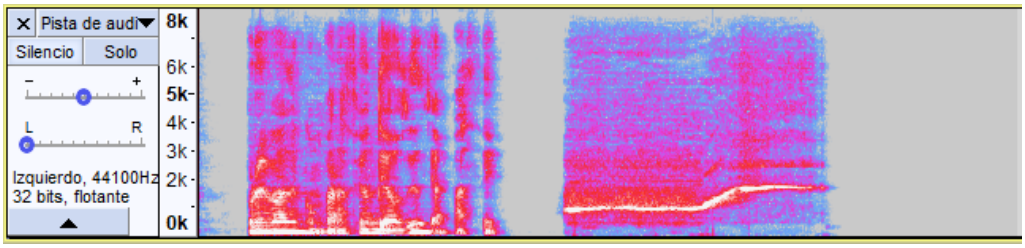

Figura 3.3: Espectrograma en escala lineal. En el eje x se representa el tiempo y en el eje y la componente espectral (frecuencia) en escala lineal.

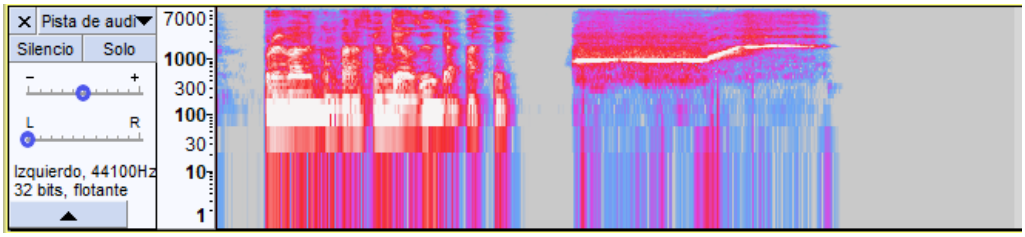

Figura 3.4: Espectrograma en escala logarítmica. En el eje x se representa el tiempo y en el eje y la componente espectral (frecuencia) en escala logarítmica.

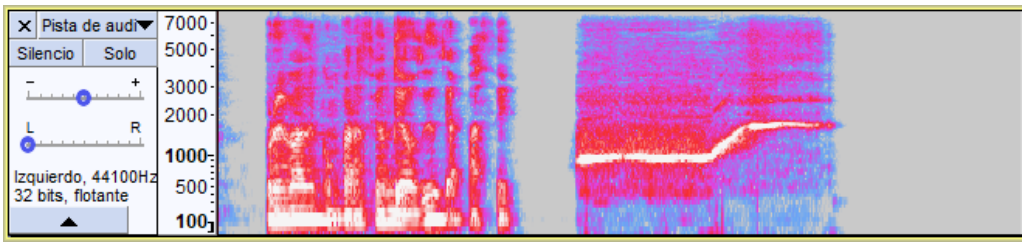

Figura 3.5: Espectrograma en escala de Mel. En el eje x se representa el tiempo, en el eje y las frecuencias relativas a la capacidad auditiva humana.

Frecuencias en escala de Mel, MFCC

Es una escala que surgió con el objetivo es la obtención de una escala orientada al sistemas auditivo del ser humano, lo que se llama una escala psicoacústica, con el fin de poder extraer características de la señal de audio para poder obtener información. Podemos ver el ejemplo del mismo audio en la figura 3.5

Esta es la fórmula de las frecuencias en escala de Mel y las frecuencias en escala lineal (f)[32, 29]:

$$Mel(f) = 2595 * \log_{10}(1 + \frac{f}{700})$$

MFCC (*Coeficientes Cepstrales en las Frecuencias de Mel*) son los coeficientes necesarios para esta representación del sonido relacionado con el sistema auditivo humano. El proceso de obtención de este coeficiente sigue los siguientes pasos [32, 35]:

- División del audio en fragmentos (*frames*).
- Minimizar las discontinuidades de la señal en el comienzo y final de cada fragmento.
- Aplicar la Transformada discreta de *Fourier* para conseguir la potencia espectral.
- Aplicar los filtros correspondientes a la escala de *Mel*.
- De cada frecuencia de *Mel* de las energías obtenemos el logaritmo.
- Aplicamos la transformada de coseno discreta a los logaritmos.

Zero Crossing Rate

Es el ratio que mide el número de veces en las cuales la señal cambia de signo.

Bitrate

Es la unidad de datos que se recogen por unidad de tiempo, los datos recogidos se miden en bits, mientras que el tiempo se mide en segundos Cuando trabajamos con sistemas de recogida de datos, en nuestro caso grabaciones de audios, el *bitrate* de esta recogida es un valor muy importante, ya que determina la cantidad de datos que vamos a obtener, esta medida puede ser muy interesante a la par de necesaria, ya que, junto con el formato y el códec, va a determinar cuánto ocupan los archivos que obtenemos además de la calidad de estos.

Sampling Rate

El Sampling Rate o frecuencia de muestreo es la tasa que marca la cantidad de muestras que se hacen por unidad de tiempo, es decir, de nuestra función continua que es la señal analógica que recoge la señal sonora, la señal de audio, pasamos a valores discretos. La unidad con la que se mide es, como en todas las frecuencias, s^{-1} o Hz [33].

Formato digital

Dentro de una señal de audio, y en general para cualquier tipo de dato que se almacena de forma digital, el formato es el estándar con el cual el dato se codifica, se decodifica y se lee.

Dentro de los formatos que existen en las señales de audios o formatos contenedores de audios podemos diferenciarlos entre los que tienen pérdida de información o *Lossy* y los que no la tienen o *Loseless*. Los formatos que tienen pérdida suelen ser formatos más ligeros en los cuales los ficheros ocupan menos y son más fáciles de procesar, en cambio como su nombre indica tienen pérdidas de información [31].

Encoder

El encoder o códec son los métodos por los cuales podemos codificar y decodificar los datos de audios almacenados en el formato seleccionado. Para cada códec disponemos una serie de formatos en los cuales se puede

Códecs	***Lossy***	***Loseless***
AAC-LC	X	
AAC-ELD	X	
HE-AAC	X	
AMR-NB	X	
AMR-WB	X	
PCM(Pulse Code Modulations)		X
FLAC		X

Tabla 3.1: Códecs de audio

codificar y decodificar, por lo que en si, la combinación códec-formato es la que puede ser *Lossy* o *Loseless* [30].

La mayoría de códecs son *Lossy*, es decir, pierden información en la codificación que luego no se puede recuperar en la decodificación, alguno de los códecs de audios más usados son los que se pueden observar en la tabla 3.1.

Base64

Método que nos permite codificar cualquier tipo de dato y/o archivo en texto ASCII y decodificarlo. Este tipo de codificación no es la más eficiente que existe, pero si que nos puede servir si queremos mandar nuestros ficheros en modo texto, como pasa en este proyecto, donde tenemos que enviar el audio grabado al servidor a través de un método post donde las variables se pasan en la URL.

Extracción de características

Proceso por el cual podemos obtener información a partir de datos. Este proceso se suele llevar a cabo en el preprocesado de información de entrada de los métodos de clasificación de minería de datos, ya que datos como pueden ser imágenes, o en nuestro caso audios, no las podemos pasar tal cual al clasificador, porque no sería capaz de interpretar la entrada, y si por un casual pudiese, esta tendría una gran dimensionalidad.

En nuestro caso, del audio se obtienen los 500 *frames* que se van a estudiar, de estos 500 se obtienen los 12 primeros coeficientes de *MFCC* 3.2 y su *Zero Crossing Rate* 3.2. Pero no podemos pasarle a nuestro clasificador una matriz de 500x13, por lo que convertimos esta matriz en una única fila

con 26 columnas, que son, las 12 medias de los coeficientes de *MFCC*, las 12 desviaciones estándar de los coeficientes de *MFCC*, la media del *Zero Crossing Rate* y su desviación estándar.

3.3. Minería de Datos, Bagging y Random Forest

La minería de datos tiene diversas definiciones válidas, pero todas ellas coinciden en que es un proceso en el cual a partir de grandes cantidades de datos a los que se aplican técnicas de inteligencia artificial o de análisis de datos, podemos obtener patrones, relaciones o en definitiva, información, con la que podemos clasificar o dividir en grupos nuevos datos.

La minería de datos entra dentro del apartado de aprendizaje automática, entendiendo como aprendizaje cuando en un sistema cambiamos el comportamiento de alguna parte o del conjunto y obtenemos una mejora en el rendimiento.

La extracción de información de la minería de datos se basa en la hipótesis de *Aprendizaje Inductivo*, que es "*Cualquier modelo que aproxime bien una función objetivo sobre un conjunto de ejemplos de entrenamiento suficientemente grande también aproximará bien la función objetivo en ejemplos no observados*", es decir, los patrones encontrados en los datos de entrenamiento de nuestros modelos de minería de datos, con un número suficientemente grande de datos, servirá para nuevos datos del mismo tipo [27].

Los métodos de minería de datos tienen distintas finalidades entre las que se destaca:

- Predicción:
  - Clasificación de datos categóricos.
  - Regresión de datos numéricos.
- Análisis de asociaciones entre los atributos que definen los datos.
- Clustering, o agrupación de los datos en distintos grupos.
- Detección de anomalías.
- Sistema de recomendaciones.

Además, dentro de la minería de datos existen diferentes métodos para llegar a las finalidades anteriormente comentadas, estos tipos de algoritmos van desde métodos basados en árboles, hasta métodos más estadísticos como los modelos de clasificación bayesiana que usan la suposición de *naïve*, en la cual se supone que los atributos de los datos no tienen ninguna relación [26].

Dentro de los algoritmos de minería de datos hay una serie de técnicas que nos permiten combinar varios modelos clasificadores para obtener un único modelo, *ensembles*, que aunque son más complejos de representar, pueden obtener mejores resultados. La combinación de modelos puede o no ser del mismo tipo de clasificador, es decir, hay técnicas que nos permiten usar por ejemplo métodos bayesianos con árboles y otros que solo nos permiten un mismo tipo de clasificador. Una de las técnicas más usada de combinación de métodos es *Bagging*, esta consiste en la combinación mediante media en problemas de regresión y mediante votación en problemas de clasificación, de los resultados obtenidos por el mismo método. El algoritmo lo que hace es a partir de varios grupos de datos o muestras del mismo problema obtenemos para cada muestra un modelo entrenado con los datos de esa muestra. El método final lo que hace es combinar los resultados, como ya se ha dicho por media o votación, de los distintos modelos de las distintas muestras.

Uno de los métodos de Bagging más usados, ya que es un método sencillo que nos puede dar una primera visión del problema ante el que estamos, es *Random Forest*, en el cual combinamos las predicciones de distintos árboles de decisión, que usa como estimador del error el *out of bag*, que consiste en intentar predecir con cada modelo los datos de las muestras con los que no se ha entrenado ese modelo [26].

Técnicas y herramientas

En este proyecto he trabajo aspectos muy distintos, es por ello que he tenido que usar técnicas y herramientas muy variadas para poder hacer las distintas tareas.

4.1. Diseño

En las distintas fases de diseño he empleado distintas herramientas según para que tipo de diseño estaba haciendo, estas herramientas son:

Pencil

Herramienta que nos permite el diseño por pantallas desde páginas webs, programas o aplicaciones móviles, con una gran variedad de elementos como botones, checkboxs... 4.6

Figura 4.6: Pencil, herramienta para el diseño de interfaces.

DIA

Programa especializado en el diseño de diagramas UML.

WebSequenceDiagrams

Herramienta online especializada en los diagramas de secuencia, www.websequencediagrams.com.

4.2. Desarrollo de las aplicaciones Android

Para el desarrollo de las diversas aplicaciones Android que tenía que realizar en el proyecto pensé en dos opciones, en cuanto IDE con el que hacerlas, estas opciones eran *Eclipse* y *Android Studio*, en principio quería hacerlo en *Eclipse* debido a que es una herramienta con la cual hemos trabajado mucho a lo largo de la carrera, pero al final me decanté con *Android Studio* ya que es el IDE más extendido para el desarrollo de aplicaciones Android y además tiene una interfaz bastante parecida a *Eclipse*, por lo que no me costó mucho adaptarme.

Android Studio

Android Studio es el IDE oficial de aplicaciones Android, fue diseñado por el propio equipo de Google para poder desarrollar de forma fácil e intuitiva en Android, basado en *IntelliJ IDEA*. Este IDE nos permite desde programar la lógica de nuestras aplicaciones, diseñar la interfaz de estas e incluso probarlas con un dispositivo o con un emulador [1].

En cuanto a cómo se puede probar una aplicación desde *Android Studio* hay dos opciones bien definidas, o usas un emulador o lo pruebas conectando directamente un dispositivo Android. En mi caso preferí desde el primer momento probar mis aplicaciones con un dispositivo propio, ya que son los dispositivos con los que me siento más seguro y voy a presentar en los distintos eventos, y porque así consumo menos recursos dentro del ordenador desde el cual programo.

Espresso

Espresso 4.7 es una herramienta que nos permite grabar ejecuciones de nuestra aplicación Android con las que poder probar las funcionalidades y sobre todo la integridad de la aplicación [2].

Figura 4.7: Espresso, grabación de test.

MonkeyTest

Es un tipo de test en el cual se pulsa de forma aleatoria en la pantalla. Este tipo de test nos da una respuesta a cómo de fuerte es nuestra aplicación antes situaciones de estrés [4].

4.3. Desarrollo del servidor

El desarrollo del servidor, como ya he comentado, se quería desde un principio hacer en *Python* ya que disponemos de una gran variedad de librerías de calidad de minería de datos. Teniendo en cuenta que quería hacerlo en *Python*, elegí *Flask* como framework para desarrollar el servidor por recomendación de los docentes de la universidad.

Flask

Microframework que nos permite diseñar y desplegar un servidor web de forma sencilla gracias a su alta abstracción.

PostMan

Herramienta que nos permite probar request HTTP, en mi caso lo he usado en las primeras versiones del servidor cuando no tenía la aplicación final conectada, para poder comprobar que los request post se hacían correctamente 4.8.

Jupyter Notebook

IDE que nos permite programar y ejecutar código *Python*.

Figura 4.8: PostMan, herramienta para realizar *request HTTP*.

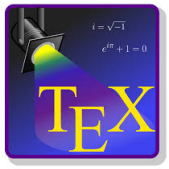

Figura 4.9: TexStudio, editor de LaTeX.

Sublime Text

Programa para editar código, usado para programar en *Python*.

4.4. Documentación

Para la documentación del proyecto se han usado distintas herramientas, para las primeras documentaciones, manuales de usuario y otro tipo de documentos realizados en las fases iniciales del proyecto se ha usado *Microsoft Office Word* y *Libre Office* ya que ya tenía un conocimiento y dominio previo. Pero para la documentación he elegido LaTeX, ya que es uno de los objetivos personales 2.3, además que como opinión personal queda bastante mejor, aunque quizás es un más difícil que otras opciones como *Microsoft Office Word*.

TexStudio

LaTeXes un sistema de composición de texto de alta calidad tipográfica, orientado y principalmente usado en artículos científicos.

TexStudio es un editor que nos permite de forma sencilla editar nuestros documentos en LaTeXy poder ver la compilación con el resultado 4.9.

Microsoft Office Word

Editor de documentos de textos dentro del paquete de ofimática *Microsoft Office*.

Libre Office Writer

Editor de documentos de texto dentro del paquete libre de ofimática *Libre Office*.

4.5. Otros

En este apartado voy a comentar otras herramientas usadas en el proyecto, pero que no entran en ninguno de los apartados anteriores.

- **Microsoft Office PowerPoint**: herramienta para realizar presentaciones, dentro del paquete de ofimática de *Microsoft Office*.
- **Audacity**: programa para el tratado de audios, donde podemos ver las distintas representaciones de la onda y espectrogramas en las distintas escalas.
- **noWifi**: aplicación que permite a la tarjeta de red de un ordenador hacer las funciones de router para desplegar de forma sencilla una red hosteada.
- **RandomKeyGen**: página web que nos permite generar claves aleatorias de distintos tamaños, `https://randomkeygen.com/`.
- **Paint**: herramienta para la edición de imágenes.
- **Gimp 2**: herramienta para la edición de imágenes.
- **AZ Screen Recorder**: aplicación móvil que nos permite grabar la pantalla de nuestro dispositivo.
- **GitHub**: control de repisotorios *Git*.
- **ZenHub**: herramienta enlazada con *GitHub* para el control de versiones y técnica *SCRUM*.

Aspectos relevantes del desarrollo del proyecto

En este apartado voy a comentar, en forma de resumen temporal, las partes más importantes en el desarrollo del proyecto, desde decisiones importantes de diseño, hasta problemas que me han llevado mucho tiempo resolver. He considero que lo mejor para este apartado es dividir los aspectos relevantes en secciones.

5.1. Comienzo con APACE e investigación

APACE Burgos presentó el proyecto AVC a los premios de Fundación Vodafone para buscar financiación para poder realizar el proyecto. Este premio se lo concedieron y fue entonces cuando se pusieron en contacto con la Universidad de Burgos para poder desarrollar el proyecto. Es en ese momento en el que me comunican la posibilidad de hacer este proyecto como Trabajo Fin de Grado.

Tuvimos una serie de reuniones con APACE donde nos expusieron el problema, que es la falta de herramientas de comunicación para personas con parálisis cerebral gravemente afectadas y la consiguiente exclusión tanto social como laboral, y discutimos sobre las posibles soluciones que tendría el problema, hasta que llegamos a la solución conceptual que define el proyecto hoy en día, una aplicación intuitiva y accesible que nos permite interpretar sonidos de personas con parálisis cerebral.

En las primeras semanas empecé a estudiar como se hacía una aplicación Android, a la par que comenzaba la parte de investigación para ayudar a mi compañero en el proyecto, Sergio Chico Carrancio. Los artículos de

investigación que estudié estaban relacionados con el problema al que nos enfrentábamos, extracción de características de audios y clasificación de sonidos. Entre los artículos que leí destacan el uso de distintos métodos de preprocesado del audio para, por ejemplo, eliminar los silencios, como se puede ver en el artículo sobre reconocimiento de sonidos de animales [35]. Después del preprocesado del audio se pasa a la extracción de características a partir de este, muchos de los métodos más usados para la extracción de características en audios usan la escala de Mel, explicada en el apartado 3.2. Y por último, tras obtener las características de los audios procesados ya tenemos la entrada para nuestros métodos de clasificación, es en estos donde podemos ver más variación ya que podemos tener métodos sencillos como puedes ser un KNN o métodos más complejos como pueden ser las Redes Neuronales [21].

5.2. Prototipo aplicación y estudio sobre el audio

Como mi parte en este proyecto no era la investigación, sino el desarrollo de las aplicaciones y el servidor, deje la parte de investigación para empezar a desarrollar la primera aplicación. En este primer prototipo me basé en una tutorial que enseñaba como usar *MediaRecorder*, una clase que he utilizado hasta el final para grabar audios con los dispositivos Android.

La finalidad de esta primera aplicación, cuya pantalla inicial se ven en la figura 5.10, en un principio, era para realizar la recogida de datos necesarios para realizar la investigación de la que se ocupaba el investigador Sergio Chico, y eran las personas encargadas de grabar los audios los que tenían que apuntar a mano y enviarnos a través de correo el conjunto de datos, es decir, el audio junto a ciertas opciones adicionales que luego se comentan en el siguiente apartado. Pero tras varias reuniones con APACE se llego a la conclusión que iba a ser muy costoso para las personas encargadas (familiares y cuidadores) generar datos y apuntar a mano, así que se planteó una segunda aplicación para la recogida de datos, pasando esta primera aplicación a ser una aplicación prototipo. Cabe destacar dentro de esta aplicación dos funcionalidad que añadí, la primera que era la generación automática del nombre que tiene el fichero de audio de la salida, y la segunda funcionalidad es la petición de permisos de acceso al almacenamiento del dispositivo y de grabación de audios. Esta funcionalidad y el código que la implementa se ha usado en las 3 aplicaciones del proyecto y me costó mucho tiempo conseguir que funcionasen, ya que la ejecución de la aplicación se

Figura 5.10: Aplicación prototipo.

paraba y cerraba por no tener los permisos concedidos, lo solucione gracias a añadir los permisos en el *AndroidManifest.xml* y a la siguiente función que pide los permisos.

```
private void askForPermissions() {
        String[] perm = {Manifest.permission.
           RECORD_AUDIO,Manifest.permission.
           WRITE_EXTERNAL_STORAGE};

        if(ContextCompat.checkSelfPermission(this.
           getApplicationContext(),perm[0]) !=
           PackageManager.PERMISSION_GRANTED ||
           ContextCompat.checkSelfPermission(this.
           getApplicationContext(),perm[1]) !=
           PackageManager.PERMISSION_GRANTED){
                requestPermissions(perm,1234);
        }
}
```

A la vez que desarrollaba el prototipo estaba investigando acerca de qué formato y qué códec de audio iba mejor para la configuración del *MediaRe-*

corder, es aquí donde realizamos el estudio del audio. Aquí nos encontramos con el problema de que en Android y en concreto con *MediaRecorder* no podíamos codificar, es decir, grabar en cualquier formato, por ejemplo, vimos que *mp3* con *MediaRecorder* en Android solo permite descodificar. Tras realizar el estudio para encontrar un formato sin pérdida, no conseguimos obtener un formato que tenga un códec sin pérdida, así que intentamos obtener el mejor formato con la mayor calidad posible. Es por ello que elegimos el códec *AAC-LC*, formato *mp4*, con un bitrate de 128kbps y un sampling rate de 48kHz [3].

5.3. Obtención de las opciones adicionales

Al comienzo del proyecto se comentó con APACE que quizás los audios no serían suficientes para poder realizar una clasificación exacta de las emociones, ya que la clasificación de una respuesta binaria si que se tomaba posible de clasificar solo con el audio. Es por ello que para este problema nos planteamos el uso de unas opciones adicionales que pudiesen dar más información al clasificador.

APACE nos mandó una primera versión de estas opciones adicionales, que obtuvieron pasando un formulario a las familias de la asociación y a profesionales en la materia, en las que teníamos un conjunto de opciones por cada una de las emociones que queríamos clasificar, que se pueden ver en la imágenes 5.11, 5.12, 5.13 y 5.14. Como se puede observar son demasiadas opciones, tantas que darían demasiadas dimensionalidades a los datos de entrada del clasificador, lo que haría muy complicado el entrenamiento y la clasificación posterior. Es por ello que decidimos tratar de resumir estas opciones, estas opciones han sido las que han llegado al final del proyecto y se pueden ver en la tabla 5.2.

5.4. Desarrollo de la aplicación de recogida de datos

Como ya he comentado, en una reunión con APACE nos comentaron el cambio que querían hacer en la aplicación prototipo para la recogida de datos, y es que querían que la selección de las opciones adicionales y de la emoción o respuesta relacionada se pudiese hacer desde la aplicación. Con este cambio tuve que empezar a crear la segunda aplicación a partir de la aplicación prototipo. Desde APACE se nos comentó que esta aplicación debía ser lo más sencillo posible de usar, ya que algunas familias que podían

Herida o escara.	Situación de enfermedad.
Fiebre.	
Enfermedad diagnosticada.	
Tiene dolor crónico.	Dolor Crónico
Si está sentado correctamente. (tanto en el momento, como si ha estado sentado incorrectamente sentado anteriormente)	Posicionamiento incorrecto (habitual o o introducción de nuevo sistema de posicionamiento
Alguna ortesis puede producirle dolor (un dafo, una sujeción, un lecho postural nuevo, corset...)	
Comprobar que haya alguna situación en la que pueda aparecer dolor o incomodidad (se está pinchando con algo, o está próximo a una fuente de calor que puede producir una quemadura...)	Situación de riesgo que puede producir dolor
Si está tomando una medicación para el dolor y no se le ha dado aún o está a punto de cumplir el tiempo para tomar la dosis prescrita	Tratamiento para el dolor Intervención quirúrgica o tratamiento que produce dolor
Ha sido intervenido quirúrgicamente recientemente.	
Está realizando algún tratamiento que puede ser doloroso (Bogta, Punción Seca, Toxina Botulínica)	

Figura 5.11: Opciones adicionales proporcionadas por APACE de la emoción dolor.

Carácter irascible.	Personalidad irascible (Base)
No responder a una demanda expresada (a su manera)	No entender su forma de comunicarse o sus demandas.
No entender su forma de expresarse.	
No aceptar un no por respuesta.	No atender a sus demandas.
Defecarse y orinarse.	
Aburrirse o no hacer una actividad que desea hacer	
Crisis epilépticas que influyan en su conducta	Estado de salud (crisis epilépticas).
Estar viviendo una situación similar a una ya vivida percibida como negativa.	Situación desagradable.

Figura 5.12: Opciones adicionales proporcionadas por APACE de la emoción enfado.

Sufre dolor.	Le duele algo
Experimentar una situación desagradable.	Experimenta una situación desagradable (Que no le gusta o porque no tiene cerca a alguien cerca que conoce.
Estimulo que no le gusta. (Visual, auditivo, táctil, alimento de su gusto, un programa de tele, una canción)	
No ve o no tiene cerca a una persona a la que conoce o aprecia.	No tiene cerca a una persona a la que aprecia.

Figura 5.13: Opciones adicionales proporcionadas por APACE de la emoción tristeza.

Horario de comida habitual.	Cambio en los hábitos de comida: Horarios
Tiempo que lleva sin comer.	
Hay comida a la vista o ve a alguien comiendo.	Presencia de alimento: vista, olfato
Hay olor a comida.	
Si hay algún alimento que le gusta a la vista.	
Habitualmente padece hambre.	Estado habitual abre.

Figura 5.14: Opciones adicionales proporcionadas por APACE de la emoción hambre.

Opción	**Posibles valores**
Actualmente está enfermo	Sí/No
Sufre dolor crónico	Sí/No
Ha sido operado recientemente	Sí/No
Ha dormido/descansado mal	Sí/No
Ha estado/está en una mala postura	Sí/No
El entorno que lo rodea no es agradable	Sí/No
Las personas que lo rodean no son conocidas	Sí/No
Ha comido	Antes/A su hora/Tarde
Ha comido	Mucho/Normal/Poco/Nada

Tabla 5.2: Opciones adicionales finales

recoger datos no tienen mucho dominio de las tecnologías. Esta aplicación fue bastante más complicada de hacer, ya que era la primera vez que iba a trabajar y tener en cuenta la navegabilidad entre pantallas de la aplicación. Además, la aplicación tenía que ser lo más lineal posible, por lo que primero me puse a diseñar una interfaz, la cual se puede ver en la figura 5.15.

Una vez diseñada la interfaz pasé a aprender a navegar por las pantallas de una aplicación Android y a desarrollar la aplicación. Del desarrollo de esta aplicación cabe destacar:

- Navegabilidad intuitiva gracias al uso de los ActivityResult, que permiten terminar un Activity con un resultado según esta haya acabado de forma correcto o incorrecta. Esta funcionalidad la he usado para poder habilitar los botones, ya que se pidió que la aplicación fuese simple y lineal.
- En la pantalla de selección de la emoción o respuesta relacionada con los datos generados no se permite, ni dejarla vacía, ni elegir una opción de la columna de emociones y otra de la columna respuesta.
- Uso de ficheros csv para la almacenar y enviar tanto las opciones adicionales como la emoción o respuesta asociado, usando la librería OpenCSV [20].
- Persistencia y carga de los datos en todas las pantallas gracias a una creación de carpetas bien definidas y al uso de una estructura de datos bien definida, lo que nos permite volver a entrar a una de las pantallas y tener los datos que se habían generado. Esto ocurre tanto en la pantalla de grabación del audio, como en las pantallas de selección de opciones y de selección de emoción o respuesta.
- Generación de comprimidos con el archivo del audio y los archivos csv, para poder enviarlos con mayor facilidad.
- Envío por correo del comprimido para poder recogerlo y estudiarlo.

Este último punto hubo que discutirlo con los tutores, porque nuestras opciones para enviar el comprimido eran o hacerlo directamente con un servidor, o usar un correo electrónico. Al final, decidimos usar el correo electrónico ya que no teníamos tiempo de desplegar un servidor, y la aplicación tenía que estar lo antes posible en marcha para que se empezase cuanto antes a generar datos con los que el investigador colaborador, Sergio Chico, pudiese hacer el estudio.

Figura 5.15: Diseño interfaz aplicación recogida de datos.

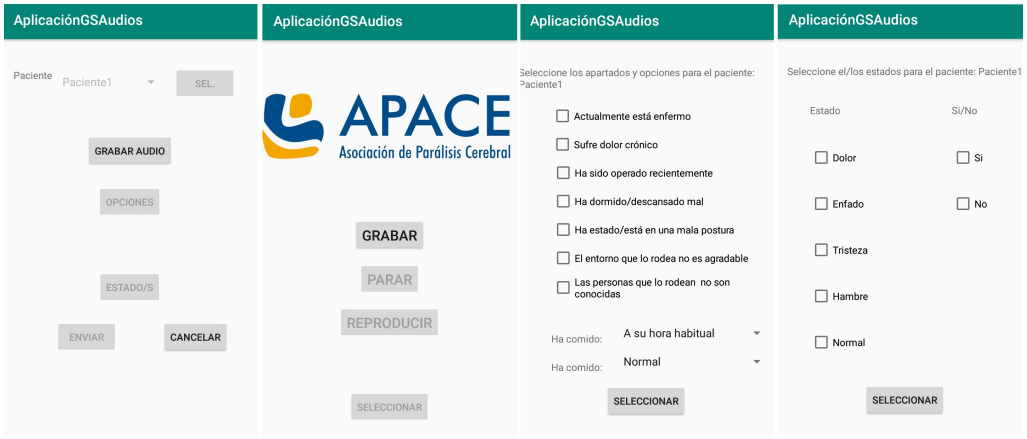

Figura 5.16: Interfaz aplicación recogida de datos.

La interfaz final de la aplicación se puede ver en la figura 5.16.

5.5. Obtención de sonidos prototipo

Tras finalizar la aplicación para la generación de datos, y sus presentaciones en APACE que posteriormente comentaré, nos encontramos con un problema que ya veíamos venir desde prácticamente el comienzo del proyecto,

y es la falta de datos. Obviamente no podíamos forzar a los pacientes a hacer sonidos, y menos cuando las emociones con las que nos pidió APACE trabajar son negativas (dolor, hambre, tristeza y enfado).

En una reunión comentamos las posibles opciones que teníamos, y estas eran o esparar a que antes de la entrega tuviésemos suficientes datos como para al menos tener un clasificador para una persona, o intentar imitar los sonidos para poder generar algunos datos con los que el investigador colaborador, Sergio Chico, pudiese empezar a trabajar. Con el fin de poder tener una aplicación prototipo final que nos permitiese clasificar audios decidí generar de forma artificial los audios con los que entrenar al clasificador.

Esta probablemente sea una de las tareas que más me ha costado en todo el proyecto, ya que tenía que seguir unos pasos, para que los datos que generase tuviesen el mayor sentido posible, estos pasos para cada una de las emociones y sí/no eran:

- Descargar algunos ejemplos que ya hubiesen enviado, para tener algo en lo que basarme.
- Grabar un audio propio intentando ceñirme lo máximo posible a los audios descargados anteriormente.
- En las emociones, estudiar las opciones adicionales para saber cuales se suelen repetir más con una emoción.
- Grabar al rededor de unos 20 audios por cada emoción y sí/no, teniendo que en cada uno escucharlo, valorar si es o no válido y poner las opciones adicionales correspondientes.

5.6. Desarrollo de la aplicación de interpretación - AVC

Una vez que ya contábamos con suficientes datos como para que el investigador colaborador, Sergio Chico, empezase a estudiar las mejores formas de extraer las características del audio y poder clasificarlo, yo empecé a desarrollar la aplicación final, la aplicación más importante en el proyecto, ya que es la que se más se usará, y la que dará esa mejora en la vida de las personas con parálisis cerebral que buscamos.

Lo primero que realicé fue el diseño de la interfaz junto con un equipo especializado de APACE, diseñé al rededor de 6 interfaces iniciales, en las

cuales se destaca el uso desde el principio de botones de información para dar mayor accesibilidad a la aplicación, y a las cuales este equipo me fue indicando cambios y mejoras hasta llegar al diseño final, que aparece en la figura 5.17. Cabe destacar una cambio muy importante que me comentaron desde APACE, y es que las opciones adicionales no se cumplimentasen cada vez que queremos hacer una interpretación, sino que estas se almacenaran en el servidor y solo se modificasen cuando sea necesario.

Esta es la aplicación más importante y compleja del proyecto, es por ello que es a la que sin duda he dedicado más tiempo. En el desarrollo de esta primera versión sin conexión al servidor cabe destacar los siguientes apartados:

- **Creación de una estructura de ficheros:** Necesario para poder controlar los distintos ficheros con los que trabaja la aplicación, siendo la raíz de esta estructura donde se almacena el audio grabado. En la carpeta */config* tenemos el fichero *csv* donde se almacena al paciente por defecto.
- **Comprobación y restauración de la estructura de ficheros**: Para que al iniciarse la aplicación lo primero que se compruebe sea que la estructura de ficheros existe, y si falta alguna carpeta y/o fichero los genera.
- **Implementación de la selección del paciente por defecto:** Que al abrir de nuevo la aplicación aparezca en el *spinner* del paciente el último paciente seleccionado antes de cerrar la aplicación. Esto se ha conseguido gracias al conocimiento sobre el control de ficheros *csv* obtenido en el desarrollo de la aplicación de generación de datos con *OpenCSV* [20].
- **Continua implicación por parte de APACE:** Necesaria para conseguir que la aplicación fuese lo más accesible posible. Esto dio como resultado algunos cambios en la interfaz, desde las primeras versiones que aparecen en la figura 5.18 donde me comentaron que los degradados no era lo mejor para personas con alguna discapacidad visual, a una segunda versión donde a parte de eliminar el degradado se modificó el color al color que ellos me dijeron el azul, teniendo que modificar todas las imágenes para que los textos estuviesen en blanco 5.19. Y por último, eliminé todos los pixeles blancos de las imágenes a mano, porque al tener las imágenes con un fondo azul los pixeles blancos se ven mucho, esto no lleva a la última versión de la interfaz 5.20.

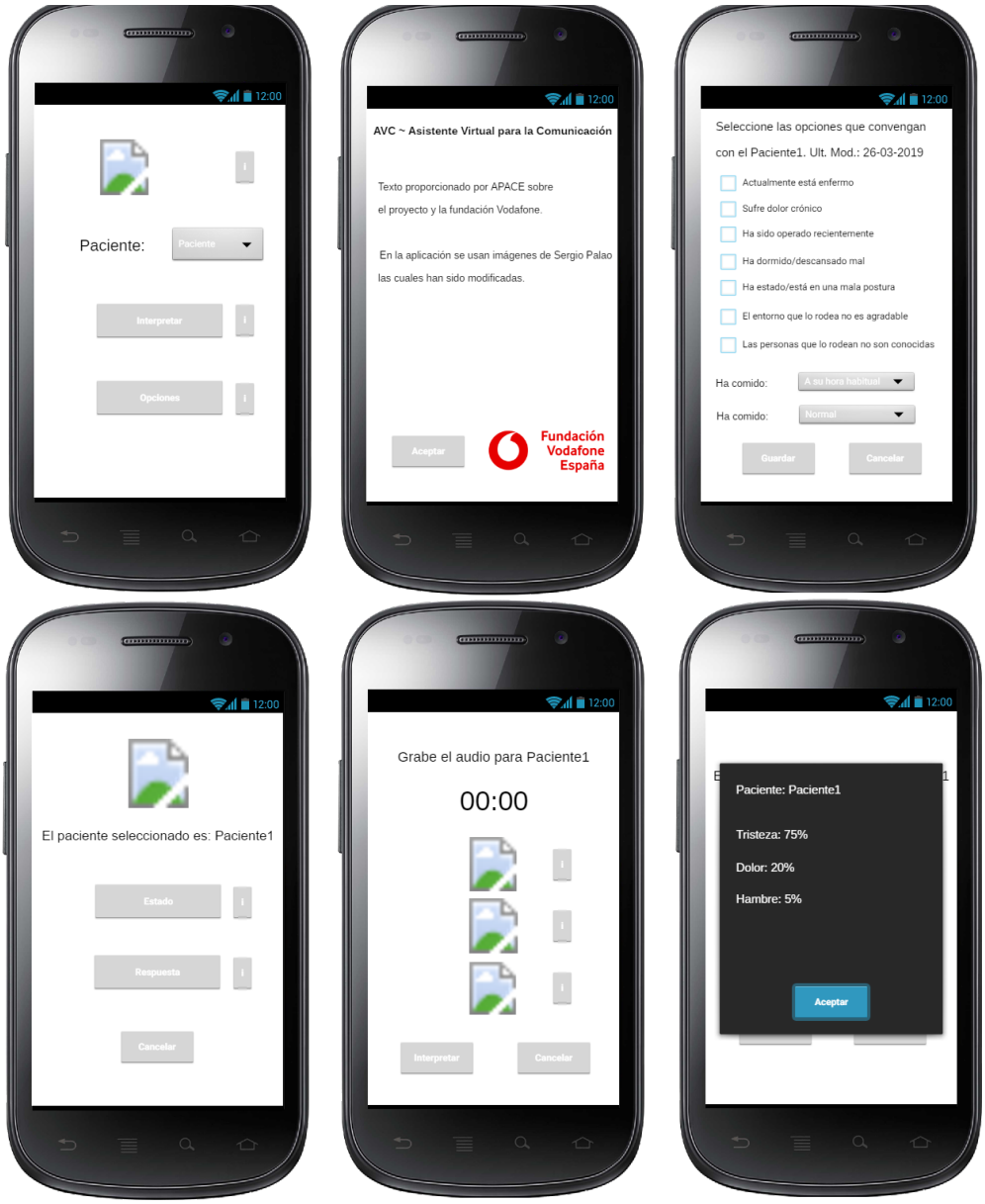

Figura 5.17: Diseño interfaz aplicación AVC.

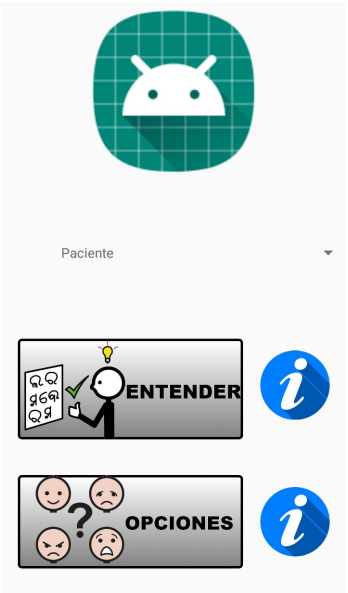

Figura 5.18: Primera versión de la interfaz.

- **Uso de diálogos para mejorar la accesibilidad:** Con *AlerDialog* en los botones de información para mostrar información sobre el botón que está al lado, estos diálogos están escritos en lectura fácil gracias a la Universidad de Salamanca. Además, al abrir el diálogo se reproduce por audio el texto escrito para dar una mayor accesibilidad 5.21.

Como se puede observar, el desarrollo de esta aplicación ha sido orientado casi totalmente en conseguir una mayor accesibilidad. Este punto ha sido tan importante en el equipo de desarrollo porque desde un principio teníamos la idea de que hasta los compañeros de estas personas gravemente afectadas del centro de día de APACE pudiesen utilizarlo, para así poder entender a sus compañeros.

Después de terminar el desarrollo de la aplicación empecé con la fase de test de esta, en ella primero diseñé los test unitarios para probar la lógica y la creación de los distintos elementos de la aplicación y los implementé, y después, usando Espresso [2], diseñé e implementé los test de integración para comprobar la correcta colaboración entre las distintas partes de la aplicación. Un ejemplo de este diseño de los test se puede ver en la figura 5.22 [24, 23, 25].

Tras probar los distintos test descubrí unos cuantos *bugs*, que es la función de los test. Hubo un *bug* en particular que me llevó mucho tiempo resolver y trataba de como se generaba la estructura de datos, ya que hay

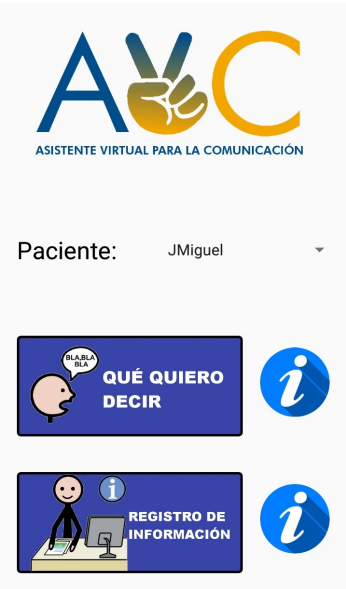

Figura 5.19: Segunda versión de la interfaz.

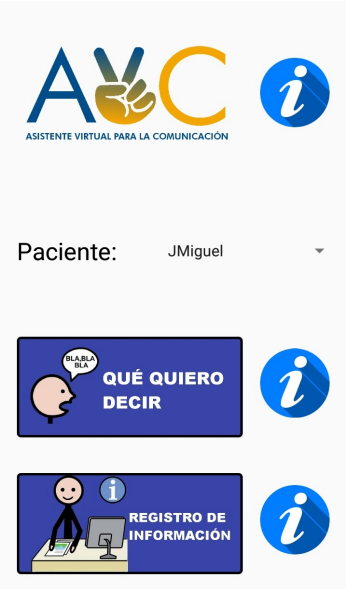

Figura 5.20: Versión final de la interfaz.

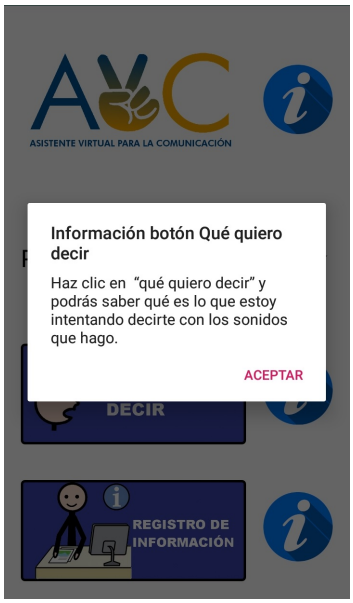

Figura 5.21: Ejemplo de diálogo en lectura fácil.

algunos test que prueban esa funcionalidad eliminando partes o toda la estructura de ficheros de la aplicación.

Además, realicé un tipo especial de test llamado *Monkey Test* [4], que consiste en a través de una prueba de estrés pulsando aleatoriamente puntos de la pantalla probar la consistencia de la aplicación. Gracias a este test descubrí un error importante en la pantalla de grabación de audios, ya que le grabación no se paraba cuando se retrocedía a la pantalla anterior pulsando el botón *back* del dispositivo.

5.7. Desarrollo del servidor

Tras tener una primera versión de la aplicación tocaba empezar a desarrollar el servidor, para ello primero tenía que aprender a usar Flask [12], ya que era la primera vez que iba a desarrollar un servidor.

El primer paso que tuve que dar es el de elegir el tipo de *request* con el cual quería hacer las peticiones al servidor, elegí el tipo *post* porque tenía más conocimiento acerca de este tipo de *request HTTP* y por recomendación de compañeros [16, 17].

El segundo paso fue el diseño de los distintos métodos del servidor. Tras el diseño se quedó en 4 métodos *post* en el servidor:

Identificación	Cliente~OpcionesActivity	**Cliente**	Apace
Proyecto	AVC		
Descripción	Pantalla para modificar las opciones adicionales para el paciente seleccionado en el menú principal.		
Descripción Larga			
El cliente desea poder ver las opciones que están guardadas para un paciente, modificarlas y si quiere guardarlas o cancelar los cambios.			
Pruebas			
• Comprobar texto informativo no nulo: testTextoNoNulo • Comprobar la visibilidad del ImageButton guardar: testVisGua • Comprobar la visibilidad del ImageButton cancelar: testVisCan • Comprobar que el ImageButton guardar no está habilitado al comienzo: testEnGua • Comprobar que el ImageButton cancelar está habilitado al comienzo: testEnCan			

Figura 5.22: Diseño de test unitarios de OpcionesActivity.

- **Nombre:** API/Nombres, **caso de uso:** CU1, **descripción:** Devuelve la lista de usuarios de los que hay un método entrenado preparado para clasificar. Esta lista está en un fichero csv en el servidor, **parámetros:** Token, **salida:** JSON con una cadena con la lista de los nombres.

- **Nombre:** API/ObtOpciones, **caso de uso:** CU2, **descripción:** Devuelve los valores de las opciones adicionales del paciente seleccionado, leyendo el fichero csv del servidor indicado, **parámetros:** Paciente y token, **salida:** JSON con una cadena con la lista de las opciones almacenadas.

- **Nombre:** API/GuaOpciones, **caso de uso:** CU3, **descripción:** Almacena las opciones que se envían desde la aplicación para el paciente seleccionado, **parámetros:** Paciente, lista de valores y token, **salida:** Resultado booleano de la operación.

- **Nombre:** API/Clasifica, **caso de uso:** CU5 y CU6, **descripción:** Ejecuta el modelo entrenado para el paciente pasado con el audio pasado y con las opciones adicionales que están actualmente guardadas en el servidor., **parámetros:** Paciente, tipo de interpretación, audio y token, **salida:** JSON con una cadena con los datos sobre la emoción o respuesta y sus porcentajes.

Después, cree la estructura necesaria en el servidor, esta estructura es:

```
root
├── Modelos
├── Opciones
├── Temp
├── pacientes.csv
├── apiserver.py
└── clip.py
```

En la carpeta *Modelos* es donde se almacenan los distintos modelos ya entrenados. En la carpeta *Opciones* se encuentran todos los csv con las opciones adicionales guardadas para cada paciente. En *Temp* tenemos los ficheros de audio temporales de cada interpretación, estos se eliminarán después de devolver al cliente el resultado. El fichero *pacientes.csv* tiene la lista de pacientes entrenados y los ficheros *apiserver.py* y *clip.py* son los ficheros encargados de la extracción de características y de la clasificación.

Por último, implementé todos los métodos en *Python*, haciendo todas las respuestas con *jsonify* [15]. Algunos aspectos que se pueden destacar del desarrollo del servidor son:

- **Control de concurrencia:** Sobre todo en el método de clasificación donde el servidor recibe varios archivos de audios que tiene que saber identificar correctamente. Esto se ha conseguido gracias a la creación dinámica de nombres para los audios en el servidor.

- **Resultado aleatorio:** A parte de obviamente tener el clasificador para los datos que generé, quise añadir más usuarios en la aplicación para así poder modificar las opciones y comprobar el correcto funcionamiento, pero para ello también necesitaba un clasificador para estos audios. Es por ello que creé una función que genera resultados válidos aleatorios con los cuales responder a las interpretaciones de pacientes no entrenados.

Tras haber implementado cada uno de los métodos tenía que probarlo, pero como la aplicación final no estaba desarrollada la parte de unir ambas partes, aplicación y servidor, tuve que usar una herramienta para poder hacer *request HTTP* desde el ordenador.

5.8. Conexión de aplicación y servidor

Tras tener la aplicación y el servidor desarrollados solo quedaba unir ambas partes, para ello primero tuve que aprender a realizar *post* en *Android*. Decidí hacer los *post* con paso de parámetros por *url*, lo que después me llevó un problema con el audio que contaré posteriormente.

Al implementar los métodos *post* me encontré con un problema. Este problema era que la conexión al servidor, que está desplegado en un portátil, tengo que acceder a través de una dirección IP. Eso cuando lo estaba implementando en casa era sencillo, ya que esta IP apenas cambiaba y solo hay un router en red, pero la presentación del proyecto se iba a hacer en sitios como la universidad, o el mismo APACE, donde nos encontramos con redes más complejas donde apuntar a un dispositivo no es tan fácil como poner la IP, es por ello que decidí usar el programa *noWifi* en el portátil en donde está desplegado el servidor, ya que me permite crear un red a la cual apuntar con los dispositivos *Android* siempre con la dirección 192.168.137.1 y al puerto definido en *Flask*, el puerto 5000.

Al implementar el cuarto método *post* en *Android*, el método que me permite enviar la petición de clasificación me encontré con un gran problema. Este problema era cómo conseguir pasar un archivo de audio en *mp4* a un servidor a través de un método en el cual le paso los parámetros por texto en la *url*. La respuesta la encontré en la codificación a Base64, que me permite transformar mi archivo de audio en texto el cual puedo pasar en la *url* y luego descodificarlo en el servidor haciendo la operación inversa.

Al implementar el paso a Base64 me encontré con otro problema, y es que la codificación que usaba en *Android* no se descodificaba bien en el servidor *Python*. Este problema me llevó mucho tiempo solventarlo, ya que intenté usar muchos métodos para codificar en Base64, pero ninguno daba resultado, todos los métodos que probaba al decodificarlo en el servidor me daba como resultado un fichero corrupto. Además, se junto otro impedimento, y es que la librería de Base64 de *Java* subía el API mínimo varias versiones, por lo que no podríamos cumplir nuestro objetivo técnico [13]. Tras probar muchos métodos pasé directamente a imprimir el resultado de la codificación

para ver si podía verse el problema, y en efecto, no se correspondía con la codificación esperada por el servidor, es entonces cuando cambié el fichero de audio para pasar a codificar un fichero de texto, encontré el método, *Base64.encodeToString* que funcionaba correctamente en *Android* y luego lo probé con el fichero de audio y por fin funcionó.

```
String baudio=null;

try {
        byte[] b = new byte[(int) aufile.length()];
        FileInputStream fis = new FileInputStream(
           aufile);
        fis.read(b);
        baudio = Base64.encodeToString(b, Base64.
           NO_WRAP);
} catch (FileNotFoundException e) {
        e.printStackTrace();
} catch (IOException e) {
        e.printStackTrace();
}
```

En el código se ve como se hace la codificación a Base64, es importante el uso del *Base64.NO_WRAP*, ya que lo que hace es codificarlo todo como una única cadena, ya que otras configuraciones lo que hacen es introducir saltos de línea que impiden la correcta descodificación.

Por último, quise dar una mayor seguridad al servidor, para que no se pueda acceder a una de las peticiones desde fuera de la aplicación, esto lo conseguí añadiendo un *token* de seguridad como parámetro a todos los métodos. Es entonces donde pienso que puede haber una serie de errores con el servidor que se podrían agrupar, codificar, describir y dar una solución. Es así como se me ocurrió hacer una estandarización de los errores de la aplicación en la conexión con el servidor, estos tienen un código asociado, y con el una descripción de por qué ocurre y como solucionarlo 5.23. Un ejemplo de un mensaje de error se puede ver en la figura 5.24.

Después de terminar de enlazar la aplicación de interpretación con el servidor modifiqué y eliminé algunos test que tenía implementados, pudiendo así probar desde los test unitario y de integración el servidor. Al final tengo un total de 82 test, como se puede ver en la figura 5.25, entre unitario y de integración, que prueban diferentes partes de la aplicación y el servidor.

Error	Descripción	Causa	Solución
Er1	No se puede acceder al servidor.	Este error se da cuando el servidor está caído.	Avisar al Administrador para que reinicie el servidor.
Er2	Token de seguridad incorrecto.	O se ha corrompido la aplicación cambiando el token de seguridad o el token de seguridad ha cambiado sin ser actualizado en su dispositivo.	Avisar al Administrador para que le pase de nuevo la aplicación.
Er3	Lista de nombres vacía en el servidor.	La lista con los nombres en el servidor está vacía.	Avisar al Administrador para que recupere el archivo.
Er4	Lista de nombres no encontrada.	El fichero con la lista de los pacientes en el servidor no está.	Avisar al Administrador para que recupere el archivo.
Er5	Opciones del paciente no encontradas.	El fichero con las opciones adicionales del paciente no está en el servidor.	Avisar al Administrador para que recupere el archivo.
Er6	No hay conexión a Internet.	No tener conexión ni Wifi ni Datos Móviles.	Conectarse a una red con acceso a Internet.
Er7	Fichero del paciente vacío.	El fichero del paciente con las opciones está vacío en el servidor.	Avisar al Administrador para que recupere el archivo.
Er8	Fichero de clasificación no existe.	No existe el fichero con el que se clasifica en el servidor a ese paciente.	Avisar al Administrador para que recupere el archivo.
Er9	Resultado vacío.	El servidor ha devuelto una solución vacía.	Avisar al Administrador para que compruebe el funcionamiento de la clasificación para ese paciente.

Figura 5.23: Errores de la conexión aplicación-servidor.

Figura 5.24: Ejemplo de error.

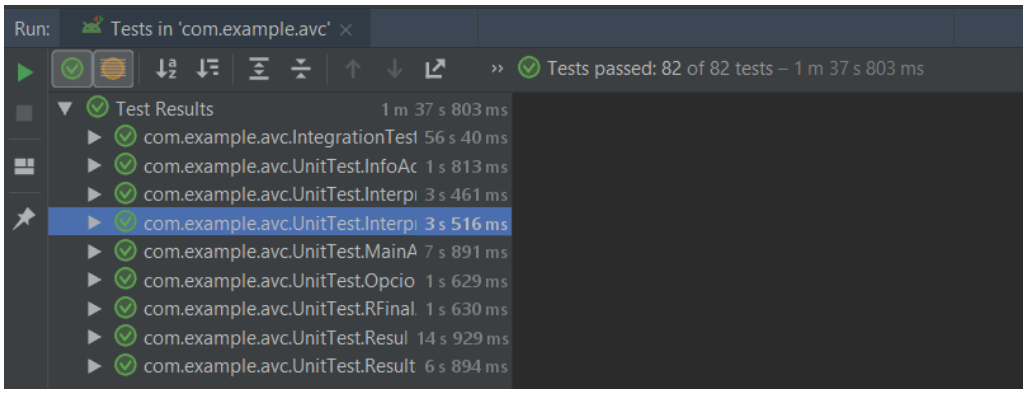

Figura 5.25: Resultado de los 82 test.

Con esto terminé el desarrollo de código del proyecto, pero no el proyecto en general ya que aún quedaba realizar presentaciones y documentarlo.

5.9. Presentaciones y Premios

Desde un principio nos pusimos como objetivo en el proyecto que este fuese mediático, es decir, intentar apuntarnos al mayor número de presentaciones donde poder mostrar los avances del proyecto. A lo largo del desarrollo del proyecto he realizado, a diferentes tipos de públicos, las siguientes presentaciones:

- Varias presentaciones en APACE de la aplicación de recogida de datos a los familiares y cuidadores del centro.
- Presentación de la aplicación final en las jornadas ASPACEnet en Madrid, 28 de Mayo de 2019.
- Presentación para la elaboración del vídeo final del proyecto, https://www.youtube.com/watch?v=6lWLMS9o8v0&feature=youtu.be.
- Presentación de la aplicación final a los cuidadores del centro.
- Presentación ante la prensa del proyecto y de la aplicación 5.26, https://www.diariodeburgos.es/Noticia/Z2D22B97F-0523-981F-1046FF5A06C5B81F/201906/Una-aplicacion-para-entender-a-quienes-no-pueden-hablar.

El proyecto ha sido galardonado con los premios de Fundación Vodafone y de Becas Prototipo.

Figura 5.26: Imagen de la presentación ante la prensa, fuente *Diario de Burgos*.

Trabajos relacionados

Dentro del proyecto podemos diferenciar dos tipos de trabajos relacionados, unos que son orientados a la comunicación de personas con parálisis cerebral, y otro los trabajos relacionados con la clasificación de audios.

Sin embargo, durante el proyecto, no hemos encontrado ningún trabajo similar al nuestro, es decir, que auné los dos tipos de posibles trabajos relacionados, un clasificador de audios para la mejora en la comunicación de personas con parálisis cerebral.

6.1. Herramientas para la comunicación

Como ya he comentado en esta memoria, existen algunos programas y aplicaciones que facilitan la comunicación a las personas con parálisis cerebral, algún ejemplo de estas aplicaciones es *Jocomunico* [14]. En esta aplicación las personas con parálisis cerebral, no gravemente afectadas, pueden seleccionar distintos botones con pictogramas accesibles, que al pulsarlos reproduce el sonido relacionado con la imagen. Este tipo de aplicaciones las he podido ver en uso en el centro de APACE Burgos, son unas herramientas de precio elevado, pero que cumplen su correcta función, ya que permite a una persona con parálisis cerebral con un a tablet poder comunicarse.

Pero existe un problema con este tipo de herramientas, se necesita que la persona con parálisis cerebral pueda pulsar de alguna forma la pantalla del dispositivo, esto como puede ver en la conferencia de ASPACENet en Madrid se puede hacer de diferentes formas, ya que capa paciente puede hacer y moverse de formas distintas, hay algunas personas que dirigen sus dispositivos con la nariz, otros con un *joystick*, otras personas lo hacen con la barbilla... Pero ninguna con ninguna de estas formas puede usar una

persona gravemente afecta un dispositivo móvil, es por ello que este tipo de herramientas no sirven para estas personas.

6.2. Clasificadores de audios

Los trabajos relacionados de clasificadores de audios son más comunes, debido a que se pueden aplicar a muchos tipos de problemas, desde clasificación de llantos de bebés para interpretar que les pasa, hasta clasificación de sonidos de animales. Como el conjunto de trabajos relacionados de clasificación de audios es muy abundante voy a pasar a explicar alguno de ellos, de los que más nos han llamado la atención o de los que hemos usado tanto el investigador colaborador, Sergio Chico, como yo.

Detección de sonidos de animales

En este proyecto [35] se trata de identificar el tipo de animal a partir del audio. Como se comenta en el artículo, ellos tuvieron el mismo problema que nosotros, la falta de datos.

En este proyecto se usan técnicas parecida a las que usamos, en concreto en la técnicas de extracción de características del audio con el uso del *Zero Crossing Rate* 3.2 y *MFCC* 3.2. Primero lo que hacen es detectar el punto de la grabación donde empieza el sonido, gracias *Zero Crossing Rate*. Después de obtener el punto de inicio de la grabación se pasa a la extracción de característica con *MFCC*, esta se divide en 5 procesos, algo diferentes a los nuestros:

- División del audio en fragmentos (*frames*).
- Minimizar las discontinuidades de la señal en el comienzo y final de cada fragmento.
- Transformada rápida de *Fourier* para obtener el dominio de frecuencias a partir del temporal.
- Obtenemos el espectro de frecuencias de *Mel*.
- Aplicar el logaritmo a las frecuencias de *Mel* para volver al dominio de tiempo.

Es en el punto de la clasificación donde encontramos la mayor diferencia entre ambos trabajos. Nosotros usamos para la clasificación, después de la

extracción de características, un *Random Forest*, mientras que ellos usan DTW (*Dynamic Time Warping*), un algoritmo que nos permite ver las similitudes entre dos señales de audio no necesariamente del mismo tiempo de duración. Este algoritmo funciona colocando en una tabla los datos de ambas señales, siendo la primera colocada en la primera columna y la segunda en la última fila, y en el resto de celdas de la tabla se calculan con la siguiente fórmula:

$$(x[t] - y[t])^2$$

Tras obtener todos los valores de la tabla se calcula la mejor ruta, por mínimo coste de ruta, para ir desde la posición superior izquierda de la tabla hasta la inferior derecha, es decir, de esquina a esquina de la tabla.

Es esta ruta a través de la tabla la que representa la similitud entre las dos señales.

Como se puede ver, es un algoritmo bastante diferente de algoritmo, ya que nosotros entrenamos un *ensemble* que forma un bosque de árboles de decisión que al clasificar un nuevo dato lo que hace es pasar este nuevo valor por cada uno de los árboles que conforman el bosque y se vota la salida, siendo la más votada la que devuelve el *ensemble*. Sin embargo, aquí lo que hacemos es comparar la nueva señal de audio con DTW con cada una de las posibles señales.

ESC-50

Este es otro de los proyectos en los cuales nos hemos inspirado para desarrollar el nuestro. ESC-50 es un *dataset* en el cual encontramos más de 2000 sonidos de diferentes categorías, como animales y sonidos humanos, con cada una de las categorías 10 tipos distintos de sonidos, como por ejemplo dentro de la categoría de animales tenemos perros, gatos, ovejas... El repositorio es https://github.com/karoldvl/ESC-50.

Este es un *dataset* muy útil ya que nos permite dentro del mismo repositorio ver qué tipos de clasificadores funcionan mejor con qué categoría de audios. Además, dentro del repositorio podemos acceder a los artículos donde se estudian estos rendimientos de los diversos métodos clasificadores.

Cabe destacar que el método que mejores resultados da es usar una red neuronal convolucionales.

Gracias a este repositorio, mi compañero, Sergio Chico, ha podido observar qué métodos son buenos para otros problemas de clasificación de audios.

Conclusiones y Líneas de trabajo futuras

En este apartado se va comentar las conclusiones del proyecto, además de una explicación sobre las mejoras que se podrían realizar sobre éste.

7.1. Conclusiones

El proyecto merece la pena porque permite comunicarse con personas gravemente afectadas que sin este no podrían, sobre todo en los aspectos más relevantes que son la interpretación de emociones negativas que nos permiten saber cuando le ocurre algo malo al paciente, que por desgracia, no nos lo puede decir por su propia voz.

El proyecto ha sido complicado, personalmente el proyecto más costoso en cuanto a tiempo requerido que he hecho, pero tanto desde el equipo de desarrollo de la Universidad de Burgos como desde APACE Burgos estamos contentos con el desenlace de este.

En primer lugar porque hemos cumplido todos los objetivos que nos habíamos propuesto al comienzo del proyecto. Al menos todos los objetivos que se ven mostrados en este documento, es decir, los objetivos que dependían directamente de mi trabajo.

Como comento en el siguiente apartado en las posibles líneas futuras del trabajo, algo que me hubiera gustado probar es el funcionamiento de la aplicación de interpretación con datos de pacientes reales, pero no dependía de nosotros el poder generar los datos suficientes y tampoco se podía forzar a los pacientes a hacer esos sonidos, principalmente porque las emociones

con las que hemos trabajado son negativas, ya que tanto de APACE como desde la universidad nos parecían más importantes de interpretar.

En segundo lugar, hemos podido durante todo el proyecto contar con un punto muy importante, las opiniones de los usuarios finales, es decir, opiniones de padres y madres, de los cuidadores, de los compañeros... Esta es la opinión que realmente nos interesaba. Tanto en las presentaciones a usuarios de la aplicación de generación de datos, como las presentaciones de laa aplicación de interpretación he obtenido muy buenas críticas, sobre todo en los apartados hacia los que enfocamos el proyecto, la simplicidad y la accesibilidad.

7.2. Líneas de trabajo futuras

Como ya he comentado en el apartado anterior, he cumplido casi todos los objetivos que tenía en este proyecto, aun así siempre hay puntos donde se puede mejorar. Algunas de estas mejoras son:

- Uso de otro tipo de clase para la grabación en *Android*, aunque *MediaRecorder* no me ha dado malos resultados, siempre es mejor usar un grabación sin pérdidas.
- Poder tener más datos con los que poder realmente hacer un estudio de la clasificación real.
- Poder incluir otro clasificador que nos permita identificar el paciente a partir del audio si tener que ser seleccionado a mano.
- Mejorar el aspecto visual de la aplicación de generación de datos para que tenga una mayor accesibilidad.
- Poder realizar un estudio, con los datos suficientes, para saber si se puede generalizar el problema y así no tener que hacer un clasificador por cada paciente.
- Poder modificar la aplicación para que se asocie a un paciente y grabe siempre que este emite algún sonido.
- Poder usar la aplicación con personas con otras discapacidad y/o enfermedades.
- Poder realizar una aplicación web para poder añadir y eliminar pacientes.

Bibliografía

[1] Android. Android studio. https://developer.android.com/studio/features.

[2] Android. Crear pruebas de iu con la grabadora de pruebas espresso. https://developer.android.com/studio/test/espresso-test-recorder?hl=es-419.

[3] Android. Mediarecorder.audioencoder. https://developer.android.com/reference/android/media/MediaRecorder.AudioEncoder.

[4] Android. Ui/application exerciser monkey. https://developer.android.com/studio/test/monkey.

[5] Charo Ariza. Parálisis cerebral infantil. https://crene.es/paralisis-cerebral-infantil/, jun 2019.

[6] ASPACE. Algunos datos. https://aspace.org/algunos-datos, jun 2019.

[7] ASPACE. Aspace parálisis cerebral. https://aspace.org/, jan 2019.

[8] ASPACE. Qué es la parálisis cerebral. https://aspace.org/que-es, jan 2019.

[9] Apace Burgos. Sobre apace burgos. http://www.apaceburgos.com/.

[10] Equipo de colaboradores y profesionales de la revista ARQHYS.com. Señales analógicas. *Revista ARQHYS*, dec 2012.

[11] Ayuntamiento de Granada. Análisis espectral: ¿que es y que se representa en las gráficas? https://www.granada.

org/inet/sonidos.nsf/d483b298c3f6a1b9c1257cdd00384c53/3fdfc36a7489b607c1257cde0024bb34!OpenDocument, apr 2014.

[12] Flask. Flask web development, one drop at time. http://flask.pocoo.org/.

[13] Java. Class base64.encoder. https://docs.oracle.com/javase/8/docs/api/java/util/Base64.Encoder.html.

[14] Jocomunico. Herramienta de comunicación. https://jocomunico.com/#/home, jun 2019.

[15] kite. jsonify. https://kite.com/python/docs/flask.jsonify.

[16] Raul Marticorena. Introducción a los sistemas distribuidos. Apuntes Sistemas Distribuidos, Universidad de Burgos.

[17] Raul Marticorena. Programación en servidor. Apuntes Sistemas Distribuidos, Universidad de Burgos.

[18] musiki. Espectrograma (sonograma) — musiki,. http://musiki.org.ar/index.php?title=Espectrograma_(sonograma)&oldid=80207, 2019. [En línea; consultado el 14-junio-2019].

[19] NICHD. Causas parálisis cerebral. https://www1.nichd.nih.gov/espanol/salud/temas/cerebral-palsy/informacion/Pages/causas.aspx, jun 2019.

[20] OpenCSV. Opencsv users guide. http://opencsv.sourceforge.net/.

[21] Yagya Raj Pandeya and Joonwhoan Lee. Domestic cat sound classification using transfer learning. *International Journal of Fuzzy Logic and Intelligent Systems*, 18(2):154–160, 2018.

[22] John R Pierce and A Michael Noll. *Señales: la ciencia de las telecomunicaciones*. Reverté, 1995.

[23] Pedro Renedo. Pruebas de caja blanca. Apuntes Validación y Pruebas, Universidad de Burgos.

[24] Pedro Renedo. Pruebas de caja negra. Apuntes Validación y Pruebas, Universidad de Burgos.

[25] Pedro Renedo. Pruebas de integración. Apuntes Validación y Pruebas, Universidad de Burgos.

[26] Juan José Rodríguez. Clasificación bayesiana, basada en instancias y por combinaciones. Apuntes Minería de Datos, Universidad de Burgos.

[27] Juan José Rodríguez. Introducción a la minería de datos. Apuntes Minería de Datos, Universidad de Burgos.

[28] Peter Rosenbaum, Nigel Paneth, Alan Leviton, Murray Goldstein, Martin Bax, Diane Damiano, Bernard Dan, and Bo Jacobsson. A report: the definition and classification of cerebral palsy april 2006. *Developmental medicine and child neurology. Supplement*, 109:8–14, 2007.

[29] T Villa-Canas, E Belalcazar-Bolaños, S Bedoya-Jaramillo, JF Garces, JR Orozco-Arroyave, JD Arias-Londono, and JF Vargas-Bonilla. Automatic detection of laryngeal pathologies using cepstral analysis in mel and bark scales. In *2012 XVII Symposium of Image, Signal Processing, and Artificial Vision (STSIVA)*, pages 116–121. IEEE, 2012.

[30] Wikipedia. Códec de audio — wikipedia, la enciclopedia libre. https://es.wikipedia.org/w/index.php?title=C%C3%B3dec_de_audio&oldid=114820037, 2019. [Internet; descargado 13-junio-2019].

[31] Wikipedia. Formato de archivo de audio — wikipedia, la enciclopedia libre. https://es.wikipedia.org/w/index.php?title=Formato_de_archivo_de_audio&oldid=116651731, 2019. [Internet; descargado 13-junio-2019].

[32] Wikipedia. Mfcc — wikipedia, la enciclopedia libre. https://es.wikipedia.org/w/index.php?title=MFCC&oldid=113070764, 2019. [Internet; descargado 14-junio-2019].

[33] Wikipedia contributors. Sampling (signal processing) — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Sampling_(signal_processing)&oldid=885036021, 2019. [Online; accessed 14-June-2019].

[34] ASPACE Castilla y León. Qué es la parálisis cerebral. https://www.federacionaspacecyl.org/quienes-somos/que-es-la-paralisis-cerebral/, jan 2019.

[35] Che Yong Yeo, SAR Al-Haddad, and Chee Kyun Ng. Animal voice recognition for identification (id) detection system. In *2011 IEEE 7th International Colloquium on Signal Processing and its Applications*, pages 198–201. IEEE, 2011.