Universidad de San Andrés

Maestría en Ciencia de Datos

Tesis de Maestría:

Footprint y clasificación de señales de audio para identificación de hablantes

Tesis presentada para optar al tıtulo de Magíster de la Universidad de San Andrés en el área Ciencias de Datos

Ing. Martin ignacio Pastorino

Directores de tesis:

- Dr. Facundo Carrillo
- Dr. Pablo Riera

Buenos Aires 2023

Índice general

Lista de figuras **3**

Lista de tablas **5**

1. Introducción **1**

1.1. La señal de habla 4

1.2. El reconocimiento automático de hablantes 6

1.3. El problema 8

1.4. Estructura de la tesis 9

2. Marco Teórico **11**

2.1. Las señales y los sistemas 11

2.2. La transformada de Fourier 13

2.2.1. La transformada discreta de Fourier 15

2.2.2. El espectrograma 16

2.3. Los Coeficientes Cepstrales Mel 17

2.4. Aprendizaje supervisado 18

2.4.1. Modelos de clasificación 19

2.4.2. Metodología de validación 24

2.4.3. Medidas de rendimiento 25

2.5. Feature embeddings 29

2.6. Reducción de dimensionalidad 31

2.6.1. Análisis de componentes principales 32
2.7. Redes Neuronales artificiales 34

3. De los modelos gaussianos a los i-vectors **36**

4. Redes Neuronales en la detección de hablantes **41**

5. La extracción de características **46**

6. Los datos **49**
6.1. La calidad de los datos 50
6.2. Construcción y estructura de los datos 52

7. Metodología **56**
7.1. Descripción general del modelo 56
7.2. Preprocesamiento 58
7.2.1. Filtro de silencios 59
7.2.2. Filtro de preénfasis 61
7.2.3. Filtro de Ruido de fondo 63
7.3. Speaker embedding 64
7.3.1. Construcción de embeddings 65
7.3.2. Reducción de dimensionalidad 67
7.4. Backend 70
7.4.1. Clasificación 71

8. Resultados **76**
8.1. Matrices de confusión 76
8.2. Indicadores de bondad de ajuste 78

9. Discusión **80**

Índice de figuras

1.1. Espectro de una señal de habla 5
1.2. Verticales en el reconocimiento de hablantes 7

2.1. Ejemplo de configuración en serie y paralelo de un sistemas discreto 13
2.2. Espectrograma de una señal modulada 17
2.3. Clasificador K - Nearest Neighbor 20
2.4. Clasificador Support Vector Machine 21
2.5. Clasificador Random Forest 24
2.6. Matriz de confusión binaria 26
2.7. Ejemplo de un modelo de generación de embeddings 30
2.8. Ejemplo de una proyección por PCA 33

3.1. Ajuste de GMM-UBM al conjunto de muestras y creación de supervectors 39

4.1. Arquitectura de red neuronal generadora de d-vectors 43
4.2. Arquitectura de red neuronal generadora de x-vectors 44

6.1. Distribución de clases (hablantes) en el conjunto de datos 53
6.2. Forma de onda de señales de habla masculina y femenina 54
6.3. Espectrograma de señales de habla masculina y femenina 55

7.1. Diagrama en bloques del sistema identificador de hablantes 58
7.2. Efecto del filtrado de silencios sobre una señal de habla 61
7.3. Efecto del filtrado de preénfasis sobre una señal de habla 62

7.4. Efecto del filtrado de ruido de fondo sobre una señal de habla 64
7.5. Diagrama en bloques del proceso de generación de embeddings 66
7.6. Construcción de embedding a partir de una señal de habla 67
7.7. Primeros dos componentes principales de un embedding 69
7.8. Reducción de dimensionalidad aplicada a un embedding 70

8.1. Matrices de confusión para clasificadores SVM y RFC 77
8.2. Matrices de confusión para clasificadores KNN 78
8.3. Matrices de confusión 79

Índice de cuadros

8.1. Indicadores de bondad de ajuste 79

Resumen

Con el auge de los medios de comunicación y las aplicaciones de redes sociales y mensajería, la producción de contenido audiovisual ha crecido exponencialmente, generando un incrementado en el interés de las organizaciones en el uso de sistemas computarizados para la interacción con sus clientes. En este contexto, el trabajo de tesis que se presenta explora la viabilidad de utilizar el habla como fuente de información para identificar a una persona, profundizando así en el análisis del habla y en las propiedades que hacen que esta sea única para cada individuo.

Para llevar a cabo este análisis, se utilizaron diferentes técnicas de procesamiento de señales con el objetivo de analizar cómo se compone y cómo se comporta una señal de habla a lo largo del tiempo. Posteriormente, se estudió la disciplina que investiga la detección de hablantes a través de su voz, en donde se analizo una diversidad de artículos escritos por investigadores con amplia trayectoria. Una vez comprendidas las diferentes áreas de investigación que la componen, se decidió enfocar el problema hacia la identificación de hablantes sin la presencia de un texto predefinido y, asumiendo que los hablantes -a detectar- se encuentran en el conjunto de entrenamiento. Luego de definir el problema, se procedió a investigar sobre las técnicas utilizadas en la detección de hablantes como son los modelos probabilísticos no supervisados, vectores de características y redes neuronales. Estas, requieren de una etapa previa de extracción de características de las señales de voz, por lo que se indagó en técnicas como los bancos de filtros y, un enfoque distinto, el cual utiliza redes neuronales llamadas SincNets.

Para la construcción de sistemas basados en *machine learning*, es imprescindible la creación de un conjunto de datos que permita llevar a cabo las etapas de entrenamiento y validación que le corresponde a cada modelo. Además, es de gran importancia que estos datos reflejen la realidad de las personas en su día a día y, de esta forma, construyan un sistema útil que resuelva los desafíos actuales de los usuarios. Por lo tanto, se focalizó en los dispositivos móviles y, en específico, en los audios de WhatsApp, ya que son una fuente de datos representativa de

las problemáticas presentes en las señales de habla generadas, recolectadas y enviadas a través de medios de comunicación.

Una vez corroborada la factibilidad de utilizar *machine learning* para la detección de hablantes y obtenida la fuente de datos de entrenamiento y validación, se decidió avanzar hacia la construcción de un sistema que valide dicha hipótesis. La estructura se describe en el capítulo Metodología, donde se destaca la utilización de SincNets como técnica de extracción de características y el análisis de tres modelos de *machine learning* en búsqueda de determinar cuál presenta un mejor rendimiento. El modelo K-Nearest Neighbor con distancia coseno, fue el elegido puesto que arrojó un Fβ mayor al 99 %. Frente a estas evidencias, es que la investigación concluye que el habla humana es rica en información y se valida la gran capacidad que poseen estos sistemas automáticos basados en *machine learning* para detectar la identidad de hablantes, ofreciendo así un gran potencial en la generación de beneficios para la sociedad.

Capítulo 1

Introducción

La capacidad de reconocer personas a través de su voz es un rasgo adquirido desde las primeras etapas en la formación de un ser humano. Según estudios realizados en trabajos como *Human voice perception* [30], *Effects of experience on fetal voice recognition* [27] y *Understanding Voice Perception* [6], se ha demostrado que los seres humanos somos capaces de reconocer la voz de nuestras madres desde la etapa fetal y, además, podemos distinguir eficientemente entre sonidos con y sin habla desde una etapa muy temprana de nuestra vidas. Esta cualidad de reconocer seres humanos por su voz resulta trivial para la mayoría de las personas en su vida cotidiana, pero acarrea numerosas limitaciones cuando se debe generalizar sobre hablantes no tan familiares o comienzan a accionar factores externos sobre la comunicación.

Al momento de entablar un diálogo sobre un medio de comunicación, como por ejemplo una llamada telefónica, la primer acción que realiza una persona es la identificación de su interlocutor y, en caso de no estar seguro, generalmente solicita algún tipo de verificación adicional para continuar con el dialogo. Esto, hace que la mayoría de las personas realicen diariamente reconocimientos de hablantes con un alto grado de precisión, especialmente cuando se interactúa con conocidos cercanos o figuras públicas. Tal es el grado de precisión que puede ostentar una persona ordinaria frente a un hablante conocido que incluso una pequeña declaración como una risa, un llanto o un grito es suficiente para efectuar una correcta identificación [17]. En contraparte, cuando se está en presencia de un hablante no conocido (se lo ha escuchado pocas

veces) o el medio de comunicación afecta en gran medida la calidad de la voz, la detección de su identidad adquiere una gran dificultad tanto para personas comunes como para especialistas; dando lugar al surgimiento de sistemas computarizados que solventan estas limitaciones y permiten construir sistemas robustos de reconocimiento automático de hablantes.

Los sistemas de reconocimiento automático de hablantes enfrentan nuevos desafíos en comparación con otras formas biometría (huellas dactilares, iris, rasgos faciales) ya que en el habla la información sobre la identidad del hablante se encuentra contenida en la forma en que se habla y no su contenido. Esto genera que las muestras de un individuo posean un alto grado de variabilidad ya que en ocasiones una persona no repite la misma frase de la misma manera, generando un efecto conocido como *style shifting* o *intraspeaker variability* [16]. Además, varios dispositivos de grabación y métodos de transmisión exacerban el problema, por ejemplo, puede resultar difícil reconocer la voz de alguien a través de un teléfono o, quizás, cuando la persona está enferma, susurrando o gritando. Teniendo esto en cuenta, se pueden agrupar a estos tipos de variabilidades en dos grandes grupos, los cuales son:

- Basado en el hablante: Es la variabilidad intrínseca al hablante y consiste en cambios en la forma en que produce el habla, como por ejemplo: el sujeto se encuentra realizando alguna tarea mientras habla (Ejem: maneja un auto), altera su fonación normal (Ejem: susurra [18]), altera su habla en presencia de ruido (Ejem: efecto Lombard [21]), posee alguna enfermedad, altera intencionalmente su voz, entre otros.

- Basado en la tecnología o externas: Abarca el cómo y el dónde se captura los audios que contienen las señales de habla, como por ejemplo: problemas en el canal de transmisión (Ejem: telefonía celular o fija [42], tipo de micrófono, ruido de fondo [43], reverberación), la calidad de datos (Ejem: frecuencia de muestreo, compresión de audio), entre otros.

En la actualidad existen diversos métodos que mitigan la variabilidad y degradación de las señales debido a factores externos, como por ejemplo los filtros adaptados y ecualizadores, los cuales aminoran el efecto nocivo del ruido aditivo y la variabilidad del canal de transmisión. Respecto a la variabilidad intrínseca del habla, esta es más difícil de solventar ya que resulta

complejo cuantificarla debido a que la voz de una persona puede cambiar por su estado de salud o por envejecimiento [25] [26]. Otro aspecto a tener en cuenta son las acciones que son naturales en un humano pero que para los sistemas de reconocimiento automático les presentan un desafío, como por ejemplo la identificación de segmentos del habla dentro de un audio. Las personas pueden distinguir de forma eficiente y desde una edad muy temprana [6] los sonidos que pertenecen a un diálogo y los que no y, además, pueden discriminar con relativa facilidad distintas voces dentro de una conversación. Estas capacidades de reconocimiento de hablantes conocidos y discriminación de voz de hablantes desconocidos son acciones cognitivas separadas [52], lo que le permite al ser humano paralelizar estas tareas; premisa que no suele ser cierta para los sistemas automáticos.

Más allá de las ventajas que presentan las personas en la identificación de hablantes, existen ciertas limitaciones en donde los sistemas de reconocimiento automático se vuelven robustos. Las personas son susceptibles a los sesgos contextuales [24] por ejemplo, si un individuo se atribuye una declaración es posible que se tienda a encontrar coincidencias entre la voz escuchada y la identidad asumida. Otra limitante es la poca capacidad de una persona para recordar la voz de un hablante durante extensos lapsos de tiempo [35], ya que la capacidad de retención de la memoria depende de la edad y de la cantidad de veces que el oyente ha escuchado dicha voz [13]. Es probable que luego de un cierto tiempo sin escuchar una determinada voz, esta pueda resultar familiar pero sin poder identificar correctamente quién es el hablante. La atención es otro factor importante ya que los humanos se desempeñan mejor mientras están atentos, sin embargo, el nivel de atención de una persona cae con el tiempo y los oyentes tienden a fatigarse después de cierto lapso. Finalmente, la familiaridad con la fonología del idioma en que fue dicha una frase también puede influir en la precisión con la que se reconoce un hablante [38].

Sobre estas limitaciones presentes en los seres humanos y, gracias a los continuos avances en el desarrollo de algoritmos de *machine learning*, es que los sistemas automáticos empiezan a tomar gran relevancia en el ámbito de reconocimiento de hablantes. Así como una persona requiere escuchar sucesivamente a un hablante para poder reconocerlo, los algoritmos de *machine learning* realizan un proceso similar ya que necesitan ser entrenados con diferentes muestras

de habla para poder extraer patrones que las distingan. Estos entrenamientos persistirán en el tiempo y, gracias a la implementación de un sistema de monitoreo continuo, será posible realizar nuevos entrenamientos cuando se detecten degradaciones en sus inferencias; generadas por cualquiera de las variabilidades antes mencionadas. Finalmente, es de vital importancia que los sistemas estén compuestos por diferentes etapas que permitan adecuar la señal con el objetivo de amainar las impurezas acarreadas desde su recolección (Ejem: ruido o reverberación) y, también, la eliminación de contenido con poca información (Ejem: silencios). Una vez transitada esta etapa, la nueva señal podrá ser procesada con mayor facilidad por diferentes modelos definidos aguas abajo.

1.1. La señal de habla

Cada hablante tiene ciertos rasgos característicos en su habla que la hacen única. Estas características individuales pueden no ser tan fácilmente distinguibles, pero son únicas principalmente debido a dos elementos: la fisiología del tracto vocal y los hábitos aprendidos en su articulación. Según estudios realizados por [33] y [51] se puede observar que incluso los gemelos idénticos con grandes similitudes en su tracto vocal y propiedades acústicas, poseen diferencias en sus voces. Por lo tanto, ya sea que el reconocimiento sea realizado por humanos (un oyente experto o no) o por sistemas automáticos, es importante definir aspectos medibles y predefinidos sobre las señales de habla para así poder realizar comparaciones entre ellas.

Si nos adentramos un poco en la construcción de la señal de habla podemos ver que esta comienza por la apertura y cierre de la glotis generando ondas sonoras periódicas, las cuales son filtradas por el tracto vocal creando una amplia variedad de sonidos entre los que se destacan las vocales y las consonantes. En el dominio frecuencial, las primeras se encuentran compuestas por una frecuencia fundamental y sus armónicos, los cuales extienden su rango espectral hasta aproximadamente los 8 KHz; mientras que las segundas no tienen fundamental, son sonidos no periódicos y su rango espectral puede extenderse mas allá de los 8 KHz, un ejemplo son los casos de la S o F. En lo que respecta a las vocales, tanto la frecuencia fundamental como los

primeros formantes son de gran importancia. La primera tiene su origen en la glotis y brinda información sobre la velocidad a la que vibran las cuerdas vocales y la segunda, se corresponde con los tonos que han recibido un énfasis al pasar por el tracto vocal. Existen una gran cantidad de formantes pero se considera que sólo los 3 primeros (asociadas a la cavidad oral, bucal y nasal) proporcionan la suficiente cantidad de información para poder diferenciar los distintos tipos de sonidos.

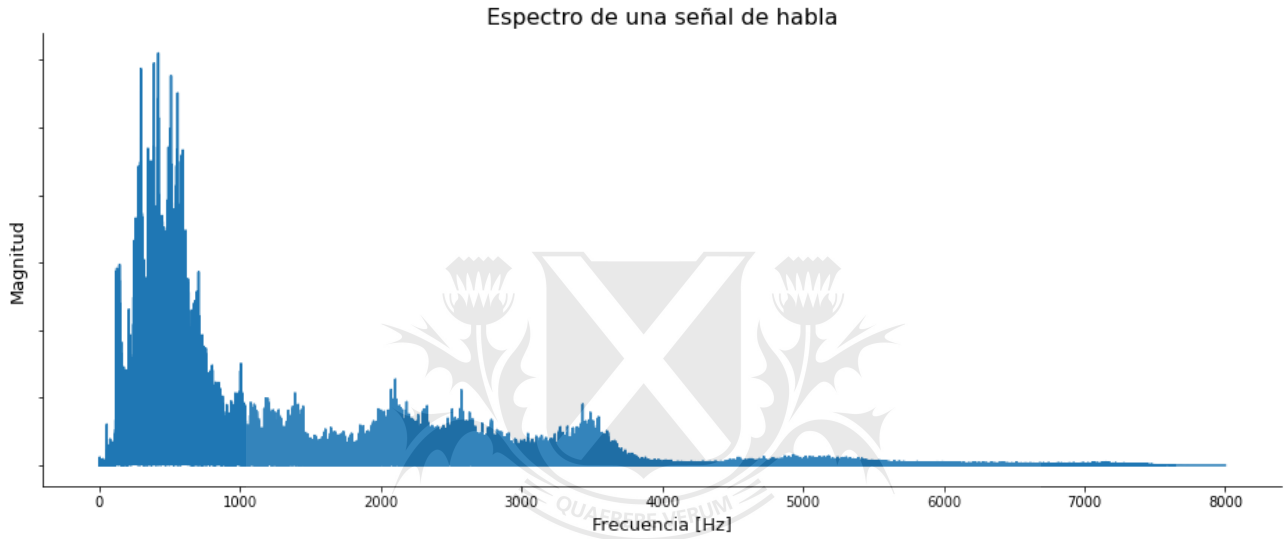

Figura 1.1: Espectro de una señal de habla

Se esperaría que una voz única deba tener todas sus características únicas pero esto no es siempre cierto. Por ejemplo, dos hablantes diferentes pueden tener la misma velocidad de habla, la cual es un *feature* de análisis totalmente válido pero, a su vez, poseer diferencias en sus tonos promedio. Dentro del universo de *features* que se pueden obtener de una señal de habla, se pueden clasificar en dos grandes familias: Bajo nivel y Alto nivel. Las primeras se corresponden con aspectos básicos de la acústica, generalmente relacionados con la fisiología del tracto vocal, como puede ser la frecuencia fundamental y los formantes; mientras que las segundas se encuentran relacionadas con los hábitos y el estilo aprendidos de un hablante, como es el uso particular de palabras o el idiolecto (manera particular de un individuo de hablar una lengua). Este universo de características hacen al habla una excelente fuente a partir de la cual se puede extraer la identidad de una persona. No obstante, es importante

tener en cuenta que aunque un oyente entrenado perciba dos voces similares, estas pueden provenir de dos hablantes distintos. Ello ocurre debido a que la posibilidad de llevar a cabo una correcta distinción aumentará ante la mayor cantidad de *features* que se tengan a disposición al momento de la detección. Esto, sumado a la variabilidad y degradaciones que sufren las señales de voz, complejiza aun mas la detección de hablantes por parte de otras personas.

Sobre las dificultades que se le presentan a una persona para detectar diferencias entre dos voces muy parecidas es que se abre un abanico de oportunidades para los sistemas automáticos de detección de hablantes. Ellos, gracias a su gran capacidad para detectar, extraer y comparar características, junto con las propiedades únicas que posee el habla de cada persona permiten que estas señales sean un excelente biomarcador para el reconocimiento de un individuo.

1.2. El reconocimiento automático de hablantes

La señal del habla transmite diferentes tipos de información, principalmente un mensaje hacia otro individuo, pero también acarrea información de la persona que se ha enviando ese mensaje. Mientras que el área de reconocimiento de voz o *speech recognition* se ocupa de extraer el mensaje lingüístico subyacente en un enunciado, el área de reconocimiento del hablantes o *speaker recognition* se ocupa de extraer la identidad de la persona que pronuncian dicho enunciado. A medida que la interacción a través del habla con sistemas computarizados se vuelve cada vez más frecuente, también aumenta la importancia de poder reconocer automáticamente a un hablante basándose en sus características vocales.

Adentrándonos en el reconocimiento de hablantes podemos observar que este es un área del procesamiento del habla que consiste en la construcción de un modelo que permita la identificación de un hablante a través de su señal de voz. Esta área de investigación es utilizada en variadas aplicaciones del mundo real como la autenticación en dispositivos inteligentes, validación por voz para transacciones bancarias o en investigaciones policiales [23]. Las características únicas que presentan las señales de habla producidas por distintos individuos le permiten a las computadoras la capacidad de identificar patrones únicos en la voz, llamado *Voice Footprint*.

Dentro de *Speaker Recognition* es posible evidenciar dos grandes verticales o campos de aplicación llamados Identificación del hablante o *Speaker Identification* (SI) y Validación del hablante o *Speaker Validation* (SV).

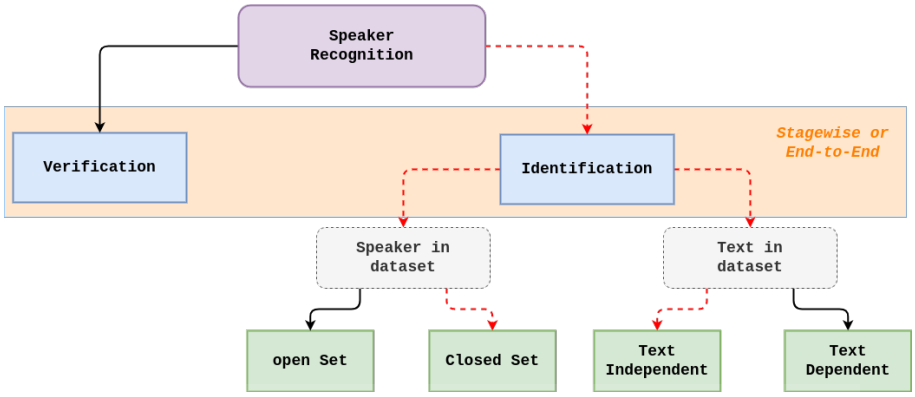

Figura 1.2: Verticales en el reconocimiento de hablantes

En los problemas de **Speaker Identification** se busca identificar un hablante desconocido dentro de un conjunto de hablantes conocido, a través de la noción de mínima distancia (es más parecido). Dentro de este campo de aplicación, se tiene dos escenarios de análisis diferentes, los cuales dependen del origen de la muestra a reconocer. Si el hablante a detectar se encuentra contenido dentro del conjunto de entrenamiento (hablante conocido), entonces nos encontramos en un escenario de tipo **close-set**. En cambio, si el hablante puede estar o no contenido dentro del conjunto de entrenamiento, se lo llama **open-set**. La diferencia entre ambos radica en que para un escenario de tipo *close-set* siempre se determinará una asignación, mientras que para un escenario de tipo *open-set* existirá un umbral de decisión a partir del cual no se le asignara una etiqueta al audio. Observando la figura 1.2 se evidencia otra ramificación mas en el universo de problemas de *Speaker Identification*, las cuales se diferencian entre sí respecto a la presencia o no de textos para enmarcar los audios o declaraciones. Una de ellas es **Text-dependent**, en donde los audios utilizan un mismo texto tanto para las muestras de entrenamiento como para las muestras a identificar. Este sistema es altamente dependiente del texto utilizado en el entrenamiento y las frases pueden ser comunes a todos los hablantes o distintas. La otra vertiente es **Text-Independent**, la cual se utiliza con mayor frecuencia ya que requieren muy poca o ninguna cooperación por parte del hablante al momento de recolectar las muestras.

En esta metodología, los textos utilizados durante el entrenamiento y la identificación son diferentes, haciendo posible el entrenamiento del modelo sin el conocimiento del usuario.

Respecto a los problemas de **Speaker Verification**, aquí se busca validar si el hablante es quien dice ser a través de la comparación de dos muestras de habla. En base a esta comparación, se decidirá si efectivamente la muestra a validar es suficientemente parecida a muestras propias del hablante que dice ser. En cierto sentido, la verificación es una coincidencia 1:1, mientras que la identificación es una coincidencia 1:N, en la que la voz se compara con varios *voice-prints* o *embeddings* diferentes. En este trabajo nos enfocaremos en una aplicación del tipo *Speaker Identification*, ya que se quiere identificar a una persona por su voz sin la necesidad de una proclamación previa de identidad.

Habiendo expuesto todas las ramificaciones presentes en el universo de problemas del reconocimiento de hablantes, vale la pena destacar que este trabajo se focalizará en la resolución de un problema de *Speaker Identification en un escenario close-set y con un marco text-Independent* ya que creemos que uno de los problemas con mayor impacto en la actualidad.

1.3. El problema

En las secciones anteriores se describieron las disciplinas que estudian las diferentes formas de lograr el reconocimiento automático de hablantes y se presentó una reseña sobre las características únicas que poseen las señales de voz de diferentes personas. Tomando estas consideraciones como punto de partida surgió la pregunta que este trabajo de tesis buscará responder, la cual es: **¿Existe un modelo teórico/práctico que permita reconocer la identidad de una persona utilizando audios sin una estructura discursiva estática o predefinida?**. Su respuesta estará direccionada hacia la creación de un sistema computarizado que logre reconocer la identidad de hablantes a través del uso y análisis de conversaciones coloquiales que este ha mantenido con otra persona. La respuesta a esta pregunta es un rotundo SI y en las secciones siguientes se detallará los pasos necesarios para construir los procesos que logren dicho objetivo.

1.4. Estructura de la tesis

La tesis se encuentra organizada en 9 capıtulos las cuales son: Introducción, Marco Teórico, De los modelos gaussianos a los i-vectors, Redes Neuronales en la detección de hablantes, La extracción de características, Los datos, Metodología, Resultados y Discusión. A continuación, una breve reseña de cada uno.

El capitulo Introducción, plantea la descripción de dos pilares fundamentales de este trabajo de tesis: La señal de habla y El reconocimiento automático de hablantes. Sumado a esto, se tendrá una sección con la descripción del problema a resolver llamada El problema.

El capitulo Marco Teórico, plantea todos los conceptos teóricos necesarios para llevar adelante este trabajo de tesis. Dentro de este tendremos: Las señales y los sistemas, La transformada de Fourier, Aprendizaje supervisado, Feature embeddings y Reducción de dimensionalidad

El capitulo De los modelos gaussianos a los i-vectors, repasa la historia de los primeros algoritmos de reconocimiento de hablantes, desde modelos probabilísticos de aprendizaje no supervisado hasta modelos que buscan la creación de vectores de características.

El capitulo Redes Neuronales en la detección de hablantes, analiza como impacto el auge de las redes neuronales en la detección de hablantes, agregando el concepto de *embeddings* como representación de una señal de habla. Dentro de esta, se repasaran diferentes arquitecturas de redes neuronales y como cada una ofrece una solución superadora al mismo problema.

El capitulo La extracción de características, plantea los diferentes enfoques utilizados para la extracción de las características de las señales de habla, que mejor representen al hablante.

El capitulo Los datos, describe como es la composición y estructura de los datos utilizados para la ejecución de las pruebas de este trabajo. Dentro de este tendremos: La calidad de los datos y Construcción y estructura de los datos

El capitulo Metodología, plantea todos los pasos necesarios para la contracción del sistema automático de reconocimiento de hablantes, que es el objetivo principal de esta tesis. Dentro de este tendremos: Descripción general del modelo, Preprocesamiento, Speaker embedding y Backend

El capitulo Resultados, ejecuta todas las validaciones de los tres modelos de *machine learning* que fueron entrenados y refinados en la sección Backend 7.4. Aquí se elegirá el modelo candidato a integrar la etapa de Backend. Dentro de este tendremos: Matrices de confusión y Indicadores de bondad de ajuste.

El capitulo Discusión, realiza un resumen de los temas mas importantes analizados en el transcurso de esta investigación. En este capitulo se tendrán conclusiones y oportunidades que han surgido durante este trabajo.

Capítulo 2

Marco Teórico

Para poder llevar adelante este trabajo es necesario comprender diferentes conceptos que conforman la conjunción entre el mundo de la señales y los algoritmos de aprendizaje maquinal o *machine learning*. En este capítulo se abordarán los temas más sobresalientes que permitirán facilitar la lectura y comprensión de la temática abordada en esta tesis.

2.1. Las señales y los sistemas

Cuando entablamos una conversación entre dos personas estamos llevando a cabo un proceso de generación e intercambio de señales de habla sobre un medio gaseoso. Aun cuando utilizamos dispositivos electrónicos como intermediarios en nuestra comunicación, el tracto vocal necesita del aire para poder generar las palabras. Esto ocurre gracias a que la señal de habla es una señal de tipo acústica, cuya recepción y comprensión ha sido optimizada por los humanos a lo largo de miles de años. Estos mismos dos procesos pueden llevarse adelante sobre sistemas computacionales gracias al desarrollo e implementación del análisis de señales y sistemas. Dada la importancia de estos dos conceptos, a continuación se detallará brevemente a qué nos referimos cuando hablamos de una señal o de un sistema.

Las **señales** están presentes en una gran variedad de campos de estudio como la comunicación, la astronomía, la geología, el procesamiento del habla, entre otras. Alan Oppenheim en su libro *Señales y Sistemas* [34] comenta que todas las señales, aun cuando difiere su naturaleza

física, comparten dos aspectos primordiales: ser funciones de una o más variables independientes y ser portadoras de información acerca del comportamiento o naturaleza de un fenómeno. Por lo tanto, podemos decir que cuando hablamos de señales estamos hablando implícitamente de funciones que almacenan información.

Tal como sucede con las funciones en el ámbito matemático, las señales también pueden clasificarse como continuas o discretas. La diferencia entre ambas radica en cómo se conforman sus dominios y codominios, ya que una señal continua tendrá tanto sus entrada como salida contenidas en un espacio continuo, mientras que las señales discretas solamente tendrán su entrada definida en un conjunto discreto. Dentro de las señales discretas, tenemos un subgrupo conformado por las llamadas "señales digitales", las cuales se caracterizan por tener tanto su dominio como codominio definido en un conjunto discreto. Estas, son las señales utilizadas por los sistemas computacionales y, por lo tanto, son de gran relevancia para el desarrollo de este trabajo.

El próximo paso luego de definir que es una señal, es definir qué es un sistema. Alan Oppenheim establece que un **sistema** es una interconexión de componentes o dispositivos, los cuales aplican transformaciones a una señal de entrada con el objetivo de obtener una señal distinta a su salida. Cuando nos referimos a la transformación que sufre la señal de entrada (X), en realidad estamos hablando de una operación matemática entre dos funciones, siendo una de estas la señal de entrada y la segunda denominada Función de transferencia (h). Esta, es la función que caracteriza al sistema y es de mucha utilidad en el diseño de filtros digitales. Tomando la definición de Oppenheim, los componentes que conforman un sistema pueden conectarse de diferentes maneras, ya sea en paralelo o en serie, con el objetivo de obtener una transformación resultante que permite construir la señal de salida deseada (Y). En la figura 2.1 se muestra un ejemplo de algunos de los distintos tipos de configuración que puede generarse a partir de tres sistemas h_i distintos; siendo i el subíndice que indica que son funciones distintas.

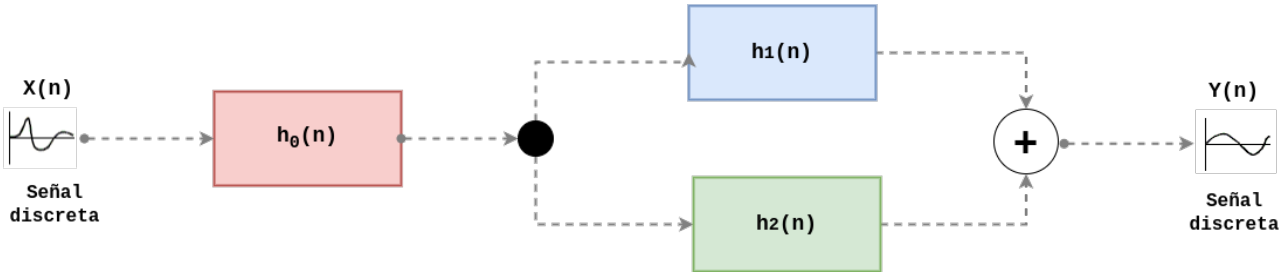

Figura 2.1: Ejemplo de configuración en serie y paralelo de un sistemas discreto

Es importante destacar que así como existen las señales continuas, discretas y digitales, también existen sistemas continuos, discretos y digitales. Por lo tanto, la posibilidad de contar con señales y sistemas digitales permite realizar su diseño e implementación sobre computadoras utilizando diferentes tipos de algoritmos. Algo a destacar es que la mayor diferencia entre una señal o sistema discreto y uno digital, es que la variable dependiente no se encuentra contenida en un conjunto continuo ya que solo puede tomar valores discretos y, además, se encuentra cuantizada.

Finalmente y situándonos en la problemática que busca atacar este informe, dispondremos de una señal acústica continua (el habla), la cual será tomada por un sistema receptor de audio (micrófono), se digitalizara y luego se almacenara. Una vez realizada esta recolección, las señales deberán ser procesadas por diferentes sistemas (filtros) que permitan implementar un correcto preprocesamiento de los datos, previo a las etapas de entrenamiento e inferencia.

2.2. La transformada de Fourier

El tiempo es una variable muy conocida y su concepto es familiar debido a que es un marco de referencia en todas las actividades que realizamos a lo largo de nuestras vidas. Nos es muy fácil entender cómo se comportan o varían distintos elementos o fenómenos en el dominio temporal. Ahora, cuando nos sumergimos en el mundo del análisis de señales, la utilización del tiempo únicamente como variable de análisis acarrea ciertas limitaciones y complejidades interpretativas; las cuales pueden reducirse en gran medida si también analizamos el comportamiento de las señales y los sistemas en el dominio frecuencial. Esta dualidad nos permite

identificar cómo interactúa un sistema con las distintas frecuencias que componen una señal (espectro) y facilita la construcción de filtros que permitan modelar una señal de salida a partir de una señal de entrada.

Joseph Fourier fue un físico y matemático francés, muy conocido en el ámbito del procesamiento de señales por sus trabajos sobre la descomposición de funciones periódicas en series convergentes llamadas Series de Fourier. A partir de esta representación nace el concepto de Transformada de Fourier. Esta transformada revolucionó el análisis de señales y generó innumerables aplicaciones [7] demostrando que cualquier señal periódica en el tiempo puede descomponerse en una combinación lineal de funciones sinusoidales o exponenciales complejas, conocidas como armónicos. Ahora, si realizamos un ejercicio mental y pensamos a una función aperiódica como una señal periódica con periodo tendiendo al infinito, tendremos que las series de Fourier convergen a una función.

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi ft} \mathrm{d}t$$

Esta función es la denominada transformada de Fourier y nos permite representar a una señal temporal en el dominio frecuencial. Así como existen diferentes tipos de funciones o señales, también existen distintas versiones de la transformada de Fourier. Estas, dependen de la naturaleza de la señal a la cual se aplican por ejemplo, si esta es continua se tendrá la Transformada de Fourier de tiempo Continuo mientras, si es discreta se tendrá la Transformada de Fourier de tiempo discreto o DTFT. Por lo tanto, independientemente de que la función de origen sea continua o discreta, ambas transformadas son continuas en el dominio espectral.

De todas las aplicaciones que nos brinda la transformada de Fourier, nos centraremos en dos casos particulares: La Transformada discreta de Fourier o DFT y el espectrograma. Estas dos son de gran utilidad para el análisis de señales y, por lo tanto, serán descritas a continuación.

2.2.1. La transformada discreta de Fourier

En la sección Las señales y los sistemas se mencionó la importancia de las señales digitales para llevar acabo diferente análisis sobre computadoras, por lo que sería de gran utilidad poder implementar la transformada de Fourier de forma digital. Para digitalizar una señal es necesario que ésta sea discreta tanto en su dominio como codominio pero, tal como se comentó en la sección anterior, la transformada de Fourier de tiempo discreto recibe señales discreta a su entrada y entrega señales continuas a su salida. Por lo tanto, para cumplir con estos requerimiento es que se creo un algoritmo que permite calcular de forma digital esta transformada, denominada Transformada de Fourier discreta o DFT [49].

La **DFT** puede interpŕetarse como el muestreo de la Transformada de Fourier de tiempo discreto en "N" puntos separados por intervalos de tiempo T (en términos de frecuencias 1/T), coincidiendo los valores de ambas transformaciones en esos puntos. Para brindar un poco de rigurosidad matemática a dicha transformada, es que se muestra a continuación la formulación que define el cálculo de la DFT sobre una señal discreta de tamaño N.

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{\frac{-j2\pi kn}{N}} \qquad n \in [0, N-1]$$

Partiendo de la base que existe una dualidad entre la función y su transformada, es que este muestreo en el dominio espectral se traduce en otro tipo de operación en el dominio temporal, llamada convolución. El entendimiento de estas dos operaciones matemáticas son de mucha utilidad al momento de la construcción de las etapas de preprocesamiento de los datos de entrada. Una vez definida la operación DFT tanto a nivel conceptual como matemático, es necesario disponer de una herramienta que facilite su implementación de una manera rápida y óptima. Para satisfacer estos requerimientos se creo el algoritmo FFT.

La Transformada rápida de Fourier o **FFT** [10], por sus siglas en inglés, es un algoritmo que permite implementar la DFT dentro de un sistema computacional de manera óptima. Este algoritmo ocupa un papel fundamental en el análisis de señales digitales ya que permite, con solo unas pocas líneas de código, la obtención de su descomposición en frecuencias. Sin ahondar

mucho en detalle, se puede comentar que la FFT optimiza el cómputo a través de la división del problema en cálculos de DFT de menor orden mediante una estructura recursiva. La famosa frase "divide y conquistarás" aplica a la perfección en este caso. A los fines de los trabajos de laboratorio de este proyecto la FFT será utilizada en la etapa de preprocesamiento de las señales de habla.

2.2.2. El espectrograma

Otra herramienta muy útil en el análisis de señales es el denominado Espectrograma [3], el cual es una representación visual de las variaciones de amplitud que presentan las diferente frecuencias de una señal a lo largo del tiempo. La posibilidad de obtener esta representación es de gran utilidad ya que las señales de habla no son homogéneas en el tiempo y pueden presentar distintos comportamientos como por ejemplo, silencios o etapas ruidosas de baja amplitud. Estos comportamientos, como tantos otros, pueden ser detectados fácilmente a través de un espectrograma.

Para calcular y obtener el espectrograma debemos hacer uso de la ya conocida FFT, la cual se aplica a la señal de entrada a lo largo de una ventana de tiempo de tamaño fijo. Esta ventana no es estática, por lo contrario, se desplazará sobre la señal seleccionando distintos subconjuntos de muestras a las cuales se les calcula la FFT. La suma de todas las representaciones obtenidas a través de las FFTs, permitirá construir un mapa que relacione la variación de la energía en cada frecuencia o armónico en el tiempo. En la figura que se presenta a continuación se tiene un ejemplo del espectrograma de una señal de radio con frecuencia modulada.

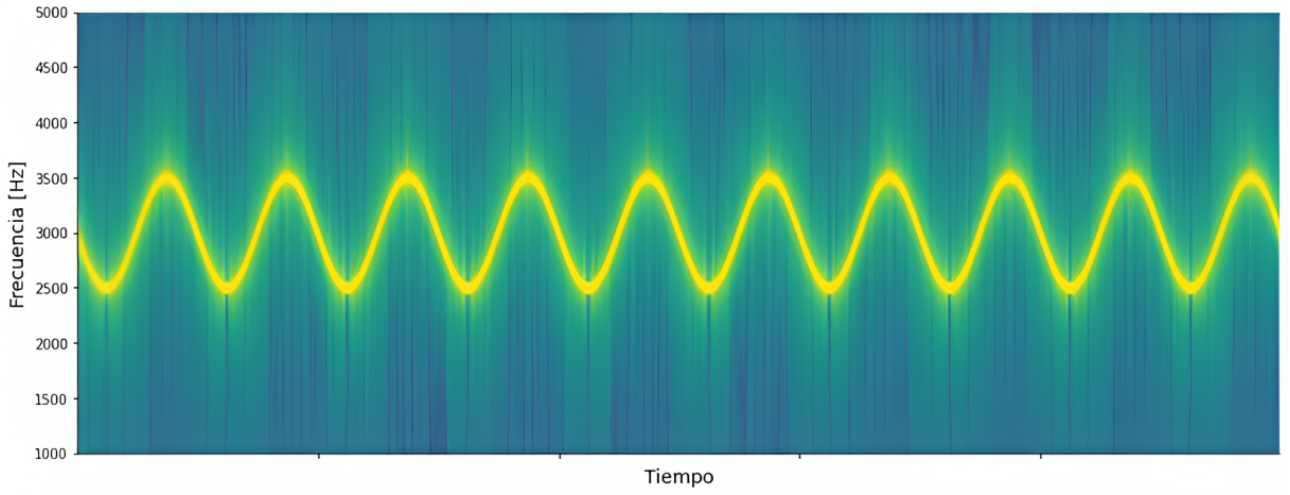

Figura 2.2: Espectrograma de una señal modulada

Cabe destacar que el tamaño de la ventana utilizada definirá el nivel de resolución que tendrá el espectrograma tanto en tiempo como en frecuencia, por lo que si se aplica una ventana larga se tendrá una mayor resolución en frecuencia (la cantidad de puntos está directamente relacionada a la cantidad de muestras de la FFT) pero no en tiempo. En cambio, si se utiliza una ventana de tiempo corta se tendrá una mayor resolución en tiempo y no en frecuencia, por lo que esta relación de compromiso debe ser tenida en cuenta al momento de utilizar espectrogramas para el análisis de señales.

2.3. Los Coeficientes Cepstrales Mel

Los **Coeficientes Cepstrales Mel** o también conocido como **MCC** [1], por sus siglas en inglés, son una técnica de procesamiento de señales de habla ampliamente utilizada en el campo del reconocimiento del habla y la síntesis de voz. Estos, son una representación matemática de la información extraída del espectro de una señal de habla, basados en la percepción auditiva humana.

Para poder llevar adelante el cálculo y extracción de los MCC se utiliza un sistema compuesto por varias etapas. Primero, se divide la señal de voz en pequeñas ventanas de tiempo, generalmente de 20-40 milisegundos, ya que como las señales de habla varían constantemente

en el tiempo, es importante obtener porciones de ella que sean aproximadamente "estadísticamente estacionarias". Es aquí donde existe una relación de compromiso, ya que si el cuadro es muy corto no se tendrá suficientes muestras para obtener una estimación espectral confiable. Una vez realizado el ventaneo, se aplica la DFT a cada una de las ventanas, para luego ser procesadas por un banco de filtros Mel, el cual utiliza una escala de frecuencias no lineal llamada "Escala de Mel", que trata de imitar la manera en que el oído humano percibe el espectro de frecuencias. Finalmente, se aplica el logaritmo de la energía de cada banda y, mediante una transformada matemática como la transformada de coseno discreta (DCT) [48], se obtiene el conjunto de coeficientes cepstrales de cada ventana de tiempo.

En resumen, los MCC son una técnica de procesamiento de señales de voz que es utilizado para extraer características de una señal de voz relevantes para la percepción humana y, que han demostrado ser muy eficaces en aplicaciones de reconocimiento del habla, de hablantes y síntesis de voz.

2.4. Aprendizaje supervisado

El sistema que se busca construir en este trabajo está compuesto por diferentes etapas, entre la que se destaca un módulo que utiliza una algoritmo de *machine learning* para determinar si un audio pertenece a un hablante conocido o no. La palabra "conocido" es de vital importancia ya que nos indica que estamos ante un problema de aprendizaje supervisado. En esta sección describiremos todos los aspectos que hacen a la resolución de este tipo de problemas.

El aprendizaje supervisado es una subárea del *machine learning* la cual busca generar sistemas que aprendan a tomar decisiones mediante el suministro de un conjuntos de datos etiquetados. En términos coloquiales, es un proceso iterativo que escanea los datos disponibles en busca de información que brinde una solución a problemas conocidos por ejemplo, ¿Esta persona es lo suficientemente confiable como para entregar un préstamo?, ¿Cuántas personas se encuentran en una foto?, ¿Cuántos mililitros de lluvia tendrá la ciudad de buenos aires en lo que resta del año?, entre otras. En términos más rigurosos, el aprendizaje supervisado busca

ajustar funciones parametrizadas y bien conocidas sobre los datos suministrados con el objetivo de vincular una variable de entrada con una variable de salida. Este proceso se lleva adelante de manera iterativa a través de la exposición del algoritmo frente a pares dominio - codominio o [X,y]. El resultado de este entrenamiento es un modelo bien definido, el cual deberá ajustar correctamente a los datos de entrenamiento y aceptablemente (se deberá definir un criterio) a los nuevos datos también conocido como generalización.

2.4.1. Modelos de clasificación

Dentro de la rama del aprendizaje supervisado se tiene dos subcategorías que se definen por el tipo de variable a inferir. Si la variable objetivo es continua nos encontraremos en un problema de regresión, mientras que si la variable objetivo es discreta estaremos dentro de un problema de clasificación. Por su parte, la clasificación también puede dividirse en binaria y multiclase, cuya diferencia se encuentra en la cantidad de respuestas válidas que pueden obtenerse en la inferencia. En el marco de este trabajo, nos enfocaremos en modelos de clasificación multiclase, ya que se buscara detectar la identidad de una persona dentro de un conjunto de hablantes conocido. A los fines de esta tesis se utilizaron los algoritmos *K-Nearest Neighbor*, *Support Vector Machine* y *Random Forest*, los cuales se detallan a continuación.

K-Nearest Neighbor

K-Nearest Neighbors o **KNN** es un algoritmo supervisado basado en instancia, lo que significa que se caracterizan por memorizar el conjunto de muestras de entrenamiento para luego realizar las inferencias. Es por esto, que son conocidos como modelos de aprendizaje basados en memoria. El concepto detrás de este algoritmo es sencillo e intuitivo y puede resumirse en el dicho “dime con quién andas y te diré quien eres” ya que su método de asignación de etiquetas se basa en elegir la clase mayoritaria dentro de sus “k” vecinos más cercanos. Es decir, calcula la distancia de las nuevas muestras frente a cada uno de sus vecinos, las ordena de menor a mayor y selecciona la clase con mayor frecuencia, tal como se muestra en la figura.

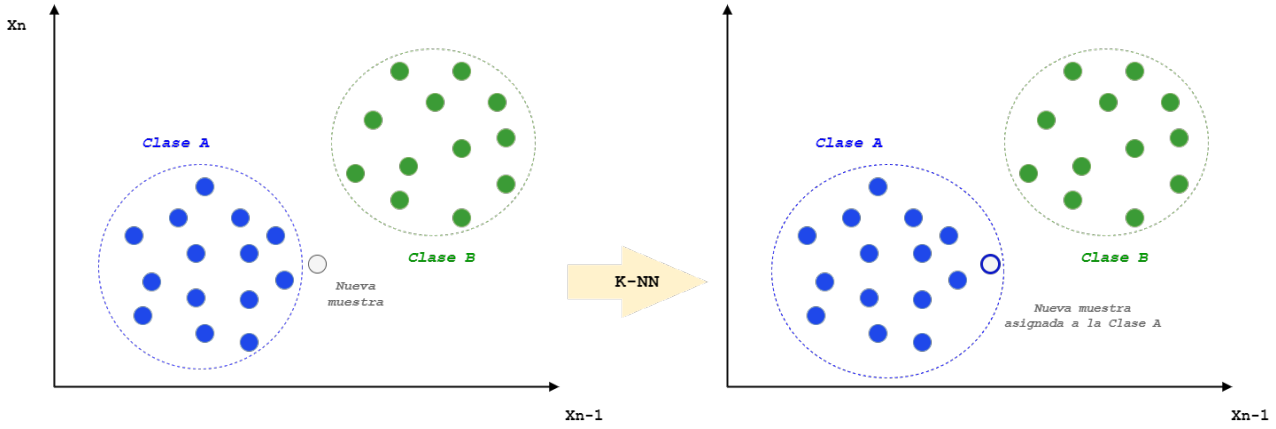

Figura 2.3: Clasificador K - Nearest Neighbor

Este método puede utilizarse tanto para problemas de clasificación como de regresión pero, en el marco de este proyecto, se utilizará su faceta clasificadora ya que se deberá clasificar un *embedding* de audio dentro de "k" hablantes conocido. Aunque parezca ser un algoritmo sencillo, es utilizado en una gran variedad de campos de aplicación como la detección de anomalías, algoritmo de recomendación, asignación de créditos, entre otros. La simplicidad de su interpretación e implementación son una ventaja considerable que, sumado a la sencillez de su entrenamiento, hacen a KNN un algoritmo de gran interés. Como contraparte, algunas de las desventajas que presenta este algoritmo son el elevado consumo de memoria y CPU y, debido a sus inferencias se basan en el cálculo de distancias (Ejem: Euclídea, Manhattan y Coseno), su susceptibilidad ante la "maldición de la dimensionalidad" [28].

$$d_{Manhattan}(p,q) = \sum_{i}^{n} |p_i - q_i| \qquad d_{Euclidea}(p,q) = \sqrt{\sum_{i}^{n} (p_i - q_i)^2}$$

Para amainar estas problemáticas existen maneras de atenúan el efecto de la dispersión de puntos en espacios de alta dimensión (maldición de la dimensionalidad) como las técnicas de reducción de dimensionalidad, por ejemplo, el Análisis de Componentes principales o PCA. Estas técnicas buscan proyectar a las muestras sobre espacios de menor dimensión, tratando de perder la menor información o variabilidad posible.

Support Vector Machine

Support Vector Machine o **SVM** en un algoritmo nacido bajo el concepto de clasificación binaria pero que ha evolucionado con el correr de los años hacia un algoritmo que permite resolver tanto problemas binarios, como de regresión y clasificación múltiple. Este, es considerado una referencia dentro del ámbito del aprendizaje estadístico y, por lo tanto, será evaluado en este proyecto en su versión multiclase.

El objetivo de SVM radica en la obtención de un hiperplano p-dimensional que permita separar, de la mejor manera posible, muestras definidas en un espacio n-dimensional (donde n = p - 1) en diferentes clases. Cuando se menciona "la mejor manera posible", se hace referencia a que existe una gran cantidad de hiperplanos que permiten separar las clases pero solo uno será el denominado *Maximal Margin Hyperplane* (MMH). Este, se define como el hiperplano que logra separar las diferentes clases y, que además, se encuentra lo más alejado posible de las observaciones fronterizas. Para una mejor visualización, supongamos que deseamos realizar una clasificación binaria sobre un conjunto de muestras constituidas por dos *features* cada una. Dado este contexto, SVM buscará la mejor recta posible de una dimensión que permita separar ambas clases.

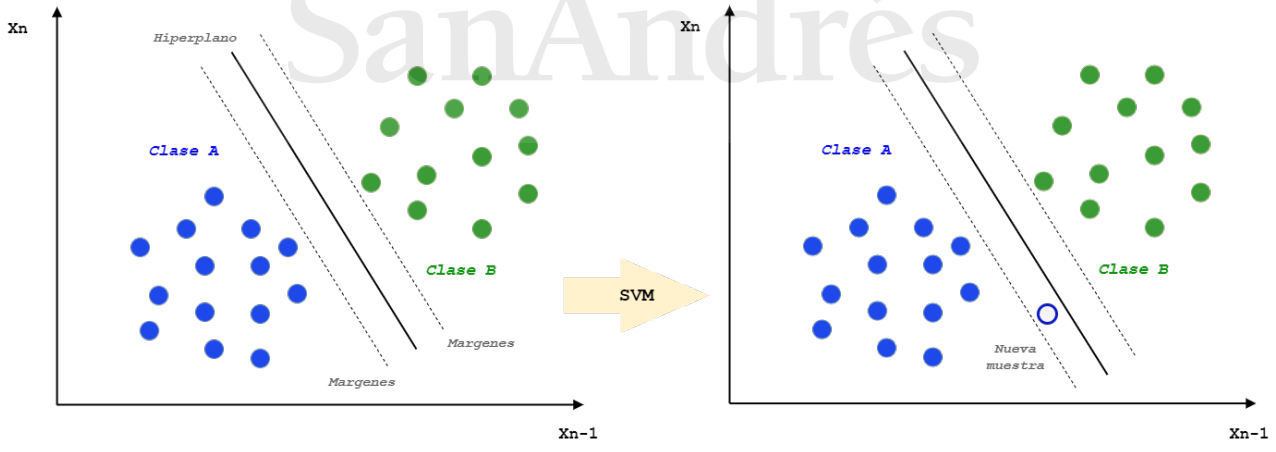

Figura 2.4: Clasificador Support Vector Machine

Tal como se muestra en la figura 2.4 el hiperplano elegido será el que maximice la distancia entre los márgenes, la cual se obtiene calculando la distancia ortogonal de cada observación

frente a este. La máxima distancia obtenida se denomina Margen y los puntos que la definen se llaman *Support Vectors*. Debido a que el conjunto de hiperplanos candidatos a ser el MMH tiende a infinito, es necesario la utilización de diferentes métodos de optimización para su selección. Más allá de que conceptualmente la elección del mejor hiperplano resulta atractiva, en la práctica esta estrategia puede presentar inconvenientes ya que el hiperplano se vuelve muy sensible a la inclusión de nuevos datos de entrenamiento, generando una tendencia hacia el *overfitting*. Estas problemáticas se atacan aplicando la metodología *Soft Margin Classifier*, la cual tolera que cierta cantidad de observaciones estén en el lado incorrecto del margen.

En el caso de que la distribución de las clases no admita una separación lineal, SVM puede subsanar esta limitante gracias a una estrategia basada en la expansión de la dimensión del espacio de *features*. Para ello, se parte de un supuesto de que si se lleva a las muestras a un subespacio de mayor dimensión, las clases puedan ser separables por un hiperplano. Esta expansión se logra gracias a los *kernels*, los cuales son funciones que permiten obtener un punto de dimensión "n", a partir de dos puntos de dimensión "n-1". Existen una gran variedad de estos, como son el Lineal, el Polinómico y el Gaussiano.

Finalmente es importante destacar algunas ventajas de este algoritmo, como es su robustez frente a muestras con alta dimensión, su eficiencia en memoria y su versatilidad gracias a los *kernels*. Respecto a algunas de sus desventajas, existe la posibilidad de obtener un *overfitting* si la cantidad de muestras es menor a la dimensión de su espacio vectorial.

Random Forest

Random Forest o **RF** es un algoritmo de aprendizaje supervisado utilizado tanto para clasificación como para regresión. Este se construye mediante la creación de múltiples árboles de decisión más los conceptos de *ensamble*, *bagging* y *bootstrapping*. En los siguientes párrafos se explicará cada uno de estos elementos y cómo se amalgaman en una solución de clasificación multiclase robusta y eficiente.

Los árboles de decisión son uno de los métodos más utilizados en la resolución de problemas de *machine learning* y generan predicciones en base a la utilización de condicionales. Estos

algoritmos se caracterizan por poseer una gran interpretabilidad, posicionándose como una buena elección a la hora de entender el porqué de sus predicciones. Si nos enfocamos en su fisonomía, los árboles de decisión están compuestos por un nodo raíz a partir del cual se generan diversas ramas de acuerdo al cumplimiento o no de una determinada condición sobre un *feature*. Estas ramificaciones terminan en los nodos hojas a una determinada profundidad, a la cual se le debe prestar atención ya que si es muy grande el árbol tiende a "memorizar" las soluciones generando *overfitting*. Por último, es importante comentar que existen diferentes criterios a la hora de particionar un nodo, entre los que se destacan el Índice Gini y Ganancia de información o Entropía.

$$Ginni = 1 - \sum_{i=1}^{Clases} (p_i)^2 \qquad Entropy = \sum_{i=1}^{Clases} -p_i * log_2(p_i)$$

Existe una relación de compromiso común a todos los algoritmos de *machine learning* la cual es el equilibrio entre sesgo y varianza. El sesgo hace referencia a cuánto se alejan en promedio las predicciones de un modelo respecto a los valores reales, en cambio, la varianza hace referencia a cuánto un modelo varía de acuerdo a pequeñas variaciones en los datos de entrenamiento. Los árboles de baja profundidad presentan poca varianza pero alto sesgo, en cambio los árboles de gran profundidad se ajustan muy bien a los datos de entrenamiento, por lo que tienen muy poco sesgo pero gran varianza. Los métodos de *ensamble* buscan aplacar estos problemas combinando múltiples modelos para la toma de decisiones. *Bagging* es un tipo de ensamble que se caracteriza por la toma de una decisión final en base a la combinación de predicciones realizadas por distintos modelos, los cuales han sido entrenados con un subconjunto de muestras de igual tamaño obtenidas a través de *bootstrapping* (método aleatorio de muestreo con reemplazo).

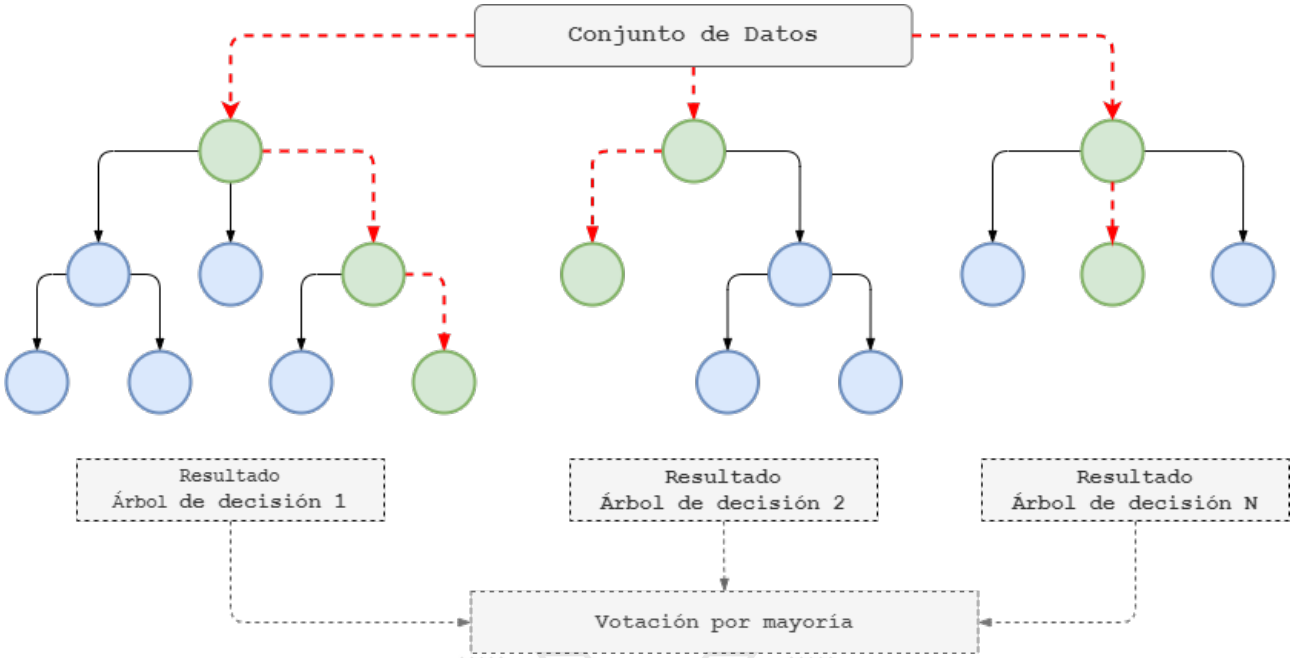

Figura 2.5: Clasificador Random Forest

Poniendo todas las cartas sobre la mesa, se puede definir a *Random Forest* como un *ensamble* de árboles de decisión simples, entrenados con muestras aleatorias extraídas a partir de los datos de entrenamiento originales mediante *bootstrapping*.

Finalmente es importante destacar algunas ventajas de este algoritmo, como es su robustez frente a *outliers* y *overfitting* y su estabilidad frente a la inclusión de nuevas muestras. Respecto a algunas de sus desventajas, tenemos los altos tiempos de entrenamiento respecto a otros algoritmos, las limitaciones de rendimiento en conjuntos de datos pequeños y su poca interpretabilidad.

2.4.2. Metodología de validación

El objetivo principal que se persigue al entrenar un modelo de *machine learning* es que pueda realizar inferencias de calidad sobre datos no vistos, también conocido como generalización. Si consideramos a la totalidad de las muestras como un único conjunto de entrenamiento, solo podremos calcular el error de entrenamiento, el cual se calcula en base a las inferencias realizadas sobre las muestras que él modelo vio y, por lo tanto, suele resultar demasiado opti-

mista. Entonces, con el fin de evitar enmascarar futuros problemas de generalización, se tiene que recurrir a un conjunto de test o emplear estrategias de validación basadas en *resampling*.

Los métodos *resampling* son estrategias que permiten estimar la capacidad predictiva de los modelos frente a nuevas observaciones, haciendo uso únicamente del conjunto de entrenamiento. La noción utilizada por estos métodos consiste en la extracción de un subconjunto de muestras de entrenamiento, para poder ser utilizadas en la evaluación de las inferencias del modelo entrenado sobre el subconjunto restante. Este proceso se repite múltiples veces, compensando posibles sesgos debido al reparto aleatorio de las muestras y agregando los resultados en una única métrica resultante.

De todos los métodos de *resampling* se destacan *K-Fold Cross-Validation* (KF-CV) y *Leave-One-Out Cross-Validation* (LOO-CV) como los más utilizados. La diferencia entre ambos métodos se sustenta en que el primero separa el conjunto de entrenamiento en "k" subconjuntos (k-1 grupos se utilizan para entrenar y uno validar), mientras que el segundo solamente separa de a un elemento por cada iteración. KF-CV presenta dos ventajas respecto a LOO-CV, las cuales son:

- Requiere menos iteraciones en su implementación, por lo que escala mejor frente a conjuntos de datos de gran magnitud o modelos complejos.
- Consigue una estimación más precisa del error de test gracias a un mejor balance entre sesgo y varianza obtenido por su mayor diversidad entre los subconjuntos de evaluación. LOO-CV emplea prácticamente todo el conjunto de datos en cada iteración, por lo que estos están altamente correlacionados (riesgo de *overfitting*).

En vistas de estas características, se eligió **K-Fold Cross-Validation** como método de *resampling* para los entrenamientos de los modelos.

2.4.3. Medidas de rendimiento

Para poder validar si un modelo generaliza bien, es necesario definir tanto un método de validación como las métricas de rendimiento, obtenidas a partir de la evaluación del conjunto

de muestras de prueba. Estas métricas son de vital importancia ya que permiten realizar la validación final del modelo antes de su despliegue al ambiente productivo. Se tiene una gran diversidad de métodos para evaluar el rendimiento de un modelo, en donde cada uno ponderara distintas cualidades, como por ejemplo la tasa de aciertos de la clase positiva (clasificación binaria) o TPR. Más allá de la gran cantidad de métricas disponibles, para los fines de esta tesis, se utilizarán la Matriz de confusión, y las métricas de *Accuracy*, *Precision*, *Recall* y *F-score*; cada una con un enfoque distinto, el cual se detalla a continuación.

Matriz de confusión

Una matriz de confusión es una representación matricial de los resultados obtenidos de las inferencias de un clasificador aplicado sobre un conjunto de prueba con etiquetas conocidas. La matriz de confusión es una matriz cuadrada de tamaño n x n, donde n es el número de clases presente en el problema. Cada fila de la matriz se corresponde a la verdadera clase de la muestra, mientras que las columnas son las clases que predijo el modelo. Si tomamos como ejemplo una clasificación binaria, tendremos como matriz resultante ejemplificada en la siguiente figura.

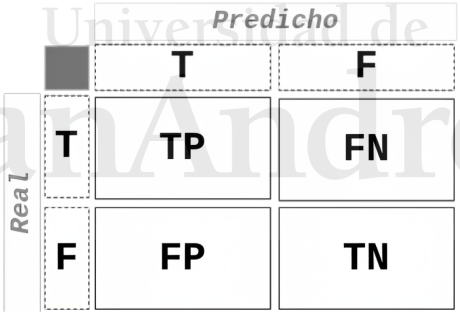

Figura 2.6: Matriz de confusión binaria

Tal como se observa, los elementos de la diagonal representan el número de puntos para los cuales la etiqueta predicha es igual a la etiqueta verdadera, mientras que cualquier cosa fuera de la diagonal fue mal etiquetada por el clasificador. Por lo tanto, cuanto mayor sean los valores de la diagonal de una matriz, mejor es el rendimiento del modelo. Dado esto, cada predicción puede encasillarse dentro de uno de cuatro resultados:

- Verdadero Positivo (TP): Predicho verdadero y verdadero en realidad.

- Verdadero Negativo (TN): Predicho falso y falso en realidad.
- Falso Positivo (FP): Predicción de verdadero y falso en la realidad.
- Falso Negativo (FN): Predicción de falso y verdadero en la realidad.

Si nos enfocamos en los errores que nos muestra esta matriz tendremos: un error de tipo I (equivalente a los FP) y un error de tipo II (equivalente a FN). Cada uno tendrá su respectiva importancia dependiendo del caso de uso a resolver debido a que rara vez ambos tienen igual peso en un mismo problema.

La matriz de confusión es una herramienta simple pero clara que nos permite entender qué tipo de errores sufre nuestro modelo y, por lo tanto, tomar acciones en respuesta. Además, es conveniente recalcar que aunque se haya utilizado un ejemplo de clasificación binario, estos conceptos se extienden de igual manera a una clasificación multiclase.

Accuracy

La métrica de *accuracy* se define como el porcentaje de predicciones realizadas correctamente por el modelo, respecto a todas las predicciones efectuadas sobre un conjunto de muestras. Para el caso de clasificaciones binarias, esta puede calcularse fácilmente dividiendo el número de predicciones correctas por el número de predicciones totales, tal como se muestra en la siguiente fórmula.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

El *accuracy* es una métrica de evaluación común para los problemas de clasificación y, cuando las clases se encuentran balanceadas dentro de un conjunto de muestras, brinda un reflejo fidedigno del rendimiento del algoritmo. Ahora, cuando estamos en presencia de conjuntos desbalanceados, el *accuracy* puede no ser la mejor elección ya que este enmascara posibles problemas en la clasificación de clases minoritarias, como por ejemplo, en un problema de detección de fraude.

Precision

La métrica de *precision* se define como el porcentaje de predicciones positivas realizadas correctamente por el modelo, respecto a todas las predicciones positivas efectuadas para un conjunto de muestras. Para el caso de clasificaciones binarias, esta puede calcularse fácilmente dividiendo el número de predicciones correctas por el número de predicciones positivas totales, tal como se muestra en la siguiente fórmula.

$$Precision = \frac{TP}{TP + FP}$$

La optimización de esta métrica permite disminuir el error de tipo I ya que un valor alto de esta nos dice que nuestro algoritmo acierta, en su mayoría, al momento de asignar una etiqueta positiva; siendo más laxo con FN.

Recall

La métrica de *recall* se define como el porcentaje de predicciones positivas realizadas correctamente por el modelo, respecto a todas las muestras positivas presentes en un determinado conjunto de pruebas. Para el caso de clasificaciones binarias, esta puede calcularse fácilmente dividiendo el número de predicciones correctas por el número de predicciones positivas correctas más el número de predicciones negativas incorrectas tal como se muestra en la siguiente fórmula.

$$Recall = \frac{TP}{TP + FN}$$

La optimización de esta métrica permite disminuir el error de tipo II ya que un valor alto de esta nos dice que nuestro algoritmo acierta, en su mayoría, al momento de asignar una etiqueta negativa; siendo más laxo con FP.

F-score

La F-score o Fβ difiere de las demás métricas dado que esta se encuentra parametrizada por un coeficiente llamado β, el cual puede tomar valores entre 0 y 1. Cuando beta es 1, el F-Score o F1 se define como la media armónica de las métricas de *precision* y *recall*, donde alcanza su mayor valor cuando estas son perfectas. Ahora, si no definimos a beta en 1, lo que obtendremos es una manera de poder compensar la importancia entre *precision* y *recall*, siendo $\beta < 1$ un valor que prioriza a la *precision*, mientras que $\beta > 1$ direcciona hacia el *recall*, tal como se muestra en la siguiente fórmula.

$$F_\beta = (1 + \beta^2)\frac{Precision * Recall}{(\beta^2 * Precision) + Recall}$$

La optimización de esta métrica permite encontrar un balance entre el error de tipo I y el error de tipo II, para así generar un métrica que nos permita prestar más atención a los FP o FN, de acuerdo a lo que el problema demande.

2.5. Feature embeddings

El objetivo que se persigue al entrenar un algoritmo de *machine learning* es la habilidad de extraer patrones de los datos para luego poder realizar predicciones lo más acertadas posibles. Por lo general se espera que dos muestras con *features* similares generen una predicción similar, por lo tanto el modo en que son representadas afectará directamente la calidad de las predicciones. El cómo representar los *features* es importante para los casos en que las muestras no son estructuradas o, que contengan algún *feature* no estructurado como el texto, las imágenes y los audios. Por lo tanto, ante la presencia de este tipo de datos es necesario crear una representación compacta a través de vectores numéricos, los cuales se denominan **embeddings**.

Un *embedding* es, en pocas palabras, la representación o transformación de una muestra sobre un espacio vectorial numérico cuyo foco se centra en capturar la mayor parte de su información contextual y semántica. Esto significa que dos muestras con significados similares

se ubican cercanas, tomando alguna métrica de distancia como referencia (Ejem: euclídea o coseno) dentro del espacio vectorial de los *embeddings*. Para brindar una explicación más clara, tomaremos como ejemplo la representación del texto de un título.

El título está compuesto por pocas palabras, por lo que si se lo quiere representar vectorialmente se podría utilizar alguna técnica en donde cada palabra del diccionario sea un *feature*. Utilizando esta técnica, se tendrá cada título representado por un vector esparzo (mayoría de sus componentes estarán en cero) y de alta dimensionalidad, ya que es poco probable que un título use más de veinte palabras. En cambio, sí se entrena un algoritmo que tome diferentes corpus de texto, extraiga su información contextual y semántica, y construya una representación vectorial de cada una de ellas, se tendrá un vector de *features* de mucha menor dimensionalidad que el anterior. Este, ubicará a las palabras con similar significado cercanas en el espacio, generando un *embedding* de mayor densidad [56] que representara al texto que componen a los título.

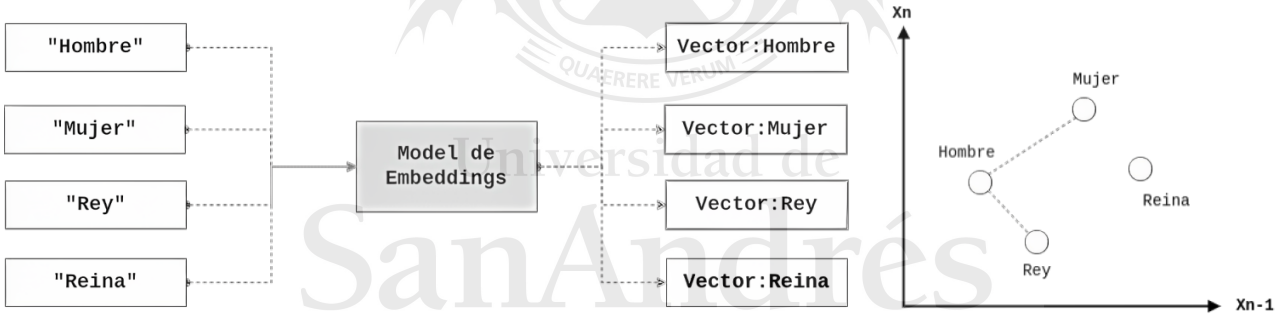

Figura 2.7: Ejemplo de un modelo de generación de embeddings

De manera similar, esta lógica puede aplicarse a distintos tipos de datos como los audios y, en particular, las señales de habla. Las redes neuronales son muy utilizadas para la obtención de vectores de *embeddings* sobre datos no estructurados. Para los fines de este trabajo, emplearemos una red neuronal preentrenada la cual nos permitirá pasar de un conjunto de audios a un conjunto de vectores numéricos. Una vez transformado todo el conjunto de datos, se utilizarán como entradas que alimenten el entrenamiento de una modelo de clasificación multiclase que buscará detectar la identidad de un hablante basándose en su nueva representación.

2.6. Reducción de dimensionalidad

La obtención de muestras con una gran cantidad de *features* es cada vez más frecuente en la actualidad. Algunas de las razones de este comportamiento van desde la presencia de una gran diversidad de fuentes de datos o, también, la utilización de técnicas de *feature embeddings* que generan vectores de alta dimensionalidad a partir de textos o audios. Es común pensar como ideales a los escenarios que presentan gran cantidad de *features*, sin embargo, en la práctica puede observarse que los espacios de alta dimensionalidad puede atentar contra la capacidad de generalización de los modelos e incrementar sus tiempos y costos de entrenamiento. Estos comportamientos pueden enmarcarse dentro del fenómeno conocido como Maldición de la dimensionalidad.

La maldición de la dimensionalidad ocurre cuando la tasa de aumentos de la cantidad de *features* es mayor a la tasa de aumento de la cantidad de muestras, generando una disminución de su densidad en el espacio de coordenadas. En pocas palabras, las muestras se empiezan a dispersar cada vez más. Además, se produce otro comportamiento que es tanto o más perjudicial que la dispersión y es que la variabilidad de la distancia entre ellas disminuye con el aumento del número de dimensiones del subespacio que las contiene. Por lo tanto, las muestras están a casi la misma distancia, perjudicando especialmente a los métodos basados en distancias, como por ejemplo KNN.

Frente a esta amenaza, una acción razonable será encontrar algún método que nos permita solventar este problema (disminuir la dimensión del espacio de muestras), perdiendo la menor cantidad de información posible. Una hipótesis a considerar en una estrategia de reducción de dimensionalidad, es suponer que no todas las variables que componen un vector de *features* contiene información única por lo tanto, pueden representarse los mismo datos en un espacio de menor dimensión. Tomando como base esta hipótesis, las técnicas que implementan reducción de la dimensionalidad se dividen en dos verticales: la eliminación de características y la extracción de características. Ambas verticales son muy utilizadas en el ámbito de la ciencia de datos y, en reiterada ocasiones, se utilizan su combinación de ellas.

En este trabajo nos enfocaremos en la metodología de extracción de características, la cual crea nuevas variables a partir de las originales, logrando una gran reducción de la cardinalidad del subespacio a cambio de la pérdida de su interpretabilidad. Análisis de Componentes Principales es un ejemplo de estas técnicas, siendo esta la utilizada en este trabajo de tesis y por lo tanto se detallará a continuación.

2.6.1. Análisis de componentes principales

El objetivo principal del **Análisis de Componentes Principales** o **PCA**, por sus siglas en inglés, es reducir la dimensionalidad de un conjunto de datos que se encuentran compuestos por *features* correlacionados entre sí. Estas correlaciones son de diversa magnitud y PCA buscará proyectar las muestras originales en un nuevo subespacio conservando la mayor cantidad de variabilidad, a través de la construcción de una transformación lineal que permita representar a los datos a un espacio de menor dimensión.

$$D \subset \mathbb{R}^p \xrightarrow{PCA} \mathbb{R}^q \; con \; q < p$$

Esta transformación se obtiene mediante la construcción de una base ortogonal compuesta por los autovectores de la matriz de covarianza de las muestras. Estos autovectores representan las direcciones en las que se proyectan las muestras y sus autovalores, la magnitud de la variabilidad o información que representan. Por lo tanto, ordenando los autovectores de acuerdo a la magnitud de sus autovalores (mayor a menor), podremos obtener la matriz de proyección de los componentes principales. Matemáticamente, si definimos un vector de muestras $X \in \mathbb{R}^p$, se tendrá:

$$\lambda_1 \geq ... \geq \lambda_p \; autovalores \; de \; la \; matriz \; de \; covarianza \; \Sigma$$

$$\gamma_1, ..., \gamma_p \; autovectores \; de \; \Sigma \; formando \; \Gamma = (\gamma_1, ..., \gamma_p)$$

$$Siendo \; v = \Gamma'(X - \mu) \; ; \; el \; vector \; de \; componente \; principales \; de \; X$$

Como resultado, la reducción de dimensionalidad estará dada por la cantidad de componentes principales que se quiera utilizar, siendo el primer componente el de mayor varianza. A medida que se agregan nuevos componentes, se irá construyendo la proyección del vector *features* original, siendo acumulativa y cada vez menor la variabilidad que aporta cada componente. Esta lógica de transformación y luego selección de componentes principales invita al desafío de encontrar la cantidad justa y necesaria de dimensiones del subespacio de *features*, mientras que se retiene la mayor parte de la información posible es decir, su variabilidad. En la siguiente imagen puede observarse un ejemplo gráfico en un espacio de dos dimensiones de los ejes de coordenadas de las proyecciones.

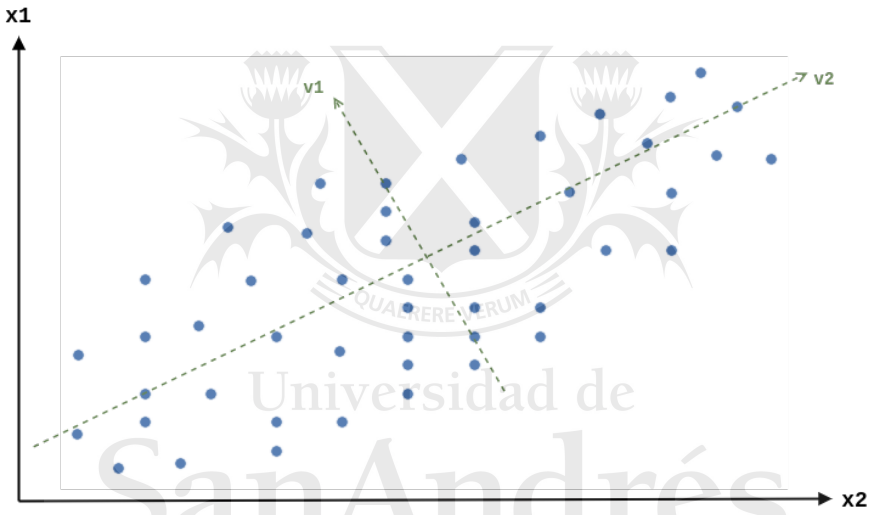

Figura 2.8: Ejemplo de una proyección por PCA

PCA es una herramienta muy poderosa para el preprocesamiento de los datos y trae grandes soluciones para el entrenamiento de algoritmos basados en distancia pero, antes de aplicar esta técnica, es necesario conocer sus bondades y limitaciones. En lo que respecta a sus limitaciones, PCA es sensible a la escala de magnitudes relativa que presentan los *features*, por lo que estos necesitan ser normalizados previamente antes de aplicarles dicha transformación. Además, ya que las nuevas coordenadas de los vectores de *features* no corresponden al subespacio original, los nuevos datos pierden su interpretabilidad. En cuanto a sus bondades, se destaca la gran disminución tanto del espacio de almacenamiento como del tiempo de procesamiento. Además

nos permite llevar las muestras a dimensiones muy bajas (2D y 3D) lo que facilita la creación de visualizaciones y, como principal valuarte, nos ayuda a combatir la maldición de la dimensionalidad.

2.7. Redes Neuronales artificiales

Las **redes neuronales artificiales** o **ANN**, por sus siglas en ingles, son un conjunto de algoritmos de aprendizaje automático inspirados en la estructura y funcionamiento del cerebro humano. Estas redes están compuestas por una serie de unidades de procesamiento interconectadas, llamadas neuronas, que se organizan en capas y procesan información para resolver problemas complejos de aprendizaje automático. Existen varios tipos de arquitecturas de redes neuronales, en donde algunas de las mas comunes son: Totalmente conectadas, Recurrentes, Convolucionales; cada una diseñada para resolver un problema específico como, por ejemplo, clasificación, procesamiento de imágenes, señales de audio, entre otros.

De las arquitecturas mencionadas, vale la pena destacar a las **redes neuronales convolucionales** o **CNN** [2], por sus siglas en ingles, las cuales son un tipo de red neuronal que se utiliza comúnmente para procesar datos como imágenes y señales de audio. Las CNN se basan en una operación matemática llamada convolución, la cual se implementa a través de filtros y permiten extraer características relevantes de los datos de entrada. Las CNN pueden estar compuestas totalmente por capas convolucionales o por una combinación de estas con otras, por ejemplo capas de *pooling* o totalmente conectadas. Siguiendo con este ejemplo, las capas de *pooling* reducirán la dimensión de los mapas de características generados por las capas convolucionales, mientras que las capas completamente conectadas serán utilizadas para llevar adelante el proceso de clasificación. Las CNN han demostrado ser muy efectivas en la clasificación de imágenes, la detección de objetos y la segmentación de imágenes, como así también en el procesamiento de señales de audio y lenguaje natural.

En lo que respecta a este trabajo, las CNN han demostrado ser de gran utilidad para el reconocimiento del hablante, al permitir la extracción de características de las señales de habla,

como es la frecuencia fundamental. Esto las hace una arquitectura atractiva al momento de utilizar redes neuronales para la construcción de sistemas automáticos de detección de hablantes.

Capítulo 3

De los modelos gaussianos a los i-vectors

En el capitulo Introducción se comentó los diferentes niveles de información que contiene una señal del habla y cómo se divide conceptualmente las áreas de *Speech Recognition* y *Speaker Recognition*. Además, se avanzó hacia un nivel mayor en la ramificación del *Speaker Recognition* obteniendo las subáreas de *Speaker Verification* y *Speaker Identification*, enfocadas en resolver problemas distintos. Ya sea cualquiera de las ramas de análisis mencionadas, algo que es común a ellas es que su éxito dependerá de cuán eficiente sea la extracción y modelado de las características que distinguen un hablante de otro. En este capitulo, se hará un breve recorrido por las técnicas más relevantes utilizadas en los comienzos del *Speaker Identification*.

El primer método automático utilizado en la identificación de hablantes se basaba en **Modelos Gaussianos Mixtos** o **GMM** [11]. Estos, son modelos no supervisados que buscan generar *clusters* de señales y asignarles una función de densidad de probabilidad a cada uno. Luego, se evaluaran estas densidades sobre los vectores de *features* obtenidos de las señales de habla para así obtener una métrica que sirva para la toma de una decisión. Reynolds y Rose en su artículo *Robust Text-Independent Speaker Identification Using Gaussian Mixture Speaker Models* [41] comentan que existen dos motivaciones principales para usar mezcla de gaussianas como representación de la identidad del hablante. La primera motivación es la noción intuitiva

de que un densidad multi-modal como GMM pueda modelar el espacio acústico de la voz de un hablante, la cual puede caracterizarse por la unión de eventos fonéticos ocultos como vocales, nasales y fricativas. La segunda motivación se basa en la observación empírica de que una combinación lineal de funciones gaussianas puede representar una gran cantidad de clases debido a su capacidad para formar aproximaciones suaves a densidades de diferentes formas.

Si nos enfocamos en la formulación matemática que sustenta esta modelización tendremos que la función de probabilidad de una mezcla gaussiana es una suma ponderada de M densidades, tal como se muestra en la siguiente ecuación:

$$p(\vec{x}|\lambda) = \sum_{i=1}^{M} p_i b_i(\vec{x})$$

Donde $\vec{x}$ es un vector d-dimensional, $b_i \in i = 1, ..., M$ son las densidades que componen la mezcla y $p_i \in i = 1, ..., M$ son los pesos de la mezcla. Cada una de las densidades que componen la mezcla se corresponden con una gaussiana d-variante con media μ_i y matriz de covarianza E_i.

$$b_i(\vec{x}) = \frac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} \, exp[\, -\frac{1}{2}(\vec{x} - \vec{\mu_i})' \Sigma_i^{-1} (\vec{x} - \vec{\mu_i}) \,] \; con \; \sum_{i=1}^{M} p_i = 1$$

Entonces, la densidad de la mezcla gaussiana quedará parametrizada mediante los vectores de medias, las matrices de covarianza y los pesos de la mezcla de todas las densidades de los componentes. Estos parámetros están representados colectivamente por la notación:

$$\lambda = \{ \, p_i, \vec{\mu_i}, \Sigma_i \, \} \; con \; i = 1, ...M$$

Por lo tanto, cada hablante está representado por un modelo GMM y su correspondiente parámetro lambda, quedando el problema circunscrito a encontrar la lambda que mejor ajuste a la distribución de los vectores de *features*.

En su modelo GMM extremo a extremo, Reynold propone un enfoque frecuencial para construir una primera etapa que extraiga de las señales de voz la mayor cantidad de información que permita representar a un hablante. Para ello, utiliza coeficientes cepstrales obtenidos de un

banco de filtros Mel o *mel-frequency filterbank*. En el articulo, el sistema calcula la magnitud del espectro de un segmento de voz utilizando la FFT, le aplica un filtro de preénfasis y luego lo procesa a través de un banco del filtros Mel. Finalmente, a la salida del filtro se aplica la transformada coseno [48] para producir los coeficientes cepstrales. El espectro del habla como fuente para la extracción de *features* resulta muy efectivo para la detección de hablantes [4] ya que, como se comento en secciones anteriores, refleja la estructura del tracto vocal que es un factor fisiológico predominante en la unicidad de la voz de una persona. Una vez obtenidos los vectores de características, se procederá a entrenar el modelo GMM para poder estimar el parámetro lambda y lograr que la mezcla de gaussianas se asemeje lo mayor posible a la densidad propia de los vectores de *features*. Para llevar adelante esta búsqueda el método comúnmente utilizado es el de máxima verosimilitud y Reynolds propone el algoritmo EM (*Expectation-Maximization*) para lograr su maximización. Gracias a este algoritmo se podrán obtener las GMM que modelan a cada hablante y luego, mediante su evaluación y posterior definición de un umbral de decisión, llevar a cabo la detección.

Ahora, este enfoque ha sido efectivo en identificación de hablantes pero cuando nos encontramos frente a un problema de verificación de hablantes (uno contra todos) es necesario poder contrastar dos hipótesis, las cuales plantean si un audio pertenece a una hablante o no. Para hacer frente a esto nace el modelo de hablante alternativo o también conocido como **Universal Background Model** o **UBM** [40] el cual propone modelar a todos los hablantes, excepto al objetivo, como una única mezcla de gaussianas parametrizadas. Este modelo universal es esencialmente un GMM entrenado para representar a todas las distribuciones independientes de las muestras que componen el universo de hablantes conocidos. Una vez entrenados ambos modelos, se procederá a realizar una evaluación del mismo modo que GMM. Posteriormente, el UBM fue utilizado en la identificación de hablantes como modelo inicial para todas las distribuciones de las muestras dando lugar el método GMM-UBM, el cual obtiene la mezcla de gaussianas de un hablante a partir de la derivación del UBM mediante adaptación bayesiana [19]. Este enfoque demostró proporcionar un mejor rendimiento que el clásico de GMM, el cual entrenan todas las mezclas de forma independiente.

Aun con las mejoras ofrecidas por GMM-UBM, existía un problema que merecía ser tenido en cuenta. Este, consiste en que las muestras audio utilizadas para el entrenamiento y prueba podían poseer diferentes duraciones, lo cual generaba problemas al momento de realizar las comparaciones entre ellas y limitaba mucho la diversidad de clasificadores a utilizar. Para subsanar esto se crearon los **GMM Supervectors** [11], los cuales consisten en vectores de longitud fija formados por la concatenación de los parámetros de las GMM (generalmente sus vectores de medias). Este nuevo vector longitud fija requirió la utilización de un algoritmo de aprendizaje automático para poder llevar adelante la identificación de hablantes. Para este método, el clasificador *Support Vector Machines* [14] fue una de las técnicas que mejor rendimiento mostró. En la figura 3.1 se ilustra un esquema GMM que utiliza un UBM sobre una mezcla de 4 densidades gaussianas, las cuales comienzan un proceso de adaptación frente a las muestras de entrenamiento (a). Una vez finalizada dicha adaptación o ajuste, se procede a la creación de los supervectors mediante la concatenación de los vectores de medias de cada densidad (b).

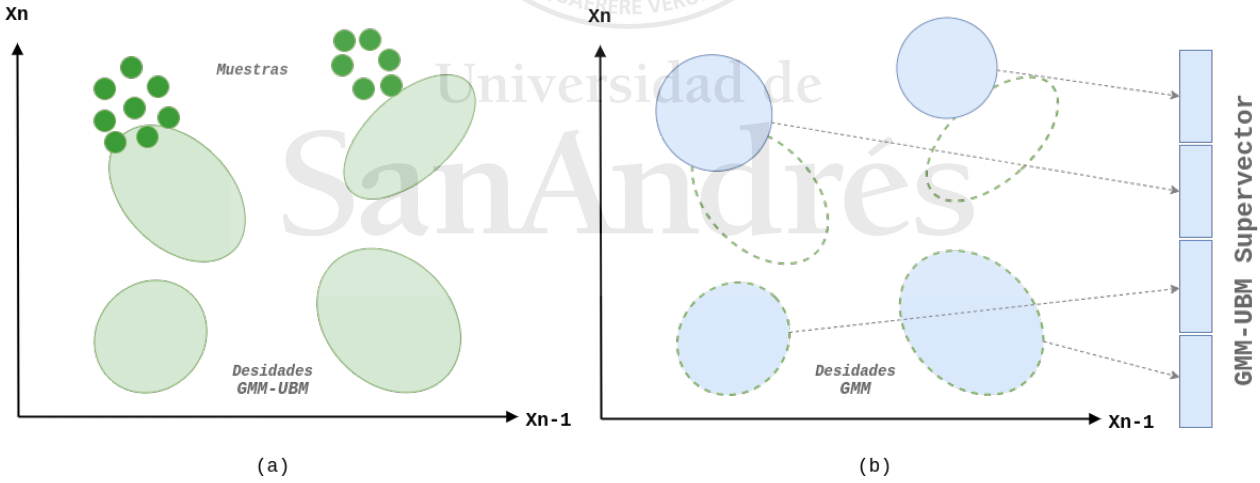

Figura 3.1: Ajuste de GMM-UBM al conjunto de muestras y creación de supervectors

Gracias a los supervectors se dio solución a la problemática de la variabilidad de la duración de las muestras pero, como contrapartida, la cardinalidad del espacio vectorial de las muestras había crecido en una gran proporción. Al ser una concatenación de vectores de medias extraídas de una mezcla de gaussianas, la dimensión de un supervector es directamente proporcional a la

cantidad de densidades utilizadas en la mezcla; por lo que podrá variar entre distintos trabajos. Mas allá de esto, con el fin de dar una noción de magnitud, se observa en el trabajo realizado por Campbell [11] la utilización de una mezcla de 2048 densidades, obteniendo así un vector de dimensión 2048 * N_μ, siendo esta ultima la cardinalidad del vector de medias. Esta alta dimensionalidad trae aparejado problemas a la hora de procesar, almacenar y clasificar dicha información. Debido a esto surgió una nueva inquietud la cual preguntaba si realmente es necesario contar con tantas dimensiones o si varias de ellas contienen información redundante que pueda ser representada en un espacio de menor dimensión. Siguiendo la dirección de este análisis nacieron los **i-vectors**, los cuales según Dehak [15] son *un enfoque data-driven para la extracción de features que proporciona un marco elegante y general para la clasificación e identificación de audios*. Por lo tanto, gracias a la utilización de herramientas de reducción de dimensionalidad como el análisis factorial, nacen estos nuevos vectores cuya dimensión es menor a la de los *GMM Supervectors* y mantienen gran parte de la información contenida en ellos. Este último, antes de la era de las redes neuronales, fue considerado el estado de arte por muchos investigadores para la construcción de vectores de *embeddings* de señales de habla.

Capítulo 4

Redes Neuronales en la detección de hablantes

En el capitulo De los modelos gaussianos a los i-vectors se realizó un recorrido sobre los orígenes del *Speaker Recognition*, particularmente sobre las técnicas más sobresalientes utilizadas en la detección de hablantes. Primero se comenzó hablando de técnicas probabilísticas no supervisadas basadas en modelos gaussianos mixtos para luego decantar en métodos enfocados a la creación de vectores de características, llamados *i-vectors* [54]; los cuales requieren de un clasificador que lleve a cabo la detección del hablante. Los *i-vectors* fueron el estado de arte de numerosos trabajos, pero con el advenimiento de las redes neuronales y su gran abanico de ventajas, se comenzaron a proponer numerosas soluciones basadas en estas.

La utilización de *deep learning* en la detección de hablantes puede dividirse en dos grandes enfoques: *deep learning* para la extracción de *features* y *deep learning* extremo a extremo [5]. El primer enfoque consiste en reemplazar los *i-vectors* por vectores de *features* obtenidos a partir del entrenamiento de una red neuronal. La red es entrenada mediante muestras obtenidas a partir de *features* acústicos (extraídos de la señal de voz) y la identidad del hablante como variable objetivo. Una vez entrenada, se elige alguna de sus capas intermedias como la nueva salida de la red, a partir de la cual se obtendrán los *embeddings* de las señales de habla. Finalmente, estos nuevos vectores serán utilizados como entrada de un clasificador ubicado

aguas abajo del sistema. La segunda estrategia consiste en utilizar las redes neuronales tanto para clasificación como para la toma de decisión. Este trabajo utilizará el primer enfoque, para el cual, se han desarrollado numerosas técnicas, entre las que se destacan los *d-vector*, *j-vector* y *x-vector*, descritas a continuación.

La técnica de los **d-vectors** o **deep-vectors** fue una de las primeras implementaciones de redes neuronales para la generación de *embeddings* de señales de habla. Tal como muestras [53], los *d-vector* se obtienen a través del entrenamiento de una DNN (*Deep Neural Network*) con múltiples capas ocultas totalmente conectadas. El objetivo del entrenamiento consiste en llevar adelante la detección de los hablantes a nivel de cuadro o *frame* ya que el conjunto de entrenamiento se encuentra compuesto por vectores de *features* obtenidos a partir de un ventaneando de la señal de habla. Para llevar adelante el entrenamiento, primero se deben extraer las características espectrales de la señal habla a través de un banco de filtros. Luego, para brindar contexto temporal a los cuadros, se concatena el cuadro a detectar con 30 cuadros mas a su izquierda y 10 a su derecha. Además, se establecerá la variable objetivo como un vector *one-hot* obtenido de la etapa de salida de la red y cuya dimensión es igual al número de hablantes que componen el universo de muestras. Una vez que la DNN se ha entrenado con éxito, se eliminará su última capa, usando las activaciones de su penúltima capa para así extraer los *embeddings* de cada uno de los cuadros. Es decir, para cada cuadro de una señal de habla se genera un vector de *embeddings* que luego se promediaran para obtener un *embedding* que represente a la muestra en su totalidad. Estos son los *d-vector* y han demostrado en numerosos trabajos una mejoría general respecto a los *i-vector*.

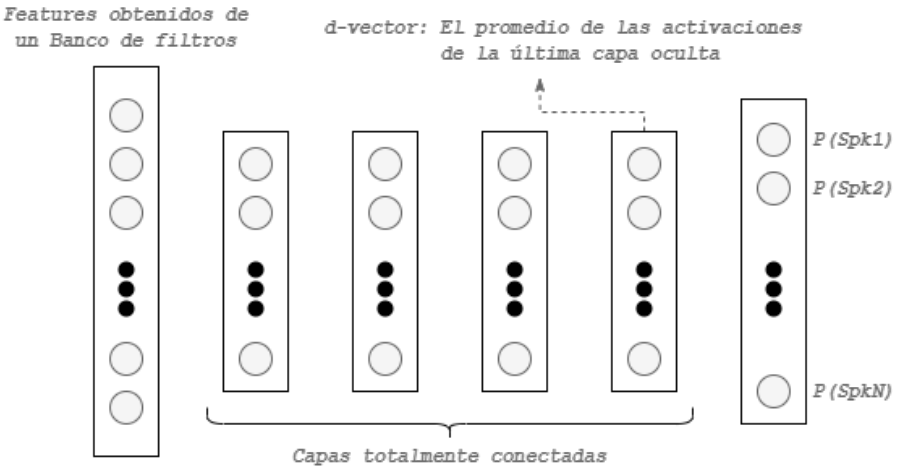

Figura 4.1: Arquitectura de red neuronal generadora de d-vectors

La técnica de los **j-vectors** o **joint-vectors** consiste en una generalización del concepto de *d-vectors* [31] la cual se basa en un entrenamiento con múltiples variables objetivo. El concepto detrás de este enfoque es la intuición de que puede resultar difícil reconocer directamente a un hablante sólo por sus vectores de *features*, ya que en realidad diferentes hablantes tendrán su propio estilo de pronunciación en cada sílaba o palabra. Por lo tanto, se propone utilizar como variable objetivo no solo la identidad del hablante sino también los textos pronunciados, generando así un aprendizaje múltiple que permite aumentar el rendimiento de la red neuronal en la identificación de un hablante. Para llevar adelante el entrenamiento se utilizará una función de costo compuesta por dos términos, una asociada a la detección del hablante y otra asociada a la detección del texto enunciado. Tal como en los *d-vectors*, una vez que la DNN se ha entrenado con éxito, se eliminará la última capa y se utilizara las activaciones de su penúltima capa para extraer los *embeddings* de cada cuadro, los cuales se promedian para obtener el *embedding* de la muestra, también llamado *j-vector*. Así como existen numerosos trabajos que muestra una superioridad de los *d-vectors* frente a los *i-vectors*, sucede los mismo con los *j-vectors* respecto a ellos dos; tal como se muestra en los experimentos realizados por Larcher [29] sobre la base de datos RSR2015.

El enfoque de **x-vectors** son una evolución de los *d-vectors*, los cuales se caracterizan por primero realizar un procesamiento a nivel de cuadro y luego, mediante un proceso de

agregación estadística, a nivel de muestra. Tal como se especifica en *x-vectors: robust dnn embeddings for speaker recognition* [47], la red neuronal que construye los *x-vectors* recibe a su entrada los vectores de *features* de T cuadro obtenidos mediante una ventana deslizante de aproximadamente uno 25 milisegundos. Luego, cada una de las capas que integran la *frame-layer* (figura 4.2) procesa los cuadros dentro de un contexto temporal, es decir que se procesa un cuadro en conjunto con sus vecinas. Por ejemplo, la primera capa tendrá un contexto de cinco *frames* (intervalo de dos cuadros a la izquierda y dos a la derecha) la cual alimentará una siguiente capa que, a su vez, impondrá un nuevo contexto temporal a su entrada. Este comportamiento expandirá la amplitud del contexto inicial y se repetirá hasta la última capa antes de la agregación estadística. Debido a este procesamiento con contextos temporales, es que la arquitectura se denomina *Time-Delay NN* (TDNN). Una vez llegado a la capa de agregación estadística (*segment-level* o *feedforward*), se extraerá la media y desvío estándar a cada uno de los T *embedding* obtenidos por cada cuadro. Luego, estos serán concatenados para así obtener un *embedding* por cada muestra. Finalmente, la arquitectura quedara compuesta por unas primeras capas con retardos de tiempo, una capa de agrupación estadística y una sección *feedforward*, las cuales serán entrenadas conjuntamente para la detección de N hablantes; quedando definidos los x-vector como los vectores extraídos desdén la segunda hasta la última capa oculta de la red *feedforward*, tal como se observa en la imagen.

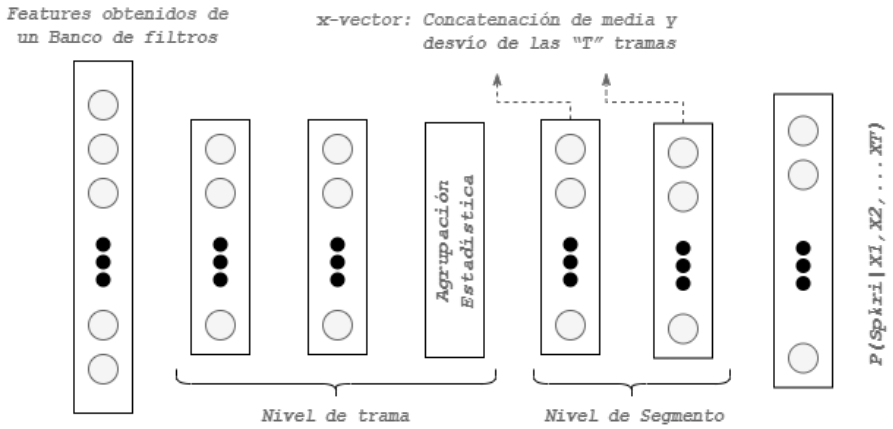

Figura 4.2: Arquitectura de red neuronal generadora de x-vectors

Es importante destacar que los *x-vectors* han sido el nuevo estado de arte frente a los *i-*

vectors, *j-vectors* y *d-vectors* gracias a la obtención de un rendimiento superior a las demás y, además, son utilizados como base de muchos de los trabajos con nuevos enfoques en la resolución del problema de identificación de hablantes. Teniendo en cuenta estas realidades es que este tipo de arquitectura será la utilizada para este trabajo de tesis.

Capítulo 5

La extracción de características

Habiendo repasado los diferentes enfoques de las secciones anteriores, se llega a la conclusión de que las redes neuronales son una solución superadora a los enfoques clásicos mostrados en el capitulo 3. Pero, un aspecto fundamental que no fue mencionado hasta el momento fue que las redes de ese capitulo 4 requieren de una etapa previa que les faciliten a su entrada vectores con las características de cada una de las señales. Estos, son los denominados bancos de extracción de características.

La mayoría de las implementaciones de estos extractores de características consisten en bancos de filtros construidos a mano, como por ejemplo los MCC (Mel-cepstral Coefficients) [12] Estos, se sustentan a partir de pruebas perceptivas y no presentan garantías de que dichas representaciones sean óptimas para todas las tareas relacionadas con el habla. Segun Ravanelli y Bengio en su artículo *Speaker recognition from raw waveform with sincnet* [39] los bancos de filtros comúnmente utilizados generan efectos nocivos sobre las señales de habla ya que suavizan su espectro, dificultando la extracción de propiedades cruciales como el pitch y los formantes. Para mitigar estos efectos, algunos artículos han propuesto alimentar directamente a la redes neuronales con *bins* de espectrogramas o incluso con las mismas señales definidas en el dominio temporal. Respecto a este último enfoque, las redes neuronales convolucionales o CNN son las arquitecturas más populares para procesar muestras de voz.

Una de las regiones de mayor criticidad de las CNN es la primera capa convolucional, la

cual alberga los filtros que se encargan de extraer los *features* de las señales de habla. Estos filtros entrenados a menudo toman formas ruidosas e incongruentes (especialmente frente a pocas muestras de entrenamiento), teniendo algún sentido para la red neuronal pero poca interpretabilidad para las personas. Para subsanar estas limitaciones es que Ravanelli y Bengio proponen en su articulo [39] una nueva arquitectura convolucional llamada **SincNet**, la cual consiste en la implementación de un arreglo de funciones Sinc parametrizadas. En la siguiente fórmula se puede observar la expresión matemática que caracteriza a una convolución entre una porción de una señal (x[n]) y un filtro (h[n]) de tamaño L, ambos discretos.

$$y[n] = x[n] * h[n] = \sum_{i=0}^{L-1} x(l).h(n-l)$$

Las CNN tradicionales deben aprender todos los componentes de cada uno de los filtros que utilizan, en cambio en el enfoque SincNet solo se deben aprender las frecuencias de corte (bajas y altas). Dado esto es que la fórmula anterior puede escribirse de la siguiente manera:

$$y[n] = x[n] * g[n, \theta]$$

Aquí surge la función $g[n, \theta]$ que describe al filtro y depende solamente de los parámetros contenidos en θ, los cuales serán aprendidos por la red en su etapa de entrenamiento. Al realizar convoluciones en el dominio temporal sobre sincs, se estará implementando un filtrado pasa-banda en el dominio frecuencial (la transformada de fourier de una sinc es una función rectangular). Este tipo de filtrado es común en el área del procesamiento de señales, por lo que Ravanelli y Bengio proponen seguir esta lógica y así poder formular matemáticamente el filtro propuesto como la diferencia entre dos filtros de pasa-bajo:

$$G[f, F_1, F_2] = rect(\frac{f}{2F_2}) - rect(\frac{f}{2F_1})$$

Donde F_1 y F_2 son las frecuencias de corte bajas y altas aprendidas en el entrenamiento, y $rect(.)$ es una función rectangular en el dominio de la frecuencia. Una vez definida la estructura del filtro, se aplicara la inversa de la transformada de Fourier para obtener su función de

transferencias en el dominio temporal.

$$g[n, F_1, F_2] = 2F_2 sinc(2\pi F_2 n) - 2F_1 sinc(2\pi F_1 n)$$

Donde la función $sinc(x) = sin(x)/x$. Las frecuencias de corte se puede inicializar dentro del rango comprendido entre el cero y la mitad de la frecuencia de muestreo (frecuencia de Nyquist)

Ahora, todas las definiciones se sustentan sobre formalidades teóricas e ideales y, al implementar filtros digitales, se esta lejos de la idealidad, por lo que estos presentaran ondulaciones en la banda de paso (donde el filtro idea seria plano) y atenuación limitada en la banda suprimida (donde el filtro ideal seria cero). Una solución popular para mitigar este problema es la creación de ventanas para luego multiplicarlas por la función de transferencia g con objetivo aminorar estos efectos.

$$g_w[n, F_1, F_2] = (2F_2 sinc(2\pi F_2 n) - 2F_1 sinc(2\pi F_1 n)).w[n]$$

Aquí se ha llegado a la formula definitiva que gobierna las redes SincNet, logrando una forma compacta y eficiente de la cual se podrá derivar un banco de filtros personalizado ajustado específicamente para la aplicación deseada. Los experimentos realizados por Ravanelli y Bengio muestran que para tareas de identificación y verificación de hablantes, la arquitectura propuesta converge más rápido y funciona mejor que los enfoques tradicionales.

En vistas a la realización de este trabajo y tomando el razonamiento generado en este capitulo, se decidió ir hacia un enfoque de extracción de características a través de una red neuronal convolucional de tipo SincNet.

Capítulo 6

Los datos

Uno de los mayores desafíos en el campo del reconocimiento del habla es la poca disponibilidad de datos abiertos a la comunidad. La mayoría de los datos de voz son propietarios (se deben abonar), de difícil acceso y en variadas ocasiones presentan un etiquetado de baja calidad y son ruidosos. Además, el entrenamiento de modelos de detección de hablantes a través de datos limpios, estandarizados y tomados de micrófonos de gran calidad carece de sentido para aplicaciones modernas debido a que los usuarios hoy en día interactúan con ella a partir de una gran diversidad de dispositivos electrónicos como tabletas, teléfonos celulares, altavoces inteligentes, entre otros. Estos dispositivos, buscan tener un alcance masivo y lograr penetrar lo mayor posible en todos los estratos de la sociedad, por lo que la calidad de sus componentes estará definida por la estrategia de mercado de su fabricante y esta, no siempre sera la mejor.

Los teléfonos celulares o *smartphones* son uno de los principales dispositivos utilizados por las personas en sus actividades diarias y son una tecnología transversal a la mayor parte de la sociedad (género, edad, nivel socioeconómico, entre otros). Estos, son utilizados continuamente para la generación y envío de contenido audiovisual, por lo que resultan una fuente muy atractiva para la construcción de *datasets* de entrenamiento. En lo que respecta a las aplicaciones utilizadas mayormente para la generación de contenido, se tiene dos grandes familias: las redes sociales y las aplicaciones de mensajería, las cuales en la actualidad poseen un límite cada vez más difuso. Dentro de las aplicaciones de mensajería, WhatsApp se encuentra en el podio de

las aplicaciones que presentan un predominio en los mercados de estas latitudes.

El objetivo de todo sistema de *machine learning* con impacto directo en la población es el de brindar un servicio que solvente necesidades de una manera efectiva y justa, por lo que es necesaria la construcción de *datasets* que reflejen la gran diversidad de dispositivos electrónicos utilizados por la sociedad para la toma y generación de contenido audiovisual. Estos *datasets* nos permitirán entrenar modelos con un menor sesgo y a su vez, construir sistemas basados en *machine learning* mas robustos. Teniendo en cuenta todas las premisas presentadas es que se ha decidido utilizar en este trabajo de tesis los mensajes de audio de WhatsApp como fuente del *dataset* de entrenamiento, lo cual tendrá acarreado conceptos de calidad y consistencia de los datos, abordados a continuación.

6.1. La calidad de los datos

Un algoritmo de *machine learning* busca definir reglas de decisión en base a patrones que extrae de los datos con los que fue entrenado. Esto implica que se debe prestar especial atención a la calidad de los datos con que se va a entrenar dicho algoritmo, ya que esto impactara directamente en su capacidad de generalización. El término *Garbage In, Garbage Out*, es bien conocido en el ámbito de las ciencias de datos y busca esquematizar que aunque tengamos un algoritmo muy sofisticado, no tendremos buenos resultados si los datos de entrenamiento son de baja calidad. En resumen, un algoritmo será tan bueno como sus datos de entrada y, por lo tanto, es un punto crítico a tener en cuenta.

Proponer que los datos utilizados para entrenar algoritmos sean de calidad parece trivial, pero según el articulo *Everyone wants to do the model work, not the data work* publicado por Google Research [46] se tiene evidencia empírica de que muchas fallas en los sistemas de predicción se deben a la propagación de problemas en los datos. Esta problemática es denominada *Data Cascades* y está directamente relacionada a prácticas convencionales de IA/ML que subestiman la calidad de los datos. Según el documento, esta subestimación se debe a que este trabajo es visto como operacional comparado con la construcción de nuevos modelos y, por lo

tanto, la mayoría de las organizaciones no logran crear o cumplir un estándar de calidad de datos.

Para afianzar la calidad de los datos es primordial realizar una buena recolección de datos detectando sus bondades y deficiencias, para así poder evitar sesgos. Existen diferentes técnicas de sintetización de datos que permiten solventar algunas de estas limitaciones como por ejemplo, técnicas para balancear clases pero en el ámbito productivo generalmente la mejor respuesta es el inicio de una nueva etapa de recolección. Otro aspecto importante es el concepto de *Data Provenance* y *Data Lineage*, el cual hace hincapié en conocer el origen de los datos y registrar las transformaciones utilizadas en la construcción de los conjuntos de datos de entrenamiento. Siguiendo esta línea, el articulo *Datasheets for Datasets* [20] destaca la importancia de realizar una correcta documentación de ellos y, que en al actualidad no existen procesos estandarizados para su documentación.

Siguiendo en el camino hacia una mejor calidad de los datos, tenemos a la etapa de preprocesamiento como un elemento de gran criticidad, el cual busca corregir deficiencias en los datos que puedan dañar el aprendizaje como son el ruido o valores extremos. En este trabajo se utilizará una etapa de preprocesamiento para adecuar las señales de habla, las cuales se utilizarán posteriormente para alimentar una red neuronal cuyo propósito es el de generar vectores de *embeddings* de los audios. Según el informe *Un Estudio sobre el Preprocesamiento para Redes Neuronales Profundas* [37] se puede observar que más allá de que las redes neuronales presentan una gran flexibilidad para el aprendizaje de patrones complejos, reduciendo la necesidad de una preparación manual de los datos, sigue siendo importante un preprocesamiento adecuado para obtener resultados de calidad. En este texto se hace referencia a un ejemplo de detección de dígitos manuscritos, pero se extiende a todo tipo de problema.

Como conclusión, podemos enfatizar que los datos de la vida real contendrán numerosas limitaciones, por lo que el camino hacia la construcción de *datasets* de calidad se encuentra acompañado de numerosos desafíos. Estos, deberán detectarse y solucionarse al momento de querer implementar sistemas basados en *machine learning* en ambientes productivos

6.2. Construcción y estructura de los datos

En el marco de la construcción de un *dataset* que sea representativo de la sociedad, WhatsApp nos brinda una oportunidad única ya que sus mensajes de voz nos permiten obtener datos de gran diversidad y diferente calidad. Estas señales de habla contendrán modismos, silencios, ruidos, reverberaciones, diferentes tamaños y contendrán sentimientos y emociones; típicos de conversaciones de la vida cotidiana. Toda esta rica información puede ser accedida fácilmente (especialmente en Android) ya que se encuentra almacenada dentro los dispositivos celulares. Es por esto, que el *dataset* que se utilizara en este proyecto estará compuesto por los mensajes de voz que fueron enviados y recibidos por el autor de este documento. Además de los audios, Whatsapp permite exportar la metadatos de cada conversación, los cuales será utilizados para obtener las etiquetas de los hablantes de cada archivo. Cabe aclarar que el contenido de los audios utilizados no serán divulgados y solo serán utilizados con fines académicos y, además, quedará enmascarado detrás de los *embeddings* que los representan.

Los audios se encuentran almacenados en un formato de archivos llamado "opus". Este es un formato de audio con pérdida, el cual fue creado por la Internet Engineering Task Force (IETF) con el objetivo de ser eficiente en la transmisión a través de internet. Esta codificación se caracteriza por ser de mejor calidad que otros formatos de audios más conocidos y permite codificar el sonido en tiempo real. Estos archivos, se encuentran almacenados en estructuras de carpetas que los segmentan a través del tiempo y no guardan ningún identificador que los relacione a un determinado mensaje de voz. Debido a las limitaciones de codecs soportados por las librerías utilizadas, se realizó un conversión de formato "opus" a "ogg", ambos, de código abierto.

Para la detección de la identidad del hablante, se utilizarán diversas técnicas de aprendizaje supervisado, por lo que es necesario contar con las etiquetas que identifiquen a los hablantes de cada audio. Por la naturaleza de los datos, no tendremos un problema de *Speaker Diarization* ya que no se cuenta con múltiples hablantes en un mismo audio. Luego de exportar los metadatos de cada conversación, se realizó un análisis gramatical ad-hoc mediante expresiones regulares

para obtener las identidades de los hablantes de cada audio. Finalmente, se realizó un filtrado de los archivos que poseen etiquetas obteniendo el conjunto de datos definitivo, el cual está compuesto por 761 audios generados por 12 hablantes el cual fue dividido en un conjunto de entrenamiento (80 %) y validación (20 %). La distribución de audios por hablante no es uniforme, por lo que se tiene una mayor proporción de muestras provenientes del dueño del dispositivo celular.

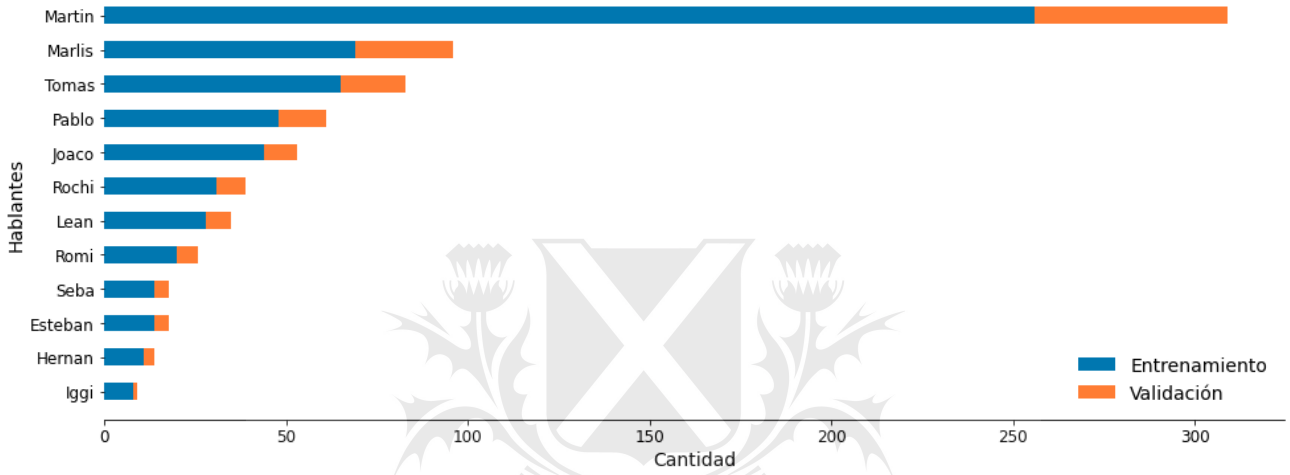

Figura 6.1: Distribución de clases (hablantes) en el conjunto de datos

En lo que respecta a la duración de los audios se evidencia una diversidad en su longitud, pero la mayoría se encuentra por debajo del minuto salvo algunas excepciones. Este comportamiento es habitual ya que los audios generalmente buscan transmitir mensajes concisos y dependen en gran medida del estilo de los hablantes y, gracias a la utilización de las redes SincNet, no existe una problemática en estas diferencias. Si nos enfocamos en la diversidad de hablantes tendremos un universo compuesto por aproximadamente 3 mujeres y 9 hombres, lo cual es importante ya que con esto se pueden estudiar características particulares de la señales de habla de hombres y mujeres como por ejemplo, los rangos de frecuencias y amplitudes de sus armónicos. Algunas de las diferencias presentes es que las mujeres tienden a tener frecuencias más altas (agudas) y los hombre más bajas (graves).

Con el objetivo de poder visualizar estos comportamientos es que, partiendo de un audio perteneciente a un hablante masculino y uno femenino, se realizara un análisis descriptivo de

cada una se la señales. Este análisis sera llevado a cabo con las herramientas mas utilizadas en el procesamiento de señales, las cuales fueron descriptas en la sección La transformada de Fourier y se verán implementadas a continuación.

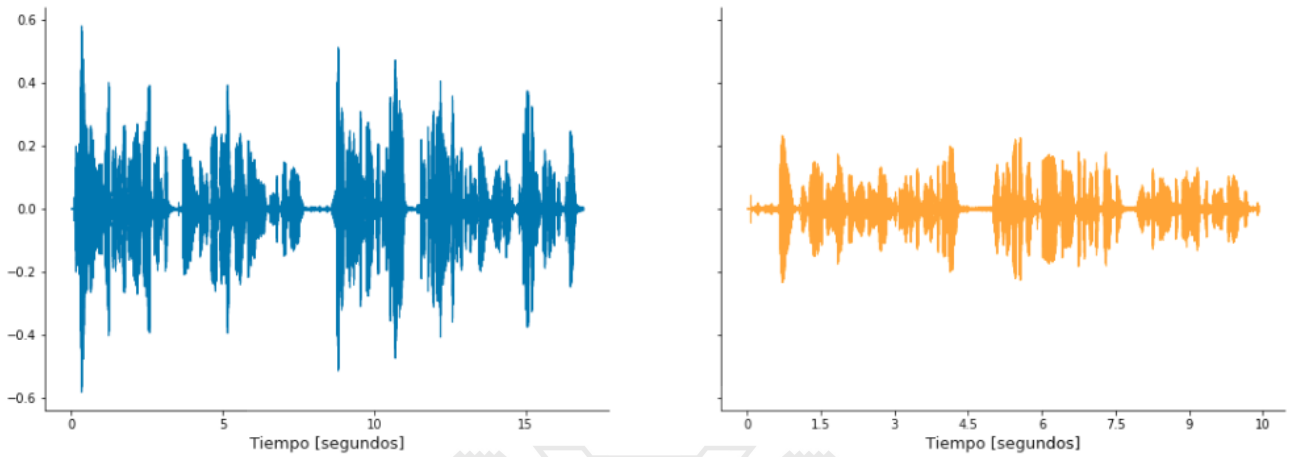

Figura 6.2: Forma de onda de señales de habla masculina y femenina

En la figura 6.2, se evidencia a la izquierda una señal de habla de un hombre (azul) mientras que la de la derecha (naranja) se tiene una hablante mujer. A simple vista, en el dominio del tiempo solo puede detectarse diferencias en sus amplitudes, pero es en el dominio frecuencial donde se enriquece el análisis. Para ello se llevara a cabo un espectrograma, el cual mediante la utilización de la FFT sobre una ventana de tiempo deslizante, permitirá detectar cualidades representativas de cada audio.

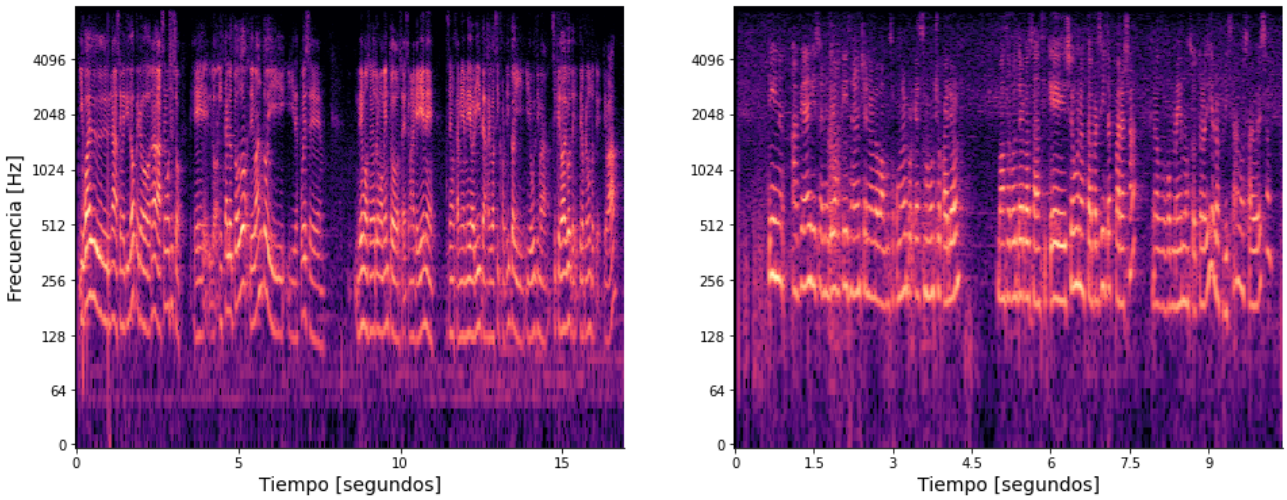

Figura 6.3: Espectrograma de señales de habla masculina y femenina

En la figura 6.3 se puede observar la ubicación, en el espectro de frecuencias, de los formantes que componen ambas señales como así también el rango de frecuencias en que se ubican los armónicos de cada señal a lo largo del tiempo. Otro aspecto a destacar es que los tonos con mayor amplitud se encuentran en la banda de frecuencia por debajo de los 4 khz, algo esperado y bien utilizado en los canales de comunicación telefónicos. Finalmente, se evidencia un corrimiento ascendente de los tonos, o armónicos, presentes el espectro de habla femenino (lineas con mayor brillo) comparado a el masculinos, mostrando así que estos poseen un sonido mas agudo.

Capítulo 7

Metodología

Habiendo recorrido los diferentes conceptos teóricos que poseen preponderancia dentro del universo de problemas en la detección de hablantes, se decidió construir un sistema que aborde todas estas problemáticas. Esto, implicara el diseño y construcción de un conjunto de etapas que integraran un sistema automático de identificación de hablantes. Tal como se comento en el documento, se implementara una etapa de preprocesamiento, seguido por una etapa de generación de *embeddings* y, finalmente, una etapa de clasificación de tipo multiclase. Esta ultima etapa sera llevada a cabo por un grupo de algoritmos bien conocidos y mencionados recurrentemente en gran parte de los artículos de investigación referenciados en este trabajo. En las siguientes secciones se detallaran las funciones de cada etapa, las herramientas utilizadas en ellas y cómo impactaran sus transformaciones sobre las señales de habla. Una vez descrita y ejemplificada esta metodología, se transitara hacia el capitulo Resultados, el cual evaluará la calidad de las clasificaciones realizadas por los distintos algoritmos seleccionados.

7.1. Descripción general del modelo

El sistema de *Speaker Identification* implementado se encontrará compuesto por tres macro-bloques que se ensamblarán en serie, cada uno con una tarea específica. Cada uno de estos macro-bloques estarán a su vez compuesto por un determinado número de micro-bloques, que se encargaran de llevar adelante tareas más específicas y acotadas. A continuación se presentan

los tres macro-bloques o etapas que componen el flujo de trabajo (también conocido como *pipeline*) necesario para poder entrenar el modelo y clasificar las señales de habla:

- Preprocesamiento: Esta etapa se encargará de recolectar un conjunto de muestras compuestas por múltiples señales de audio de diferentes hablantes, para luego limpiarlas y acondicionarlas. Esto, buscara resaltara los aspectos distintivos de un hablante particular y eliminar contenido de baja utilidad. Dentro de esta etapa se tendrá un proceso de eliminación de silencios, luego un proceso de preénfasis y finalmente un filtrado de ruido de fondo.

- Speaker embedding: Esta etapa se encargará de transformar las señales de habla, definidas en el tiempo, en vectores de *embeddings*. Los *embedding* serán representaciones vectoriales de los audios, los cuales buscarán captar los atributos que mejor distinguen un hablante de otro. Dentro de esta etapa se tendrá primero un proceso de extracción de *features*, luego un proceso de generación de *embedding* y finalmente un proceso de reducción de dimensionalidad.

- Backend: Esta etapa estará compuesta por un algoritmo de clasificación, el cual deberá ser entrenado con los *embedding* obtenidos de las muestras, para así ajustar sus inferencias los mejor posible a los hablantes que componen el conjunto de datos. Una vez entrenado, será el encargado de dictaminar que hablante ha generado cada audio. Esta clasificación se basará en las similaridades que presenten los diferentes vectores en el subespacio que los alberga.

Cada una de estas etapas impactarán directamente sobre el rendimiento de su sucesora, por lo que es de vital importancia respetar su orden de ejecución y su forma de interacción. En la siguiente figura se ejemplifican dichos bloques.

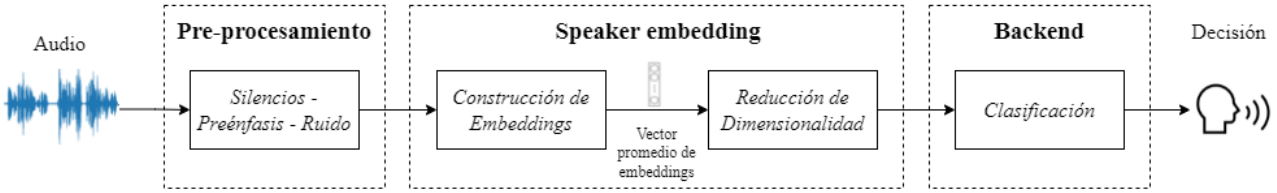

Figura 7.1: Diagrama en bloques del sistema identificador de hablantes

Tal como se puede observar el sistema contará con una primera etapa adaptación y limpieza de señales de audio, para luego ingresar al bloque de generación de *embeddings* de voz. Este bloque subdividirá a la señal en diferentes cuadros y generará un *embedding* por cada una de ellas para luego construir mediante un promedio el vector de *embeddings* que represente al audio en su totalidad. Luego, estos vectores transitaran una etapa de reducción de dimensionalidad, la cual bajara notablemente su cardinalidad. Finalmente, los vectores conformarán un nuevo conjunto de datos que será utilizado para entrenar y validar diferentes tipos de algoritmos de clasificación supervisada. En esta última etapa se realizaron diversas pruebas con el objetivo de encontrar la mejor combinación de un algoritmo con sus correspondientes hiperparámetros. Es importante comentar que una vez elegido el clasificador, todas las etapas mencionadas deberán implementarse tanto para un nuevo reentrenamiento del modelo como para el sistema de inferencias.

En las siguiente secciones se detallarán cada uno de los diferentes bloques funcionales que componen el sistema y, también, los módulos que los integran. Además, se presentan las herramientas computacionales utilizadas en cada una de las etapas junto con ilustraciones que muestran como impactan cada una de estas transformaciones en una señal de habla.

7.2. Preprocesamiento

Tomando como referencia lo expuesto en la sección Los datos, se evidencia que en los comienzos de todo pipeline de reconocimiento de hablantes es muy importante incorporar una etapa de preprocesamiento que permita limpiar y acondicionar las señales de audio. Sobre ellas se deberán aplicar una serie de transformaciones que permitan eliminar todo fenómeno no

deseado que se incorpora a la señal de habla desde que es generada hasta que es capturada por un micrófono. De esta forma se podrá enviar a etapas subsiguientes una señal con mayor porcentaje de información en el tiempo que permita realizar una extracción de *features* eficiente y focalizada solo en la porción de habla, la cual contiene atributos específicos y distintivos que identifican al hablante.

En la actualidad los micrófonos de los dispositivos móviles han mejorado su calidad, logrando captar con mayor nitidez la señal objetivo (Habla) que, junto con la utilización de redes neuronales, permiten la construcción de sistemas de extracción de *features* cada vez más robustos. Más allá de estas consideraciones, el campo de preprocesamiento de señales sigue estando en constante evolución permitiendo el mejoramiento de las señales de audio independientemente de la calidad con que fueron captadas. Para poder llevar a cabo este acondicionamiento, se utilizaron diferentes tipos de filtros digitales, cada uno centrado en una tarea específica. En la literatura científica [23] se encuentra una gran variedad técnicas de preprocesamiento, de las cuales se utilizaron las siguientes:

- Filtro de Silencios o VAD.
- Filtro de Preénfasis.
- Filtro de ruido de fondo.

Todos ellos serán de gran importancia en el tratamiento de las muestras y deberán llevarse a cabo en un orden en que fueron nombrados. A continuación se detallarán cada uno de esto y de ejemplificaba cómo impacta su aplicación sobre una señal testigo elegida aleatoriamente del conjunto de entrenamiento.

7.2.1. Filtro de silencios

El habla es discontinua ya que a menudo tenemos pausas entre las oraciones e incluso dentro de las mismas, por lo que un audio puede contener silencio en varias posiciones como en el comienzo, entre las palabras de una oración o al final del audio. Si se eliminan estos

silencios se conseguirá reducir el tiempo y la complejidad de los cálculos algorítmicos en el proceso de extracción de características. Por lo tanto, existe un potencial ahorro de recursos de cómputo si se logran eliminar las mayor cantidad de porciones de audio que no contengan habla. Típicamente, el consumo de recursos computacionales de un proceso de eliminación de silencios se encuentra entre uno y tres órdenes de magnitud [55] respecto al consumo de los modelos de *machine learning*, por lo que pueden convivir dentro del mismo sistema sin degradar su rendimiento. Es por esto, que estas etapas se encuentran presentes en la gran mayoría de los sistemas y es fundamental en cualquier proceso de preparación de datos relacionados con el análisis de señales de habla; pasando desapercibida su presencia en la mayoría de los casos por los usuarios finales.

El Filtrado de silencios o también conocido como VAD consiste en la determinación de intervalos de tiempo en donde una señal de audio presenta o no habla, siendo una decisión binaria. Para ello, primero debe definirse la probabilidad de que una señal de entrada contenga voz o no, denominada probabilidad de presencia de voz, para luego implementar un esquema de clasificación sobre un umbral de decisión. Este problema parece ser sencillo de resolver pero la dificultad se encuentra en lograr sistemas que tenga una buena generalización, debiendo funcionar razonablemente bien en ambientes con diversidad de fuentes de audio, ruido y niveles de relación señal a ruido [55]. Un VAD de buena calidad debería ser portable, logrando un equilibrio entre un bajo consumo computacional y latencia. A continuación se muestra un ejemplo de una señal de habla perteneciente al conjunto de datos de entrenamiento, la cual fue sometida a este tipo de filtrado.

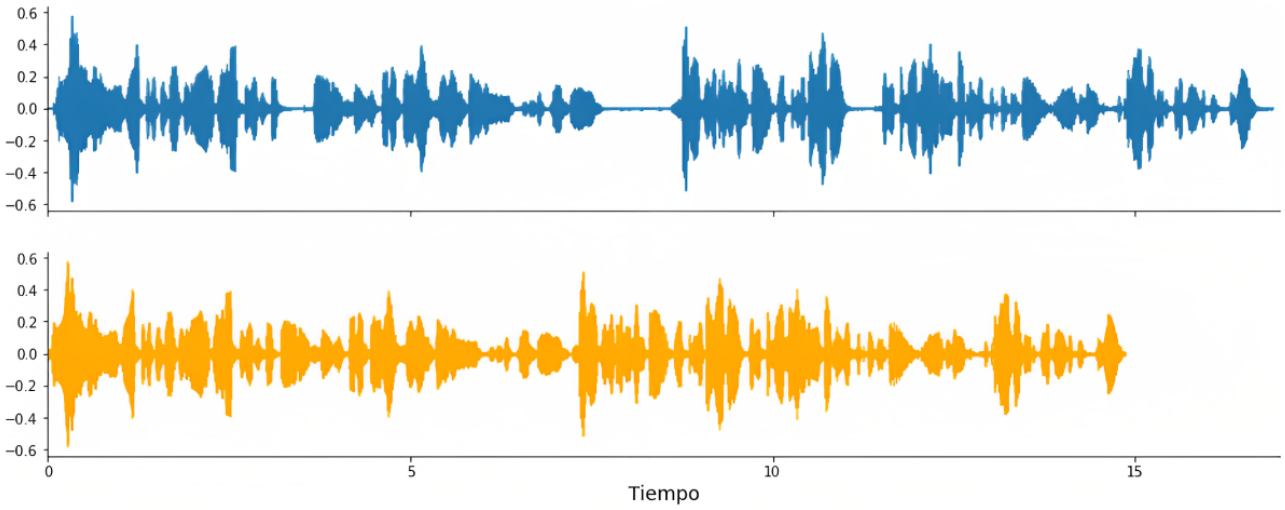

Figura 7.2: Efecto del filtrado de silencios sobre una señal de habla

En la figura 7.2 se puede observar el cambio sufrido por una señal de habla cruda (azul) luego de aplicarle un filtrado VAD (naranja). Este filtro genera una reducción del tamaño de la señal de habla a través de la eliminación de las secciones con silencios, conservando las porciones de la señal con mayor información. En este trabajo se utilizó un modelo preentrenado de uso libre llamado **Silero VAD** [50] el cual se construyó sobre una red neuronal basada en MHA (Multi-Head Attention) [22], la cual utiliza *features* obtenidos a través de la transformada de fourier de corto tiempo. Según la bibliografía, se eligió esta arquitectura debido al hecho de que las redes basadas en MHA han mostrado resultados prometedores en muchas aplicaciones como el procesamiento del lenguaje natural y procesamiento del habla. Esta se encuentra disponible en el repositorio de modelos Pytorch-Hub y consiste en un paquete de detección de voz de alto rendimiento que soporta diferentes lenguajes, entre los que se destacan Ruso, Inglés, Alemán y Español.

7.2.2. Filtro de preénfasis

El filtrado de preénfasis busca aumentar los niveles de energía que presentan las frecuencias altas del espectro de la señal de voz, debido a que a menudo los niveles de energía de las bandas altas son menores que en las bajas. Al aplicar este tipo de transformación se logra un mayor

equilibrio sobre todo el espectro de la señal mejorando la relación señal-ruido en las frecuencias altas y, además, se reduce el rango dinámico de la magnitud del espectro ayudando a disminuir potenciales problemas numéricos al momento de cuantizar estos valores.

El filtro de preénfasis se lleva adelante gracias a la implementación un filtro diferencial pasa-alto de primer orden, parametrizado con un coeficiente definido entre cero y uno; donde cero significa que la señal no sufrirá cambios y uno implicara la implementación del filtro en su totalidad. A continuación se muestra la fórmula, en el dominio del tiempo, que gobierna este tipo de filtrado.

$$y(t) = x(t) - \alpha x(t-1)$$

Es importante tener en cuenta que la utilización de un preénfasis excesivo puede generar problemas sobre las señales fricativas, ya que estas poseen altos niveles de energía en las frecuencias altas. Por lo tanto, el grado de preénfasis aplicar dependerá en gran medida del caso de uso que se requiera atacar. Para poder observar el efecto del preénfasis se tomó la señal de la sección anterior, se le aplicó el dicho filtrado y se calculó su transformada discreta de Fourier (algoritmo FFT) antes y después ello.

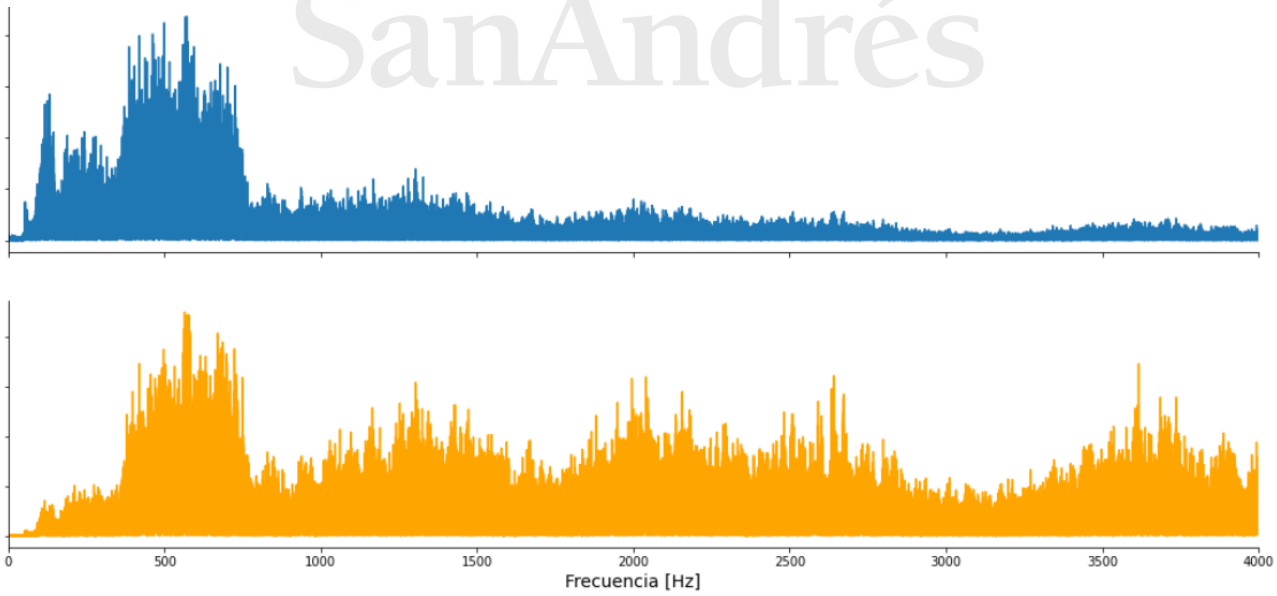

Figura 7.3: Efecto del filtrado de preénfasis sobre una señal de habla

En la figura 7.3 se puede observar el cambio sufrido por una señal de habla de la sección anterior (azul) luego de aplicarle un filtrado de preénfasis (naranja). Este filtro genera un balance más equitativo entre las magnitudes de las frecuencias altas y bajas del espectro. En este trabajo se utilizó una función llamada **effects.preemphasis** [32] perteneciente a la librería **LibROSA** y consiste en un paquete desarrollado en python para análisis de audios y videos. Esta función implementara el tipo de filtrado antes mencionado utilizando al coeficiente alpha en su valor por defecto de 0.97, el cual se toma como referencia de la herramienta HTK (Hidden Markov Model Toolkit) [57]. Esta herramienta es utilizada en el ámbito académico en la investigación de reconocimiento de voz, síntesis de voz, reconocimiento de caracteres, entre otros.

7.2.3. Filtro de Ruido de fondo

El ruido aditivo es una composición de señales espurias de alta frecuencia que se adhieren a las señales que transmiten información. Este, se encuentra presente en todos los entornos por donde transitamos ya sea por interferencia con otras señales o alteraciones generadas por el propio entorno y, por lo tanto, estará presente en las señales de habla contenidas en los archivos de audio. Es por esto que el ruido es una de las principales perturbaciones presentes en el trasfondo acústico de una comunicación hablada. Estas señales espurias son captadas por los micrófonos y, dependiendo de la calidad de los mismos podrán estar atenuadas en mayor o menor medida. En lo que respecta al *dataset* que se conformó para este trabajo, es importante destacar que no se contó con una flota de celulares de alta categoría, por lo que todos los audios presentaban algún nivel de ruido.

El oído humano puede identificar fácilmente la señal de voz en entornos ruidosos, es más, se encuentra desarrollado para llevar adelante esta tarea de forma muy eficiente pero, cuando utilizamos sistemas computacionales, esta perturbación puede generar gran impacto sobre el rendimiento del sistema. Por lo tanto, es de vital importancia la inclusión de una etapa de preprocesamiento que permita apaciguar los efectos de este fenómeno. Para llevar adelante esta tarea, se utilizó una herramienta de uso libre, la cual permitió eliminar señales reunidas de cada

muestra, en la siguiente figura se muestra el espectrograma de la señal de habla de la sección anterior, la cual fue sometida a un filtro de ruido de fondo.

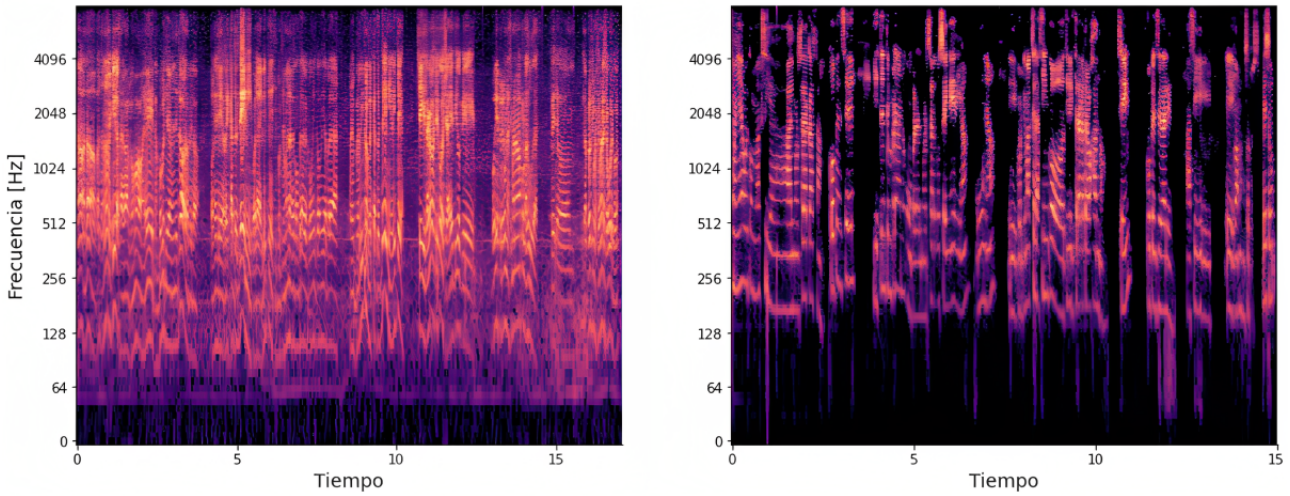

Figura 7.4: Efecto del filtrado de ruido de fondo sobre una señal de habla

En la figura 7.4 se puede observar el cambio sufrido por una señal de habla de la sección anterior (izquierda) luego de aplicarle un filtrado de ruido de fondo (derecha). Este filtro presenta una atenuación sobre las frecuencias que poseen una magnitud mucho mas baja respecto a la señal predominante, permitiendo limpiar el ruido presente en cada audio. En este trabajo se utilizó la función **nr.reduce_noise** perteneciente a la librería **Noisereduce** [45] [44], la cual consiste en un paquete desarrollado en python que utiliza un método llamado *Spectral Gating*, el cual calcula el espectrograma de la señal y estima un umbral de ruido para cada banda de frecuencia, para luego utilizarlo en la definición de qué sección del audio se debe filtrar. Finalmente, es importante aclarar que el *dataset* de entrenamiento no contaba con muestras demasiado ruidosas por lo que la herramienta utilizada fue de gran utilidad para poder eliminar las porciones de este efecto no deseado.

7.3. Speaker embedding

Una vez que las muestras transitan la etapa de preprocesamiento se inicia el proceso de construcción de los *embeddings* de la señal de habla, el cual es el corazón del sistema de reco-

nocimiento de hablantes. En esta etapa transformaremos las señales de audio en vectores de *embedding* con el objetivo de captar la mayor cantidad de rasgos distintivos de cada una, por lo cual se utilizará un red neuronal preentrenada y de alto rendimiento.

7.3.1. Construcción de embeddings

El procesamiento de señales basa gran parte de su análisis en el espectro de frecuencias, también conocido como espectro de Fourier. En este nuevo dominio, es posible detectar aspectos fundamentales que hacen al comportamiento de las señales, por lo que para lograr captar las distintas características innatas de cada señal se utiliza una herramienta conocida como Banco de Filtros. El banco de filtros, formado por un arreglo de varios filtros pasa-banda, separará la señal de entrada en varias componentes, cada una de las cuales transportara una sub-banda de frecuencias de la señal original. Al aislar diferentes componentes frecuenciales, algunas tendrán mayor importancia y podrán ser codificadas con una mayor resolución, por lo que el esquema de codificación deberá ser sensible a las pequeñas diferencias existentes entre unas y otras. Existen diversas propuestas para construir bancos de filtros, algunas completamente basadas en redes neuronales y otras más artesanales. En este trabajo, utilizaremos un enfoque llamado **SincNets**, el cual convoluciona la señal de entrada con un conjunto de funciones parametrizadas de tipo sinc, en donde las frecuencias de corte de estos filtros pasa-banda son los únicos parámetros a aprender. Por lo tanto, al utilizar SincNets se podrá obtener vectores de *features* a partir de señales definidas en el tiempo.

Luego de seccionar los audios en ventanas de tiempo y extraer sus *features* a través de la SincNet, se tendrá a su salida un conjunto de vectores de *features* los cuales son tomados por una etapa posterior encargada de la construcción de los *embeddings*. Está etapa estará constituida por una red neuronal compuesta por capas totalmente conectadas pero con la particularidad de que cada una abarca contextos temporales distintos. A estas redes se las conoce como **TDNN** o **Time Delay Neural Networks** y fueron descritas en sección Redes Neuronales en la detección de hablantes. Por lo tanto la arquitectura que compone la etapa de construcción de *embeddings* estará compuesta por dos redes neuronales, las cuales integraran una estructura

de tipo X-Vector TDNN alimentada por *features* obtenidos a través de un banco de filtros tipo SincNet.

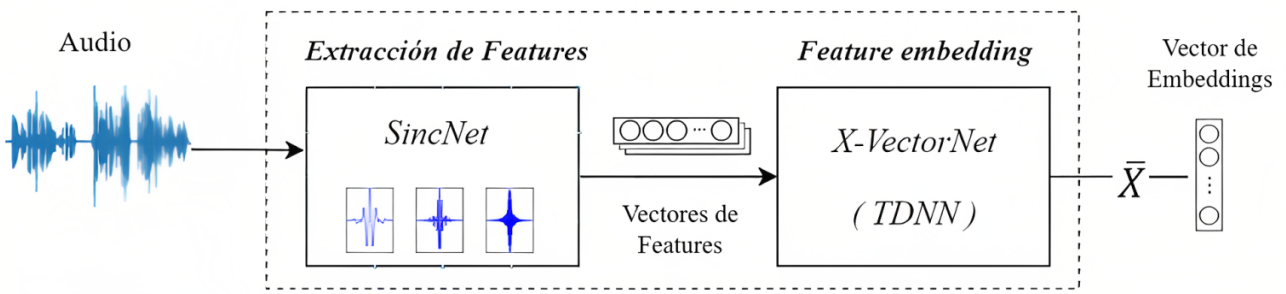

Figura 7.5: Diagrama en bloques del proceso de generación de embeddings

Para implementar esta arquitectura se utilizó la herramienta de código libre **pyannote.audio** [9] [8] construida sobre un *framework* basado en python que proporciona un conjunto de módulos compuestos por redes neuronales, las cuales pueden combinarse para construir sistemas de detección y reconocimiento de voz. Este conjunto de librerías permitirá implementar el diseño elegido para este trabajo y el cual se ejemplifica en la figura 7.5. *Pyannote.audio* incorpora una diversidad de modelos preentrenados que cubren una amplia gama de dominios en la detección y análisis de señales de voz y, en particular, en este trabajo hará uso del módulo **Speaker Embedding**, el cual se encuentra preentrenado con un conjunto de datos de uso abierto llamado VoxCeleb.

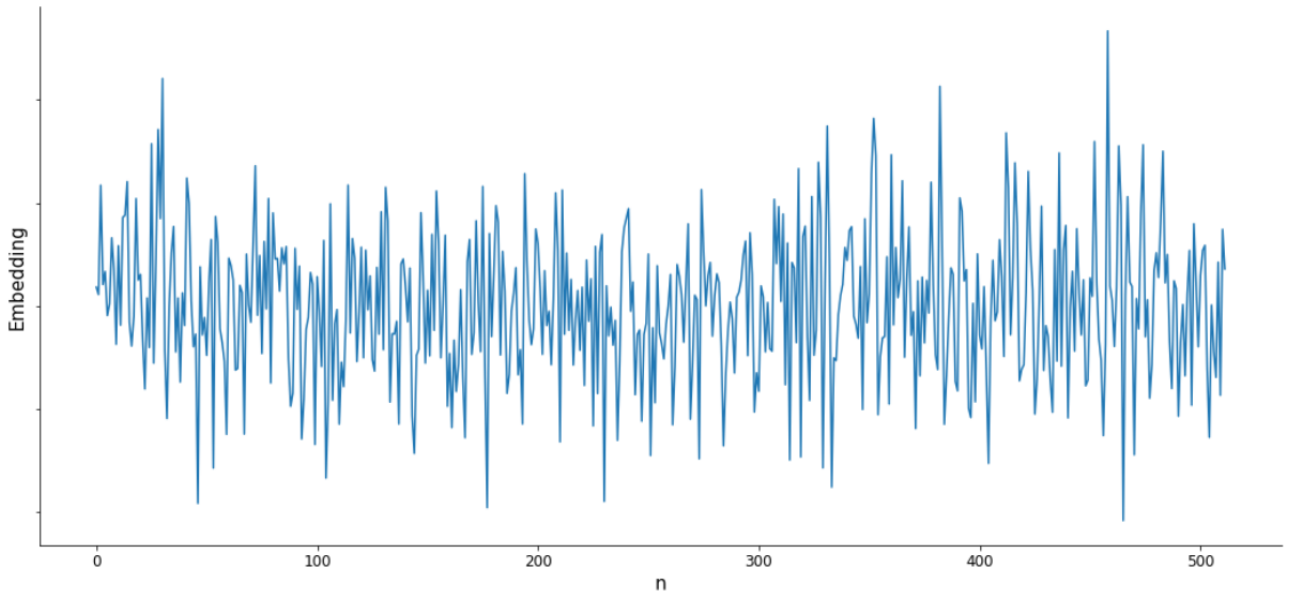

Figura 7.6: Construcción de embedding a partir de una señal de habla

Finalmente, al procesarse cada audio con el modelo provisto por *pyannote.audio* se obtendrá un vector de *embeddings* de dimensión 512 por cada sección o cuadro que compone un audio. Luego, se tomará el promedio de ellos para generar un *embedding* general que represente al audio en su totalidad. En la figura 7.6 se muestra un ejemplo de un vector de *embeddings* que resulto del procesamiento de una señal de audio obtenida a la salida de la etapa de preprocesamiento.

7.3.2. Reducción de dimensionalidad

Existen ocasiones en que poseer un gran número de características o *features* puede ser contraproducente para la generalización de los modelos en las tareas posteriores al entrenamiento. Así como es real que todas las dimensiones aportaran algo de información a la muestra, también es real que en reiteradas ocasiones esta información puede representarse en gran medida a través de su proyección sobre un subespacio de menor dimensión. Para lograr esto existe una variedad de herramientas que permiten disminuir la dimensión a costa de pérdida de interpretabilidad, algo que no nos es de gran interés aquí porque esta ya se ha perdido en su conversión a *embeddings*. Esta reducción, además de afectar la interpretabilida también genera la pérdida de un cierto porcentaje de información, definiendo una relación de compromiso entre la dimensión

del vector y el porcentaje de información capturada.

Volviendo al sistema a construir, se había comentado que se encontraba disponible a la salida de la etapa de extracción de características un vector de *embeddings* de tamaño 512, por lo que el nuevo conjunto de muestras contendrá una dimensión de esa misma magnitud. Por lo tanto las muestras estarán alojadas en un espacio vectorial de alta dimensionalidad, el cual podría generar problemas en los algoritmos de clasificación utilizados en las etapas posteriores, sobre todo en los basados en distancias. Dado esto, nos hacemos la siguiente pregunta ¿Es posible proyectar los *embeddings* hacia subespacios de menor dimensión, reteniendo la mayor información posible?

Para responder esta pregunta será necesario sumergirse en el universo de algoritmos de reducción de dimensionalidad, para luego realizar diferentes pruebas sobre el conjunto de muestras y así validar si es redituable aplicar alguna de estas técnicas. De todas las técnicas, se decidió ir por una de fácil implementación, bien conocida por el autor de este trabajo, simple de entender y, además, presente en la mayoría de la bibliografía utilizada como referencia [23]. Bajo estas consideraciones es que se decidió utilizar el algoritmo no supervisado Análisis de componentes principales, a través de la clase **sklearn.decomposition.PCA** provista por la librería de código python llamada **Scikit-Learn** [36]. Gracias a esta librería es posible implementar proyecciones que permitan obtener visualmente un panorama de cómo se encuentran distribuidas las clases y así intuir de antemano cuán fácil o difícil será su clasificación. En la siguiente ilustración se puede observar la proyección del conjunto de entrenamiento en sus dos primeros componentes principales.

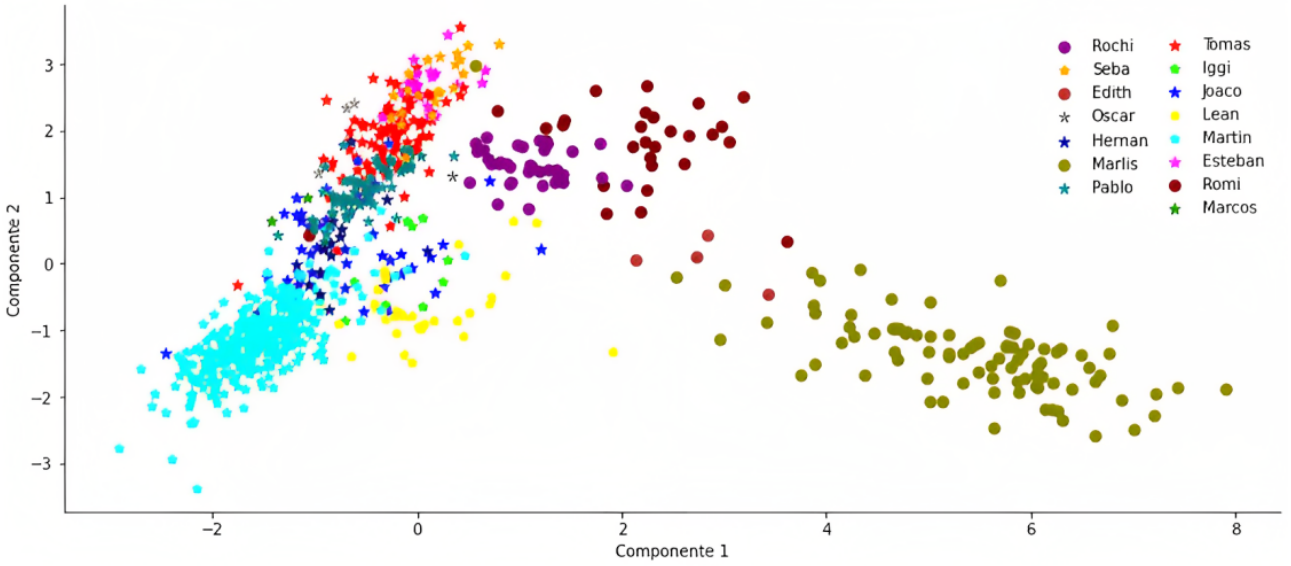

Figura 7.7: Primeros dos componentes principales de un embedding

Tal como se puede observar en la figura 7.7, ya con los primeros dos componentes se tiene una separación notable entre los audios de personas de género masculino (punto en forma de estrella) y femenino (punto en forma de circulo). Además, tomando como ejemplo los audios femeninos, se puede notar que existe poca superposición entre las clases que componen ese grupo. Esto, brinda perspectivas alentadoras en vistas a lograr una fuerte reducción de dimensionalidad, manteniendo la mayor parte de la información posible y permitiendo una optimización de los recursos de compuesto del sistema a través de la mejora de los tiempos de entrenamiento del modelo.

Una vez concluido el estudio preliminar sobre el impacto de la reducción de dimensionalidad sobre los vectores de *embedding*, se decidió incluir la etapa de reducción de dimensionalidad como parte del *pipeline* del sistema y la configuración elegida para ella sera el algoritmo PCA junto con una retención de variabilidad del 95 %. Esto decanto en una reducción del tamaño del vector que representa a las muestras a sus primeros 40 componentes principales, por lo tanto gracias a esta etapa se logrará una reducción de más del 92 % de su tamaño. A continuación se muestra el nuevo *embedding* de la señal de habla de la sección anterior obtenido a partir de la etapa de reducción de dimensionalidad.

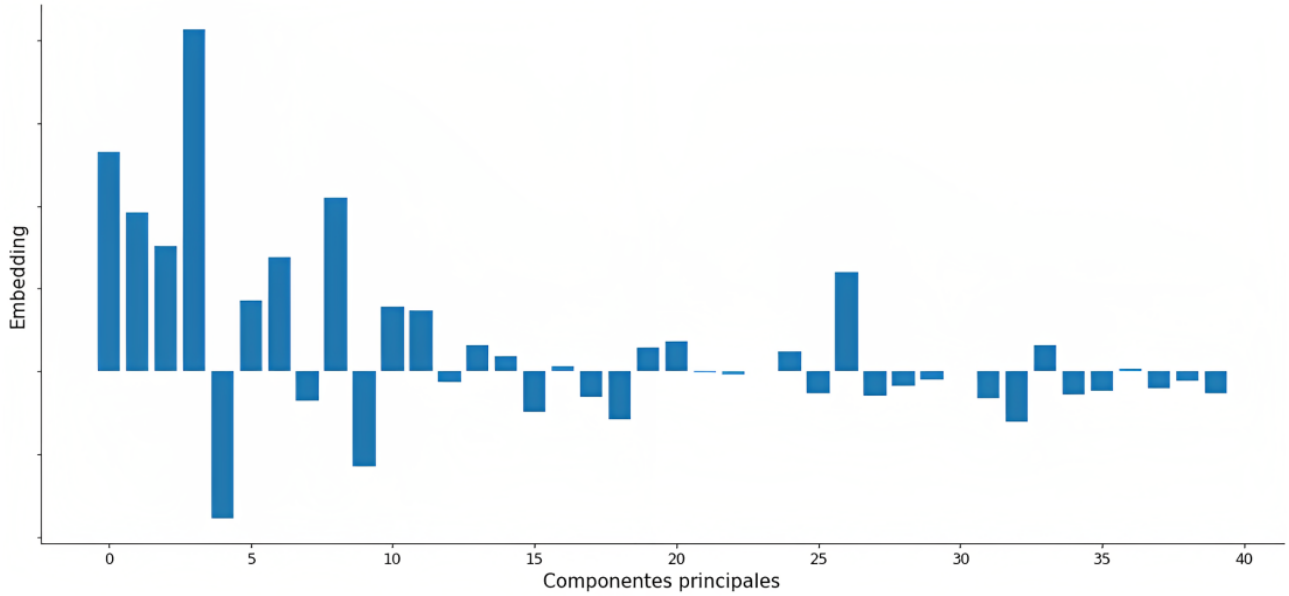

Figura 7.8: Reducción de dimensionalidad aplicada a un embedding

Este proceso se aplicará a todos los *embedding* que componen el conjunto de muestras, obteniendo así un nuevo *dataset* compuesto por muestras de dimensión 40, el cual será utilizado como para realizar el entrenamiento del clasificador que integra la etapa de Backend.

7.4. Backend

Hasta el momento, el flujo de trabajo construido comenzó con la recolección de un archivo de audio con la voz de una persona. Posteriormente, este audio tránsito hacia una etapa de preprocesamiento donde fue acondicionado, para luego ingestar una etapa de extracción de *features* y posteriormente crear un *embedding*. Una vez finalizadas estas modificaciones sobre la muestra original, se tendrá una última etapa encargada de llevar adelante la identificación de la identidad del hablante, también conocida como el **Backend** del sistema. Dentro de esta etapa se deberá primero entrenar y evaluar distintos algoritmos de clasificación, para luego elegir uno de ellos en base a métricas objetivo. Una vez elegido dicho algoritmo, se entrenará con un modelo con las muestras de entrenamiento y se dispondrá como parte del sistema de inferencias.

7.4.1. Clasificación

Al encontrarnos en un problema de identificación de hablantes con un escenario de tipo *close-set* y un marco *text-Independent*, la etapa de clasificación se enmarcará en un esquema de aprendizaje supervisado. Además, basándonos en la bibliografía expuesta para este trabajo, no se creyó necesario la utilización de redes neuronales para esta etapa, ya que existen algoritmos con mayor explicabilidad y simplicidad que cumplirán con creces los requerimientos en sus métricas de objetivo. Tal como se comentó, en los diferentes trabajos académicos referenciados se observa la utilización de variados algoritmos, como por ejemplo Modelos Mixtos Gaussianos, *Support Vector Machine*, *Naïve Bayes*, Árboles de Decisión, entre otros. Basándose en la literatura consultada para el armado de este trabajo de tesis (por ejemplo [23]) se decidió utilizar los algoritmos **K-Nearest Neighbor**, **Support Vector Machine** y **Random Forest** pertenecientes a la librería **Scikit-Learn**; en donde cada uno de ellos brindaran diferentes fortalezas y debilidades a la mesa de discusión.

Para llevar adelante la elección del algoritmo que integrará la etapa de Backend se implementara una búsqueda de hiperparámetros a través de función **GridSearchCV** provista por la librería Scikit-Learn, la cual realizara una búsqueda exhaustiva sobre distintos valores de parámetros que definen a un estimador. El objetivo de esta búsqueda es el de encontrar un modelo que se ajuste lo mejor posible a los datos de entrenamiento y, además logre generalizar correctamente sobre las muestras de validación. Tal como se comentó en la sección Construcción y estructura de los datos se tienen un conjunto de datos compuesto por clases desbalanceadas, por lo que al momento de definir la búsqueda de hiperparámetros se realizó un **k-Fold Cross Validation** con $F_{0,7} - score$ como métrica objetivo y metodo de . La utilización de esta métrica permite dar una mayor equidad a la detección de las diferentes clases, ya que el uso de la métrica de *accuracy* en estos casos puede ser contraproducente y brindar falsas expectativas. A continuación se detallará la implementación de cada algoritmo evidenciando sus ventajas y limitantes, los hiperparámetros elegidos y los resultados de la búsqueda realizada.

K-Nearest Neighbor

Este modelo se basa en la noción de similaridad a través de distancias, por lo tanto asume que existe similitud entre puntos próximos o vecinos más cercanos. Este método no aprende explícitamente un modelo, en cambio memoriza las instancias de entrenamiento que luego son utilizadas para la fase de predicción. La ventaja de este algoritmo radica en su fácil interpretación e implementación, mientras que sus desventajas están íntimamente relacionadas con la maldición de la dimensionalidad debido a que cuanto más esparsos se encuentren los puntos, mayores problemas tendrá al momento de computar las distancias. Para implementar este algoritmo se utilizó la clase **sklearn.neighbors.KNeighborsClassifier**, la cual disponibiliza una variedad de hiperparámetros, de los cuales ajustaremos los siguientes: "n_neighbors" (número de vecinos cercanos), "weights" (ponderación que reciben los vecinos, igual o inversa de su distancia), "p" (parámetro de potencia de la distancia Minkowski) y "Metric" (tipo de medida de distancia).

Para poder evaluar qué parámetros se deberá definir una grilla de valores, que luego evaluará la función GridSearchCV mediante validación cruzada. A través de ella se obtendrá un modelo entrenado juntos con los hiperparámetros que mejor métrica objetivo brinden. Los rangos de valores elegidos para ellos son: *'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9], 'p': [1, 2], 'weights': ['uniform', 'distance'], 'metric': ['cosine', 'minkowski']*. De todos estas propuestas, cabe destacar la utilización de la distancia Coseno ya que es una distancia muy utilizada en la bibliografía y la Minkowski la cual, en conjunto con el parámetro de potencia "p" define si será una distancia de tipo Euclídea o Manhattan.

Una vez definido el tipo de algoritmo, la métrica objetivo y el rango de búsqueda, se procedió a ejecutar la función para obtener el mejor modelo posible. Finalmente, La combinación de hiperparámetros que maximiza el $F_{0,7} - score$ es: *'metric': 'cosine', 'n_neighbors': 5, 'p': 1, 'weights': 'distance'*, siendo 0.979 el valor de la métrica objetivo obtenida. Se puede observar que tal cual se menciona en la bibliografía, que las distancia coseno en una de las medidas con mejor rendimiento en la disciplina del reconocimiento de hablantes.

Support Vector Machine

Este modelo busca una frontera y un margen en el espacio que contiene a las muestras que les permita delimitar de manera exitosa las regiones que aglutinan la mayor parte de las muestras de cada clase. Este algoritmo permite definir la forma que tendrá las fronteras, como por ejemplo, tomar la forma de un hiperplano o optar por objetos con mayor curvatura (Ejem: kernel polinómico). Además, permite cierta flexibilidad al momento de definir los márgenes de decisión, al poseer cierto porcentaje de tolerancia ante muestras mal clasificadas (ubicarse dentro del margen o en la región de otra clase) y, al no depender de todas las muestras para la construcción de su frontera, es un algoritmo robusto frente a *outliers* y alta dimensionalidad. Para implementar este algoritmo se utilizó la clase **sklearn.svm.SVC** la cual disponibiliza una variedad de hiperparámetros, de los cuales ajustaremos los siguientes: "C" (grado de observaciones mal clasificadas permitido), "kernel" (tipo de frontera utilizada en la calificación), "gamma" (parámetro del kernel radial que indica cuánta influencia posee una muestra individual) y "degree" (grado del kernel polinómico).

Para poder evaluar qué parámetros se deberá definir una grilla de valores, que luego evaluará la función GridSearchCV mediante validación cruzada. A través de ella se obtendrá un modelo entrenado juntos con los hiperparámetros que mejor métrica objetivo brinden. Los rangos de valores elegidos son: *'C': [1, 10, 100, 1000], 'gamma': [1, 0.01, 0.001, 0.0001], 'kernel': ['linear', 'poly', 'rbf'], 'degree': [2, 3]*. De todos estas propuestas cabe destacar la recomendación por parte de la documentación oficial de Scikit-Learn, en donde se especifica que los valores de los hiperparámetros "C" y "gamma" son críticos para el rendimiento del algoritmo y se sugiere espaciar sus valores en la grilla de forma exponencial.

Una vez definido el tipo de algoritmo, la métrica objetivo y el rango de búsqueda, se procedió a ejecutar la función de búsqueda del mejor modelo. Finalmente, la combinación de hiperparámetros que maximiza el $F_{0,7} - score$ es: *'C': 1, 'degree': 2, 'gamma': 1, 'kernel': 'linear'*, siendo 0.975 el valor de la métrica objetivo obtenida.

Random Forest

El objetivo de Random Forest radica en poder llevar adelante un proceso de clasificación a través de la implementación de múltiples árboles de decisión individuales, que operan en conjunto. Esta metodología se corresponde con la familia de *ensambles*, específicamente *bagging*, la cual busca generar una decisión por cada árbol individual para luego implementar una votación que convierte a la clase con más votos en la predicción del modelo. Este algoritmo posee numerosas ventajas ya que al corregir el problema de correlación entre árboles, reduce el riesgo de *overfitting*, es robusto frente a *outliers* y no se ve gravemente afectado por las diferencias de escala en los datos. Estas bondades logradas por el método de ensamble son de gran utilidad pero se logran a costa de la pérdida de interpretabilidad del modelo y de un mayor tiempo de procesamiento. Para implementar este algoritmo se utilizó la clase **sklearn.ensemble.RandomForestClassifier**, la cual disponibiliza una variedad de hiperparámetros, de los cuales ajustaremos los siguientes: "n_estimators" (número de árboles que integrarán el clasificador), "max_features" (máxima cantidad de features que se tomará para cada árbol), "max_depth" (profundidad máxima de cada árbol), "min_samples_split" (número mínimo de muestras para dividir un nodo) y "criterion" (criterio para medir la cuán buena fue la elección de una división).

Una vez definido los hiperparámetros a ajustar, se deberá definir una grilla de valores que luego sera evaluada por la función GridSearchCV, mediante validación cruzada. A través de ella se obtendrá un modelo entrenado juntos con los hiperparámetros que mejor métrica objetivo brinden, siendo estos los siguientes: *'n_estimators': [400, 600, 800], 'criterion': ['gini', 'entropy'], 'max_features': ['auto', 'sqrt'], 'max_depth': [4, 8, 16, 24], 'min_samples_split': [4, 8, 12]*. De todos estas propuestas cabe destacar que el parámetro "bootstrap" se encuentra habilitado por defecto, por lo que se realizará un muestreo con reemplazo del conjunto de datos original, generando nuevos grupos de muestras utilizados para entrenar cada árbol. El tamaño de estos conjuntos está dictaminado por el parámetro "max_samples", el cual por defecto define que todos los conjuntos sean del mismo tamaño que el original.

Una vez definido el tipo de algoritmo, la métrica objetivo y el rango de búsqueda, se proce-

dió a ejecutar la función de búsqueda del mejor modelo. Finalmente, La combinación de hiperparámetros que maximiza el $F_{0,7} - score$ es: *'criterion': 'gini', 'max_depth': 8, 'max_features': 'auto', 'min_samples_split': 4, 'n_estimators': 400*, siendo 0.936 el valor de la métrica objetivo obtenida.

Resumiendo, una vez finalizados todos los entrenamiento con búsqueda de hiperparámetros, se puede destacar que los tres modelos evaluados han mostraron un gran rendimiento sobre el conjunto de entrenamiento. Esto es esperanzador pero no definitorio, ya que resta analizar cómo será su desempeño frente a muestras que no fueron utilizadas en la etapa de entrenamiento. En la siguiente sección se analizarán los resultados obtenidos por los modelos cuando se los evalúa sobre un conjunto de pruebas, observando así su grado de generalización.

Capítulo 8

Resultados

En esta sección se presentarán los resultados obtenidos de los diferentes modelos sobre las muestras de validación, por lo que el rendimiento de los clasificadores en esta etapa mostrará que tan bien generalizan. La metodología de evaluación estará compuesta, en primera instancia por un análisis de las matrices de confusión, para luego transitar hacia el análisis de los indicadores de bondad de ajuste, de los cuales se utilizaran el *Accuracy*, *Recall*, *Precision* y $F\beta$. Es importante analizar todo ellos ya que cada uno brinda información única, por ejemplo, evaluar las métricas de *Precision*, *Recall* y $F\beta$ es muy importante ya que estamos en presencia de un problema de multiclasificación desbalanceado. Para estos análisis, el paquete **Scikit-Learn** proporciona un módulo llamado **Metrics**, el cual contiene una gran variedad, entre ellas las antes mencionadas, lo nos permitirá evaluar nuestros modelos. Por lo tanto, para el cálculo de las matrices de confusión se utilizó la función **confusion_matrix**, mientras que para los indicadores de bondad de ajuste, se utilizaron las funciones **accuracy_score**, **recall_score**, **fbeta_score** y **precision_score**.

8.1. Matrices de confusión

La matriz de confusión es una técnica que permite evaluar visualmente el rendimiento de un algoritmo de clasificación para un conjunto de datos en particular, concediendo un análisis explícito acerca de los tipos de errores cometidos y el grado de mezcla entre las clases predichas

y las verdaderas. Al momento de realizar las predicciones con los distintos algoritmos se calculó el grado de asertividad de sus clasificaciones sobre un total de 153 muestras que integraban el conjunto de datos de validación, obteniendo: 1 muestras mal clasificada por el algoritmo K-Nearest Neighbours, 3 para Support Vector Machine y 4 para Random Forest. Luego, para poder analizar más a detalle la confusión ocurrida entre las distintas clases, se construyeron las matrices de confusión.

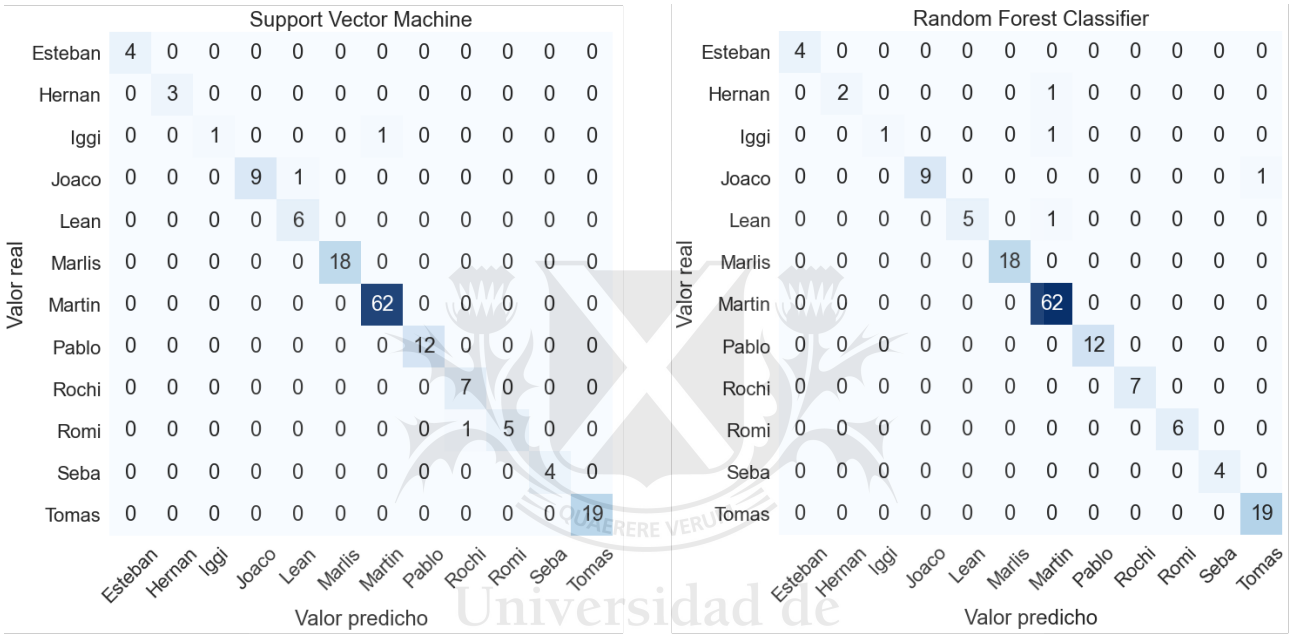

Figura 8.1: Matrices de confusión para clasificadores SVM y RFC

En la figura 8.1 se observar que el algoritmo SVM clasificó erróneamente a la hablante Romi con Rochi, Iggy con Martin y Joaco con Lean, de lo que se puede destacar que todas las confusiones fueron intra-género y entre personas similares desde un punto de vista generacional. Esto, refuerza empíricamente el marco teórico descrito en secciones anteriores, en donde se muestra a la edad y género como factores influyentes en las características del habla humana. En Random Forest, se confunde a los hablantes Hernan, Iggy y Lean con Martin, mientras que Joaco con Tomas.

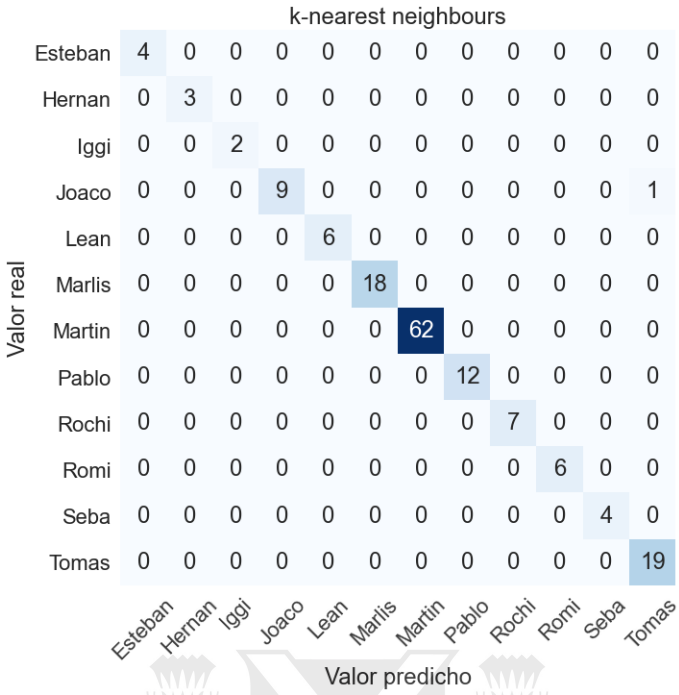

Figura 8.2: Matrices de confusión para clasificadores KNN

Finalmente, con muestra la figura 8.2 para el caso de K-Nearest Neighbours se confunde solamente el hablante Joaco con Tomas. Es importante destacar que en los tres clasificadores se respeta las premisas mencionadas en el marco teórico provisto y las personas que fueron repetidamente mal clasificadas por los modelos, se corresponde con los mismos audios.

8.2. Indicadores de bondad de ajuste

Al enfrentarse a un problema multiclase desbalanceado es importante definir los indicadores de bondad de ajuste de manera que permita tener en cuenta estas condiciones de contorno. Al momento de construir estos indicadores en python, es importante definir el parámetro **average** ya que este determina el tipo de promediado que se realizará sobre los datos y sus clases. El valor asignado a este parámetro será **Macro**, la cual no favorece ninguna de las clases minoritarias y así se podrá tener en cuenta si existiese algún efecto generado por el desbalance de las etiquetas. Este parámetro, permite calcular una métrica por cada una de las clases y luego realiza un promedio no ponderado sobre ellas. Una vez definido estas configuraciones se procedió al cálculo de los indicadores, los cuales se pueden observar en la siguiente tabla.

Indicador	K-nearest neighbours	Support Vector Machine	Random Forest
Accuracy	99.34 %	98.03 %	97.38 %
Precision	99.58 %	97.63 %	99.19 %
Recall	99.16 %	93.61 %	90.83 %
F[0.7]	99.42 %	95.47 %	95.40 %

Cuadro 8.1: Indicadores de bondad de ajuste

Tal como se observa en la tabla, las inferencias obtenidas por los clasificadores resultaron tener gran acertividad, superando el 90 % en todas las métricas medidas sobre el conjunto de validación o prueba. De los tres modelos evaluados se destaca en algoritmo de vecinos más cercanos, tal como se puede observar en la siguiente figura que ilustra todas las métricas evaluadas.

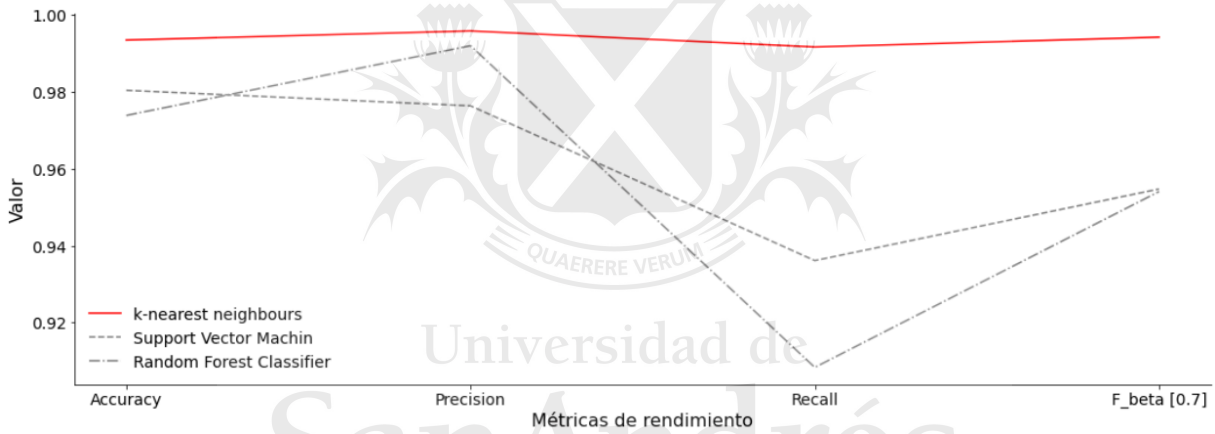

Figura 8.3: Matrices de confusión

Es importante recalcar que la base de datos no presentaba problemáticas como *outliers*, datos faltantes o alta dimensionalidad. Más allá de la generalidad de buenos resultados, se destaca que el clasificador que mejor generaliza sobre las muestras de validación es K-nearest neighbours con un rendimiento en todas sus métricas superando el 99 %. Esto significa que logró clasificar casi a la perfección todos los hablantes de la muestra de validación. Dado esto es que finalmente **el calificador elegido para completar el sistema de reconocedor de hablantes es K-nearest neighbourss con una métrica basada en distancia Coseno, con 4 como el número de vecinos más cercanos y una ponderación uniforme en las distancias evaluadas**.

Capítulo 9

Discusión

El habla es un rasgo distintivo que caracteriza a los seres humanos respecto a los demás seres vivos de este planeta. A través de ella podemos comunicarnos, asociarnos y formar sociedades colectivas, lo que nos permitió lograr los avances culturales y científicos que hoy en día disfrutamos. La comunicación es una actividad fundamental para desarrollarnos como individuos dentro de una sociedad, con la globalización ya sobre nosotros la comunicación se ha vuelto más fácil e instantánea, sin barreras geográficas o culturales. Gracias al auge de los medios de comunicación y sobre todo de las aplicaciones de redes sociales y mensajería, es que la producción de contenido audiovisual ha crecido exponencialmente. En este contexto es que el *machine learning* y sus diferentes ramas de aplicación presentan una gran oportunidad para poder extraer o inferir cierta información sobre la persona que producen ese contenido. En la actualidad, es cada vez mas frecuenta la automatización en los canales de comunicación que una empresa establece con sus clientes a través el uso de *chatbots*, de autenticación por biometría y, también, el almacenamiento de los diálogos para luego realizar distintos análisis, como por ejemplo de sentimientos. Todas estas oportunidades generan un mayor interés el uso de sistemas computarizados que faciliten la interacción con sus clientes y permitan obtener un rédito de negocio.

Siguiendo esta dirección, en este trabajo de tesis se profundizó sobre la utilización del habla como fuente de extracción de información que identifique unívocamente a una persona. Para

ello, primero fue necesario entender a que nos referimos cuando hablamos de el habla, valga la redundancia, y qué particularidades posee este tipo se señal. Esto decantó en un estudio sobre las propiedades que hacen a una señal de habla única para cada persona y, también entender los diferentes sonidos que la componen, como se estructuran y cómo pueden ser afectados por factores externos. Para llevar adelante este análisis fue necesario utilizar diferentes técnicas que nos permitan analizar cómo se compone y cómo se comporta una señal a lo largo del tiempo, llevándonos a adentrarnos un poco en el mundo de las señales y sistemas. Aquí, es importante destacar que el autor de este trabajo posee un conocimiento previo en esta área, específicamente sobre las señales de habla, lo que facilitó la curva de aprendizaje y logro una mayor celeridad en la implementación de las primeras etapas del proyecto. Una vez afianzados es estos conceptos, fue necesario entender cómo se encuentra conformada la disciplina que estudia la detección de hablantes a través de la voz, por lo que comenzó una etapa de estudio sobre las metodologías mas preponderantes utilizadas en las áreas de investigación. Para ello se analizaron varios artículos escritos por investigadores con un amplia trayectoria en en análisis del habla, lo que nos permitió recorrer diversos enfoques históricos utilizados para proponer al habla como una fuente de biometría.

Una vez investigados todos los componentes y particularidades que enmarcan la detección de hablantes y, comprobando la factibilidad de utilizar *machine learning* para este tipo de detecciones, es que se decidió avanzar hacia la contracción de un sistema automático de reconocimiento de hablantes basado en *machine learning*. La estructura de este sistema se encuentra descripta en el capitulo Metodología, por lo que a continuación presentaremos una discusión de cada capítulo, en particular donde se abordan áreas claves para llevar adelante la construcción de este sistema.

En la sección La señal de habla se realizo una investigación enfocada en detectar si efectivamente el habla es factible de ser un identificador unívoco de una persona. Tomando como referencia numerosas investigaciones se puede observar que el habla está gobernada mayoritariamente por la fisiología del tracto vocal y los hábitos aprendidos en su articulación y que, aún en gemelos idénticos, se siguen percibiendo diferencias entre sus voces. Esto potencio las

expectativas para la utilización de la voz como un biomarcador. Con esto, surgió la pregunta de ¿qué? y ¿cómo? extraer las características que permitan diferenciar lo mejor posible una señal de otra. Según la literatura consultada, las características mas preponderante son el pitch, la frecuencia fundamental y los tres primeros formantes. Estas, definidas como *features* acústicos, puede ser extraídas a través de algoritmos computacionales y permiten detectarse diferencias aun teniendo dos voces similares.

En la sección El reconocimiento automático de hablantes se presentaron todas las verticales de investigación relacionadas al universo de problemas de reconocimiento de hablantes. Dentro de estas disciplinas se tienen dos grandes familias, una enfocada en la verificación de hablantes, la cual implica un contraste entre una muestra con una identidad asumida contra un conjunto de muestras etiquetadas con esa misma identidad. Por otro lado, la identificación de hablantes se enfoca en detectar la identidad de una persona solamente por el audio provisto. Está, a su vez, pueda ramificarse aún más si se tiene en cuenta si existe un texto que acompañe ese audio o si el hablante a detectar se encuentra contenido en el conjunto de entrenamiento. Todas estas aristas fueron evaluadas y se definió enfocar el trabajo a un problema de identificación de hablantes sin la presencia de texto y, a los fines de simplificar las implementación, se asumió que todas las personas a detectar tenían presencia en el *dataset* de entrenamiento

En los capítulos De los modelos gaussianos a los i-vectors y Redes Neuronales en la detección de hablantes se repasó la historia de las técnicas utilizadas en la detección de hablantes por numerosos investigadores. Primero se comenzó por modelos probabilísticos no supervisados basados en mezcla de gaussianas, para luego continuar hacia modelos orientados a la construcción de vectores de *features* sobre estadísticos obtenidos de las señales. Esto logró subsanar numerosas problemáticas en cuanto a diferencias en las duración de cada audio, pero trajo aparejado problemas por la alta dimensionalidad de los vectores resultantes. Mas tarde, mediante técnicas de análisis factorial se subsano esta problemática dando lugar a los i-vectors. Este tipo de enfoques son de gran utilidad ya que permiten desacoplar los sistemas en dos etapas, el primero encargado de generar representaciones vectoriales y el segundo para clasificarlos. Luego, con el auge de las redes neuronales se empezó a remplazar ambas etapas por este tipo de arquitectu-

ras, dando lugar al concepto de *embeddings*. Estas redes neuronales son alimentadas por bancos de filtros que se encargan de extraer las características mas preponderantes de las señales de voz para luego despenalizarlas en vectores. De todas las técnicas investigadas, los análisis más modernos muestran a las arquitectura TDNN (x-vectors) con un rendimiento superador a las demás, por lo que se opto por esta dirección. Ahora, debíamos definir cómo alimentaremos esa red neuronal, osea, qué método utilizaremos para extraer las características de las señales.

En los capítulos La extracción de características se realizo una contraposición entre los métodos clásicos y los nuevos enfoques. Dentro de los primeros tenemos los bancos de filtros que permiten recolectar información de los tonos que componen la señal, algo que brinda mucha información. Los bancos de filtros pueden construirse a partir de diversos enfoques como el MCC (Mel Cepstral Coefficients) pero investigadores como Ravanelli y Bengio argumentan que la implementación de estos bancos de filtro son a menudo resultados empíricos y no existen garantías de que dichas representaciones sean óptimas. Además, sostienen que tienen efectos nocivos sobre las señales de habla, por ejemplo suavizan su espectro dificultando la extracción de propiedades como el pitch y los formantes. Para mitigar estos efectos plantean un nuevo enfoque donde se utilizan redes bien definidas y con pocos parámetros llamadas Sincnet, las cuales han demostrado una convergencia más rápida y un mejor funcionamiento en las tareas de identificación y verificación de hablantes. Para implementar esta arquitectura, la herramienta de código libre que fue utilizada es *pyannote.audo*.

Con el objetivo de construir un sistema útil que resuelva problemáticas actuales relacionadas con el *voice footprint* es que en el capítulo Los datos hacemos hincapié en la importancia de tener datos que reflejen la realidad de las personas en su día a día. Esta realidad se encuentra marcada por el uso de dispositivos móviles como medio para interactuar con sistemas digitales externos, siendo estos una de las fuentes de recolección de información biométrica. Ya sea el uso de huellas digitales, el reconocimiento por rostros o la recolección de audios, los dispositivos celulares deben ser fuertemente considerados como fuentes para la obtención de *datasets*. Teniendo este en mente es que se considero a los audios de WhatApps, intercambiados entre el autor de este trabajo y sus conocidos, como fuente de datos representativa de las problemáticas presentes en

las señales de habla. Un aspecto a destacar es que, debido al método de recolección, el conjunto de muestras estará desbalanceado hacia el autor de la tesis, pero esto no represento un problema ya que la herramienta *pyannote.audo* disponibiliza modelos preentrenados de alto rendimiento.

Aun siendo las redes neuronales muy robustas frente a la calidad de las muestras, se ha demostrado en numerosos trabajos que un correcto preprocesamiento es gran utilidad para mejorar el rendimiento del sistema. Es por esto que en sintonía con las problemáticas que acarrean las señales se decidió implementar una etapa de preprocesamiento, la cual está integrada por un filtro de silencios, un filtro de preénfasis y un filtro de ruido de fondo. En el capitulo Metodología se muestra como fueron implementadas las etapas que desde el preprocesamiento, hasta la generación de *embeddings* y finalizando en el Backend. En esta ultima es donde se entrenaron y evaluaron tres tipos distintos de algoritmos de clasificación, para así seleccionar el de mayor rendimiento y completar todas las etapas que integran el sistema automático de identificación de hablantes.

En el capítulo Resultados se analizaron tres modelos de *machine learning* con el objetivo de decir cual de ellos integrara la etapa de Backend. Como nos encontramos ante un problema de clasificación múltiple con clases desbalanceadas, se debió elegir una métrica de evaluación que tenga en cuanta este contexto. Para ello se eligió el $F_{0,7}$ con el parámetro *average* definido en *macro*, el cual realiza un promedio entre las métricas de cada clase permitiendo tratar a todas de igual manera. El rendimiento general de todos los modelos fue alto (superior al 90 %) pero se destaca el modelo de *K-Nearest Neighbor*, el cual arrojo un $F_{0,7}$ de 99.42 % (solo erro una vez). Además, la búsqueda de hiperparámetros arrojo que este modelo K-nn utiliza la distancia coseno, distancia muy destacada en la bibliografía que se especializa en la detección de hablantes.

Cerrando este capitulo de discusión podemos concluir que, sobre las dificultades que presenta la detección de diferencias entre dos voces muy parecidas es que se abre un abanico de oportunidades para los sistemas de detección de hablantes automáticos basados en *machine learning*. Ellos, gracias a su gran capacidad para detectar, extraer y comparar características, junto con las propiedades únicas que posee el habla de cada persona, permiten que estas señales

sean un excelente bio-feature para el reconocimiento de un individuo. La investigación realizada y luego validada con trabajos de laboratorio ha demostrado con gran firmeza que el habla humanas es altamente rica en información y que, con el avance de las ciencias de datos, es un terreno fértil para continuar desarrollando innovaciones que generen nuevas oportunidades y servicios para la sociedad en todo su conjunto.

Bibliografía

[1] Zrar Kh Abdul and Abdulbasit K Al-Talabani. Mel frequency cepstral coefficient and its applications: A review. *IEEE Access*, 2022.

[2] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*, pages 1–6. Ieee, 2017.

[3] Richard A Altes. Detection, estimation, and classification with spectrograms. *The Journal of the Acoustical Society of America*, 67(4):1232–1246, 1980.

[4] Bishnu S Atal. Automatic recognition of speakers from their voices. *Proceedings of the IEEE*, 64(4):460–475, 1976.

[5] Zhongxin Bai and Xiao-Lei Zhang. Speaker recognition based on deep learning: An overview. *Neural Networks*, 140:65–99, 2021.

[6] Pascal Belin, Patricia EG Bestelmeyer, Marianne Latinus, and Rebecca Watson. Understanding voice perception. *British Journal of Psychology*, 102(4):711–725, 2011.

[7] Ronald Newbold Bracewell and Ronald N Bracewell. *The Fourier transform and its applications*, volume 31999. McGraw-Hill New York, 1986.

[8] Hervé Bredin and Antoine Laurent. End-to-end speaker segmentation for overlap-aware resegmentation. *arXiv preprint arXiv:2104.04045*, 2021.

[9] Hervé Bredin, Ruiqing Yin, Juan Manuel Coria, Gregory Gelly, Pavel Korshunov, Marvin Lavechin, Diego Fustes, Hadrien Titeux, Wassim Bouaziz, and Marie-Philippe Gill. Pyannote. audio: neural building blocks for speaker diarization. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7124–7128. IEEE, 2020.

[10] E Oran Brigham. *The fast Fourier transform and its applications*. Prentice-Hall, Inc., 1988.

[11] William M Campbell, Douglas E Sturim, and Douglas A Reynolds. Support vector machines using gmm supervectors for speaker verification. *IEEE signal processing letters*, 13(5):308–311, 2006.

[12] Paresh M Chauhan and Nikita P Desai. Mel frequency cepstral coefficients (mfcc) based speaker identification in noisy environment using wiener filter. In *2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE)*, pages 1–5. IEEE, 2014.

[13] Susan Cook and John Wilding. Earwitness testimony: Never mind the variety, hear the length. *Applied Cognitive Psychology: The Official Journal of the Society for Applied Research in Memory and Cognition*, 11(2):95–111, 1997.

[14] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20:273–297, 1995.

[15] Najim Dehak, Pedro A Torres-Carrasquillo, Douglas Reynolds, and Reda Dehak. Language recognition via i-vectors and dimensionality reduction. In *Twelfth annual conference of the international speech communication association*, 2011.

[16] Penelope Eckert and John R Rickford. *Style and sociolinguistic variation*. Cambridge University Press, 2001.

[17] Anders Eriksson. Tutorial on forensic speech science. In *Interspeech, Lisbon, Portugal*, 2005.

[18] Xing Fan and John HL Hansen. Speaker identification within whispered speech audio streams. *IEEE transactions on audio, speech, and language processing*, 19(5):1408–1421, 2010.

[19] J-L Gauvain and Chin-Hui Lee. Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains. *IEEE transactions on speech and audio processing*, 2(2):291–298, 1994.

[20] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii, and Kate Crawford. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92, 2021.

[21] John HL Hansen and Vaishnevi Varadarajan. Analysis and compensation of lombard speech across noise type and levels with application to in-set/out-of-set speaker recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(2):366–378, 2009.

[22] Miquel India, Pooyan Safari, and Javier Hernando. Self multi-head attention for speaker recognition. *arXiv preprint arXiv:1906.09890*, 2019.

[23] Rashid Jahangir, Ying Wah Teh, Henry Friday Nweke, Ghulam Mujtaba, Mohammed Ali Al-Garadi, and Ihsan Ali. Speaker identification through artificial intelligence techniques: A comprehensive review and research challenges. *Expert Systems with Applications*, 171:114591, 2021.

[24] Saul M Kassin, Itiel E Dror, and Jeff Kukucka. The forensic confirmation bias: Problems, perspectives, and proposed solutions. *Journal of applied research in memory and cognition*, 2(1):42–52, 2013.

[25] Finnian Kelly, Andrzej Drygajlo, and Naomi Harte. Speaker verification with long-term ageing data. In *2012 5th IAPR international conference on biometrics (ICB)*, pages 478–483. IEEE, 2012.

[26] Finnian Kelly, Andrzej Drygajlo, and Naomi Harte. Speaker verification in score-ageing-quality classification space. *Computer Speech & Language*, 27(5):1068–1084, 2013.

[27] Barbara S Kisilevsky, Sylvia MJ Hains, Kang Lee, Xing Xie, Hefeng Huang, Hai Hui Ye, Ke Zhang, and Zengping Wang. Effects of experience on fetal voice recognition. *Psychological science*, 14(3):220–224, 2003.

[28] Mario Köppen. The curse of dimensionality. In *5th online world conference on soft computing in industrial applications (WSC5)*, volume 1, pages 4–8, 2000.

[29] Anthony Larcher, Kong Aik Lee, Bin Ma, and Haizhou Li. The rsr2015: Database for text-dependent speaker verification using multiple pass-phrases. In *Annual Conference of the International Speech Communication Association (Interspeech)*, 2012.

[30] Marianne Latinus and Pascal Belin. Human voice perception. *Current Biology*, 21(4):R143–R145, 2011.

[31] Yuan Liu, Yanmin Qian, Nanxin Chen, Tianfan Fu, Ya Zhang, and Kai Yu. Deep feature for text-dependent speaker verification. *Speech Communication*, 73:1–13, 2015.

[32] Brian McFee, Colin Raffel, Dawen Liang, Daniel P Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, volume 8, pages 18–25, 2015.

[33] Francis Nolan and Tomasina Oh. Identical twins, different voices. *International Journal of Speech, Language and the Law*, 3(1):39–49, 1996.

[34] Alan V Oppenheim, Alan S Willsky, Syed Hamid Nawab, and Jian-Jiun Ding. *Signals and systems*, volume 2. Prentice hall Upper Saddle River, NJ, 1997.

[35] George Papcun, Jody Kreiman, and Anthony Davis. Long-term memory for unfamiliar voices. *The Journal of the Acoustical Society of America*, 85(2):913–925, 1989.

[36] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[37] Daniel Peralta, A Herrera, and Francisco Herrera. Un estudio sobre el preprocesamiento para redes neuronales profundas y aplicación sobre reconocimiento de dígitos manuscritos. 2020.

[38] Tyler K Perrachione, Stephanie N Del Tufo, and John DE Gabrieli. Human voice recognition depends on language ability. *Science*, 333(6042):595–595, 2011.

[39] Mirco Ravanelli and Yoshua Bengio. Speaker recognition from raw waveform with sincnet. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 1021–1028. IEEE, 2018.

[40] Douglas A Reynolds, Thomas F Quatieri, and Robert B Dunn. Speaker verification using adapted gaussian mixture models. *Digital signal processing*, 10(1-3):19–41, 2000.

[41] Douglas A Reynolds and Richard C Rose. Robust text-independent speaker identification using gaussian mixture speaker models. *IEEE transactions on speech and audio processing*, 3(1):72–83, 1995.

[42] Douglas A Reynolds, Marc A Zissman, Thomas F Quatieri, Gerald C O'Leary, and Beth A Carlson. The effects of telephone transmission degradations on speaker recognition performance. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 329–332. IEEE, 1995.

[43] Richard C Rose, Edward M Hofstetter, and Douglas A Reynolds. Integrated models of signal and background with application to speaker identification in noise. *IEEE Transactions on Speech and Audio Processing*, 2(2):245–257, 1994.

[44] Tim Sainburg. timsainb/noisereduce: v1. 0. *Zenodo, Jun*, 2019.

[45] Tim Sainburg, Marvin Thielk, and Timothy Q Gentner. Finding, visualizing, and quantifying latent structure across diverse animal vocal repertoires. *PLoS computational biology*, 16(10):e1008228, 2020.

[46] Nithya Sambasivan, Shivani Kapania, Hannah Highfill, Diana Akrong, Praveen Paritosh, and Lora M Aroyo. "everyone wants to do the model work, not the data work": Data cascades in high-stakes ai. In *proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–15, 2021.

[47] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. X-vectors: Robust dnn embeddings for speaker recognition. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5329–5333. IEEE, 2018.

[48] Gilbert Strang. The discrete cosine transform. *SIAM review*, 41(1):135–147, 1999.

[49] Duraisamy Sundararajan. *The discrete Fourier transform: theory, algorithms and applications*. World Scientific, 2001.

[50] Silero Team. Silero vad: pre-trained enterprise-grade voice activity detector (vad), number detector and language classifier. https://github.com/snakers4/silero-vad, 2021.

[51] WD Van Gysel, J Vercammen, and F Debruyne. Voice similarity in identical twins. *Acta oto-rhino-laryngologica Belgica*, 55(1):49–55, 2001.

[52] Diana Van Lancker and Jody Kreiman. Voice discrimination and recognition are separate abilities. *Neuropsychologia*, 25(5):829–834, 1987.

[53] Ehsan Variani, Xin Lei, Erik McDermott, Ignacio Lopez Moreno, and Javier Gonzalez-Dominguez. Deep neural networks for small footprint text-dependent speaker verification. In *2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4052–4056. IEEE, 2014.

[54] Pulkit Verma and Pradip K Das. i-vectors in speech processing applications: a survey. *International Journal of Speech Technology*, 18:529–546, 2015.

[55] Alexander Veysov and Dimitrii Voronin. One voice detector to rule them all. *The Gradient*, 2022.

[56] Shirui Wang, Wenan Zhou, and Chao Jiang. A survey of word embeddings based on deep learning. *Computing*, 102:717–740, 2020.

[57] Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, et al. The htk book. *Cambridge university engineering department*, 3(175):12, 2002.