Sistema Híbrido para el Análisis de Sentimientos en Llamadas

– –

Hybrid Call Sentiment Analysis Systems

TRABAJO FIN DE GRADO
GRADO EN INGENIERÍA INFORMÁTICA
CURSO 2020–2021

Francisco Quiñones Bellón
Mario Sánchez de Paz
Emilio Perera Fernández de Pedro

Directores

Francisco Javier Crespo Yáñez
Luis Javier García Villalba

Departamento de Sistemas Informáticos y Computación
Facultad de Informática
Universidad Complutense de Madrid
Madrid, Junio de 2021

Agradecimientos

A Javier Crespo y Javier García, nuestros directores de proyecto que desde el primer momento, mostraron una gran confianza en lo que podríamos conseguir con este trabajo de fin de grado, y supieron transmitirnos esa ilusión.

A Esteban Armas, Ana Lucila Sandoval y Daniel Povedano por su notable ayuda y constante apoyo.

A diversos profesores a lo largo de estos años de carrera por habernos ayudado a adquirir los conocimientos necesarios que hemos aplicado para desarrollar este trabajo con éxito.

A la Facultad de Informática de la Universidad Complutense de Madrid por los recursos ofrecidos.

A nuestra familia y amigos, cuyo apoyo ha sido fundamental en los momentos más difíciles de este viaje.

Este Trabajo de Fin de Grado ha sido realizado dentro del Grupo de Análisis, Seguridad y Sistemas (Grupo GASS, https://gass.ucm.es/, Grupo 910623 del catálogo de grupos reconocidos por la UCM) como parte de las actividades del proyecto de investigación THEIA (Techniques for Integrity and Authentication of Multimedia Files of Mobile Devices) con referencia FEI-EU-19-04.

Índice General

Índice de Figuras

Índice de Tablas

Lista de Acrónimos

AA	Aprendizaje Automático
AI	Inteligencia Artificial
ALPAC	*Automatic Language Processing Advisory Committee*
ATN	Redes de Transición Aumentadas
BLSTM	*Bidirectional Long Short-Term Memory*
CNN	Redes Neuronales Convolucionales
CV	*Cross Validation*
DCT	Transformada de coseno discreta
DNN	*Deep Neural Networks*
EEND	*End-to-end Neural Diarization*
FCSE	Fuerzas y Cuerpos de Seguridad del Estado
FFT	Transformada rápida de Fourier
FN	Falsos Negativos
FP	Falsos Positivos
GB	Gradient Boosting

HMM *Hidden Markov Model*

IDF *Inverse Document Frecuency*

IVR *Interactive Voice Response*

KNN K Vecinos más Próximos

LSTM *Long Short-Term Memory*

MFC *Mel-Frequency Cepstrum*

MFCCs Coeficientes Cepstrales en las Frecuencias de Mel

ML *Machine Learning*

PLN Procesamiento de Lenguaje Natural

RF *Random Forest*

RNF Filtros Neuronales Recurrentes

RNN Redes Neuronales Recursivas

RPN *Region Proposal Network*

RRSS *Redes Sociales*

SGD *Stochastic Gradient Descent*

SST *Stanford Sentiment Treebank*

SVM Support Vector Machine

TF Frecuencia de Término

TF-IDF *Term Frecuency - Inverse Document Frecuency*

VN	Verdaderos Negativos
VP	Verdaderos Positivos

Abstract

Over the last few years, there has been a considerable increase in communications and operations carried out through the Internet. Social media and electronic marketing increased their influence in our society exponentially, creating a huge amount of data about our preferences. Videos, audios, images and texts we share online can be analized to extract that kind of information. That sentiment analysis have great potential in the marketing world due to the feedback sent by users in multiple platforms, which is processed by technologies based on data analysis like Machine Learning and Deep Learning. We can use these techniques in online marketing to obtain great valued information, something that companies already know and apply on social media and market studies with the purpose of giving a close experience to the user. There are also some other applications that could offer lots of possibilities in areas like medicine, psychology and sociology. Speech Emotion Recognition and Natural Language Processing are two of these technique families that look for sentiment and emotion on data, but they work on different media. These technologies' goal is to estimate the subjective connotation inside a speech or a document. Dividing this project into to separated parts, we have designed a set of experiments to study and analize sentiment on audio and the text that is possible to extract from it with a transcription tool. Along with this systems, other tool that has been studied and developed is Speaker Diarization. Speaker Diarization is the process of partitioning an input audio stream into homogeneous segments according to the speaker identity. The conjunction of all the systems developed would make a complete analysis of the original audio, offering more details about the emotions expressed that a single appreciation could ignore.

Keywords: Aritificial Intelligence, Machine Learning, Natural Language Processing, Sentiment Analysis, Speaker Diarization, Speech Emotion Recognition.

Resumen

A lo largo de los últimos años ha existido un incremento considerable en comunicaciones y operaciones llevadas a cabo en la red. Las *Redes Sociales* (RRSS) y el comercio electrónico han visto crecer su influencia en nuestro entorno de manera exponencial, generándose una ingente cantidad de datos sobre las preferencias de cada usuario. Los vídeos, audios, imágenes y textos que compartimos pueden ser analizados con detalle para extraer información sobre nuestros gustos. Esos sentimientos tienen un gran potencial dentro del comercio, debido al *feedback* que es transmitido por las opiniones de los usuarios en múltiples plataformas, que son procesadas por tecnologías basadas en análisis de la información: *Machine Learning* y *Deep Learning*, cuyo origen es la Inteligencia Artificial. Aplicando estas técnicas al *marketing* y al comercio online se puede obtener una gran cantidad de valor, algo que conocen bien la mayoría de empresas en la actualidad, que hacen uso de las RRSS y de estudios de mercado con Big Data para ofrecer una experiencia más cercana y personalizada para sus clientes. También existen otras aplicaciones que ofrecen infinidad de posibilidades en campos como la medicina, psicología o sociología. Dos campos, que en cierto modo comparten objetivo aunque sus análisis se centran en medios diferentes, son el Reconocimiento de Emociones en Audio y el Procesamiento de Lenguaje Natural (PLN). Estas tecnologías tienen como objetivo analizar y estimar el connotación subjetiva que alguien plasma en audio y texto con la mayor precisión posible. Dividiendo este proyecto en dichas partes, se han diseñado una serie de experimentos con el fin de estudiar y analizar los sentimientos en audio y el texto extraído gracias a una herramienta de transcripción. Junto a estas aplicaciones, otra herramienta estudiada y desarrollada es la de *Speaker Diarization* o separación de interlocutores. Este es el proceso que particiona un audio de entrada en segmentos homogéneos, según la actividad de los hablantes. La conjunción de los sistemas desarrollados nos llevaría a un análisis completo de un audio original, ofreciéndonos más detalles sobre el sentimiento y las emociones reflejadas en el mismo.

Palabras Clave: Aprendizaje Automático, Análisis de Sentimientos, Inteligencia Artificial, Procesamiento de Lenguaje Natural, Segmentación de Interlocutores, Reconocimiento de Emociones.

Capítulo 1

Introducción

Para comunicarnos, los seres humanos usamos la voz, y junto a esta añadimos información no verbal que depende de los sentimientos que queramos transmitir al resto de interlocutores. Las emociones que sentimos condicionan nuestra manera de vivir y relacionarnos. Se puede decir que el ser humano está gobernado por sus emociones, ya que influyen en nuestras decisiones, acciones y la propia percepción de la realidad que tenemos. Por ello a lo largo de la historia ha habido varios intentos de clasificar y categorizar dichas emociones basándose en diferentes aspectos. El psicólogo Paul Ekman en 1983 [Ekm83] identificó, tras un estudio de la expresión facial humana, seis tipos de emociones básicas: felicidad, tristeza, miedo, ira, disgusto y sorpresa. Su clasificación fue usada como base por otros psicólogos que pensaban que había más emociones y que algunas surgen como combinación y diferentes intensidades de otras más básicas. Es el caso de Robert Plutchik, que confeccionó y explicó la Rueda de las Emociones en sus obras [Plu80] y [Plu91] desde 1980. Es probablemente la clasificación de emociones más reconocida y usada por especialistas a día de hoy. En su taxonomía hay 4 pares de emociones principales opuestas de las que surgen otras secundarias, resultantes de la combinación de estas y los diferentes niveles de intensidad, formando la rueda mostrada en la Figura 1.1.

Otro modelo postulado por Orthony, Clore y Collins[GLO13], en el que se afirma: *"la intensidad de una emoción y qué la origina son aspectos que una teoría de la emoción tiene que examinar"*. Sin embargo, dado que este modelo clasifica el espectro en 22 emociones diferentes, se ha obviado su uso dentro del procesamiento de la información, debido a un coste desmesurado, puesto que cuantas más emociones se intenten clasificar, más información se necesita, por ende, más tiempo de procesamiento para obtener unos resultados precisos.

Los seres humanos somos capaces de sentir por empatía la emoción sugerida por la entonación en el discurso de otra persona. Es algo que hacemos de forma natural, sin pensar en las características físicas de la onda de sonido. Sin embargo, un estudio sobre los cambios en la voz al transmitir diferentes emociones puede ser de gran utilidad, como el realizado en [AZS17], que se explica en la siguiente Tabla 1.1.

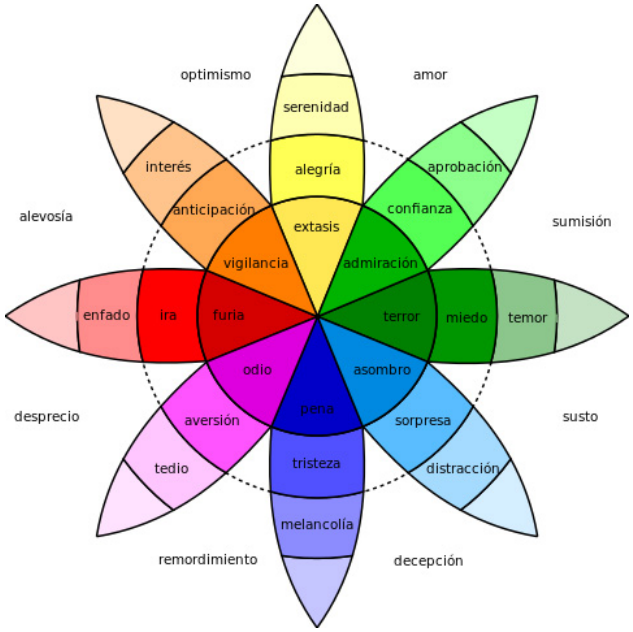


Figura 1.1: Esquema de Rueda de Emociones, Robert Plutchik. [Plu91]

Tabla 1.1: Características de las Emociones Expresadas en la Voz

	Enfado	**Felicidad**	**Tristeza**	**Miedo**	**Disgusto**
Ratio (Velocidad)	Ligeramente más rápido	Rápido o lento	Ligeramente más lento	Muy rápido	Mucho más rápido
Tono medio	Mucho más alto	Muy alto	Ligeramente más bajo	Mucho más alto	Mucho más bajo
Rango de Tono	Muy amplio	Muy amplio	Ligeramente más estrecho	Muy amplio	Ligeramente más amplio
Intensidad	Alta	Alta	Baja	Normal	Baja
Calidad de voz	Desde el pecho, con respiraciones	Con respiraciones, estridente	Resonante	Pronunciación irregular	Desde el pecho, quejicoso
Cambios de Tono	Abrupto	Suave, inflexiones hacia arriba	Inflexiones hacia abajo	Normal	Amplios, inflexiones hacia abajo
Articulación	Tensa	Normal	Trabada	Precisa	Normal

El habla humana contiene tanto información lingüística, que se expresa de forma explícita, como para-lingüística, que se da de forma implícita. La información lingüística identifica los patrones cualitativos que el hablante ha articulado (palabras, sílabas), mientras que la información para-lingüística se mide por características cuantitativas que describen la onda física del sonido transmitido.

Además de analizar la información no verbal transmitida por la voz es posible hacer que una máquina procese el contenido de un texto para estimar el sentimiento reflejado por el mismo y clasificar documentos en función de si transmite una connotación positiva o negativa. A este campo se le denomina Análisis de Sentimiento, y para extraer la información subjetiva de un documento hace uso de herramientas de Procesamiento de Lenguaje Natural. Para realizar un procesamiento completo de un discurso en forma de

audio, un modelo debe tener en cuenta estos siete puntos, propuestos en [TA19]:

- Nivel fonético de la pronunciación.
- Nivel morfológico de las palabras, que contienen un significado con prefijos y sufijos.
- Nivel de significado léxico de las palabras y estructuras de la oración.
- Nivel sintáctico de gramática de oraciones.
- Nivel semántico de los significados de las oraciones.
- Nivel de discurso, centrada en diferentes partes del texto.
- Nivel pragmático para analizar el conocimiento externo al documento.

La comunicación entre máquina y humano es compleja y hasta ahora las emociones no han jugado el papel esencial que tienen, ya que en la actualidad las máquinas pueden reconocer "lo que se dice" o "quién lo dice", como podemos observar en múltiples interfaces de voz que nos hacen la vida un poco más fácil. Si una máquina es capaz de entender también el "cómo se dice" podremos hacer que la interacción entre humano y máquina sea mucho más agradable y natural. Esto podría ser de especial utilidad en escenarios como el diagnóstico psiquiátrico, la ayuda a personas con diversidad funcional o la detección de mentiras, entre muchos otros campos.

En este trabajo se ha desarrollado un análisis completo del audio, incluyendo el análisis de sentimientos a partir de técnicas de PLN, separación entre hablantes y reconocimiento de emociones en el propio audio. Creemos que la conjunción de estos tipos de clasificación podrán aportarnos una mejor visión de las emociones y sentimientos expresados por una persona en una pieza de audio. En la Figura 1.2 se muestra un ejemplo del proceso de reconocimiento de emociones tanto en audio como en texto.

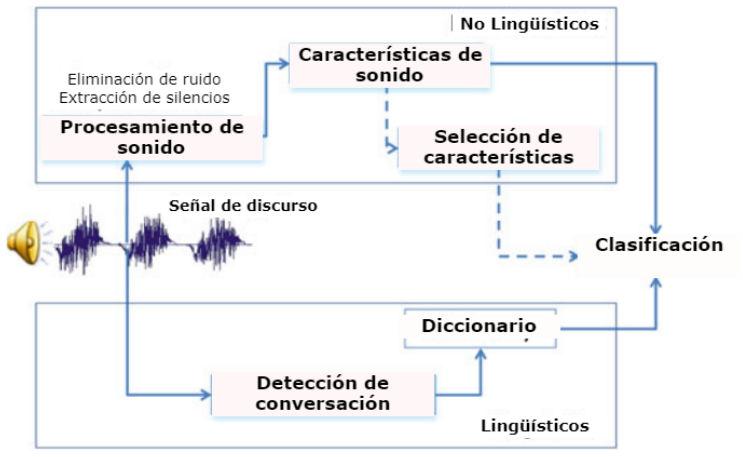


Figura 1.2: Proceso de Reconocimiento de Emociones [Ana12]

Las emociones expresadas en redes sociales, audio o conversaciones son procesadas de manera inconsciente o conscientemente en el caso de los humanos, pero las máquinas, a pesar de ser infinitamente más rápidas, requieren de una información y entrenamiento previos antes de poder sacar cualquier tipo de conclusión. El fin de este análisis es detectar

las emociones dentro de cualquier ámbito, con múltiples fines, como el marketing de productos y servicios. Las redes sociales llegan hasta el rincón más recóndito del planeta y en ellas todo tipo de usuarios publican masivamente sus pensamientos, experiencias y opiniones, haciendo posible analizar su sentimientos hacia cualquier aspecto en concreto. Las técnicas de PLN se basan en dos modelos principales:

1. Los **Modelos Lingüísticos**, basados en gramáticas y recursos lingüísticos.
2. Los **Modelos Probabilísticos**, basados en datos anotados y aprendizaje automático. A su vez, se pueden dividir en **análisis léxico** (peso y connotación de ciertas palabras) y **análisis más complejos**, con los que también se les da importancia a las estructuras que pueden expresar emociones.

Dentro del campo de esta tecnología se diferencian dos campos a la hora de categorizar los sentimientos para mejorar las predicciones de los algoritmos. El modelo categórico asocia datos con etiquetas, y es gracias a este que se pueden hacer más "tangibles.[es]tas emociones para un ordenador, pero como inconveniente presenta una falta de etiquetas que abarquen la gran cantidad de emociones que se pueden categorizar. Además, a mayor cantidad de etiquetas, mayor es la necesidad de una base de datos más grande y más poder computacional. El modelo dimensional representa los datos, basándose en un conjunto de medidas cuantitativas, y es capaz de detectar cambios más sutiles dentro de las emociones que para el modelo categórico habrían pasado desapercibidas.

Una investigación muy similar a una parte de este trabajo fue realizada por Chuen-Hsien Wu [CHW06], donde se propone un enfoque aplicable al reconocimiento de emociones mediante contenido textual. Dividió el estudio en dos fases: entrenamiento y prueba, y a su vez analizó las emociones del corpus respecto al componente del dominio (dependiente o independiente). El componente independiente del dominio es la parte semántica “obtiene”,mientras que el componente dependiente del dominio es la parte semántica “algo que es beneficioso”.

En los últimos años, la atención al cliente ha aumentado, con el avance de las empresas cimentadas dentro de la red. Hasta hace poco, se daba mayor importancia a la calidad de los productos y administración en sí, antes que al servicio al cliente. De acuerdo con una investigación realizada por la consultora *Lee Resources*, el 80 % de las empresas afirman que ofrecen una buena atención al cliente, mientras que solamente un 8 % de los clientes concuerdan con tal dato. SalesForce, una compañía de software, que cuenta con productos con enfoque en la atención al cliente, marketing, inteligencia artificial y gestión de comunidades publicó los siguientes datos:

- El 70 % de la experiencia de compra online de un cliente se basa en la forma en que fue atendido, más allá del precio o producto.
- Si durante el proceso de atención al cliente, se consigue resolver un problema, el 70 % de los clientes volverá a comprar otro producto.
- Se estima para 2020, que el 90 % de los negocios en internet usarán las redes sociales para brindar una mejor atención al cliente.
- Los cliente se gastan entre un 20 y 40 % más cuando las empresas se involucran con su problema y responden a través de las redes sociales.

Recientemente se ha realizado una investigación de llamadas ([Seh18]) mediante el análisis automático de sentimientos, siguiendo la estructura de la Figura 1.3.

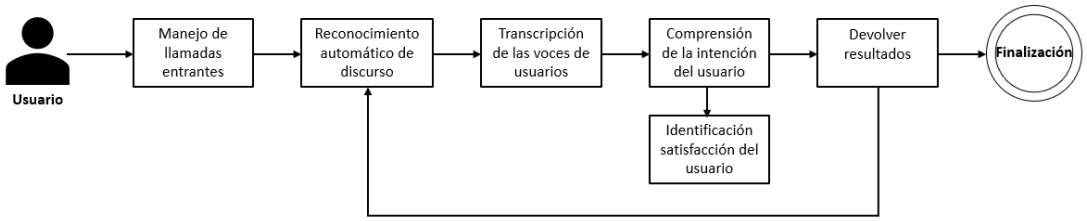


Figura 1.3: Estructura ASR combinada con un sistema IVR.

El objetivo de esta investigación es analizar las diferentes técnicas aplicables en este campo para observar su precisión y funcionamiento. Se han usado señales de tono dual y multi-frecuencia, sistemas de reconocimiento de audio automáticos combinados con Sistemas de Respuesta Interactiva (del inglés *Interactive Voice Response* (IVR)), clasificadores varios (Bayes, Entropia y Boosted Trees), y *datasets* de partes como centros de atención virtuales y computación en la nube. Se busca resolver el problema del servicio al cliente, como se ha mencionado anteriormente, y para ello se ha entrevistado a 100 personas. Las estadísticas mostraban que un 60 % de los usuarios estaban satisfechos con el servicio ASR, mientras que un 25 % lo llegó a rechazar. A su vez, una gran mayoría sintió un vacío de comunicación con el sistema IVR.

Para llegar a este punto, se procesó la información mediante una bolsa de palabras multinomial de Bayes. Tras obtener la información y pasarse a texto, tokenizada y agrupada según el peso, similar a los procesos de PLN. A partir de aquí el sistema aprende a clasificar con mayor precisión, llegando a saber si el cliente está o no satisfecho, ya que hay partes despreciadas del texto al no tener peso en el sentimiento.

Después de analizar los trabajos citados, se descubrió la inexistencia de un artículo, o experimento que introdujese las múltiples tecnologías que han sido asignadas para este, y es por eso que desarrollar cada parte ha sido un desafío desde el principio, pero con resultados tangibles en cada campo.

1.1. Motivación

Como se ha mencionado antes, el objetivo principal de este proyecto es entrenar y mejorar una serie de modelos que detecten las emociones y quiénes las producen dentro de un audio. La mayoría de trabajo previo revisado solo se centra en el reconocimiento de emociones en audio a través de la extracción de Coeficientes Cepstrales en las Frecuencias de Mel (MFCCs), pero creemos que plantear un sistema híbrido que lo integre con el análisis de sentimiento en texto puede mejorar significativamente su fiabilidad. Para ello será necesario diseñar y entrenar varios modelos que se centren en cada tarea del proceso.

El primer modelo a diseñar es el de *Speaker Diarization*, que es la parte de este sistema centrada en analizar audios con el fin de detectar la separación entre hablantes. El fin de

este modelo es recortarz transmitir las partes de audio de interés a los otros dos procesos para conseguir un análisis completo. La parte de reconocimiento de emociones en audio recibirá los audios segmentados, de los que extraerá sus características para crear un modelo capaz de predecir la emoción expresada en una pieza de audio.

Paralelamente, los audios segmentados son introducidos en una herramienta de transcripción de audio a texto que nos devolverá los datos de entrada para nuestro sistema de análisis de sentimientos. Esta parte procesará los textos transcritos de los audios, creando un modelo cuya función será predecir la positividad de un mensaje.

La combinación de los resultados de ambas partes podría mostrar un punto de vista innovador al reconocimiento de emociones en audio clásico, además de aportarle robustez y fiabilidad. Creemos que otros campos de conocimiento como la medicina, psicología o ciencias forenses podrían encontrar aplicaciones útiles a partir de la información que nuestros experimentos pueden proporcionar.

1.2. Objetivos

Desde el inicio, se han pautado pequeñas metas, para avanzar poco a poco, y de la mejor forma posible. Inicialmente, se organizó al grupo, para investigar cada campo que se trata en este trabajo, siendo el primer objetivo: aprender las bases e ir avanzando. Tras múltiples semanas de lectura, y recopilación de datos, se expusieron las herramientas que se iban a utilizar, con el fin de comprender su funcionamiento, y cómo obtener los resultados necesarios para cada sección del trabajo. Podemos concluir los objetivos con la siguiente lista:

1. Investigar y aprender las bases de *Machine Learning* e Inteligencia Artificial, así como sus múltiples técnicas de extracción de características, pre-procesamiento y clasificación.

2. Decidir qué herramientas y programas en concreto serán necesarias para diseñar los modelos.

3. Mejorar los modelos con la optimización de sus parámetros.

4. Optimizar el rendimiento de modelos implementados.

5. Extraer conclusiones sobre los sistemas creados y su posible combinación en trabajo futuro.

1.3. Estructura de la Memoria

Este proyecto comienza con una explicación de la clasificación de nuestras emociones y su expresión en la voz, para después describir las tecnologías que se encargan de su análisis, repasando su historia y explicando sus herramientas. Una vez finalizado el marco teórico se expondrá el diseño, resultados y conclusiones de los experimentos desarrollados. Tras esta introducción los próximos Capítulos se abordarán siguiendo la siguiente estructura:

En el Capítulo 2 se cubre el marco teórico más general en temas de Inteligencia Artificial, donde se explican los principales conceptos, así como las particularidades dentro de los experimentos de Segmentación de Interlocutores y Reconocimiento de Emociones en Audio.

El siguiente Capítulo, el número 3, se aborda con más detalle el experimento de Análisis de Sentimiento en texto y las técnicas de Procesamiento de Lenguaje Natural usadas en el mismo, ofreciendo y comparando diferentes herramientas de uso abierto.

En el Capítulo 4 se repasa el estado del arte, donde se presenta el trabajo previo revisado y los resultados de las aproximaciones más relevantes realizadas para cada uno de los experimentos, comentando también las bases de datos, extractores de características y clasificadores empleados.

El Capítulo 5 presenta nuestra contribución, consistente en los experimentos desarrollados que darán lugar a los modelos que buscamos tras el entrenamiento. Aquí se exponen las decisiones y consideraciones que se tomaron en el proceso de diseño de cada experimento, así como las herramientas que fueron necesarias.

Los propios experimentos se cubren en extensión en el Capítulo 6, donde se expone el procedimiento que se ha seguido en el procesamiento de los datos utilizados por los clasificadores para entrenar y los resultados obtenidos en forma de tablas y figuras para su fácil comparación y comprensión.

A las conclusiones extraídas de todo el proceso de investigación y desarrollo de cada experimento les corresponde el Capítulo 7, donde también se habla de los objetivos no logrados y que quedan pendientes para trabajo futuro.

El Capítulo 8 recoge las aportaciones individuales de cada miembro del grupo.

Finalmente, los Capítulos 9 y 10 se presentan la introducción y conclusiones en inglés.

Capítulo 2

Marco Teórico

El objetivo de este capítulo es proporcionar una visión general de los conceptos más relevantes para el desarrollo de la investigación. En la Sección 2.1 se presenta una breve introducción a la inteligencia artificial y la evolución que ha tenido. Seguidamente, se describen las técnicas de aprendizaje automático más relevantes para el desarrollo de este trabajo en la Sección 2.2. En la Sección 2.3 se detallan los conceptos fundamentales y las tecnologías utilizadas del proceso de segmentación de interlocutores. Finalmente, los principales sistemas de reconocimiento de emociones audio se describen en la Sección 2.4.

2.1. Inteligencia Artificial

La Inteligencia Artificial (AI) se centra en desarrollar y mejorar el comportamiento de las máquinas para que se asemejen más al humano. En nuestro caso, la AI tiene como función asemejar datos que se introducen, aprender sobre ellos y saber usarlos para la tarea designada. Los ejemplos más comunes son el reconocimiento de escritura, de patrones y del habla, aunque por excelencia es mucho más famoso el caso de los videojuegos.

Dentro de esta tecnología, existen dos ramas de pensamientos: Inteligencia Artificial Convencional, e Inteligencia Computacional. La primera está basada en el análisis formal y estadístico del comportamiento humano, y el aprendizaje mediante datos empíricos y toma de decisiones. La segunda implica un desarrollo o aprendizaje interactivo, intentando evitar los métodos heurísticos de la primera rama, centrándose en el estudio de mecanismos adaptativos para permitir el comportamiento inteligente de sistemas complejos y cambiantes.

2.1.1. Historia de la Inteligencia Artificial

La creación e investigación de esta ciencia comienza alrededor de 1843, cuando Ada Lovelace publica un artículo sobre la Máquina Analítica de Babbage, donde ponía en duda si una máquina podría pensar en algún futuro no muy lejano, a pesar de que la concepción social era que las máquinas solo podían hacer lo que un humano les ordenase. Un siglo más tarde, otro punto clave fue la creación de dos máquinas de ajedrez, por Leonardo Torres y Quevedo [Red04], las cuales, despertaron ideas sobre la capacidad de

pensar de estas. Esta no fue su única invención, pero sí la más próxima a AI; el resto eran creaciones algebraicas y de cálculo. Tiempo después se daría a conocer una de las personas más famosas en este campo: Alan Turing. En 1939, con el inicio de la Segunda Guerra Mundial, Turing, junto con Bletchley Park, creó el *Bombe*, para poder descifrar la máquina alemana Enigma [Zab95].

15 años después, el término Inteligencia Artificial había sido oficialmente acuñado por Marvin Minsky y John McCarthy, en 1956. Ambos científicos organizaron un evento de 8 semanas, donde se reunirían investigadores de múltiples campos para centrarse en crear máquinas capaces de simular el comportamiento humano. A estos investigadores se les denomina fundadores de la AI,con el ejemplo de Nathaniel Rochester, quien en un futuro no muy lejano crearía el primer ordenador científico IBM 701 [AR52]; y Claude Shannon quién fundó la teoría de la información [Gap99]. Los avances siguieron llegando, con un gran hito en 1964. Joseph Weizenbaum creó el programa ELIZA [Wei66a], una herramienta de procesamiento del lenguaje natural capaz de simular una conversación sencilla con un humano, y siendo uno de los primeros programas en intentar pasar el test de Turing. A su vez, otros programas, como el de Herbert Simon también tuvieron mucho éxito en la resolución de programas sencillos como las Torres de Hanoi. Estos sucesos hicieron que se afirmase la posible creación de una máquina capaz de pensar en los próximos 3 a 8 años.

Sin embargo, poco después comenzó un gran rechazo hacia este campo, con artículos donde se exponían opiniones negativas sobre sus limitaciones, afirmando que se había alcanzado el límite, y no llegaría más lejos que una experiencia *amateur* en los juegos como el ajedrez, y el sentido común sencillo.

Pasaría bastante tiempo hasta que se volviese a indagar la Inteligencia Artificial. En 1997, Deep Blue, de IBM, sería capaz de ganar a Gary Kasparov en una partida de ajedrez, gracias a los árboles de búsqueda, que permitían al programa analizar 200 millones de movimientos e ir siempre 20 por delante del oponente [Wei66b]. Pero este método utilizado no era óptimo en otras áreas como el reconocimiento de caras u objetos totalmente distintos, y se necesitaba un algoritmo capaz de entender la información externa, y ser capaz de usarla sin confusiones plasmadas en los resultados. Este problema ya fue debatido por Donald Hebb en los años 40, con una teoría que replicaba las neuronas humanas para aprender, siendo este, el inicio de las Redes Neuronales.

En 2015, las Redes Neuronales, volvieron dentro del campo *Deep Learning*, con el programa AlphaGo, creado por Google, que fue capaz de ganar al campeón mundial en el juego *Go* [Che16]. Este programa tendría que realizar cálculos mucho más densos que en el ajedrez, y gracias a su eficiencia, es una de las herramientas más usadas y mejoradas hoy en día.

Actualmente, el potencial completo no ha sido descubierto, y su aplicabilidad es inmensa. Autores como Tambe y Capelli, con su artículo [TCY19], explican cómo puede usarse AI para organizar recursos en el ámbito del trabajo; o como Overgoor, Chica, Rand, and Weishampel, con su artículo [OCRW19], en el que se comenta cómo se podría solucionar y manejar de forma más eficiente los negocios y marketing mediante esta tecnología.

2.2. Aprendizaje Automático

El Aprendizaje Automático (AA), o como es ampliamente conocido en inglés *Machine Learning* (ML), estudia algoritmos capaces de mejorar de manera autónoma a través de la experiencia y de la exposición a datos. Los algoritmos pertenecientes a este paradigma tienen en común la capacidad de extraer conocimiento y construir un modelo con el que realizar predicciones a nuevos datos. Durante muchos años, e incluso en la actualidad, el AA ha sido visto como un sub-campo dentro de la AI pero a partir de los años 90 los objetivos que perseguían ambos campos empezaron a diferir. Mientras que la AI perseguía el aprendizaje de un sujeto para lograr relacionarse exitosamente con su entorno, el AA se inclinó más hacia el lado probabilístico y estadístico para resolver problemas prácticos.

Los diferentes algoritmos dentro de este paradigma se agrupan de varios tipos dependiendo de la forma en la que aprenden, siendo los tres principales los siguientes:

2.2.1. Aprendizaje Supervisado

El aprendizaje supervisado es una técnica de AA en la que para deducir una respuesta los modelos deben ser entrenados con vectores de pares de objetos, siendo un componente los datos de entrada y el otro los resultados deseados. Tras el entrenamiento el modelo creará una función de inferencia con la que será capaz de hacer una predicción a cualquier dato nuevo que reciba. En este tipo de aprendizaje es fundamental que el conjunto de entrenamiento sea consistente y representativo de todo su dominio a la vez que no contiene excesivo ruido, ya que algunos de estos algoritmos son muy sensibles a sobre-ajustar.

Dependiendo de la naturaleza de los datos a analizar podemos diferenciar dos tipos de aprendizaje supervisado: clasificación y regresión. La principal diferencia entre un clasificador y un regresor está en el dominio de los datos sobre los que trabajan.

Los datos de entrada y la predicción de un clasificador pertenecerán a una categoría dentro de un conjunto determinado de categorías. Durante el entrenamiento el clasificador tratará de dividir el espacio de muestras intentando minimizar el número de objetos que quedan fuera del sub-espacio perteneciente a su categoría (ver Figura 2.1). En este proceso se construyen y optimizan las “fronteras” entre distintas clases, dando lugar a un modelo que nos dará una predicción de clase a la que pertenece cualquier muestra nueva a la que se le exponga.

En cambio un regresor trabaja sobre un dominio continuo, lo que significa que iteración a iteración irá minimizando el error de sus predicciones dando lugar a una función matemática que, tras analizar los valores de las características de una muestra, nos devolverá un valor numérico dentro de un rango como predicción. La función extrapolada de los datos puede ser de muchos tipos o una combinación de distintos tipos. La regresión lineal es muy común debido a su sencillez y bajos tiempos de procesamiento, aunque también se suelen usar funciones cuadráticas, logarítmicas y exponenciales, dependiendo de la forma de distribuirse de los datos como se observa en la Figura 2.2.

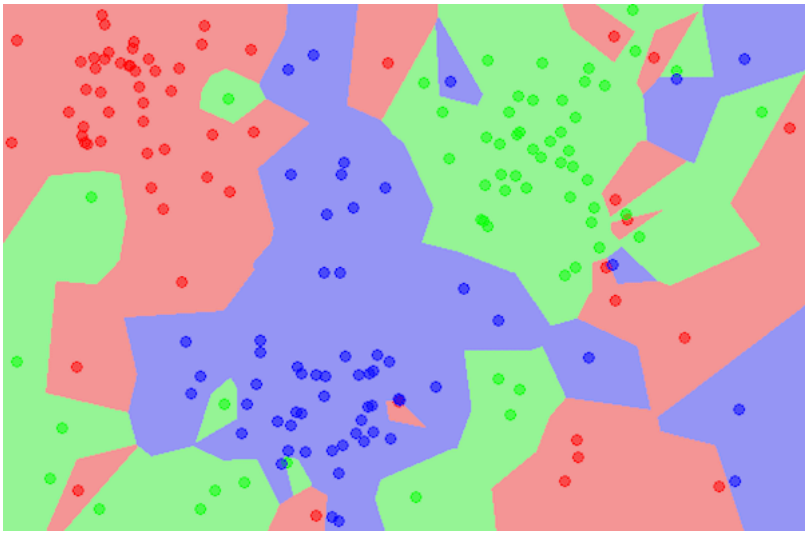


Figura 2.1: Ejemplo de clasificación de 3 categorías [clf21]

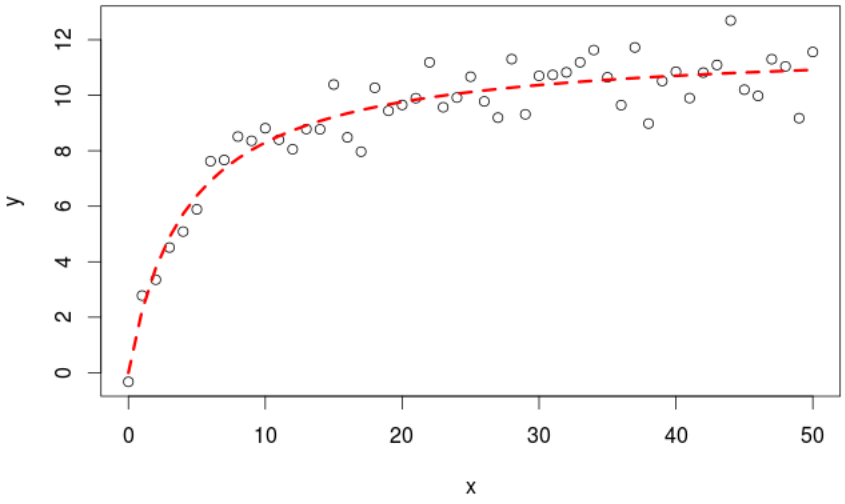


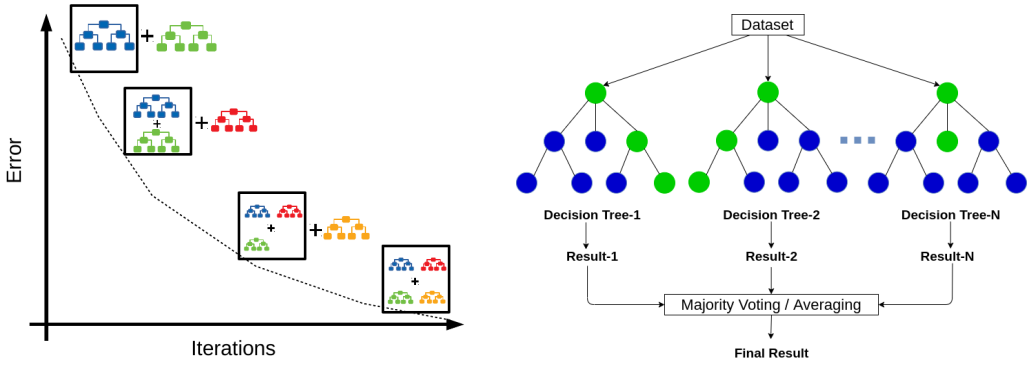
Figura 2.2: Ejemplo de regresión no lineal [Her18]

Existen muchos tipos de algoritmos diferentes que pertenecen al paradigma del aprendizaje supervisado, y muchos de ellos se inspiran en procesos que podemos apreciar en la naturaleza. Estos son algunos de los tipos de clasificador más destacados:

- **Reglas de asociación**: Se utilizan para descubrir hechos y relaciones entre las variables de un conjunto de datos.
- **Árboles de decisión**: Esos algoritmos usan árboles de decisión como modelo predictivo para pasar de observaciones de un objeto a conclusiones sobre su etiqueta. En las estructuras de árbol las hojas representan etiquetas de clase y las ramas son las conjunciones de características que conducen a esas clases. Un mayor

número de árboles de decisión proporciona un modelo más robusto y previene el sobreajuste (overfitting), aunque puede aumentar drásticamente el tiempo de ejecución y entorpecer el proceso. Por ello es muy recomendado reducir el número de características introducidas al clasificador. En este proyecto trabajaremos con dos algoritmos basados en árboles de decisión: GB y RF Las dos principales diferencias entre ambos son la forma en que los árboles son construidos y la forma de realizar la predicción:

- El clasificador Gradient Boosting entrena los árboles de manera secuencial, de forma que cada nuevo árbol intenta mejorar los errores del anterior, combinando los resultados en el proceso para obtener la predicción final. Esto permite ir reduciendo el error en cada iteración, como es posible ver en la Figura 2.3(a)
- Por su parte, Random Forest entrena a cada árbol con una muestra ligeramente distinta de los datos de entrenamiento, agregando al final del proceso los resultados de todos los árboles para obtener una predicción, como queda representado en la Figura 2.3(b). La combinación final de resultados puede variar pero es común hacer una media o que escoja la mayoría.



(a) Gradient Boosting [Pal20] (b) Random Forest [Kal21]

Figura 2.3: Comparación del funcionamiento de Gradient Boosting y Random Forest

- **K Vecinos más Próximos**: El algoritmo K Vecinos más Próximos (KNN) o k-vecinos más próximos estima la similitud entre el objeto a clasificar y los `k` objetos más cercanos a él, asignándole la clase predominante en sus vecinos (Figura 2.4). La clasificación depende de la distancia entre los objetos, por lo que normalizar las diferentes escalas de las características puede mejorar drásticamente la precisión. Elegir un buen valor para `k` es importante ya que normalmente grandes valores pueden reducir la cantidad de ruido pero a la vez el modelo puede exponerse a sobreaprendizaje. Además un valor más bajo hace que las fronteras entre clases sean más nítidas, por lo que dichos valores suelen tener más efectividad a la hora de agrupar clases.

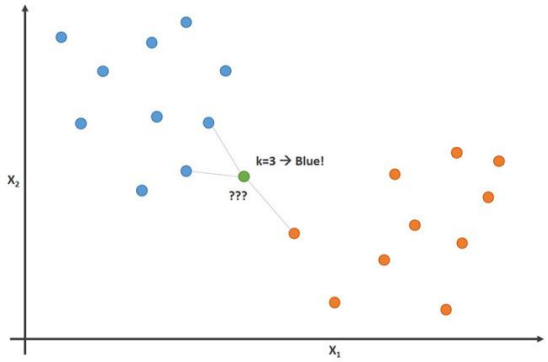


Figura 2.4: Representación sencilla del algoritmo KNN [McC19]

- **Algoritmos genéticos**: Simulan la evolución y la selección natural de las especies. Para ello usa mecanismos de mutación, selección y cruce para que solo los mejores individuos continúen en la población activa y tengan descendientes.

- **Redes neuronales artificiales**: Tratan de imitar la estructura y funcionamiento de las neuronas de nuestro cerebro. Las neuronas artificiales están conectadas entre sí para transmitir la información, que es regulada por los pesos de las conexiones (sinapsis) en el proceso de aprendizaje, y su salida es calculada por una función no lineal de la suma de sus entradas. Normalmente las neuronas se dividen en capas que dan lugar a diferentes tipos de transformaciones a las señales que reciben.

- **Máquinas de vectores de soporte** (SVM): Se caracterizan por realizar una regresión lineal en un espacio cuyas dimensiones son las características de los objetos, creando un hiperplano que divide el espacio en el número de categorías deseado intentando mantener la mayor distancia entre ellas. A dicho hiper plano se le denomina vector de soporte, que es la línea divisoria observable en la Figura 2.5. El clasificador SGD es una implementación de SVM y la principal diferencia que aporta SGD es su modo de aprender, en el que la pérdida es calculada en cada muestra, disminuyendo cada vez el ratio de aprendizaje.

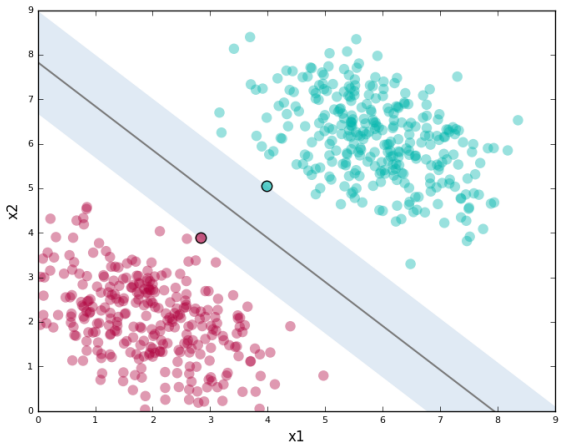


Figura 2.5: Vector de soporte generado por una SVM [Kum20]

- **Redes Bayesianas**: Son modelos gráficos probabilísticos que representan un conjunto de variables aleatorias y sus dependencias a través de un grafo acíclico dirigido.

- *Modelo oculto de Markov:* Su funcionalidad consiste en intentar determinar los parámetros ocultos en base a otros observados. En un modelo de Markov habitual los estados son directamente visibles por el observador, pero en este caso sí es posible ver las variables a las que estos han influenciado. A veces esta técnica es considerada como una versión simplificada de una red bayesiana.

Para que los algoritmos de aprendizaje supervisado puedan mejorar en su proceso de entrenamiento es necesario establecer una serie de **métricas** que servirán que medirán cuánto se parecen las predicciones de los modelos con las etiquetas verdaderas. Dichas métricas se basan en la comparación entre los valores predichos y reales de cada categoría en matrices de confusión binarias, dando lugar a predicciones de cuatro tipos: Verdaderos Positivos (VP), Falsos Positivos (FP), Verdaderos Negativos (VN) o Falsos Negativos (FN).

- **Precisión**: La precisión (en inglés *precission*) mide la calidad del modelo en tareas de clasificación, es decir, la cantidad de aciertos en las predicciones realizadas.

$$precission = \frac{VP}{VP + FP} \tag{2.1}$$

- **Exactitud**: La exactitud (o *accuracy*) mide el porcentaje de los casos en los que el modelo ha acertado. Sin embargo, si estamos ante un mal modelo, esta métrica puede tener un alto valor, llevando al error de pensar que el rendimiento es mejor de lo que en realidad es.

$$accuracy = \frac{VP + VN}{VP + FP + VN + FN} \tag{2.2}$$

- **Exhaustividad**: La exhaustividad (en inglés *recall*) nos informa sobre la proporción de muestras que el algoritmo es capaz de identificar del número total de positivos reales.

$$recall = \frac{VP}{VP + FN} \tag{2.3}$$

- **F1 score**: Se utiliza para combinar la precisión y exhaustividad en un solo valor que es representativo de la fiabilidad de un modelo. Se calcula haciendo la media armónica entre ambas métricas.

$$F1 = 2 \cdot \frac{precission \cdot recall}{precission + recall} \tag{2.4}$$

Tras el proceso de entrenamiento es momento de evaluar resultados. Para ello es comúnmente usada la técnica de la validación cruzada, o *Cross Validation* (CV) en inglés, que evalúa los resultados de un clasificador entrenado con diferentes particiones

del conjunto de datos. Su finalidad es garantizar que el éxito del modelo es independiente de una partición particular de los datos de entrenamiento y de prueba, ofreciendo los resultados de cada ejecución, así como la media y la desviación típica. Es especialmente útil para detectar casos de **sobre-ajuste**, en los que el modelo no se ajusta a los datos de validación tan bien como a los de entrenamiento. Esto puede ser debido a un reducido tamaño de los datos de entrenamiento o a un alto número de parámetros o características.

2.2.2. Aprendizaje no Supervisado

El aprendizaje no supervisado se distingue del supervisado en que los conjuntos de datos con los que se entrenan los modelos no están etiquetados previamente por un humano. De esta manera el algoritmo trata los objetos de entrada como un conjunto de variables aleatorias, construyendo un modelo de densidad con los que extraer características intrínsecas de los datos. La técnica más común dentro del aprendizaje no supervisado es el *clustering* o agrupamiento, un tipo de clasificación donde no conocemos la naturaleza de las muestras. Los datos de entrada se agrupan en "clusters" según las relaciones entre sus parámetros y al exponer al modelo a un nuevo objeto es capaz de clasificarlo en uno de ellos por sus semejanzas.

Estos son los principales modelos que pertenecen a este tipo de aprendizaje:

- **K-means**: el objetivo de este método es agrupar los n datos en k grupos, donde cada elemento pertenece a uno de los grupos según su valor medio con estos.
- **Agrupamiento por Jerarquía**: mayormente aplicado en minería de datos, este agrupamiento se centra en crear grupos puntuales. Existen dos tipos: Aglomerativas y decisivas. La primera es un agrupamiento ascendente, cada observación se analiza comenzando con un grupo, y a este se le añaden mas observaciones progresivamente. La segunda funciona al contrario, considerando inicialmente grupos de observaciones, que se van dividiendo, o agrupando con mejor precisión *a posteriori*.

Un ejemplo de estos modelos se pueden ver en las Figuras 2.6 y 2.7.

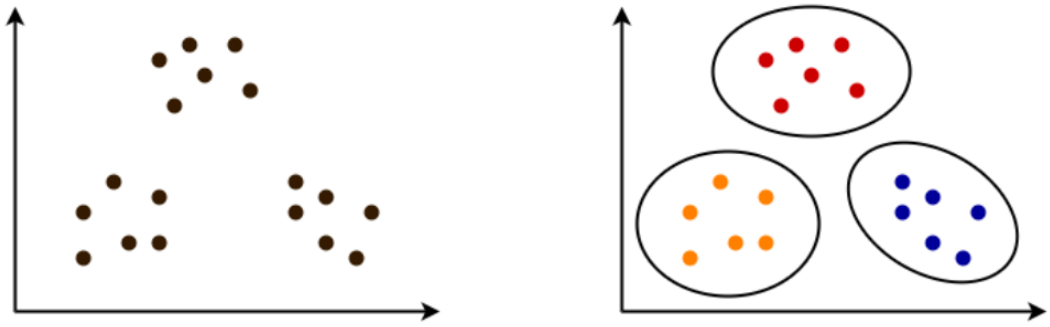


Figura 2.6: Ejemplo del funcionamiento de k-means [Sin20]

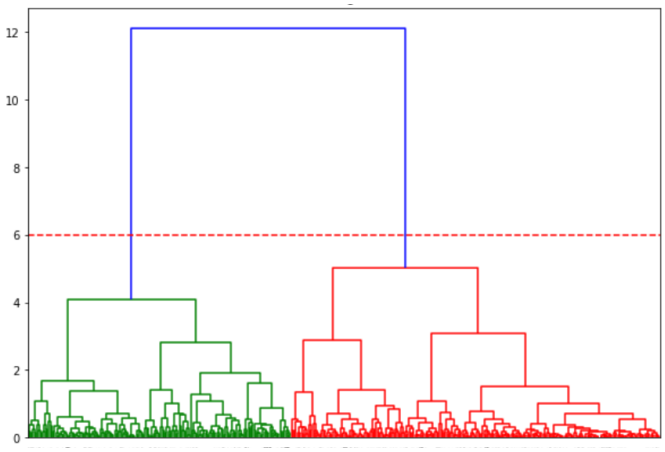


Figura 2.7: Ejemplo de agrupación por jerarquía [Has20]

2.2.3. Aprendizaje por Refuerzo

El algoritmo aprende a través del mundo que le rodea, ya que su información de entrada es el *feedback* o retroalimentación que obtiene del mundo exterior como respuesta a sus acciones. Por lo tanto, el sistema aprende a base de ensayo-error, con el objetivo de maximizar la señal de recompensa para así optimizar su comportamiento dentro del entorno. Es un tipo de aprendizaje muy usado en robótica. Se podría decir que el aprendizaje por refuerzo es un caso más general que las previas categorías. En vez de que un instructor indique al agente qué hacer, el agente inteligente debe aprender cómo se comporta el entorno mediante recompensas (refuerzos) o castigos, derivados del éxito o del fracaso de sus acciones. Entre los métodos de aprendizaje por refuerzo se encuentran:

- **Fuerza Bruta**: divido en dos etapas, en las que se muestran todos los posibles resultados, y se elige el mejor. Sin embargo, si se trabaja con grandes cantidades de resultados, no es la opción más recomendable.

- **Búsqueda Directa**: en este método convierte la búsqueda en una optimización estocástica. Con un conjunto de políticas, el algoritmo seleccionará un sub-conjunto de estas (el que sea más óptimo). Aún así, se puede dar el caso de que su progresión converja a un punto lentamente si la información es ruidosa, impidiendo la evolución rápida del algoritmo.

2.3. Segmentación de Interlocutores

El objetivo general del presente trabajo es el análisis de archivos de audio con ayuda de diferentes técnicas de inteligencia artificial para extraer las emociones expresadas por el hablante. Para ello varios tipos de clasificación entran en juego, como el análisis de sentimientos y el reconocimiento de emociones en audio, que esperamos que en conjunción nos permitan evaluar las emociones reflejadas en un discurso más allá de las denotadas por las palabras. Además será necesario explorar otras facetas del procesamiento del

audio, como es la separación entre hablantes, la extracción de características a través de coeficientes de Mel o la transcripción de audio a texto.

2.3.1. Historia del Análisis de Audio

La historia del análisis de audio y conversaciones se remonta a 1930, cuando Homer Dudley [Seh86]propuso un sistema para replicar la voz humana, y realizar el análisis pertinente. Esta máquina se componía de un analizador, que detectaba los niveles de energía transmitidos por unas teclas, y un sintetizador, que revertía el proceso escaneando la información del analizador, para transmitirla a una red de filtros que producían el sonido. Después de haber realizado tal invención, los estándares pasaron a estar fundamentados mayoritariamente sobre la cuantificación del espectro del poder que genera la voz (junto con variantes como cepstrum).

Posteriormente, las investigaciones tomaron un rumbo algo distinto, apuntando al reconocimiento de la voz. Un ejemplo sería el estudio y experimentos de 1952 realizados por Davis, Biddulph y Balashek [KHDB52] para reconocer números dictados por un único interlocutor. La precisión de este proyecto llegó a ser de 97-99 %, ajustando los parámetros a cada hablante; si se ponía a prueba con varias personas dictando la secuencia de números, la precisión descendía al 50-60 %. Internamente, el algoritmo se basaba en dividir la frecuencia del audio en dos canales y comparar los datos de referencia con los números dichos de forma oral (los coeficientes que se sacan de la voz del interlocutor). En el caso de que un dígito desconocido se hubiese dictado, el programa buscaba la mejor coincidencia entre los datos mencionados anteriormente, y se daba como resultado final. Pero este experimento dejaba un caso sin resolver: ¿qué pasaba si no se dictaban dígitos?.

En 1960 varios laboratorios japonenses investigaron sobre el análisis de vocales dentro de sílabas de un discurso, y se consiguieron resultados notables, como los que presentaron Suzuki y Nakata [JS61]. Su algoritmo se dividía en los siguientes pasos: transformación del audio en frecuencia de espectro, agrupación de los canales según la detección de vocal, dicotomización del agrupamiento anterior, operación lógica basada en la salida cuantizada, y finalmente el código con resultado final. En otras partes del mundo, se investigó también sobre la detección de vocales y consonantes, y se vieron los primeros avances propiamente dichos en el reconocimiento de discursos. Un ejemplo sería Tom Martins, con la fundación de Threshold Technology, y el uso de ASR (Automatic Speech Recognition) en aplicaciones sencillas, como la organización de paquetería dentor de FedEx.

El dictado por voz también fue desarrollado por IBM, en la década de los 70 [ABB+39], con un sistema llamado Tangora-5, que posteriormente pasó a ser Tangora-20, debido a la diferencia de 5000 a 20000 palabras que contenía en su memoria. Se basaba en Modelos Ocultos de Markov (HMM). Al recibir un dictado exterior, la máquina, con su entrenamiento y HMM predecía las palabras que tenía que procesar con robustez y precisión. La robustez era el aspecto referido a ruidos y factores externos al interlocutor que podían afectar al dictado, denominado como efecto Lombard, que podía modificar el tono de voz de cada interlocutor, según los factores que le rodeasen. Este problema se resolvía en mayor medida mediante una compensación realizada en el procesamiento del audio: bandas críticas (para separar el ruido del audio), y transformadas de Fourirer

(acotar la frecuencia del audio).

Los modelos de Markov mantuvieron y aumentaron su fama dentro de este campo de investigación, puesto era posible analizar el mismo audio reproducido por voces distintas (acústicos diferentes y estructura lingüística similar). Es por esto que HMM es una medida de probabilidad que usa las cadenas de Markov y sus distribuciones para cuantificar la variabilidad en la realización acústica en los sonidos del audio. La idea de este modelo apareció alrededor de los años 60, en el Instituto de Análisis de Defensa, donde Len Baum y Lloyd Welch crearon el algoritmo Baum-Welch. Se referían al modelo Oculto de Markov como un set de funciones probabilísticas, que abarcaban dos distribuciones: una de pre-entrenamiento y otra asociada a la distribución de probabilidad. La aplicabilidad de este modelo fue notoria en predicciones de mercado, y el cripto-análisis; pero no fue hasta los años 80 que no se combinó HMM con "densidades variadas"(mixture densities), y se probó la habilidad que presentaba en tareas de reconocimiento de audio. También hubo otra tecnología que no era nueva, pero se volvía a presentar, puesto que en 1950 falló al producir los resultados esperados. Las Redes Neuronales (ANN) despertaron interés por su similitud con la estructura de un experimento de Procesamiento Distribuido, en el que se conectaban una serie de computadoras, y trabajaban con un sencillo método de propagación de errores, siendo esta una símil con la mente humana. Los objetivos de las redes neuronales eran identificar fonemas o palabras, conseguidos con éxito. Pero el reconocimiento de voz requiere una variación temporal, y esto hace que una red neuronal en su forma más simple no sea suficiente, y se propuso combinar ANN con HMM.

A partir de los años 90, los avances fueron notables en la mejora de los sistemas de vocabulario con modelos de lenguaje sin restricciones, y modelos sintácticos de detección de audio. Todos los anteriormente mencionados, estaban fundamentados en comprensión de lenguaje estocástico, aprendizaje estadístico de modelos de lenguaje, y finalmente la introducción de máquinas de estado para la implementación eficiente de sistemas capaces de comprender un audios con un gran contenido de vocabulario. El aprendizaje no supervisado de segmentación de audio y *clustering* de eventos internos relacionados con el ruido de fondo también tuvo un desarrollo extenso, como se explica en artículos como [SYG92] o [SMW93], donde el principal objetivo es dividir los segmentos de audio entre interlocutores, y el ruido o sonido de fondo no perteneciente a la conversación.

Finalmente, en estos últimos años, se han visto modelos semánticos completos, sistemas de transformación de texto a audio, y el aumento del uso de "Machine Learning"para mejorar los algoritmos de aprendizaje. Deep Learning ha sido considerado como el siguiente paso para poder seguir avanzando en este campo, junto con las redes neuronales y x-vectors [SGRS+18] o d-vectors [JHY+12], que son normalmente representaciones de *embeddings* dentro de Redes Neuronales Profundas (del inglés *Deep Neural Networks* (DNN)).

2.3.2. Tecnologías Usadas

A continuación se detallan las herramientas usadas a lo largo de la historia de las investigaciones de Reconocimiento de voz, con mayor detalle y en profundidad.

2.3.2.1. Vectores de Cuantización

Inicialmente se usaron Vectores de Cuantización, donde se compara un vector introducido de datos ordenados, con un patrón similar, y así comparar resultados. A su vez, esta tecnología es conocida como compresión de data (*data-clustering*). Esta compresión busca reducir el ratio de bits que debe procesar, sin afectar la fidelidad de la información. Un ejemplo es el estudio realizado por [KHDB52] (de los primeros en utilizar este método), o [MAVC14], publicado en 2014, con una aproximación más evolucionada de los vectores de Cuantización, como se muestra en la Figura 2.8. R es el número de elementos que se introducen, S elevado a 1 la cantidad de neuronas competitivas, y S elevado a 2 la cantidad de neuronas lineales.

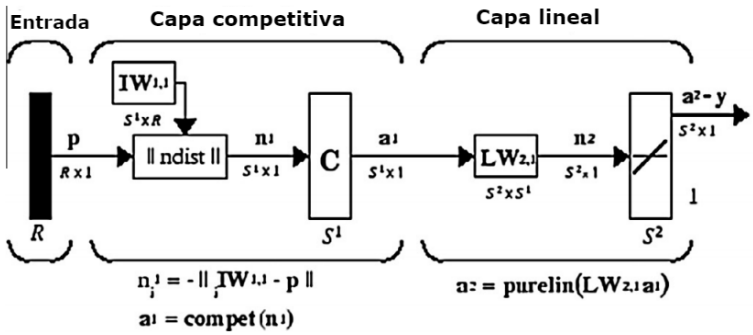


Figura 2.8: Circuitos para reconocer dígitos. [MAVC14]

Internamente, la señal introducida, se dividía en dos partes (dividas por los CPS), para posteriormente analizar la precisión y variedad de entonación cuando se dictaban los números. En el caso de recibir un número desconocido, se daba un resultado, dependiendo de la aproximación de esa información introducida con las Figura 2.9.

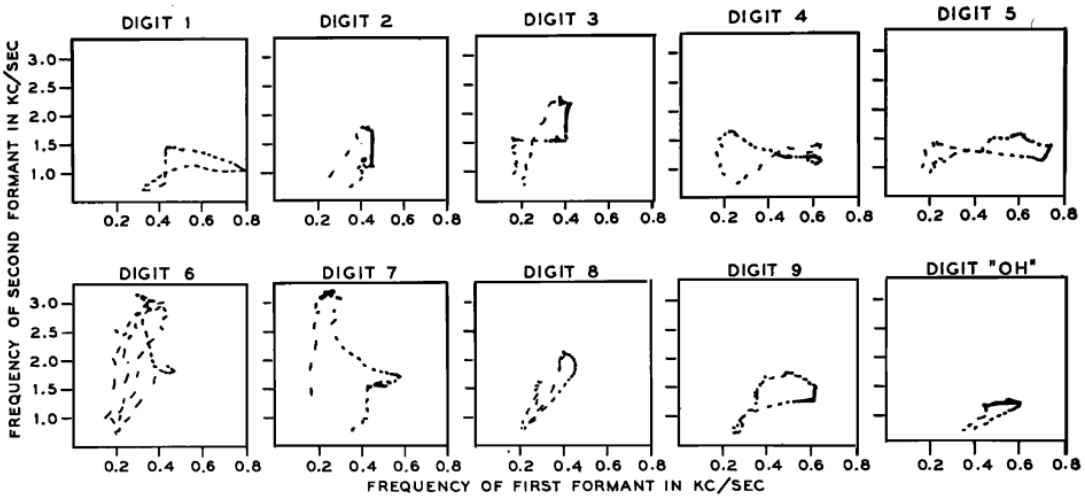


Figura 2.9: Frecuencias establecidas para cada dígito. [KHDB52]

2.3.2.2. Modelos Ocultos de Markov y Redes Bayesianas

Siguiendo en la misma línea de dictado por voz, otra herramienta, más compleja y usada años después es el modelo oculto de Markov, en inglés *Hidden Markov Model* (HMM). Se parte de un modelo con parámetros desconocidos, y el objetivo es calcular esos parámetros a partir de la información visible. El estado actual no es visible, pero las variables influenciadas por este sí, y gracias a estos se puede obtener información de la secuencia que el modelo está siguiendo. HMM crea modelos a partir de la comparación de enunciados (unidad mínima del habla) conocidos, y los desconocidos generados por el modelo. Con el uso de estadística, se ordenan los vectores que almacenan las probabilidades de transición entre estados de HMM. El entrenamiento práctico que sé le da consiste en introducir trozos de corta duración de cada audio, los transforma estados y encuentra la probabilidad de pasar de uno a otro. En el caso del dictado, analiza las sílabas e intenta predecir qué palabra se está tratando; y para el reconocimiento de voz, quién está hablando. En múltiples trabajos HMM está combinado con diferentes tipos de Redes Bayesianas, ya que son una herramienta muy útil a la hora de representar procesos estocásticos (procesos Gaussianos, de Markov, etc), y presentan incluso una mejora en cuanto a la cantidad de información que necesitan para operar con un alto porcentaje de precisión. La Figura 2.10 muestra un esquema muy sencillo de cómo funciona un modelo de Markov, con y siendo las salidas observables, x los estados ocultos que transmiten los resultados a los siguientes estados, a la probabilidad de transición y b la probabilidad de salida.

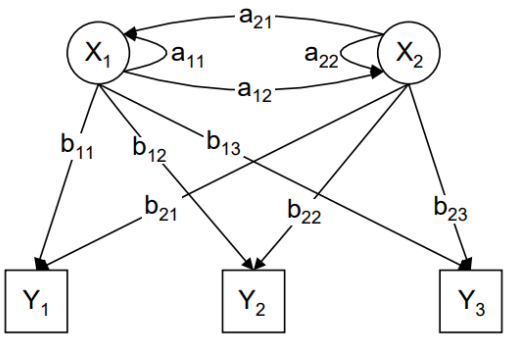


Figura 2.10: Esquema de Modelo de Markov. [TK09]

Las redes Bayesianas son grafos dirigidos donde las variables que se plasman en ella, tienen dependencia entre ellas. Los grafos están formados por nodos (variables aleatorias) y vértices (dependencia condicional). Para completar la relación entre cada nodo, se hace uso de inferencia Bayesiana: la probabilidad de las hipótesis se actualiza cada vez que se descubre más información de estas; la inferencia puede estar representada de dos formas:

- La primera es la evaluación de la relación entre valores dados de cada variable.
- La segunda busca encontrar la probabilidad de un suceso entre variables, dadas las probabilidades del resto de variables.

Las limitaciones computacionales que este método tiene ante grandes cantidades de información son considerables, y esto hace que se requieran métodos auxiliares para filtrar los datos, con el fin de disminuir su volumen (por ejemplo eliminación de variables).

2.3.2.3. Redes Neuronales

Esta tecnología consiste en formar una red de nodos que, conectados entre sí, transmiten información. Los enlaces que unen a cada nodo varían el valor que se comunica mediante pesos, y es por esto, que esa información tendrá más o menos peso, afectando a diferentes nodos. Progresivamente, esta red irá aprendiendo, y aumentarán los nodos para mejorar la precisión final. Dos variantes muy usadas en la última década dentro de el reconocimiento de voz son las Redes Neuronales Convolucionales (CNN) y las Redes Neuronales Recursivas (RNN). CNN se compone de múltiples capas, donde los datos son modificados y transmitidos por diferentes niveles. Durante la primera fase, las neuronas convolucionales extraen las características, y dan como salida el resultado de la Fórmula 2.5.

$$Y_j = g * (b_j + \sum_i (K_{ij} \otimes Y_i)) \tag{2.5}$$

La letra Y representa la salida de cada neurona, calculada a partir de la salida de las neuronas de la capa anterior y el núcleo convolucional (representado por K_{ij}). Finalmente esta cantidad es sumada a una influencia b_j y tratada por la función $g(\Delta)$. Tras este paso, las características del audio son filtradas, mediante procesos como *subsampling* o *max-pooling*, siendo este último más eficaz. *Max-pooling* consiste en detectar el valor más alto dentro del vector que se le pasa, y así reducir el espectro de valores que puedan repercutir en la red convolucional en los siguientes niveles. Por último, se realiza una clasificación final, con los datos completamente depurados, para enlazarlos con las etiquetas del entrenamiento respectivas, que en nuestro caso serían los identificadores de los interlocutores.

RNN componen una arquitectura en la que el mismo set de pesos es usado recursivamente dentro de un gráfico acíclico direccionado como se observa en la Figura 2.11. La entrada es tratada de forma que se visita cada nodo con pesos establecidos, dando tras pasar por varias capas, salida, que se usa de nuevo como entrada, cumpliendo la recursividad. Si el algoritmo debe tomar una decisión, se basará en la entrada y salida anteriores, haciendo a cada nodo dependiente de los próximos a este, a diferencia de las redes neuronales anteriores, que presentan independencia entre los datos que tratan.

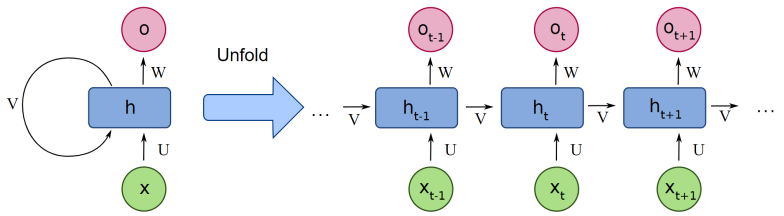


Figura 2.11: Esquema Básico de una RNN. [Con21]

Junto a RNN, se encuentra *Long Short-Term Memory* (LSTM). Puesto que RNN presenta una gran dificultad para mantener la dependecia entre grandes cantidades de información, LSTM solventa este problema. Su estructura se compone de varias celdas,

con una puerta de entrada, una de salida, y una para olvidar. En la entrada se distinguen los valores que van a ser tratados y los que serán descartados (función *Sigmoid*); en la salida ocurre lo mismo, descartando los valores que no son pertinentes, y dando diferentes pesos a los que pasarán a la siguiente celda. Por último, la puerta utilizada para olvidar desecha los valores que se han introducido, comparados con los del estado anterior, y que no son aceptados por la función *Sigmoid*. Como se indican los autores de pyannote.audio, la estructura del programa contiene dos capas bidireccionales de LSTM, con 128 unidades internas, sin *temporal pool*, y una capa final de clasificador.

2.4. Reconocimiento de Emociones en Audio

Para los sistemas de interacción tanto humano – computadora como humano – humano, cada vez se considera más importante poder reconocer el estado emocional con el que intervienen los interlocutores en los audios para poder mejorar los servicios que estos sistemas ofrecen, puesto que no sólo influye el qué se dice, sino también el cómo. Permitir que las máquinas puedan reconocer el estado en el que se encuentra el humano que está interactuando con ella le da la posibilidad de poder adaptarse a esto.

2.4.1. Historia

Los inicios del Reconocimiento de Emociones se remontan al siglo XX, con Charles Darwin afirmando que las emociones son heredadas por cada individuo. Aunque esto pueda ser completamente falso, dio paso a posteriores investigaciones que intentaban desmentir tal afirmación (como es el caso de la teoría de James-Lange [Can87]), que postulaba que las emociones son reacciones ante estímulos externos. Las investigaciones en el campo de la psicología continuaron con multitud de teorías que variaban las conclusiones, desde los años 50, hasta que alrededor de 1980, la teoría de de Red, (en la que se explica que las emociones ocurren cuando las personas relacionan hechos que ocurren con memorias y las sensaciones vividas entonces); y la teoría de emociones discreta, (donde se afirma que las emociones proceden de otras más básicas cimentadas en cada individuo) consiguieron crear un punto de vista diferente, que sería investigado más a fondo con el fin de aplicarlo en el reconocimiento facial y en el Sistema Nervioso Automático (ANS).

Los estudios y artículos relacionados con el segundo intentaban medir las emociones mediante multitud de experimentos como respuestas generadas presentando comida, o comentando recuerdos de la persona. Esto, sin embargo, no habría sido el punto final, sino más bien, un punto y coma en el campo. En la medicina, el análisis del pulso ([Sie18]), o la conductividad de la piel ([Kre10]) fueron dos pilares en los que se apoyó ANS para tener más puntos de vista, juntándose con Electroencefalogramas, y Resonancias Magnéticas, consiguiendo una mejoría en el avance del Reconocimiento de Emociones. Desde esa década de los 90, hasta hoy en día, la evolución más notable ha sido en el campo del Machine Learning, en el que se han introducido algoritmos como CNN, LSTM o DNN para intentar innovar y debatir sobre sus puntos fuertes a la hora de analizar las emociones.

En la última década, se ha propuesto analizar la información mediante un multi-modelo, divido en parte visual y parte de discurso, para continuar mejorando estas

investigaciones. Fue gracias a esto, y la combinación de los algoritmos anteriores que se vieron las diferencias al tratar la información de forma individual o en conjunto. Esto gira de nuevo en torno al uso y elección de los métodos que mayor acierto puedan ofrecer al usuario, consiguiendo así un avance constante, y nuevas combinaciones para facilitar el progreso.

2.4.2. Información Extraída: Acústica y Lingüística

Para el reconocimiento de emociones en audio, se utiliza información acústica y lingüística

- **Acústica:** Constituye la información recopilada sobre la señal de voz a analizar. Una gran cantidad de trabajos de investigación sobre este tema hacen uso exclusivamente de esta parte. Dentro de esta destacan los datos:

  - *Espectrales:* Aportan información respecto a las frecuencias de un ruido o sonido y de sus niveles de presión acústica. El análisis de estos componentes se conoce como análisis frecuencial para el que se utilizan las transformadas de Fourier dando como resultado el espectro de la señal analizada, como se observa en la Figura 2.12.

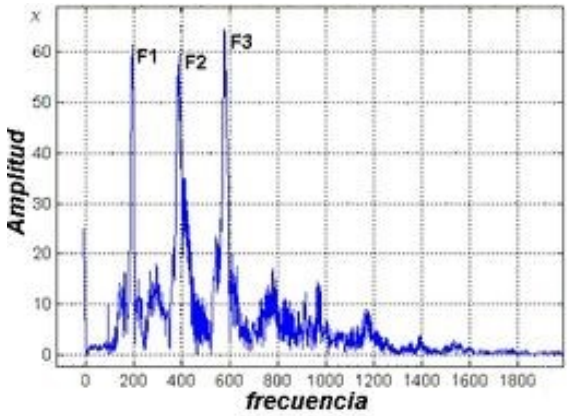


Figura 2.12: Espectro de frecuencias [spe18]

  - *De estilo*: Identifican si el interlocutor está susurrando, jadeando, gritando, etc.
  - *Prosódicos*: Analizan los elementos supra-segmentales, es decir, a los segmentos más largos que los fonemas, como la duración, el ritmo, el volumen, etc.

- **Lingüística:** Por otro lado, se encuentra la información lingüística, que es la extraída del texto transcrito del audio. Esta va muy ligada al contexto en el que se han usado y por lo tanto al corpus utilizado, es por ello por lo que estos no se suelen poder reutilizar para otras aplicaciones.

  Dentro del uso de estos datos, los dos planteamientos más utilizados son:

  - *Bolsa de palabras:* Se analiza la frecuencia con la que aparecen las palabras sin tener en cuenta el orden con el que lo hacen.

- *Palabras clave:* Este planteamiento se basa en la identificación de las palabras clave del texto, para ello, se suele usar el concepto de "palabra relevante", el cual consiste en detectar las palabras cuya frecuencia se dispara en determinadas secciones, en contraposición del resto, cuya frecuencia es muy similar a lo largo de todos los textos del corpus que se está usando.

2.4.3. Coeficientes Cepstrales en las Frecuencias de Mel

Los MFCCs son usados como una técnica de extracción de características del audio, formando una representación codificada del espectro de una onda sonora. Un conjunto de MFCCs forman colectivamente el llamado *Mel-Frequency Cepstrum* (MFC). El cepstrum de una señal es el resultado de calcular la transformada de Fourier inversa del espectro de la misma en escala logarítmica. La particularidad del MFC es que las bandas de frecuencia están espaciadas en la escala de Mel (ver Figura 2.13), que se aproxima mucho más a la respuesta del sistema auditivo humano que un cepstrum con bandas de frecuencia linealmente espaciadas. Es por ello que esta técnica es ampliamente usada en el procesamiento del discurso humano.

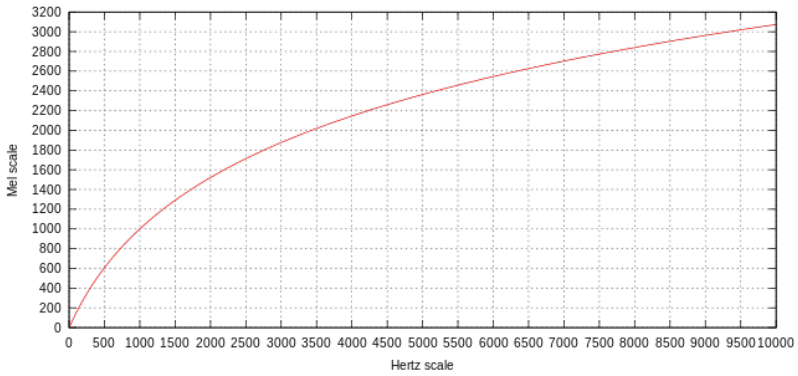


Figura 2.13: Escala de Mel, comparativa entre ganancia y frecuencia [VM20])

Los pasos para obtener los coeficientes de una señal acústica son:

1. **Pre-énfasis**: Consiste en pasar la señal por un filtro que enfatice las frecuencias altas, ya que estas poseen baja energía pero contienen más información que depende del hablante, equilibrando así el espectro de frecuencias.

2. ***Framing***: Es necesario dividir la señal en frames más pequeños,normalmente de 20-30 ms, con una superposición del 40-60 %.

3. ***Windowing***: En la mayoría de sistemas que trabajan con superposición de frames se debe suavizar la transición entre ellos. Por ello a cada frame le es aplicada una función de Hamming Windowing, que básicamente reduce la señal gradualmente a cero tanto al principio como al final.

4. **Transformada rápida de Fourier (FFT)**: Este proceso permite extraer la magnitud de frecuencia de cada frame, obteniendo la potencia espectral de la señal.

El objetivo de la transformada de Fourier es transformar una función matemática con dominio en el tiempo a otra con dominio en la frecuencia, donde (x_n) es la señal recibida del *Hamming Windowing* y (N) es el número de muestras dentro del frame y es definida por la Fórmula 2.6.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi}{N} kn} \qquad k = 0, ..., N-1 \tag{2.6}$$

5. **Procesamiento de Mel**: Es el paso donde se aplica la escala de frecuencia de Mel, que se basa en la percepción auditiva humana. El ser humano es más discriminativo en las frecuencias bajas y menos en las más altas, por lo que se aplican dos tipos de filtros diferentes. Por debajo de la frecuencia de 1000 Hz los filtros están espaciados linealmente, y a parir de esa frecuencia los filtros son espaciados logarítmicamente. Por lo tanto existen más filtros en las frecuencias bajas y menos en las regiones altas. Las frecuencias (f) son transformadas a la escala de Mel (m) con la siguiente ecuación (2.7)

$$m = 2595 \times \log_{10}(1 + \frac{f}{700}) \tag{2.7}$$

6. **Logaritmo**: El objetivo de esta fase es realzar las diferencias en las variaciones de la señal para que las estimaciones sean menos sensibles a pequeños cambios y al ruido. Para ello se calcula el logaritmo de las frecuencias en escala de Mel elevadas al cuadrado.

7. **Transformada de coseno discreta (DCT)**: Esta transformada nos permite volver del dominio de frecuencias al de tiempo, haciendo el proceso inverso a la FFT, obteniendo como resultado un vector de coeficientes Mfcc, también conocido como vector acústico. Por lo tanto podemos pensar que tras este último paso los valores que se obtienen serían los mismos que los de la señal original, pero escuchado por un humano.

2.4.4. Aplicaciones en Sectores

Desde hace bastantes años, diferentes aplicaciones de muchos sectores distintos han ido haciendo uso de estas técnicas para mejorar sus servicios [yCAR10].

Por ejemplo, en el ámbito médico, en 1999, el Dr. Gerardo M. González de la Universidad de California desarrolló un IVR (*Sistema de Respuesta Interactiva por Voz*) [Gon99] que tenía como finalidad detectar signos de depresión en la voz de los pacientes que llamaban, si estos eran preocupantes, se avisaba a un experto para que le atendiera. Este fue preparado para servir tanto en inglés como en español.

Más adelante han ido surgiendo nuevas con otros objetivos, como detectar el nivel de angustia o dolor de los pacientes para dar preferencia a sus llamadas o poder diferenciar si sufren de ansiedad o de depresión.

Otro marco en el cuál su uso resulta muy interesante, es en el de los *call centers*, ya que facilita mucho su control de calidad, tanto para comprobar si las llamadas atendidas

por máquinas resultan satisfactorias para los clientes, como para supervisar si los humanos que se encargan de otras tienen comportamientos inadecuados con estos.

En el campo de la música también aparece el MER (*Music Emotion Recognition*), que, aunque tenga otras aplicaciones, se utiliza principalmente para automatizar las formas de ordenar las canciones por estilo musical y en base a esto, para la recomendación de música.

Capítulo 3

Procesamiento del Lenguaje Natural

PLN es el sub-campo de la AI y de la lingüística que estudia la relación entre computadora y lenguaje humano con el objetivo de que la máquina procese y analice el contenido de documentos para hacer posible la comunicación entre ambas partes. Este capítulo se divide en la Sección 3.1, con el uso y evolución de PLN, hasta la actualidad; Sección 3.2, con preprocesamiento y extracción de características, y las diferentes técnicas para conseguir estas tareas; Sección 3.3, con una definición de los tipos de análisis existentes; Sección 3.4, donde se detallan las diferentes herramientas usadas en PLN y finalmente Sección 3.5, su uso dentro del Ámbito Policial y Judicial Español.

3.1. Historia del PLN

El PLN surge con la finalidad de conseguir una mayor fluidez en las comunicaciones entre el hombre y la máquina que en las realizadas a través de los lenguajes formales; lo hace con la aparición de las primeras computadoras a finales de los años 40 mediante la búsqueda de la ***"traducción automática"***, aunque hay quien considera que estos avances son independientes de la disciplina del PLN y que esta no surge hasta las décadas de los 50 o 60.

Durante las primeras décadas de historia del PLN la premisa que perseguía estaba bien representada por el experimento de la habitación china de John Searle [Col20], donde la máquina emulaba comprensión de lenguaje natural aplicando una colección de reglas que un humano previamente le había preparado.

Es en 1950 cuando **Alan Turing** publica un artículo en el que especifica una prueba (prueba de Turing) para las computadoras, en la que se trata de comprobar si son capaces de hacer que un interrogador humano no logre distinguir, de forma confiable, si se trata de una persona o de una máquina.

Una prueba que surge más tarde y cuya finalidad es la contraria, es decir, busca comprobar que se trata de un humano y no de una máquina, son los CAPTCHA.

Unos años más tarde, en 1954, tiene lugar el ***experimento de Georgetown*** (ver Figura 3.1), desarrollado por la Universidad de Georgetown e IBM, en el cual se realizó la

traducción de más de 60 oraciones en ruso al inglés de forma completamente automática (ver Figura 3.2). además los autores confiaban en que, a lo largo del siguiente lustro, el problema de la traducción automática quedara totalmente resuelto [Hut04]).

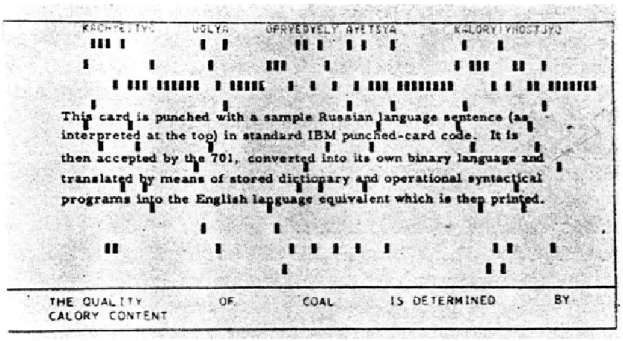


Figura 3.1: Frase a traducir escrita a máquina en el experimento de Georgetown. [Hut04]

Figura 3.2: Momento en el que se redactaron las frases a traducir en el experimento de Georgetown. [Hut04]

Sin embargo, el progreso de esta área estaba tardando más de lo previsto, lo cuál llevó al gobierno de los Estados Unidos a crear en 1964 el comité *Automatic Language Processing Advisory Committee* (ALPAC) para analizar el avance de la lingüística computacional y en particular, el de la traducción automática.

Publicaron en 1966, un informe escéptico de la investigación realizada en estas áreas y esto supuso que se redujera de forma drástica la inversión en ellas hasta la década de 1980 cuando comenzó el desarrollo de sistemas estadísticos de traducción automática.

Es en el año 1970, cuando William A. Woods introdujo las Redes de Transición Aumentadas (ATN), que se tratan de tipos de estructuras teóricas de grafos que se convirtieron en uno de los métodos más usados para escribir gramáticas de análisis de lenguajes naturales. Durante esta época también se programaron muchos asistentes virtuales.

En los años 80, se produjo un gran avance en el PLN con la aparición de nuevos algoritmos automáticos para el procesamiento del lenguaje, ya que, hasta entonces, casi todos los sistemas PLN se basaban en complejos conjuntos manuales de reglas.

Desde los años 90, las investigaciones se fueron centrando cada vez más en los modelos

estadísticos, que toman decisiones probabilísticas basadas en lo que se ha denominado como **"bolsa de palabras"** (*Words Bag*), que consiste en tratar todas las palabras del documento como términos índice y asignarles un peso en función del papel que tengan en el texto, por lo general se basa en la frecuencia con la que aparecen.

Estos modelos son muy utilizados en los sistemas de recuperación de información debido a su facilidad para implementarlos y a su gran eficacia.

Otro ejemplo de estos modelos estadísticos son los modelos de lenguaje caché en los que se basan muchos sistemas de reconocimiento de voz.

Pronto, los modelos estadísticos comenzaron a dar buenos resultados, sobre todo en el campo de la traducción automática, en gran parte debido al trabajo realizado por **IBM Research** y a la gran cantidad de corpus *(conjunto de fragmentos de textos naturales guardados en formato electrónico)* multilingües procedentes de organismos oficiales como en el caso del Parlamento canadiense, donde todas las intervenciones de sus representantes pueden ser realizadas tanto en inglés como en francés y al terminar, son traducidas al otro idioma, o como en el caso del proyecto europeo **NERC*(Network of European Reference Corpora)***, que surge con el objetivo de establecer unas normas básicas para la construcción de corpus semejantes en distintas lenguas europeas.

Últimamente las investigaciones PLN se han ido centrando cada vez más en **algoritmos semi supervisados**, que utilizan parte de datos de los cuales se han establecido las respuestas esperadas de forma manual y parte que no, y en **algoritmos no supervisados** que son aquellos que no utilizan ningún dato preparado de forma manual.

Los resultados no son tan precisos en estos tipos de algoritmos como en los **supervisados** cuando se utiliza la misma entrada de datos, sin embargo, tienen la ventaja de que al poderse utilizar datos que no han sido preparados manualmente, es mucho mayor la cantidad que hay disponible de estos.

3.2. Preprocesamiento y extracción de características

En PLN para hacer más fácil el análisis de los textos y mejorar el rendimiento de los modelos, se aplican distintos procesos, algo a lo que en conjunto se conoce como ***"preprocesamiento de los datos"*** [dC]. Vamos a ver brevemente en qué consisten algunos de los más comunes y útiles:

- **Eliminación de signos de puntuación**
- **Lowercasing:** Transforma todos los caracteres del texto de los documentos a su versión en minúscula.
- **Tokenización:** En este proceso se descompone el texto en los componentes más básicos posibles, descartando los espacios en blanco y los saltos de línea. Esto puede resultar muy útil, entre otras cosas, si por ejemplo queremos analizar la frecuencia con la que aparecen ciertas palabras.
- **Lematización:** Es el proceso en el cual se trata de encontrar la "forma de diccionario" de la palabra. Es decir, pretende llevar las formas flexivas a una forma

básica común.

- **Stemming o derivación:** Proceso encargado de encontrar la raíz de las palabras.
- **Etiquetado POS (Part of Speech Tagging):** Consiste en asignar etiquetas a cada palabra del texto en función del tipo de palabra que sea: sustantivo, adjetivo, verbo, adverbio, etc.
- **Reconocimiento de entidades nombradas:** Proceso que identifica y etiqueta palabras que se refieren a entidades reales, como personas, lugares, organizaciones, etc.
- **Filtrado de stop words**: El objetivo es eliminar del diccionario palabras típicas que podrían estar presentes en cualquier tipo de documento. Este tipo de palabras no aportan a la capacidad de discriminación del modelo y se pueden considerar como "ruido" dentro del conjunto de entrenamiento.
- **Otros**: En casos como el preproceso de tweets sería también interesante aplicar otros filtros para eliminar URLs, tags HTML y caracteres especiales, además de tratar con otro tipo de representación como son los emojis, transcribiéndolos a texto o bien eliminándolos.

Una vez realizado el preprocesamiento oportuno para nuestro conjunto de datos existen diferentes técnicas que pueden ser aplicadas para extraer las características del cuerpo de documentos dependiendo la finalidad de nuestro proyecto.

Una de las técnicas más habituales es la llamada ***bolsa de palabras***, que es una ponderación para representar el contenido de un cuerpo de documentos. La frecuencia de términos consiste en contar el número de veces que una palabra o n-grama aparece en cada uno de los documentos del conjunto, formando una matriz donde las filas son los documentos de nuestro corpus y las columnas o características son las diferentes palabras encontradas en dicho corpus o de un diccionario introducido.

Un ***n-grama*** es una subsecuencia de n elementos del texto a procesar. En nuestra tarea de extraer la frecuencia de los términos de nuestro conjunto podemos decidir qué tipos de n-gramas usar. Si dividimos cada palabra por separado obtenemos una serie de *monogramas* pero también podemos medir la frecuencia de pares (*bigramas*) o ternas (*trigramas*) de palabras, etc.

Aunque la matriz generada por la bolsa de palabras considera la frecuencia de las términos que aparecen en un documento, no se tiene en cuenta cuán común o inusual es dicha palabra dentro del corpus. Esta es precisamente la función de los transformadores *Term Frecuency - Inverse Document Frecuency* (TF-IDF) que, además de incluir la frecuencia de las palabras en los documentos, también considera la frecuencia de documento inversa, la cual sirve como factor de distinción. El valor de *Inverse Document Frecuency* (IDF) es calculado de la siguiente manera:

$$IDF = \log\left(\frac{N}{1 + |\{d \in D : t \in d\}|}\right) \tag{3.1}$$

Donde N representa el número total de documentos y la expresión $\{d \in D : t \in d\}$ es el número de documentos donde t aparece.

La métrica de IDF nos indica si el término es común o no en el conjunto de documentos, aportando una mejor visión de la relevancia de la palabra si se hace el producto con Frecuencia de Término (TF).

3.3. Tipos de Análisis

El proceso de compilación de un programa está compuesto por distintas fases (ver Figura 3.3) entre las cuales podemos encontrar 3 tipos de análisis del texto: léxico, sintáctico y semántico [Dom19].

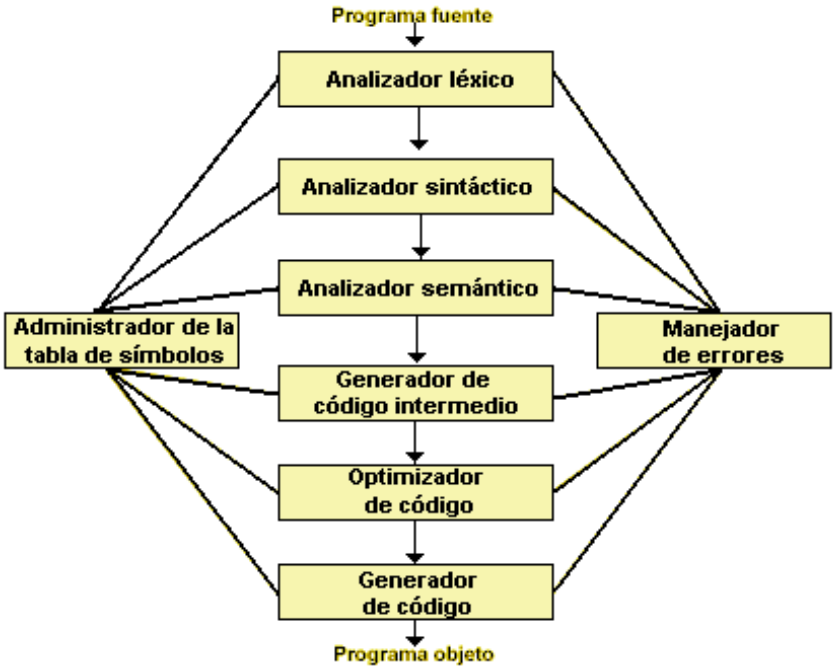


Figura 3.3: Fases de un compilador.

- **Léxico:** Esta primera etapa es la encargada de leer cada símbolo del texto que forma el código fuente y de generar los componentes léxicos, también conocidos como tokens, que serán utilizados como entrada del analizador sintáctico. En esta fase se revisa que todos los elementos del lenguaje sean correctos y, además, se elimina la información del fichero que se considera irrelevante para el análisis.
- **Sintáctico:** En esta fase, se comprueba que los tokens de entrada que le ha enviado el analizador léxico se ajustan a las reglas gramaticales establecidas por el lenguaje de programación utilizado y genera el árbol sintáctico que se usará en el analizador semántico como fichero de entrada.
- **Semántico:** En este último tipo de análisis, se revisa que el árbol resultado del analizador sintáctico tiene sentido semántico y deja preparada la generación de código.

3.4. Herramientas y Servicios PLN en Español

Aunque en el mundo existen varios miles de lenguas habladas y todas podrían ser tratadas virtualmente por los ordenadores, por practicidad, los programas se centran en el correcto funcionamiento de las más utilizadas, esto se puede comprobar en los asistentes de voz de las grandes compañías, como en el caso de *Siri* de Apple que se encuentra disponible en 21 idiomas o el *Google Assistant* que contiene 30 diferentes. La aplicación que puede usar más lenguas es *Google Translate* con más de un centenar de ellas. De hecho, a pesar de que estas aplicaciones trabajen con decenas de lenguas, la realidad es que la mayor parte de los avances y de la documentación acerca de este tema se encuentra desarrollada, sobre todo, en inglés.

Sin embargo, debido a la mejora de las computadoras y algoritmos, y a la mayor disponibilidad de big data, cada vez es más habitual encontrar herramientas y servicios con versión multilingüe e incluso modelos específicos para cada idioma.

3.4.1. Herramientas PLN para Python en Español

Actualmente disponemos de una gran variedad de herramientas PLN, tanto en distintos idiomas como en diferentes lenguajes de programación, es por ello por lo que en este apartado vamos a repasar algunas de código abierto que se encuentran disponibles en español y que se puedan utilizar en programas de Python, ya sea en forma de librería o a través de una API o un plugin [LT19]. Algunas de las herramientas de código abierto disponibles para Python y en español son:

- **Freeling:** Se trata de un conjunto de herramientas de análisis de lenguaje de código abierto (análisis morfológico, tokenización, detección de entidades con nombre, etiquetado POS y de roles semánticos, etc.). Este proyecto, liderado por Lluís Padró, profesor de la Universidad Politécnica de Cataluña, surgió con el objetivo de poner a disposición de todo el mundo los resultados obtenidos por el grupo de investigación del PLN de dicha universidad. Está disponible para una gran cantidad de idiomas entre los que se encuentra el español, y aunque se trata de una biblioteca en C++, se puede adaptar a Python mediante el uso de APIs, cuya instalación, a pesar de poder resultar un poco larga, permite el acceso a todas las funcionalidades de estas herramientas.

- **NLTK:** El NLTK **(kit de herramientas de lenguaje natural)** es un conjunto de bibliotecas y programas PLN que tiene como objetivo desarrollar programas Python para análisis de texto. Cuenta con distintas interfaces para poder trabajar con los más de 50 corpus y recursos léxicos que proporciona, e incluso tiene un foro. Está diseñado para utilizarse en inglés, pero se puede usar en español o bien entrenándolo con archivos que incorpora el propio NLTK o bien usando un etiquetador externo como el de Stanford. Está desarrollado en Python por lo que no se necesita ninguna API externa para poder utilizarlo en este lenguaje.

- **Pattern:** Pattern es un módulo de Python que cuenta con herramientas para análisis de redes, minería de datos o PLN entre otras. Está publicada en GitHub y aunque está preparado para su uso en inglés también permite el etiquetado POS en español, aunque la versión en inglés tiene una interfaz extra para *WordNet*, una base de datos léxica de este idioma.

- **SPACY:** Se trata de una biblioteca programada en Python cuyo objetivo es el procesamiento avanzado de lenguaje natural. Proporciona pipelines pre entrenados y se encuentra disponible para más de 60 idiomas. Permite realizar etiquetados, tokenización, análisis o reconocimiento de entidades con nombre entre otras funcionalidades. Para poder utilizarlo en español basta con añadir el modelo para este idioma desde un programa Python.

- **Stanza:** Stanza, antes conocido como Stanford NLP, es una biblioteca desarrollada en Java que ofrece diversas herramientas para tareas del procesamiento de lenguaje natural, así como acceso al software *CoreNLP*, también de Stanford, el cual se trata de otra biblioteca de PLN basada en *pipelines* (ver Figura 3.4). Aunque se encuentra disponible en español, no todas sus funciones lo están. Al estar desarrollado en Java, se necesita hacer uso de una API, que requiere de un servidor Java y de definir variables de entorno, para poder utilizarlo en programas Python.

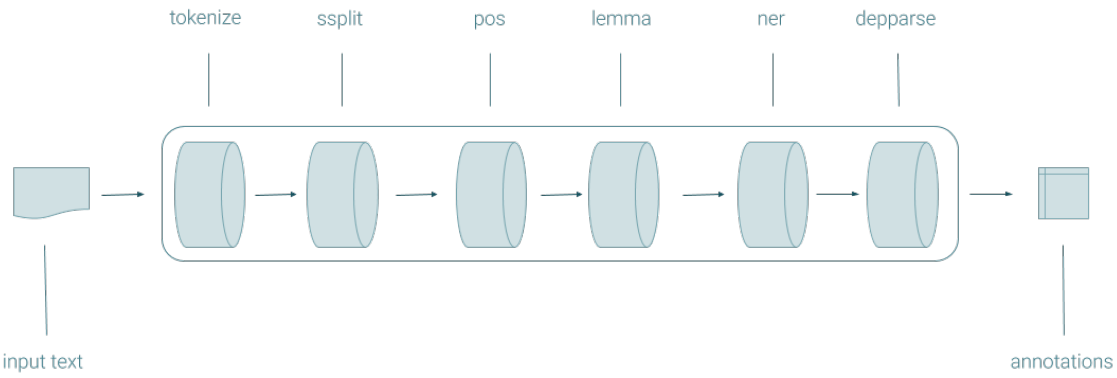


Figura 3.4: Estructura del pipeline utilizado por CoreNLP

La Tabla 3.1 resume algunos de los aspectos más destacables de las herramientas vistas.

Tabla 3.1: Aspectos de cada herramienta

Herramienta	Tokenización	Lematización	POS Tagging	Español	Lenguaje
Freeling	Sí	Sí	Eagles	Sí	C++ (API Python)
NLTK	Sí	No	Eagles	Sí	Python
Pattern	Sí	Sí	Penn TreeBank	Sí	Python
Spacy	Sí	Sí	Penn TreeBank	Sí	Python
Stanza	Sí	Sí	Eagles	No del todo	Java (API Python)

3.4.2. Servicios PLN en Español

El **RAH (*servicio de reconocimiento automático del habla*)** consiste en el proceso por el cual se logra transcribir a texto el habla humana mediante la interpretación de las ondas sonoras. Estas primeras herramientas son servicios de pago, aunque todas ellas tienen versiones gratuitas para poder probarlas.

- **Amazon Transcribe:** Garantiza transcripciones de alta calidad de entradas de audio de distintas fuentes como micrófonos, vídeos y audios. Sus principales funcionalidades son:
  - *Signos de puntuación automáticos y normalización del texto producido:* Añade signos de puntuación de forma automática y transcribe los números en dígitos.
  - *Transcripción de audio en tiempo real:* Se puede utilizar el servicio tanto con entradas de sonido ya grabadas, como directamente con un micrófono en tiempo real mientras el programa va generando y mostrando el texto por escrito.
  - *Marcas de tiempo:* El sistema añade marcas de tiempo a cada palabra que se pueden utilizar o bien para facilitar la búsqueda de palabras o frases concretas o bien para subtitular un vídeo.
  - *Añadir vocabulario:* Se pueden añadir nuevas palabras o frases para que las transcripciones resulten más precisas, por ejemplo, nombres de personas, productos, términos técnicos, etc.
  - *Filtrado de vocabulario:* También se pueden incluir palabras que no se quiere que se transcriban.
  - *Reconocimiento de interlocutores:* Identifica que interlocutor interviene en cada momento y lo deja indicado.
  - *Identificación automática del idioma:* Identifica de forma automática el idioma del audio, por lo que, si empleamos audios de diferentes idiomas, no será necesario especificarlo uno por uno.
  - *Amazon Transcribe Medical:* Servicio específico para aplicaciones relacionadas con el ámbito médico y de la salud.
- **API Speech de Google Cloud:** Hace uso de los sistemas de algoritmos más avanzados de Google para reconocer de forma automática la voz. Sus principales funcionalidades son:
  - *Vocabulario internacional:* La API es capaz de reconocer más de 125 idiomas con variedades lingüísticas.
  - *Transcripción de audio en tiempo real.*
  - *Añadir vocabulario y mejorar la precisión de la transcripción.*
  - *Reconocimiento multicanal:* Puede distinguir entre los distintos canales que intervengan en el audio.

- *Tratamiento del ruido:* Puede procesar archivos de audio en entornos ruidosos sin necesidad de tratarlos previamente.
- *Modelos para dominios específicos:* Se puede utilizar una selección de modelos optimizados entrenados para distintas finalidades, como recibir comandos de voz o transcribir vídeos y llamadas de teléfono.
- *Filtrado:* Se pueden filtrar las partes que incluyan palabras seleccionadas como inapropiadas y que no se quiere que aparezcan en los resultados finales.
- *Identificación automática del idioma (fase beta).*
- *Signos de puntuación automáticos (fase beta).*
- *Reconocimiento de interlocutores (fase beta).*

- **Microsoft Speech Service:** Los principales servicios son:
  - *Speech to text:* Transcripción de audio a texto tanto de archivos como en tiempo real. Se puede añadir vocabulario y se adapta a patrones, acentos y ruidos de fondo.
  - *Text to speech:* Permite pasar los textos a audios prácticamente en tiempo real. Se puede adaptar el modelo de voz de la API como se quiera o bien crear uno propio cargando datos de entrenamiento.
  - *Traducción:* Traducción en tiempo real para los idiomas admitidos, esta puede ser por voz o escrita.
  - *Escenarios de empresas basadas en servicios de voz:* Servicios optimizados para centros de llamadas.
- **IBM Watson Speech to Text:** Proporciona una API que sirve para añadir funcionalidades de transcripción de voz a las aplicaciones. Sus principales funcionalidades son:
  - *Reconocimiento de voz en tiempo real:* Transcripción automática del audio en tiempo real de 7 idiomas.
  - *Motor de voz altamente preciso:* Permite adaptar el modelo para mejorar la precisión añadiendo vocabulario. Identifica los distintos interlocutores que intervengan y también permite detectar palabras clave previamente especificadas.
  - *Soporte para varios casos de uso:* Aparte de poder realizar transcripciones tanto de archivos como en tiempo real, también permite el análisis de hasta 1000 segundos de audios de call centers para su análisis.
- **CMU Sphinx:** Esta herramienta es de código abierto y se encuentra publicada en GitHub, por lo que su uso es totalmente gratuito. Se trata de un conjunto de sistemas PLN, desarrollado en la Universidad de Carnegie Mellon, que está formado por varios programas de reconocimiento de voz y un entrenador de modelo acústico. El proyecto recibía constantes actualizaciones hasta que en los últimos años esto

se ha ido frenando debido a que el equipo de desarrollo se ha centrado en otros proyectos, también relacionados con el PLN, entre ellos, el último que han hecho público ha consistido en desarrollar un kit de herramientas de código abierto escrito en C++ para el reconocimiento de voz y el procesamiento de señales (**Kaldi**) para Android.

Tabla 3.2: Servicios PLN en español

Servicio	**Funcionalidad**	**Precio de uso**	**Servicio gratuito**
Amazon Transcribe	Signos de puntuación automáticos y normalización del texto producido	0'00040$ por segundo transcrito	60 minutos mensuales el primer año
	Transcripción de audio en tiempo real		
	Marcas de tiempo		
	Añadir vocabulario		
	Filtrado de vocabulario		
	Reconocimiento de interlocutores		
	Identificación automática del idioma		
	Amazon Transcribe Medical		
API Speech de Google Cloud	Vocabulario internacional	0'006$ por cada 15 segundos transcritos (0'009$ en modelos mejorados). También da la opción de permitir que guarden datos de los audios utilizados en la API para mejorar el funcionamiento de esta, rebajando así el coste del uso a: 0'004$ por cada 15 segundos transcritos (0'006$ en modelos mejorados)	60 minutos gratis cada mes
	Transcripción de audio en tiempo real		
	Añadir vocabulario y mejorar la precisión de la transcripción		
	Reconocimiento multicanal		
	Tratamiento del ruido		
	Modelos para dominios específico		
	Filtrado		
	Identificación automática del idioma (fase beta)		
	Signos de puntuación automáticos (fase beta)		
	Reconocimiento de interlocutores (fase beta)		
Microsoft Speech Service	Speech to text	"Speech to text": 0'844€/hora "Text to speech": 3'374€/millón de caracteres "Traducción": 2'11€/hora de audio	"Speech to text": 5 horas al mes "Text to speech": 5 millones de caracteres al mes "Traducción": 5 horas de audio gratis al mes
	Text to speech		
	Traducción		
	Escenarios de empresas basadas en servicios de voz		
IBM Watson Speech to Text	Reconocimiento de voz en tiempo real	0'02$ por minuto	500 minutos gratis cada mes
	Motor de voz altamente preciso		
	Soporte para varios casos de uso		
CMU Sphinx	Conjunto de sistemas PLN: Conjunto de programas de reconocimiento de voz y un entrenador de modelo acústico	Completamente gratuito	Completamente gratuito

3.5. Transcripción de Registros de Audio en el Ámbito Policial y Judicial Español

Uno de los sectores donde el uso del PLN ha ido avanzando considerablemente es en el judicial, ya que, la transcripción de los registros de audio en este ámbito resulta de gran ayuda para buscar intervenciones concretas o bien para permitir su legibilidad cuando no se encuentra disponible el soporte de audio. A pesar de esto, los estándares seguidos por los profesionales que se encargan de realizar estas transcripciones son muy distintos unos de otros [EGR19].

Para mejorar la eficiencia de estas, se cree que lo mejor sería asentar unos patrones comunes, por ello, 4 expertas en la materia, con la ayuda de un grupo de 115 transcriptores y de las Fuerzas y Cuerpos de Seguridad del Estado (FCSE) junto a los intérpretes y traductores del Ministerio del Interior que les auxilian, prepararon y propusieron un conjunto de estándares que veremos más adelante, en el que, además, analizaron los beneficios que podría suponer seguirlo. Las transcripciones en el ámbito judicial tienen una finalidad principalmente práctica y no tanto de investigación como sucede en otros casos.

3.5.1. Ley de Enjuiciamiento Criminal, 2015

En España, hasta que tuvo lugar la *Ley Orgánica 13/2015, de 5 de octubre, de modificación de la Ley de Enjuiciamiento Criminal para el fortalecimiento de las garantías procesales y la regulación de las medidas de investigación tecnológica*, la información recogida por la legislación sobre las transcripciones durante los casos no era muy específica, por ello, estas quedaban relegadas a un segundo plano en favor de las grabaciones que siempre eran tratadas como la prueba principal del caso [dE15].

Únicamente se utilizaban cuando el audio no estuviera disponible durante la vista oral y las transcripciones hubieran sido certificadas por el *Letrado de la Administración de Justicia*. Tras esta Ley, se recogió mucha más información acerca de cómo tratar las transcripciones durante los juicios, como por ejemplo la forma en la que se deben presentar y en qué situaciones puede sustituir la transcripción de una grabación a su escucha.

En muchos casos, el uso de las transcripciones facilita el trabajo de los jueces debido a la gran cantidad de documentación que conlleva cada juicio además de los problemas que suponen a veces las audiciones de las grabaciones (problemas para distinguir a los interlocutores, interrupciones entre ellos, ruidos de fondo, mala calidad del audio, etc.). Es por esto por lo que cada vez es más recurrente el uso de estas transcripciones.

3.5.2. Problemas Comunes en este Tipo de Transcripciones

Las transcripciones en el ámbito judicial y policial suponen ciertas dificultades para los profesionales que se encargan de hacerlas, tales como la falta de tiempo para realizarlas, la responsabilidad que supone no cometer errores (ya que, al tratarse de archivos delicados, un fallo como asignar una intervención a un interlocutor al cuál no le corresponde por no haberle podido identificar correctamente puede acarrear graves consecuencias), la falta

de una metodología común que puedan seguir y en algunos casos, falta de formación específica acerca de este tipo de transcripciones. Obviamente, cuando el audio es de difícil comprensión, es cuando las transcripciones son más necesarias, pero si estas dificultades vistas a la hora de ejecutarlas impiden que esta sea de buena calidad, se puede llegar a poner en duda su utilidad e incluso llegando al caso de invalidarlas. Es por esto, que se considera que establecer unos estándares comunes acerca de cómo realizar las transcripciones sería realmente beneficioso para la mejora de su eficiencia y del tiempo requerido para hacerlas.

3.5.3. Propuesta de Metodología para las Transcripciones Judiciales y Policiales en Español

Las 4 expertas que propusieron estas pautas trabajaron en la impartición de cursos de formación en transcripción y traducción forense, que tuvieron lugar en 2011 en el *Instituto de Ciencias Forenses y de la Seguridad de la Universidad Autónoma de Madrid*, en 2013, en el *Máster de Traducción e Interpretación de la Universidad de Alcalá de Henares* y otros 3 en los años 2015, 2017 y 2018 solicitados por el Ministerio.

Durante estos cursos, tras ver las ganas de acudir a estos cursos de los profesionales encargados de realizar estos trabajos de transcripción, se dieron cuenta de la falta formación específica que reciben en muchos casos y de los pocos avances que había en este tipo de procedimientos, por lo que se pusieron a ver cómo podían intentar mejorar esto, permitiéndoles ahorrar tiempo y esfuerzo y mejorando la calidad de los resultados.

Por ello, junto a la información recibida por parte de los profesionales encargados de realizar transcripciones vinculados a las FCSE, desarrollaron 20 pautas a seguir en este procedimiento y las compartieron con 115 expertos en transcripción forense para que las utilizaran y les comentaran si habían sido de utilidad y además, en los cursos formativos pedían a los profesionales que acudían que desarrollaran transcripciones utilizando sus métodos propios y después que siguieren las pautas propuestas y así comprobar si de verdad eran de ayuda. Tras la colaboración recibida y de ver el funcionamiento de esta metodología, comprobaron que los resultados estaban siendo muy positivos y siguieron retocando los puntos de esta metodología, ya que algunos daban lugar a información irrelevante o a partes confusas.

Finalmente, tras todos estos retoques, el conjunto de patrones propuesto quedó en 11 aspectos que tratar y 15 pautas a seguir. Estos aspectos son los siguientes:

1. Determinar cuándo interviene cada interlocutor.

2. Indicar solapamientos entre interlocutores, para aclarar que en esos casos se pueden producir atribuciones de intervenciones de forma incorrecta.

3. En caso de que se considere relevante, señalar si la intervención es susurrada o gritada.

4. Indicar interrogaciones y exclamaciones mediante el uso de los signos ortográficos establecidos por la normativa de la lengua española.

5. Resaltar los alargamientos de vocales y consonantes.
6. Destacar las auto correcciones y auto interrupciones.
7. Reflejar en la transcripción las pausas usando los signos ortográficos correspondientes, pero cuando estas sean de más de 30 segundos, utilizar otro símbolo diferente.
8. Señalar el uso de formas incompletas de palabras.
9. Indicar el uso de palabras en otra lengua.
10. Destacar los tramos del audio, que por problemas del mismo (ruido de fondo, audio de mala calidad, etc.), no se han podido transcribir.
11. Permitir al transcriptor poner anotaciones aclaratorias, sobre todo en los casos en los que intervengan otras culturas y/o lenguas.

Las pautas a seguir se presentan en la Tabla 3.3.

Tabla 3.3: Pautas a seguir

Pauta	**Código**	**Significado**	**Ejemplo**
1	A:	Indica el locutor que está interviniendo.	Paula: Hola
2	PALABRA	Elevación de tono.	DEJA DE GRITAR
3	Aa/ss	Alargamiento de vocal o consonante.	A veeer
4	¿?	Frases interrogativas.	¿Estás seguro?
5	¡!	Frases exclamativas.	¡Claro!
6	Pal-	Interrupciones o correcciones realizadas a sí mismo por el interlocutor.	Deja ya de fastid-, me estás poniendo de los nervios.
7	[ ]	Solapamiento.	A: ¿Qué tal está el tiempo [por allí? B: Pues últim]amente genial, soleado todo el día.
8	(( ))	Fragmento que no se ha podido transcribir.	Estoy casi (( )), ¿hola? ¿me escuchas? Que estoy casi sin cobertura.
9	((duda))	Palabra dudosa.	He comido ((melón)), pero sigo teniendo hambre.
10	Pa'	Uso de forma incompleta de palabra.	Ese es su cuña'o.
11	//	Pausas de más de 30".	Nunca había estado en su piso. // Sólo nos habíamos cruzado alguna vez en la empresa.
12	Cursiva.	Uso de otra lengua.	Ahí hay un *Burger King*, por si os apetece cenar ya.
13	*Voz susurrada*	Parte del audio susurrada.	Vale, recojo ahora la mesa *aunque te tocaba a ti.*
14	{}	Explicación breve, onomatopeya.	Estamos en la mesa del fondo {silbido largo} ¿lo has escuchado?
15	Anotaciones a pie de página.	Anotaciones del transcriptor.	N.T. Expresión usada por los jóvenes para referirse a las personas mayores.

Capítulo 4

Estado del arte

En la práctica existen múltiples aproximaciones que se centran en los puntos anteriores y posibles herramientas necesarias a la hora de analizar el audio, siendo las principales del trabajo las siguientes: reconocimiento de emociones en audio, análisis de sentimiento en texto con PLN, segmentación de interlocutores y, finalmente, el planteamiento de la implementación de estos diferentes componentes en un sistema híbrido. En la Sección 4.1 se explica qué es el Reconocimiento de Emociones, junto a los datasets más utilizados y varios artículos que han servido para entender y realizar los experimentos. De forma muy similar en la Sección 4.2 se detalla la evolución de esta tecnología y los trabajos que fundamentan este progreso, a la par que las técnicas más utilizadas. Como tercera parte, la Sección 4.3, tras una breve definición del concepto, se mencionan los algorirmos más punteros (en su mayoría variaciones de Redes Neuronales) y los trabajos más recientes que hacen uso de ellos. Para finalizar, la Sección 4.4 diferencia el origen de la información en estos campos de estudio, pudiendo ser audio o texto. Esto permite tener dos puntos de vista diferentes, para poder analizar a posteriori los resultados, pero presenta una gran dificultad a la hora de juntar, como se explicará más adelante.

4.1. Reconocimiento de Emociones

El campo de reconocimiento de emociones tiene un amplio recorrido pero el uso de técnicas de Machine Learning y Deep Learning para el procesamiento de audio con este fin apenas tiene un par de décadas de evolución. Se han llevado a cabo gran variedad de estudios para crear conjuntos de audio diseñados para el reconocimiento de emociones en diferentes idiomas. A pesar de que el español es una de las lenguas más habladas en todo el mundo, resultó bastante complicada la labor de encontrar un buen conjunto de acceso abierto en nuestra lengua, ya que a mayoría de estos datasets se presentan en lengua inglesa o alemana. Algunas de las bases de datos sobre las que hemos encontrado mayor cantidad de trabajo previo han sido RECOLA [RSSL13], IEMOCAP [BBL+08], que también incluye otros medios como captura facial para analizar las emociones, y RAVDESS [LR18], dataset con la que trabajaremos de aquí en adelante por el gran número de muestras variadas y etiquetadas que contiene, además de su fácil accesibilidad.

En cuanto a la extracción de las características del audio, prácticamente la totalidad

de trabajo revisado usa la técnica de extracción de vectores de coeficientes de Mel, la cual permite codificar el audio en una escala logarítmica similar a la que el sistema auditivo humano está acostumbrado.

Las primeras aplicaciones de algoritmos de clasificación tradicionales mostraron resultados muy prometedores pero fueron las nuevas técnicas las que en los últimos años consiguieron precisiones muy significativas. Estos avances están permitiendo aplicar el reconocimiento de emociones en voz de manera fiable en múltiples campos de conocimiento. Se puede remarcar el uso de técnicas de redes neuronales de Deep Learning como LSTM y su variante bidireccional (*Bidirectional Long Short-Term Memory* (BLSTM)), que pertenecen a la familia de redes neuronales recurrentes, como es el caso del artículo [TRB+16] llevado a cabo sobre el dataset RECOLA. También está muy extendido el uso CNN por su comprobado excelente rendimiento en esta función, como podemos comprobar en varios artículos y experimentos [FLC17] [ZMC19] que consiguen precisiones por encima del 90 % y hasta del 97 % ([HAHM+17]). Por último destacar el uso HMM que presenta resultados comparables a los anteriores [NFDS03].

4.2. Análisis de Sentimientos de Texto basado en el Procesamiento de Lenguaje Natural

Estas tecnologías han tenido una gran repercusión en los últimos años, ya que existe un gran potencial dentro de las redes sociales y otras páginas web dónde las opiniones tienen un gran peso. Como mencionan R.Das y Y.Chen ([Das07]) en su artículo relacionado con Yahoo y su evolución, en 1998 había alrededor de 70000 mensajes de Amazon dentro del tablón de Yahoo, y este número creció hasta 900000 en el año 2005. Este crecimiento no el el único que ha desbordado la capacidad humana de procesamiento, ya que las redes sociales llegan cada vez más lejos, y son utilizadas por cualquiera que tenga conexión a Internet. Una de las más famosas dentro de este contexto es Twitter, donde los usuarios publican mensajes que normalmente contienen algún tipo de opinión o sentimiento. A pesar de esta cantidad de información disponible, se ha optado por el dataset Treebank de Stanford, que nos aportará el conjunto de documentos para entrenamiento, desarrollo y tests. Cuenta con reviews cortas sobre películas y se clasifican en cinco distintos grados de opinión (muy negativo, negativo, neutral, positivo, muy positivo).

Algunos de los métodos más usados para trabajar sobre este dataset por su buen funcionamiento han sido BLSTM, CNN y RNN. El primero, usado en [ZS19], consta de varias capas: introductoria, capa de lapso, y capa de interpretación; con 5 clases a analizar. Juntando este método con un modelo pre-entrenado llamado RoBERTa, consiguiendo una precisión máxima de 92 % como se observa en la Figura 4.1. Las Redes Neuronales no han podido mejorar los resultados publicados en [RHL16], aunque tienen un 87 % de acierto y que es casi igualado por [Yan18] con Redes Neuronales Convolucionales y Filtros Neuronales Recurrentes (RNF).

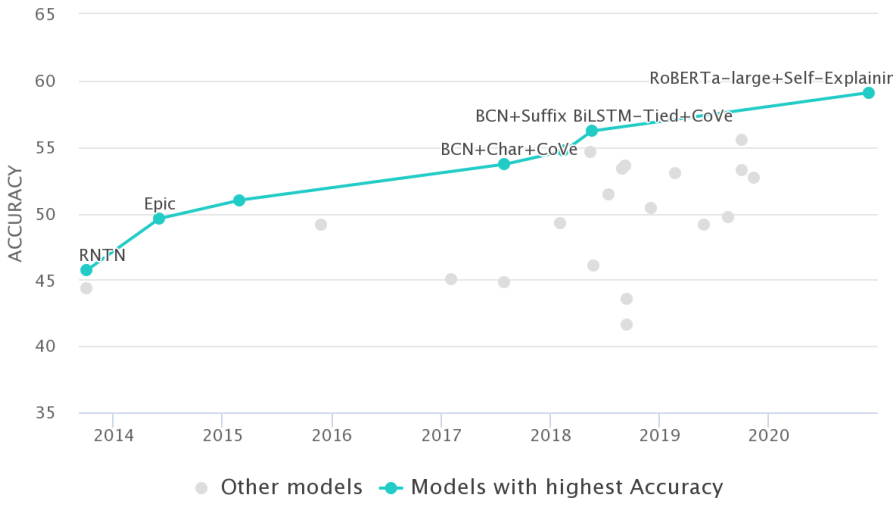


Figura 4.1: Evolución de los algoritmos y sus precisiones sobre el dataset SST [sst]

4.3. Técnicas de Segmentación de Interlocutores

Otra área que ha requerido de nuestro estudio ha sido *Speaker Diarization*, ya que es necesario saber quién transmite las emociones a la hora de hablar. Esta tecnología ha dado como resultado diferentes caminos para investigar la mejor forma de conseguir la segmentación deseada entre los interlocutores. Con el objetivo común de reducir las tasas de error, cambiando los mecanismos internos necesarios para tal tarea. Variadas Redes Neuronales son el punto de discusión principal desde hace pocos años. En el trabajo realizado por Fujita y Kanda ([FKH+19]) modelos basados en *End-to-end Neural Diarization* (EEND) muestran su efectividad siendo entrenados por separado o con BLSTM acoplado, arrojando una diferencia insignificante, pero que supone un avance en la investigación. Otro punto de vista, [SGRS+18] que también se ha dado con Redes Neuronales (DNN) junto con i-Vectores, consigue también reducir el ratio de error, aunque son más altos que el trabajo mencionado anteriormente. Por último, mencionar una aproximación ([HWF+20]) que consigue uno de los ratio de error más bajos (teniendo en cuenta que tratan los márgenes entre interlocutores con más suavidad), usa *Region Proposal Network* (RPN), y compara los resultados con EEND, para demostrar cómo las diferentes aproximaciones que se hagan pueden conllevar mejores resultados en las partes de *Speaker Diarization*.

4.4. Técnicas basadas en el Análisis de Audio y Texto

Durante la etapa de investigación se revisaron múltiples trabajos de reconocimiento de emociones, sentimientos y opinión que hacían uso de técnicas de AI en los que se ha inspirado este proyecto. Los principales experimentos a tener en cuenta son el de reconocimiento de emociones en audio y el de análisis de sentimiento en texto, de los que hay infinidad de aproximaciones diferentes documentadas, pero apenas pudimos encontrar

trabajo previo sobre un sistema que use ambas técnicas. Fue entonces cuando pensamos en proponer un modelo que combinase el reconocimiento de emociones con herramientas de procesamiento de lenguaje natural con el objetivo de obtener una visión más completa de la información verbal y no verbal que transmite un interlocutor.

El sistema híbrido recibiría como entrada piezas de audio en las que pueden participar más de un hablante con la función de aislar los fragmentos en los que participa la persona deseada y analizar las emociones y sentimientos reflejados en lo que dice y en cómo lo dice. Para ello es necesario un sistema de transcripción de audio a texto que extraiga el contenido lingüístico del archivo de audio, lo que servirá como datos de entrada para la Sección de PLN. El mayor dilema se presenta la hora de combinar la información de ambas secciones para obtener una valoración conjunta y final. Para esta cuestión encontramos dos aproximaciones posibles realizadas en trabajos previos.

La primera opción presentada en [YBJ18] combina las características resultantes de la extracción de las mismas tanto en audio como en texto, posteriormente introduce el resultado de dicha combinación al clasificador para entrenar el sistema y generar un modelo capaz de producir una respuesta a nuevos datos introducidos. Este proceso se puede observar en la Figura 4.2.

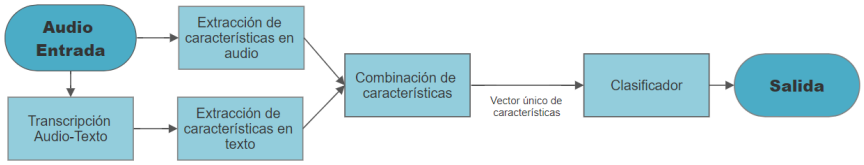


Figura 4.2: Aproximación propuesta en [YBJ18]

La investigación realizada en [BSN15], consiste en utilizar un clasificador diferente para cada Sección y una regresión logística múltiple (*softmax*) común que evalúa las predicciones de diferente naturaleza de los dos clasificadores para dar una única respuesta. El funcionamiento del método propuesto se describe en la Figgura 4.3.

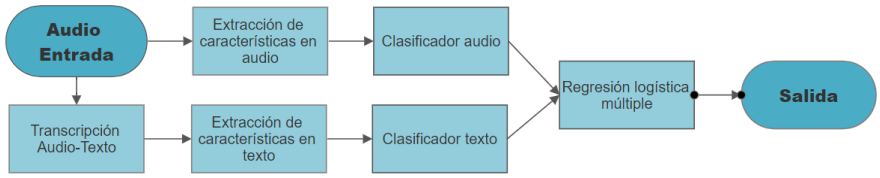


Figura 4.3: Aproximación propuesta en [BSN15]

La propuesta que más nos convence es la que evalúa las características de audio y texto por separado y luego combina los resultados de los clasificadores, ya que primero probaremos cómo funciona la clasificación en ambas secciones en experimentos separados para después intentar poner los resultados en común.

Aún así vemos bastante complejo el hecho de combinar los resultados aportados por los dos itinerarios en una respuesta conjunta y creemos que incluso podría ser merecedor de

un estudio propio. Si el análisis de audio detecta la emoción felicidad y el de texto también un sentimiento positivo es sencillo aglomerar los resultados en uno solo, pero es muy difícil determinar una única respuesta si las predicciones de los dos clasificadores entran en conflicto. Este conflicto podría estar originado por un fallo de clasificación de uno o ambos clasificadores, aunque también se puede tratar de un caso en el que la persona miente pero se delata en su forma de expresarse. Además es posible que no estemos ante un verdadero conflicto y simplemente se trate de que no disponemos del suficiente conocimiento sobre el modo en el que los humanos usamos el habla y el lenguaje.

Capítulo 5

Sistema Propuesto para el Análisis de Sentimientos de Llamadas

En este Capítulo se explica el sistema propuesto, mostrando la decisiones tomadas en el diseño de los experimentos y el proceso de elaboración de los mismos. Las tres principales áreas en las que hemos trabajado son la segmentación de interlocutores, el reconocimiento de emociones en audio y el análisis de sentimientos en texto, y cada una contará con su propio experimento. En la Sección 5.1 se introducen los clasificadores elegidos para realizar los experimentos. La siguiente Sección 5.2 explica cómo se ha comenzado con la parte de Segmentación de Interlocutores, junto con la base de datos usada, su descripción y los pasos iniciales para preparar los experimentos. La Sección 5.3, similar a la anterior, contiene una explicación de la base de datos elegida, de las herramientas escogidas, y parámetros de cada clasificador explicados. La Sección 5.4 contiene la misma estructura que la anterior, basada en el Análisis de Sentiemiento en Texto. Por último, en la Sección 5.5 se comentan las librerías escogidas.

5.1. Clasificadores Evaluados

Antes de entrar en detalle con el diseño de experimentos creemos oportuno explicar más acerca de los tipos de clasificadores que hemos decidido usar. Todos ellos vienen dados por el tool kit ScikitLearn y son relativamente sencillos de configurar. Los parámetros más importantes a la hora de regular el comportamiento de los algoritmos utilizados son:

1. **K-vecinos más Próximos**: Los parámetros ajustados del algoritmo KNN son:
   - **n_neighbors**: Número de vecinos más cercanos.
   - **weights**: Usar 'uniform' para dar el mismo peso a cada vecino o 'distance' para que los pesos sean inversamente proporcionales a la distancia entre vecinos.
   - **algorithm**: 'kd_tree', 'ball_tree', 'brute' o 'auto'. El algoritmo de fuerza bruta nos garantiza un resultado óptimo.
   - **metric**: Tipo de distancia usada (euclideana o manhattan).
2. **Stochastic Gradient Descent Classifier**: Sus principales parámetros son:

- **loss**: Función de pérdida ('hinge' para SVM lineal, 'log' para regresión logística)
- **penalty**: Método de regularización ('l2' es el regularizador estándar para SVMs lineales).
- **alpha**: Constante que multiplica el método de regularización.
- **max_iter**: Número máximo de iteraciones.
- **learning_rate**: Evolución del ratio de aprendizaje.

3. **Clasificadores Basados en Árboles de Decisión**: Potenciación del gradiente y Random Forest se caracterizan por estar compuestas de múltiples algoritmos de aprendizaje ensamblados para obtener una mejor predicción, siendo normalmente árboles de decisión. En general ambos modelos requieren menos preproceso de datos y no es necesario que se cumpla ninguna distribución específica en ellos, a diferencia de otros métodos. Es cierto que el algoritmo de GB puede resultar en un mejor rendimiento que RF pero podría no ser una buena elección si el conjunto de entrenamiento tiene demasiado ruido, ya que resultaría en sobreajuste. Debido a su similitud ambos clasificadores comparten varios parámetros posibles de modificar, siendo algunos de los más importantes los siguientes:
   - **n_estimators**: Número de árboles de regresión utilizados.
   - **max_depth**: Profundidad máxima de cada árbol individual.
   - **max_features**: Número máximo de características a considerar al buscar la mejor partición.
   - **min_samples_split**: Mínimo número de muestras para dividir un nodo interno.
   - **min_samples_leaf**: Número mínimo de muestras requeridas para formar un nodo hoja
   - **tol**: Criterio de parada. Si no es None, el algoritmo parará si no mejora al menos el valor de tol durante n_iter_no_change iteraciones consecutivas.

5.2. Segmentación de Interlocutores

Los datasets que buscamos en este experimento son bases de datos de audios habitualmente de larga duración que además incluyen su correspondiente transcripción y/o la diferenciación entre los diferentes hablantes a lo largo del tiempo. Nos han sido útiles para probar nuestros sistemas de separación de hablantes y posterior transcripción del audio a texto, que son los datos de entrada del análisis de sentimientos. Normalmente suelen ser audios y vídeos de conferencias o reuniones donde múltiples personas participan dinámicamente, o de entrevistas y discursos de personas famosas. Para esta sección hemos seleccionado el siguiente dataset:

- **AMI Corpus** [fAp06]: Consiste en 100 horas de grabaciones de reuniones realizadas en tres habitaciones con diferente acústica en inglés hablado principalmente por personas no nativas. Incluye múltiples anotaciones para usar como refuerzo en el entrenamiento de los modelos, como división de tema, gestos de la cabeza, manos e incluso emociones.

Inicialmente, se ha descargado el toolkit de *pyannote.audio*, que contiene los tutoriales necesarios para conseguir la base de datos, y saber cómo tratar los audios. Internamente, esta herramienta se divide entre modelos y *pipelines*, que se pueden entrenar de forma manual o usar los que ya vienen entrenados de antemano. Cómo se busca un análisis profundo de un audio externo a la base de datos, se compararan los resultados de los modelos pre-entrenados y los de entrenamiento propio con los audios y su referencia para saber cuál funciona mejor.

Internamente, la base de datos contiene 170 audios, divididos en 118 para el entrenamiento, 26 para el desarrollo, y 24 de test. Cada uno de estos audios tiene un archivo donde se especifica quién habla cuándo, y durante cuántos segundos. Gracias a esta referencia es más fácil analizar los resultados. Junto a la base de datos de reuniones, se incluye también un conjunto de archivos con ruido y música para hacer que el programa sea capaz de distinguir los momentos en los que, a pesar de existir ondas de sonido, se obvien si solamente es ruido y no conversación entre interlocutores.

La herramienta de *pyannote* tiene la opción de añadir bases de datos externas mediante *pyannote.audio.database*, donde se especifican los audios de entrenamiento, desarrollo y tests. También contiene una sub-herramienta con la que analizar los datos para obtener resultados precisos como error de detección, pureza del análisis, falsas alarmas, etc (explicación completa en la sección de experimentos).

De cada audio se extraen las características, y se almacenan en archivos npy, que son más tarde manejados por la librería Numpy. Estos archivos contienen arrays con los coeficientes de Mel que se necesitan para la red neuronal de entrenamiento.

La disponibilidad del hardware ha hecho que las precisiones sean más limitadas, y no se haya podido alcanzar un muy buen resultado. Los modelos se basan en archivos de configuración, donde están los detalles de cómo se trata la información: tipo de entrenamiento, tamaño de los vectores a procesar, dimensión de la red neuronales (*hidden layer*) y número de capas de profundidad.

5.3. Reconocimiento de Emociones en Audio

El primer paso en el diseño y uno de los más determinantes es la elección de un buen dataset que esté focalizado en la función a desempeñar por nuestro modelo. Las bases de datos de audio para reconocimiento de emociones son complejas de diseñar, ya que existen diferencias significativas entre el habla actuada y el habla espontánea si los actores de voz no son capaces de capturar la emoción requerida o si es demasiado exagerada y no coinciden con las emociones captadas en un entorno real. Además debe existir una buena representación de los diferentes registros y tonos de la voz humana, por lo que es necesaria la presencia de múltiples actores de voz, masculinos y femeninos. Existen varias bases de datos de emociones en audio de acceso abierto de entre las que hemos seleccionado la siguiente por los buenos resultados en trabajo previo realizado sobre ella:

- **RAVDESS (Ryerson Audio-Visual Database of Emotional Speech and Song)** [LR18]: Contiene 7356 archivos de audio, audio-video y sólo video de

frases y canciones siendo interpretadas por 24 actores (12 voces masculinas y 12 femeninas) expresando 8 diferentes emociones: felicidad, calma, tristeza, enfado, miedo, neutralidad, disgusto y sorpresa. El lenguaje empleado es el inglés, aunque no creemos que sea algo determinante ya que nos interesa la entonación y modulación de la voz a la hora de expresar emociones. Dos diferentes frases son utilizadas: "kids are talking by the door" y "dogs are sitting by the door". Todos los intérpretes realizan dos repeticiones para cada emoción, cada frase y con dos tipos de énfasis, uno con menor y otro con mayor nivel de intensidad. En total son 1440 muestras de audio etiquetadas con su correspondiente emoción expresada que pueden ser de gran utilidad para el experimento.

Tras escoger el dataset podemos centrarnos en buscar las herramientas para extraer las características de los datos. En la mayoría de trabajo previo revisado se usa la técnica de extracción de coeficientes de Mel (MFCCs), la cual es una buena forma de representar el audio en una escala logarítmica similar a la que usa el sistema auditivo humano. La librería que nos aportará esta funcionalidad es la llamada Librosa [MLM+20] con la función `mfcc`. Los parámetros que modificaremos de dicha función son:

- **y**: Archivo de audio.
- **sr**: Sample rate (frecuencia de muestreo). Asignado a 44100 Hz.
- **n_mfcc**: Número de mfcc a devolver.
- **norm**: Normalización en la fase de DCT. Asignado a 'ortho'.

Una vez extraídas las características de nuestros datos podemos introducirlas a un clasificador para entrenarlo y probar la efectividad de sus predicciones. Para escoger un clasificador hemos decidido probar varias opciones y comparar su rendimiento con diferentes conjuntos de entrenamiento. También realizaremos un proceso de optimización de hiperparámetros (GridSearch) para escoger la mejor combinación en los valores de los parámetros. Los tres clasificadores que han sido seleccionados son KNN, SGD que es un tipo de máquina de vectores de soporte y Random Forest, un algoritmo basado en árboles de decisión. Los valores preseleccionados para cada parámetro han sido ajustados tras una serie de pruebas menores con todos los algoritmos. Los conjuntos de parámetros y los posibles valores preseleccionados para su inclusión en el algoritmo GridSearch son:

1. Parámetros preseleccionados para el algoritmo KNN:
   - **n_neighbors**: [1, 3, 5, 10]
   - **weights**: ['uniform', 'distance']
   - **algorithm**: 'brute'.
   - **metric**: ['euclidean', 'manhattan', 'minkowski']
2. Parámetros preseleccionados para el algoritmo SGD:
   - **loss**: ['hinge', 'log']
   - **penalty**: ['l1', 'l2']
   - **alpha**: [$1e^{-2}$, $1e^{-3}$, $1e^{-4}$]

- **max_iter**: [1000, 2000, 5000]

3. Parámetros preseleccionados para el algoritmo Random Forest:
   - **bootstrap**: [True, False]. Habilitar el uso de muestras boostrap.
   - **max_depth**: [10, 30, 50, None]
   - **min_samples_leaf**: [1, 2, 4]
   - **min_samples_split**: [2, 5, 10]
   - **n_estimators**: [200, 600, 1000]

Para evaluar los resultados en nuestro proyecto recurriremos a la funcionalidad ShuffleSplit, también aportada por ScikitLearn, que es una validación cruzada de permutación aleatoria. Los parámetros que necesita son los tamaños de los conjuntos de entrenamiento y validación, además del número de iteraciones de partición y remezcla.

En la Figura 5.1 se muestra de forma esquemática los pasos para el reconocimiento de emociones en audio.

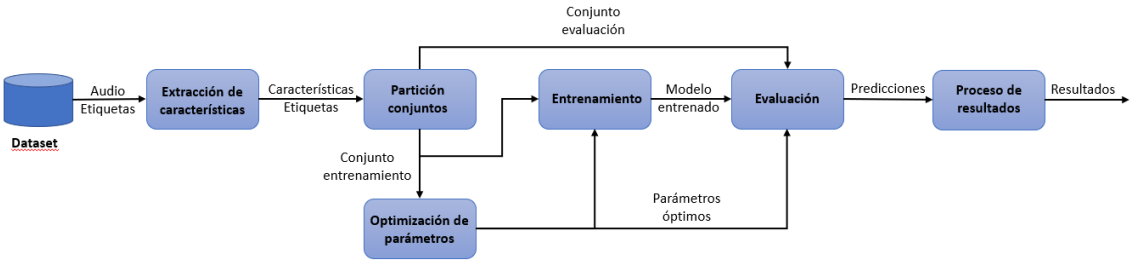


Figura 5.1: Estructura del Experimento de Reconocimiento de Emociones en Audio

5.4. Análisis de Sentimiento en Texto

El uso de modelos de análisis de sentimientos está creciendo enormemente en los últimos tiempos para monitorear las redes sociales, los gustos de las masas y las demandas del mercado. Estos modelos se basan en métodos de PLN (Procesamiento de Lenguaje Natural) y diferentes técnicas de Machine Learning para aprender de este tipo de datasets. Es necesaria una gran cantidad de datos para confeccionar una base de datos de sentimientos, aunque el mayor desafío reside en recopilar datos que sean relevantes y que cubran un gran área del análisis de sentimientos. Normalmente el corpus de este tipo de datasets está extraído de grandes sitios web en los que cualquier persona puede expresarse, como trends de Twitter, reviews de películas y series, comentarios en productos de Amazon, etc. La base de datos de sentimientos utilizada en este periodo es:

- **Stanford Sentiment Treebank (SST)** [RSP06]: Consiste en 11855 oraciones extraídas de críticas a películas publicadas por los usuarios del sitio web `rottentomatoes.com` en idioma inglés, disponible a través de la librería `pytreebank`. Este dataset fue diseñado con el propósito de tener en cuenta el orden y la jerarquía de las palabras para hacer análisis de sentimientos más complejos en un rango de

clasificación del 1 al 5 (1 muy negativo, 5 muy positivo). También existen varias pruebas realizadas sobre este mismo conjunto pero con 2 categorías solamente, una positiva y una negativa.

Una vez tenemos el dataset cargado podemos empezar a modificarlo aplicándole diferentes técnicas de preprocesamiento. Utilizaremos varias herramientas de NLTK, como el tokenizador TweetTokenizer, el lematizador WordNetLemmatizer y el conjunto de stop words que ofrece. Todas estas técnicas nos servirán para filtrar el corpus de documentos antes de extraer sus características e introducirlas en los clasificadores.

El extractor de características de este experimento tiene como objetivo determinar la frecuencia de términos, para lo cual usaremos la herramienta CountVectorizer de Sklearn. Este extractor de frecuencias nos ofrece métodos de preprocesamiento de lenguaje natural integrados que podemos usar cambiando el valor de un parámetro, pero hemos decidido hacer el procesamiento por separado con ayuda de métodos de NLTK. Los principales parámetros de CountVectorizer que modificaremos son:

- **token_pattern**: Expresión regular que determina el patrón de los tokens. Asignado a `r'\b\w\w+\b'`.
- **encoding**: Codificación de los documentos. Asignado a `'utf-8'`
- **ngram_range**: Tupla que representa los límites inferior y superior de los n-gramas a extraer. Tras realizar varias pruebas rápidas hemos comprobado que la división en palabras individuales (monogramas) es la más efectiva para todos las posibles versiones del dataset, por lo que será asignada la tupla (1,1).

Además haremos uso del transformador TfidfTransformer, también aportado por Sklearn, que transformará la matriz resultante del paso anterior y muñtiplicará el valor de frecuencia de términos (TF) por la frecuencia inversa de documento (IDF).

En este experimento realizaremos el mismo proceso de selección de un clasificador entre un conjunto de candidatos cuyos parámetros además serán optimizados para cada diferente conjunto de entrenamiento. Los clasificadores escogidos en este caso fueron: SGD, Random Forest y Gradient Boosting, otro tipo de clasificador basado en árboles de decisión. Los parámetros en concreto y los posibles valores de cada uno de ellos han sido seleccionados tras algunas pruebas intentando encontrar un equilibrio en el que los tiempos de ejecución no sean demasiado altos y el rendimiento sea el más aceptable. A continuación explicamos los conjuntos de parámetros y sus posibles valores preseleccionados para introducirlos en el algoritmo GridSearch:

1. Parámetros preseleccionados para el algoritmo SGD:
   - **loss**: ['hinge', 'log']
   - **penalty**: ['l1', 'l2']
   - **alpha**: [$1e^{-3}$, $1e^{-4}$, $1e^{-5}$, $1e^{-5}$]
   - **max_iter**: [500, 1000, 2000, 5000]

2. Parámetros preseleccionados para el algoritmo Gradient Boosting:
   - **loss**: ['hinge', 'log']. Función de pérdida que será optimizada.

- **max_depth**: [1, 5, 10]
- **n_estimators**: [200, 500, 1000]

3. Parámetros preseleccionados para el algoritmo Random Forest:
   - **bootstrap**: [True, False]. Habilitar el uso de muestras boostrap.
   - **max_depth**: [100, 200, 500, 1000]
   - **n_estimators**: [100, 200, 500]

La Figura 5.2 muestra el diagrama de flujo del de los principales procesos y los datos que se transmiten:

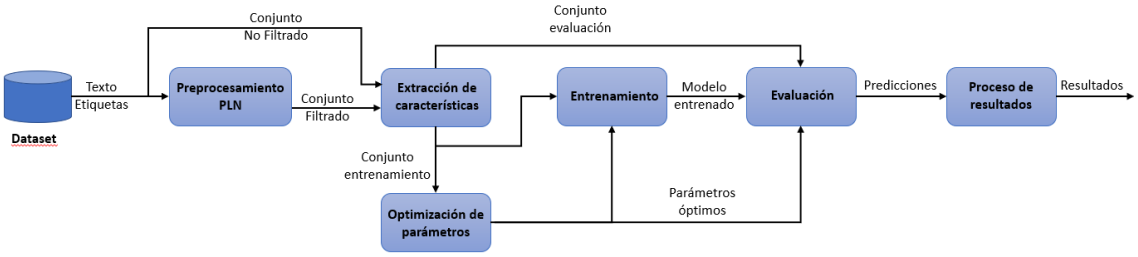


Figura 5.2: Estructura del Experimento de Análisis de Sentimiento en Texto

5.5. Librerías

Las librerías que se han usado han sido:

1. **Sklearn** (scikit-learn) [PVG+11]. Nos aportará herramientas que cubren la mayoría de técnicas usadas en *Machine Learning*: varios datasets, técnicas de extracción y selección de características, todo tipo de modelos de aprendizaje supervisado y no supervisado así como múltiples opciones de preprocesamiento y visualización de los datos.

2. **Librosa** [MLM+20]. Se centra en el análisis de audio y música. Permite extraer los MFCCs necesarios como datos de entrada para el clasificador de reconocimiento de emociones en audio.

3. **MatplotLib** [Hun07]. Paquete de gráficos 2D usado para generar imágenes que sirven de representación de los resultados de las pruebas.

4. **NLTK** (Natural Language Tool Kit) [BKL09]. Aporta las herramientas de procesamiento de lenguaje natural necesarias para el preprocesamiento de nuestro experimento.

5. **Pyannote** [BYC+20]. Paquete que contiene varios tipos de utilidades para el procesamiento de audio, como databases, pipelines y modelos preentrenados

Capítulo 6

Experimentos y Resultados

Este capítulo se dividirá en tres experimentos diferenciados que persiguen diversos objetivos, mostrando en todos ellos el procedimiento seguido a la hora de procesar la información, así como un análisis de los resultados obtenidos tras dicho procesamiento. En primer lugar, en la Sección 6.1 se explicará el experimento de reconocimiento de emociones en audio, en el cual el objetivo es crear un modelo que sea fiable y robusto a la hora de reconocer la emoción de una pieza de audio dentro de un abanico de 8 emociones. Tras esto, en la Sección 6.2 el experimento de análisis de sentimientos en texto será descrito. Durante el proceso de experimentación se ha hecho uso de técnicas de procesamiento de lenguaje natural para extraer las características de un conjunto de documentos, que sufrirá una serie de modificaciones para encontrar la mejor variante con la que entrenar el modelo. El último experimento, en la Sección 6.3 consistirá en el análisis del audio para reconocer la separación entre interlocutores. Se analizarán varias características del audio, como la actividad y superposición de discursos o el cambio de hablante, y la conjunción de los datos extraídos podrá darnos una mejor idea de cómo se divide el texto y quién interactúa en cada parte extraída. La combinación de los mejores resultados de estos tres experimentos en un sistema único permitiría analizar la emoción y sentimiento expresados por cualquier hablante en un segmento de audio.

6.1. Reconocimiento de Emociones en Audio

Este experimento tratará de entrenar y comparar varios modelos para comprobar cuál es el clasificador y sus parámetros óptimos que obtiene los mejores resultados en el reconocimiento de la emoción expresada en una pieza de audio.

6.1.1. Dataset y Extracción de Características

La base de datos con la que se han realizado los experimentos de esta sección es la dataset RAVDESS [LR18], la cual es abierta, está etiquetada y es relativamente sencilla de usar. Se utilizará la muestra que contiene 1440 audios en los que múltiples actores de doblaje de ambos sexos interpretan dos sencillas frases en ocho connotaciones emocionales diferentes. Las ocho emociones reflejadas en la voz a reconocer son: neutral, calma,

felicidad, tristeza, enfado, miedo, disgusto y sorpresa. Todos los experimentos se realizarán con una partición de 80 % para el conjunto de entrenamiento y un 20 % para el test.

Las librerías más útiles en este caso han sido Librosa y Sklearn, que proporcionan la forma de obtener las características del audio y los clasificadores para procesarlas, respectivamente. La función mfcc de Librosa nos permite extraer el número deseado de MFCCs de un audio concreto, que son los que posteriormente se introducen en el clasificador para entrenarlo y evaluar su comportamiento.

Los parámetros modificados de la función mfcc extractora de características han sido:

- y: Datos de entrada
- sr (Sampling Rate): 44100 Hz
- n_mfcc: 15/20/25 coeficientes
- norm: 'ortho' (Normalización)

La longitud de los vectores depende del tiempo del audio, lo que combinado al gran número de muestras puede ser un problema para los tiempos de procesado de los modelos. Es por ello que se ha decidido reducir el número de características reduciendo cada vector a dos datos: la media de sus valores y la desviación estándar. Por lo tanto los conjuntos tendrán las dimensiones [1440, 2*n_mfccs], lo que ahorrará en tiempos de ejecución obteniendo resultados fiables.

6.1.2. Clasificadores y Procedimiento

Los clasificadores escogidos para el experimento fueron KNN, SGD y Random Forest, todos ellos aportados por la librería Sklearn. Cada uno se probará con cuatro conjuntos de entrenamiento que varían en el número de vectores de coeficientes MFCC extraídos por audio. Inicilamente se probarán conjuntos de 20, 25, 30, 40 vectores de coeficientes para comprobar si un aumento del número de características extraídas mejora la precisión de los modelos.

Para todos ellos se llevará a cabo un proceso de optimización de hiperparámetros, que consiste en elegir la mejor combinación de valores de los parámetros de un algoritmo de aprendizaje. Esta función es implementada por la herramienta GridSearch de Sklearn, que realiza una búsqueda exhaustiva dentro de un conjunto de parámetros y valores previamente seleccionados. El algoritmo se debe guiar por una métrica de rendimiento, la cual será en este caso el F1 score, una medida que combina las métricas de precisión y exhaustividad. Además, para estimar unos resultados generales sin depender de la partición de los datos se hará uso de una validación cruzada.

Tras conocer los mejores valores posibles para los parámetros de los clasificadores se procederá a desarollar la etapa de entrenamiento, donde los clasificadores apropiadamente configurados crearán sus modelos a partir de los datos de entrenamiento. Finalmente se realizarán varias pruebas: 10 validaciones cruzadas de 10 particiones para ver un promedio de la respuesta de los modelos ya entrenados y una última validación cruzada de 5 particiones en la que se comprobará el rendimiento de los mismos según diferentes métricas del error.

A continuación se explican los conjuntos de parámetros, sus respectivos posibles valores que fueron preseleccionados si dicho parámetro se incluyó en la optimización de *GridSearch*,

así como los valores finalmente seleccionados para los diferentes clasificadores:

1. Conjunto de parámetros para el algoritmo KNN:
   - **n_neighbors**:
     - Posibles valores: [1, 2, 3, 5, 10]
     - Valor seleccionado: 1. Esto significa que el algoritmo funcionará como un *Nearest Neighbor* (vecino más próximo), por lo que cada muestra adquirirá la clasificación de su único vecino más cercano.
   - **weights**:
     - Posibles valores: ['uniform', 'distance']
     - Valor seleccionado: 'uniform'. Los pesos serán uniformes, por lo que todos los puntos del vecindario influirán a los demás de la misma manera.
   - **metric**:
     - Posibles valores: ['euclidean', 'manhattan']
     - Valor seleccionado: 'manhattan'. La distancia Manhattan, a diferencia de la distancia euclediana (que es la mínima posible entre dos puntos), dice que la distancia entre puntos es la suma de las diferencias absolutas de sus coordenadas.
   - **algorithm**:
     - Valor seleccionado: 'brute'. El algoritmo de fuerza bruta nos garantiza que la respuesta resultante del entrenamiento será óptima .Aunque puede ser poco eficiente, no es un problema en nuestro proyecto al no ser excesivo el tiempo de ejecución.
2. Conjunto de parámetros para el algoritmo SGD:
   - **loss**:
     - Posibles valores: ['hinge', 'log']
     - Valor seleccionado: 'hinge'. La función de pérdida *hinge* es propia de las SVM
   - **penalty**:
     - Posibles valores: ['l1', 'l2']
     - Valor seleccionado: 'l1'. El método de regularización elegido es el llamado *l1* o *regresión Lasso*, en el que la penalización depende del valor absoluto de la magnitud de los coeficientes de regresión.
   - **alpha**:
     - Posibles valores: [0.01, 0.001, 0.0001]
     - Valor seleccionado: 0.01. Es el valor que multiplica el término de regularización para darle más o menos importancia.
   - **max_iter**:
     - Posibles valores: [500, 1000, 2000, 5000]
     - Valor seleccionado: 5000. Número máximo de iteraciones.
3. Conjunto de parámetros para el algoritmo Random Forest:
   - **bootstrap**:

- Posibles valores: [True, False]
- Valor seleccionado: False. El dataset completo es usado para construir cada árbol

▪ **max_depth**:
- Posibles valores: [10, 30, 50, None]
- Valor seleccionado: 50. Profundidad máxima de cada árbol. Si no definimos este parámetro los nodos se extienden hasta que la hoja es pura o contiene menos de *min_samples_split* muestras.

▪ **min_samples_leaf**:
- Posibles valores: [1, 2, 4]
- Valor seleccionado: 1. Mínimo número de muestras requeridas para formar una hoja.

▪ **min_samples_split**:
- Posibles valores: [2, 5, 10]
- Valor seleccionado: 2. Número mínimo de muestras necesarias para partir un nodo.

▪ **n_estimators**:
- Posibles valores: [200, 600, 1000]
- Valor seleccionado: 600. Número concreto de árboles en el Random Forest.

6.1.3. Resultados

Se ha comenzado mostrando los resultados de 10 validaciones cruzadas de 10 particiones en gráficos de caja para poder apreciar cómo la eficacia de cada clasificador se distribuye estadísticamente. La caja de cada clasificador está compuesta por **cuartiles** acotados entre el 25, 50, y 75 % de los datos (bordes de la caja respectivamente). Las líneas que representan el inicio y fin del gráfico (conocidas como bigotes) son los límites superior e inferior donde se pueden esperar valores típicos. Finalmente, los círculos que aparecen fuera del rango establecido por el clasificador, son los valores atípicos, también conocidos como *outliers*. Estos gráficos proporcionan una visión más general de la distribución del rendimiento de los clasificadores, ya que agrupan todos la información en una dimensión y es más sencillo así ver donde está la mediana y cómo se reparten el resto de valores. Los resultados de los tests realizados con conjuntos diferentes en su número de coeficientes se muestran en la Figura 6.1.

Como se puede observar en las Figura 6.1, con 20 y 25 MFCCs los gráficos varían ligeramente, dejando en ambos al clasificador SGD por debajo de los otros dos. KNN mantiene su mediana alrededor del 65 %, siendo además su dispersión menor que la de los otros dos en el caso de los 20 Mfccs. Sin embargo, el mejor clasificador de estas dos variantes parece ser RF, con una mediana muy próxima al 70 % y una dispersión reducida.

Los gráficos con 30 y 40 coeficientes presentes en la Figura 6.2 confirman que SGD no es la herramienta más óptima para este experimento, ya que la precisión se encuentra alrededor del 45 % y la dispersión es considerable. Las mayores similitudes se dan entre KNN y Random Forest ya que, como se observa, el rendimiento de KNN ha mejorado notablemente con el aumento en el número de coeficientes, llegando a ser comparable al

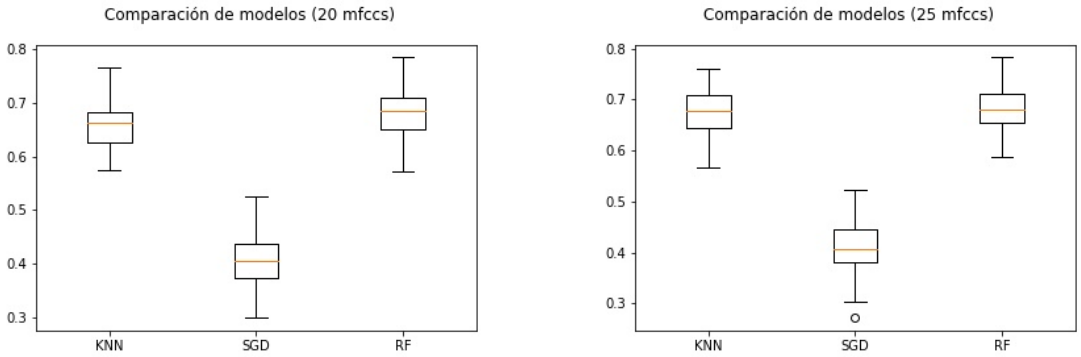


Figura 6.1: Comparación F1 score. Izquierda: 20 Mfcss. Derecha: 25 Mfccs

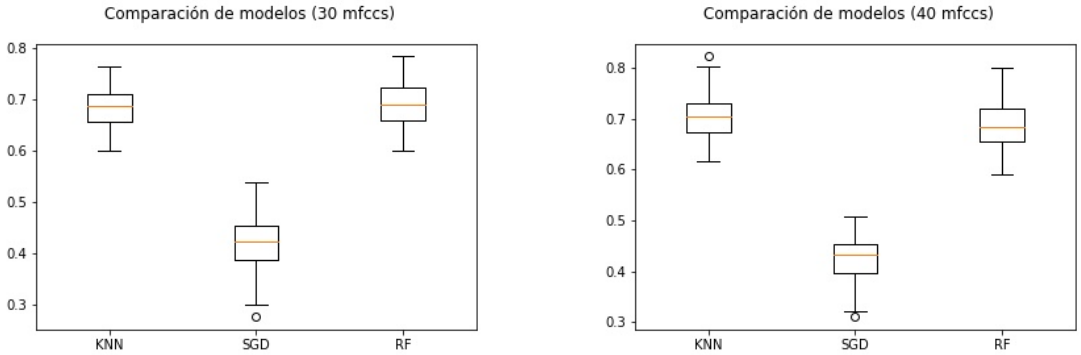


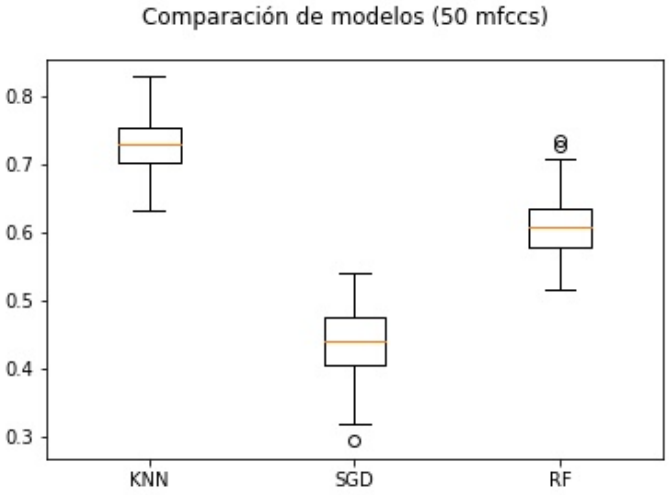
Figura 6.2: Comparación F1 score. Izquierda: 30 Mfcss. Derecha: 40 Mfccs

rendimiento de RF, el cuál no ha sufrido grandes cambios. Con estos datos no es posible hacer una valoración final y elegir un clasificador como el mejor, por lo que las pruebas se han repetido con un conjunto de número de coeficientes aún mayor para ver si persiste la tendencia creciente en el rendimiento de KNN. Dicho conjunto tendrá 50 vectores de coeficientes MFCCs y de igual manera se harán 10 validaciones cruzadas para cada clasificador para comprobar su rendimiento y fiabilidad:

Como se comprueba en la Figura 6.3, la precisión del algoritmo KNN ha mejorado con el aumento en el número de coeficientes y ha dejado atrás al Random Forest, consiguiendo validaciones con un valor alrededor del 75 %. Esto probablemente se deba a que KNN funciona mejor cuando los conjuntos de datos no son demasiado amplios, como es el caso, ya que tiene que calcular las distancias entre las muestras. Además el modelo de RF presentaba una buena respuesta pero ésta se ha visto perjudicada con el aumento de características, indicando que el modelo ha sido objeto de *overfitting* o sobreajuste.

Por último, la Tabla 6.1 muestra una última validación cruzada de 5 particiones para todos las combinaciones de modelos y conjuntos de entrenamiento con el objetivo de aportar algunas métricas del error. Dichas métricas son la precisión (en inglés *precision*), la exactitud (*accuracy*) y la exhaustividad (*recall*).

La mejor respuesta que podemos esperar de uno de estos clasificadores es la que nos ofrece KNN entrenada con un conjunto de considerable número de características. La



ll

Figura 6.3: Comparación F1 score con 50 Mfcss

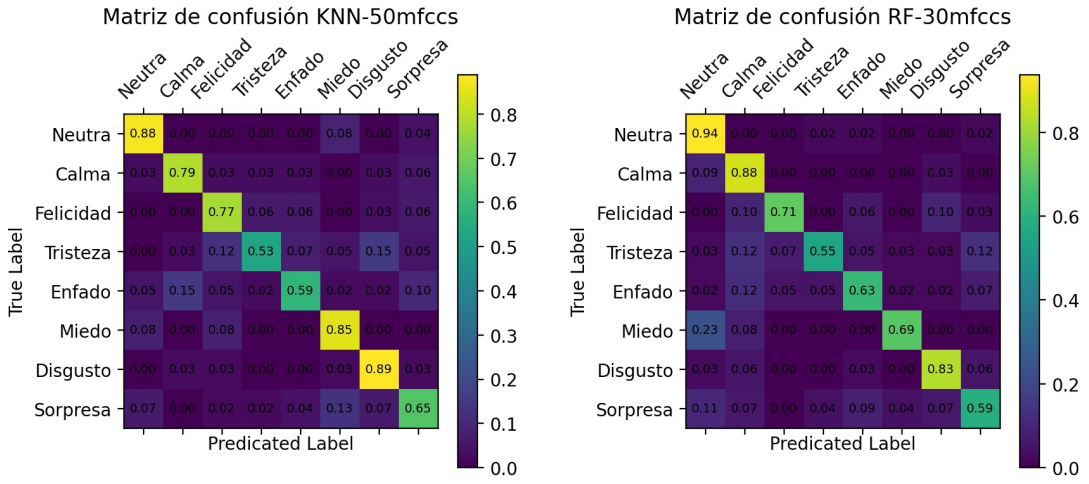
Tabla 6.1: Métricas para los diferentes clasificadores y conjuntos

Clasificador	**KNN**			**SGD**			**RF**		
Número de Mfccs	Precisión	Exactitud	Exhaustividad	Precisión	Exactitud	Exhaustividad	Precisión	Exactitud	Exhaustividad
20 Mfccs	64.66	63.98	63.98	43.26	41.55	41.99	68.60	67.18	67.62
25 Mfccs	66.96	66.06	66.06	47.58	42.07	38.44	69.34	67.27	72.22
30 Mfccs	67.59	66.67	66.67	42.93	41.64	41.82	69.52	68.22	67.44
40 Mfccs	70.46	69.7	69.7	46.78	44.07	43.98	70.52	68.22	67.88
50 Mfccs	72.19	71.34	71.34	49.4	42.59	44.15	69.92	67.01	67.96

manera más eficaz de apreciar el comportamiento de un modelo es representando sus predicciones en una matriz de confusión, ya que es posible comparar directamente las asignaciones que hace el modelo con las etiquetas de referencia. En la Figura 6.4(a) es el aspecto de una predicción sobre el conjunto de test realizado por un modelo KNN entrenado con un conjunto con 50 vectores de características MFCCs.

La matriz de confusión mostrada en la Figura 6.4(b) representa el comportamiento del modelo de Random Forest que mejor resultado ha obtenido, el entrenado con 30 MFCCs. En ella se aprecia un comportamiento de modelo bastante fiable, destacando la diagonal en la que coinciden valores verdaderos y predichos, es decir, cuando la predicción es acertada. En ninguna categoría se baja del 50 % de precisión y en algunas emociones como neutra, miedo y disgusto se supera incluso el 80 %. Las clases que menos ha podido diferenciar el modelo han sido las de tristeza, enfado y sorpresa, aunque ninguna de ellas ha sido confundida por otra categoría determinada.

En el caso de Random Forest presenta una matriz algo más irregular al contener valores más dispares en la diagonal principal. Existen categorías que tienen una precisión mayor en este caso que en el de KNN y otras que se quedan por debajo. Es curioso el aumento de precisión en connotaciones como neutra y calmada, cerca de la totalidad de las predicciones acertadas, pero también es visible la ligera confusión que genera en algunas emociones como puede ser el caso de las clases neutra y miedo.



(a) KNN entrenado con 50 MFCCs

(b) RF entrenado con 30 MFCCs

Figura 6.4: Comparación de las matrices de confusión de los mejores modelos obtenidos

Estos resultados indican que el modelo de KNN entrenado con 50 mfccs es bastante robusto y fiable aunque si no se dispone de demasiados coeficientes el algoritmo Random Forest podría ser también una opción inteligente.

6.2. Análisis de sentimientos con PLN

El análisis de sentimiento o minería de opinión se refiere al uso de PLN para extraer información subjetiva de documentos de texto. Dicha información es obtenida por la AI basándose en relaciones estadísticas, no en un análisis lingüístico. El objetivo de este experimento será entrenar varios clasificadores con un *dataset* específicamente preparado para obtener un modelo que sea capaz de determinar la actitud de un usuario con respecto a cualquier tema. Para elegir el modelo óptimo se procesará el conjunto de datos de diversas formas para obtener los diferentes datos de entrada de los extractores de frecuencia y los clasificadores. En concreto se compara el rendimiento del texto en crudo con el de texto filtrado y lematizado, además de crear una variante alternativa del dataset original en la que las muestras están clasificadas en tres categorías en vez de cinco.

6.2.1. Preprocesamiento

El conjunto de datos escogido en este experimento ha sido el llamado *Stanford Sentiment Treebank* [RSP06], un dataset que recopila un total de 11855 frases extraídas de reseñas de películas publicadas por los usuarios del sitio web `rottentomatoes.com` en idioma inglés. Cada muestra tiene adjunta una etiqueta que representa el grado de positividad denotada por la frase en una escala de cinco posiciones. Fue diseñada para tener en cuenta el orden de las palabras representando la estructura de la frase en forma de árbol. Esto aumenta la precisión frente a los clasificadores de bolsa de palabras regulares, ya que estos se basan en la presencia de palabras de fuerte sentimiento (como alucinante

o fatal) y fallan a la hora de reconocer negaciones más complejas. Está disponible a través de la librería `pytreebank` en un formato que divide los conjuntos de entrenamiento, prueba y desarrollo. Hemos pensado en probar también el rendimiento si reducimos el número de categorías a 3, para lo que hemos preparado una función que combina tanto las dos categorías positivas como las negativas, transformando el conjunto inicial. Esto nos deja con aproximadamente la mitad de elementos con la etiqueta de connotación neutra comparado con las nuevas clases combinadas, lo que podría llegar a ser un problema para el aprendizaje de nuestros modelos.

En adición a la librería Sklearn, que aporta las herramientas de Inteligencia Artificial, ha sido de especial utilidad la librería NLTK (Natural Language ToolKit [BKL09]), que facilita diversas funciones y contenido útil en el procesamiento de lenguaje natural. En concreto se ha hecho uso de su tokenizador y lematizador así como de las *stopwords* necesarias para filtrar el texto. El tokenizador permite dividir el texto en crudo en palabras o *tokens*, que serán transformados a letra minúscula si no se trata de un carácter de puntuación. A continuación las palabras son introducidas al lematizador que reducirá el término a su raíz o lema ya sea el mismo un nombre, verbo, adjetivo o adverbio.

Una vez eliminadas las ambigüedades entre palabras que comparten significado es posible tomar la decisión de filtrar el texto. Si se decide filtrar el conjunto, se eliminarán las *stopwords*, palabras que son muy comunes en los documentos, que apenas denotan una connotación y no son determinantes a la hora de generar un modelo. El resultado de este proceso de filtrado es una lista cuyas muestras son las frases de las reseñas en minúscula y limpias de signos de puntuación y palabras genéricas. La Figura 6.5 es una nube de palabras, donde el tamaño de cada palabra representa su número de apariciones en el conjunto de textos.

Figura 6.5: Nube de palabras del conjunto filtrado

6.2.2. Frecuencia de Término y Frecuencia Inversa de Documento

Antes de insertar los datos en los clasificadores es necesario extraer sus "características", que en este caso será averiguar cuán relevante es una palabra en un documento para que de este manera los algoritmos sean capaces de asociar ciertas palabras clave a categorías determinadas. Calcular métricas como la frecuencia de término y la frecuencia inversa de documento puede ayudar en esta labor. La frecuencia de término se refiere al número de veces que una palabra aparece en un documento mientras que la frecuencia inversa de documento mide lo común que es un término en el conjunto de documentos. El producto de ambas métricas proporciona una mejor idea de las palabra más importantes a la hora de determinar la categoría de un texto.

El extractor de frecuencias que se usará es el CountVectorizer de Sklearn, que permite obtener una matriz de frecuencias donde las columnas son los documentos y las filas todas las palabras existentes en el diccionario creado a partir de los datos introducidos. El parámetro introducido más relevante a comentar es el de `ngram_range`, que determina el rango de *ngramas*, al que se ha asignado como valor la tupla (1,1). Esto significa que el texto se dividirá en monogramas ya que, como se ha comentado en el sistema propuesto, se ha comprobado empíricamente que el rendimiento es mejor usando palabras individuales en vez de grupos de palabras.

Para aplicar el cálculo de TF-IDF se usará el transformador TfidfTransformer, también de Sklearn, que transformará la matriz de frecuencias multiplicando cada valor TF con su correspondiente valor de IDF. Esta matriz transformada contiene los datos de entrada que se introducirán a los clasificadores, a partir de los cuales aprenderán y construirán un modelo.

6.2.3. Clasificadores

Los clasificadores seleccionados en este experimento han sido SGD, Gradient Boosting y Random Forest, y cada uno de ellos ha sido probado con los cuatro conjuntos resultantes de combinar filtrado o no filtrado con la división en 3 o las 5 categorías originales. También se llevará a cabo el mismo proceso de optimización de hiperparámetros que en el apartado anterior gracias a la herramienta GridSearch de Sklearn, con la diferencia de que en este caso los conjuntos de entrada divergen más entre sí. Es por ello que habrá una búsqueda de la mejor combinación de parámetros de los clasificadores para cada conjunto de entrada.

A continuación se muestran los parámetros y sus posibles valores que el algoritmo de optimización de hiper-parámetros ha combinado para encontrar el mejor modelo de cada clasificador, así como los valores finalmente seleccionados para cada conjunto de entrenamiento:

1. Conjunto de parámetros a optimizar para el algoritmo SGD:
   - **loss**:
     Posibles valores: ['hinge', 'log']
   - **penalty**:
     Posibles valores: ['l1', 'l2']

- **alpha**:
  Posibles valores: [$1e^{-3}$, $1e^{-4}$, $1e^{-5}$, $1e^{-6}$]
- **max_iter**:
  Posibles valores: [500, 1000, 2000, 5000]

Tabla 6.2: Valores seleccionados en la optimización de hiperparámetros para SGD

Parámetro		loss	penalty	alpha	max_iter
SST-5	Filtrado	'hinge'	'l2	$1e^{-3}$	2000
	No filtrado	'hinge'	'l2	$1e^{-4}$	5000
SST-3	Filtrado	'hinge'	'l2	$1e^{-4}$	5000
	No filtrado	'log'	'l2'	$1e^{-5}$	5000

2. Conjunto de parámetros a optimizar para el algoritmo Gradient Boosting:
   - **loss**:
     Posibles valores: ['deviance', 'exponential']
   - **max_depth**:
     Posibles valores: [1, 5, 10]
   - **n_estimators**:
     Posibles valores: [200, 500, 1000]

Tabla 6.3: Valores seleccionados en la optimización de hiperparámetros para GB

Parámetro		loss	max_depth	n_estimators
SST-5	Filtrado	'deviance'	5	1000
	No filtrado	'deviance'	10	1000
SST-3	Filtrado	'deviance'	10	1000
	No filtrado	'deviance'	5	1000

3. Conjunto de parámetros a optimizar para el algoritmo Random Forest:
   - **bootstrap**:
     Posibles valores: [True, False]
   - **max_depth**:
     Posibles valores: [100, 200, 500, 1000]
   - **n_estimators**:
     Posibles valores: [100, 200, 500]

Tabla 6.4: Valores seleccionados en la optimización de hiperparámetros para RF

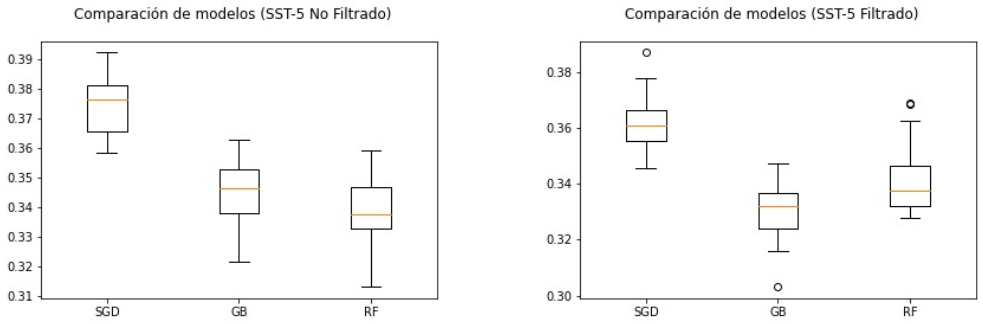
Parámetro		bootstrap	max_depth	n_estimators
SST-5	Filtrado	False	500	100
	No filtrado	False	1000	100
SST-3	Filtrado	True	1000	100
	No filtrado	False	1000	200

6.2.4. Resultados

Para este experimento se repetirá el mismo proceso que en el apartado anterior, mostrando los resultados en gráficos de caja que facilitarán ver el comportamiento de los modelos, aunque en este caso serán 5 validaciones cruzadas de 5 particiones por los altos tiempos de entrenamiento de los clasificadores GB y RF. Se evaluará la métrica de *F1 score*, que es una buena medida de la consistencia de las predicciones, con el objetivo de dar con el mejor clasificador para construir nuestro modelo.

Primero se compararon los resultados obtenidos por los clasificadores con el conjunto original de 5 categorías, en sus variantes filtrada y no filtrada, y posteriormente se comparan para el conjunto de 3 categorías.

La Figura **??** muestra el comportamiento de los modelos entrenados con conjuntos de cinco clases. El rango en el que se encuentran la mayoría de los valores se encuentra entre el 32 % y el 38 % de precisión en ambas variantes, aunque en general el conjunto que no ha sido filtrado ha obtenido resultados ligeramente mejores en términos absolutos. Los clasificadores GB y RF tienen un comportamiento comparable al ser su respuesta algo irregular e inconsistente al compararla en los dos conjuntos. En cualquier caso es evidente que es el clasificador SGD el que mantiene la mejor salida para los conjuntos de entrenamiento de 5 categorías de positividad, además de demostrar cierta consistencia en caso de ser entrenado con documentos filtrados.

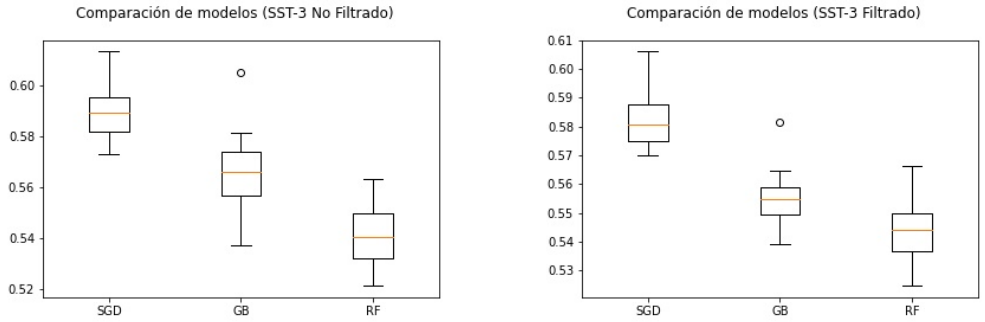


(a) SST-5 No Filtrado (b) SST-5 Filtrado

Figura 6.6: Comparación F1 score con 5 categorías

El primer aspecto a destacar del funcionamiento de los clasificadores con los conjuntos de 3 clases (Figura 6.7) es el esperable aumento en la precisión al haber menos categorías entre las que escoger. Los valores más altos, conseguidos de nuevo por el clasificador SGD, llegan a superar el 60 % y los más bajos han quedado por encima del 50 %. La precisión de todos los modelos es muy similar sin importar si los datos de entrenamiento fueron filtrados o no. Como vemos, el comportamiento de GB y RF sigue siendo algo inestable y quedan por detrás de SGD, que además demuestra una buena consistencia en ambas partes.

A partir de estos datos es lógico elegir el clasificador SGD para construir el modelo dado su buen rendimiento en el entrenamiento con cualquier tipo de conjunto. Es digno de mencionar que este ha sido también el clasificador que menos tiempo de ejecución ha



(a) SST-3 No Filtrado (b) SST-3 Filtrado

Figura 6.7: Comparación F1 score con 3 categorías

requerido en su entrenamiento, a diferencia de los modelos basados en árboles de decisión. Esto se debe a que las SVM son muy eficientes clasificando vectores como los creados en una bolsa de palabras, al representarlos como coordenadas en un espacio para generar un hiperplano de clasificación que tenga el máximo margen con los datos de entrenamiento.

Al igual que en el apartado anterior, se han recopilado en la Tabla 6.5 las métricas de precisión, exactitud y exhaustividad de una validación cruzada de 5 particiones con 80 % del volumen para entrenamiento y el 20 % restante para test:

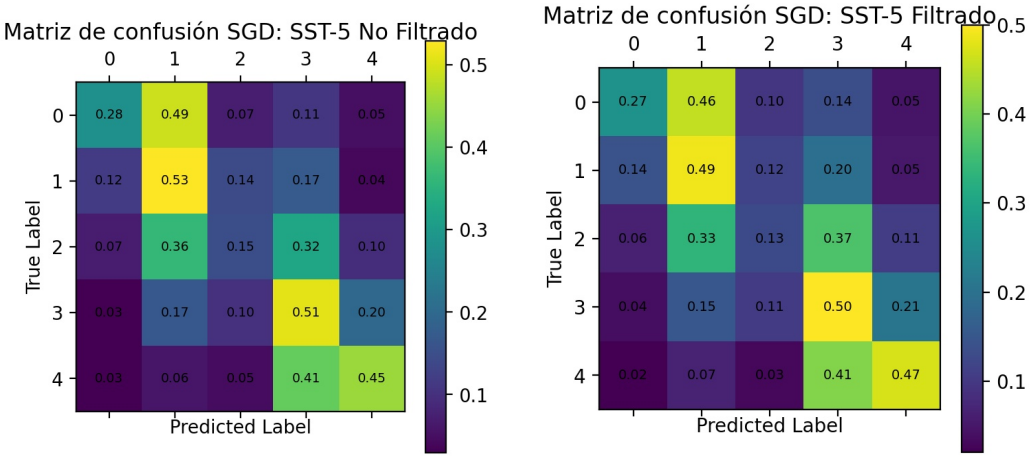
Tabla 6.5: Métricas para los diferentes clasificadores y conjuntos

Clasificadores		SGD			GB			RF		
		Precisión	Exactitud	Exhaustividad	Precisión	Exactitud	Exhaustividad	Precisión	Exactitud	Exhaustividad
SST-5	Filtrado	38.96	39.38	40.08	34.01	34.11	33.12	37.26	36.51	36.63
	No filtrado	38.05	38.74	38.97	34.8	35.28	34.64	37.93	37.16	35.98
SST-3	Filtrado	57.91	61.67	61.55	56.1	59.98	58.98	53.68	58.34	58.75
	No filtrado	58.57	61.75	61.67	57.1	60.4	59.41	53.14	57.63	57.93

Finalmente se analiza el comportamiento del modelo seleccionado entrenado con cada uno de los diferentes grupos de datos por medio de su matriz de confusión. Los valores de las etiquetas van de cero al número de clases menos uno, siendo el valor más bajo el más negativo y el más alto el de grado más positivo. En primer lugar se compararán las matrices de confusión de los entrenamientos con conjuntos con las 5 categorías originales.

En la Figura 6.8 se observa que ambas matrices de confusión son bastantes similares entre sí, ya que varían ligeramente en algunos valores pero se mantiene la misma estructura. Es posible observar que en la diagonal principal la mayor precisión se da en las clases de positividad y negatividad moderada, mientras que este valor baja en las categorías más extremas y sobre todo en la neutral. Dicha clase neutra es bastante confundida con las clases que la rodean y la mayoría de predicciones ante un texto neutro apuntarán hacia una de las dos categorías moderadas, dando lugar a una gran cantidad de falsos negativos para esta clase. Las categorías más extremas también son altamente confundidas con sus semejantes más moderadas, algo más comprensible debido a la ambigüedad en la división entre diferentes grados de positividad y negatividad.

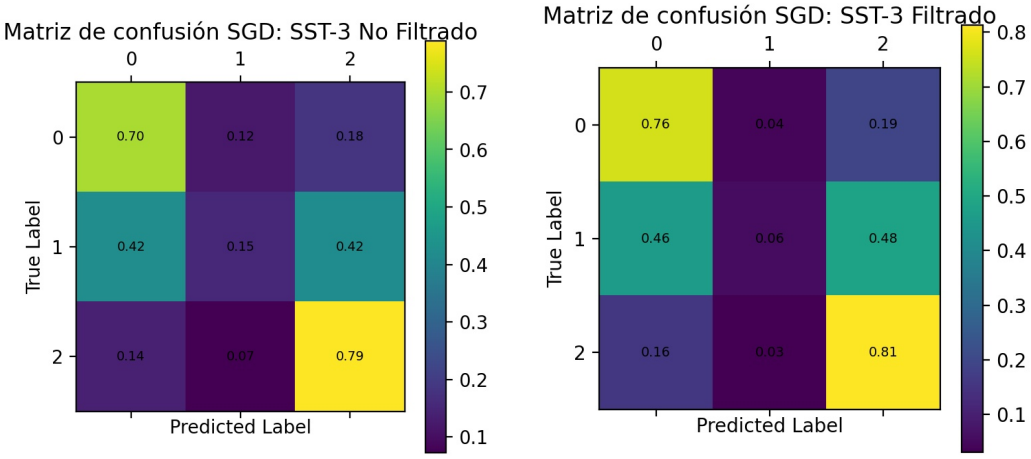
Las matrices de confusión de los modelos entrenados con datos divididos en 3 categorías



(a) SST-5 No Filtrado (b) SST-5 Filtrado

Figura 6.8: Matrices de confusión de SGD entrenado con conjuntos de 5 categorías

(Figura 6.9) obtienen mejores resultados que en el caso anterior como es lógico, aunque también se presentan los mismos problemas de confusión de clases, sobre todo focalizados en la clase neutra. El reducido número de elementos en dicha clase comparada con el volumen de las demás ha supuesto que el modelo no pueda aprender lo suficiente de una clase ya en sí bastante ambigua. Por lo tanto la gran mayoría de predicciones del modelo irán hacia una de las dos categorías mayoritarias, positiva o negativa, siendo ciertamente efectivo en la diferenciación entre ambas, aunque errando si la etiqueta verdadera se trata de la clase neutra.



(a) SST-3 No Filtrado (b) SST-3 Filtrado

Figura 6.9: Matrices de confusión de SGD entrenado con conjuntos de 3 categorías

Del análisis de las matrices de confusión se pueden extraer varias conclusiones del funcionamiento del modelo. La primera y más evidente es el pobre rendimiento en la clase neutra. Era algo esperable en el entrenamiento con 3 clases debido al menor número de

muestras, pero también ha ocurrido con el conjunto original. Esto denota la dificultad que existe para clasificar textos sin connotación positiva o negativa, tarea que probablemente sería también ambigua para un ser humano. Otro aspecto a tener en cuenta es el reducido impacto del filtrado de palabras irrelevantes en los documentos, ya que ha aumentado levemente los números del modelo entrenado con el conjunto de 3 clases pero ha ocurrido lo contrario con el de 5 clases. Se cree que la eliminación de este tipo de palabras no tiene impacto debido a que estos términos filtrados suelen ser irrelevantes y no están cargados de connotación, por lo que no permitirá clasificar mejor, simplemente ayuda a recortar el volumen de la bolsa de palabras para ahorrar recursos y tiempo de ejecución.

El modelo resultante no ha obtenido los mejores resultados en términos absolutos pero aún así la variante entrenada con 3 clases es bastante fiable discriminando entre comentarios positivos y negativos. Comparado con resultados del rendimiento de SST-5 en otros trabajos similares anteriores, el comportamiento observado es el esperable con los clasificadores y conjuntos usados, aunque otros tipos de modelo combinados con técnicas más complejas y avanzadas son capaces de obtener mejores resultados. En cuanto a la división en 3 categorías, no se han encontrado trabajos anteriores que hayan entrenado sus clasificadores reduciendo las clases del conjunto original de una forma similar, por lo que esta será la principal propuesta en este apartado. A pesar de todo se cree que el modelo del clasificador SGD entrenado con un corpus de 3 categorías es útil para reconocer textos que claramente denoten una connotación positiva o negativa y que, junto al modelo de reconocimiento de emociones en audio permitirán esclarecer la emoción y sentimiento expresados por una persona en su discurso.

6.3. Análisis de Audio

Para los experimentos de audio, se ha hecho uso de la herramienta Pyannote Audio. Esta herramienta de Open-Source está basada en Pytorch, y a su vez, en redes neuronales convolucionales. Inicialmente, se han probado los modelos pre-entrenados que proporciona.

Estos modelos son: actividad de conversación, solapamiento de interlocutores y cambio de interlocutor. En la Figura 6.10 se muestran las comparaciones entre predicción y solución correcta de un audio dentro de la base de datos AMI.

Cada uno de ellos se puede entrenar de forma individual para analizar la evolución de la pérdida que tiene, e incluso mejorar el porcentaje de acierto respecto de los modelos ya pre-entrenados mostrados anteriormente.

Puesto que el coste computacional hace que un entrenamiento completo sea desmesurado, se ha optado inicialmente por usar una parte de la base de datos, eligiendo un grupo de estos. Se compone de 28 audios para la parte de entrenamiento, 4 para la parte de desarrollo, y 4 para el test.

Figura 6.10: Referencia y predicciones de cada modelo

6.3.1. Modelos

Los modelos en los que se divide el programa son: *Overlap Speech, Speech Change Detection y Speech Activity*, y mediante las características que saca cada uno, posteriormente se realiza la diarización. Todos ellos están basados en Redes Neuronales Recurrentes (RNN), de 2 capas y *batches* de tamaño 32 y 16 debido al límite computacional a la hora del entrenamiento.

- Overlap Speech: esta tarea se centra en detectar cuando dos o más personas hablan al mismo tiempo.
- Speech Change Detection: se centra en analizar y encontrar las zonas de una conversación donde dos o más interlocutores hablan a la vez.
- Voice Activity Detection: busca detectar las zonas donde existe conversación en un audio,

Tras varios entrenamientos, los porcentajes de precisión obtenidos se muestran en la Tabla 6.6.

Tabla 6.6: Resultados de entrenamiento de modelos

Modelo	Epoch	Precisión	Horas de entrenamiento
Overlap Speech	10	68.68 %	7 horas
Speaker Change	10	67.54 %	6.5 horas
Voice Activity	15	70.8 %	6 horas %
Total	35	69 %	19,5 horas

6.3.2. Pipelines

Para realizar una diarización, pyannote sigue varios procesos, mostrados en la siguiente Figura 6.11. La primera sección se centra en eliminar los silencios y las zonas sin conversación. El cambio de interlocutor transforma los segmentos de los interlocutores de forma homogénea; la siguiente fase es agrupar esos segmentos anteriores según el interlocutor al que representen, y finalmente un reconocimiento supervisado para analizar más detalladamente esos grupos hechos e identificar a los interlocutores.

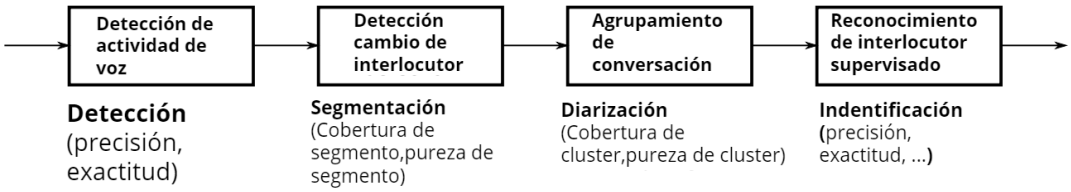


Figura 6.11: Pipeline

La precisión de actividad de conversación se evalúa mediante el *ratio de error de detección* 6.1.

$$\text{detection error rate} = (\text{false alarm} + \text{missed detection})/\textit{total} \tag{6.1}$$

donde falsa alarma indica la duración de las zonas sin conversación clasificadas como conversación, *missed detection* las zonas donde sí hay conversación, pero han sido clasificadas de forma contraria, y el total es la unidad que indica la cantidad de conversación en el archivo de referencia dentro de la base de datos. Tras un breve entrenamiento del pipeline se consigue un 5.6 % de pérdida en los 23 audios de la parte de test, que es relativo a la precisión que el programa muestra con *pyannote.metrics* en la Tabla 6.7.

Primero se extraen los datos en crudo de los modelos pre-entrenados, para después hacer una diarización completa, según esos datos. Tras de un entrenamiento indefinido, en el que la precisión varía según los parámetros que el algoritmo elija cada vuelta, se consigue una precisión y un ratio de error de los audios de la parte test. Se parte con un 65.25 % de pérdida tras una vuelta de entrenamiento. Con esta precisión se hace una evaluación final de los audios elegidos, que se muestra en la Tabla 6.8.

Tabla 6.7: Resumen análisis de Speech Activity Pipeline

Actividad de conversación	Ratio Detección Errónea	Falsa Alarma	Missed Detection
Audio EN2002a	3.21 %	1.78 %	1.43 %
Audio ES2004b	4.62 %	3.03 %	1.59 %
Audio IS1009a	5.43 %	4.78 %	0.66 %
Total	4.42 %	3.51 %	2.6 %

Tabla 6.8: Resultados de Speaker Diarization Pipeline

Diarización	Ratio de Error de Diarización	Pureza	Coverage	Ratio de Acierto	False Alarm	Missed Detection	Confusión	Vueltas
Sin restricciones	60.22 %	47.22 %	81.92 %	42.62 %	2.84 %	21.09 %	36.29 %	1
Greedy + Collar 0.2 ms	58.09 %	48.64 %	82.20 %	44.06 %	2.15 %	18.34 %	37.60 %	1
Greedy + Collar 0.5 ms	55.42 %	50.61 %	82.70 %	46.13 %	1.55 %	14.57 %	39.31 %	1
Greedy + Collar 1 ms	52.18 %	53.12 %	83.50 %	48.98 %	1.16 %	10.18 %	40.84 %	1
Sin restricciones	41.22 %	63.05 %	74.87 %	61.62 %	2.84 %	21.09 %	17.29 %	3
Greedy + Collar 0.2ms	37.69 %	65.85 %	75.10 %	64.46 %	2.15 %	18.34 %	17.20 %	3
Greedy + Collar 0.5ms	33.07 %	69.94 %	75.40 %	68.48 %	1.55 %	14.57 %	16.96 %	3
Greedy + Collar 1ms	27.57 %	75.24 %	75.88 %	73.60 %	1.16 %	10.18 %	16.23 %	3

La primera columna representa el tipo de versión utilizada para conseguir diferentes aproximaciones al resultado más óptimo. Cuando no hay restricciones incluidas, el programa opta por el uso de un método Húngaro [Kuh55], para reducir la confusión creada; sin embargo, cuando se introduce una búsqueda *greedy*, busca de forma voraz las coincidencias entre los audios y sus referencias de forma iterativa. Después de haber comparado los resultados de la Tabla **??**, y su duración de procesamiento, la diarización **Sin restricciones** tarda más, pero no sobre-estima los ratios de error, a diferencia de la búsqueda *Greedy*, la cual tarda menos, pero devuelve resultados redondeados a la baja.

Además, para conseguir una mejor precisión, es posible añadir un *collar*, con el que se eliminan los milisegundos estipulados (mitad del tiempo al inicio y mitad al final de cada interlocutor) para suavizar el fallo que pueda tener la evaluación final. Esta opción tiene un resultado de doble filo, ya que al permitir una clasificación más laxa, los porcentajes de precisión son más altos, pero menos fieles a la referencia.

La segunda columna se calcula mediante la fórmula 6.2.

$$DER = \frac{(\text{false alarm} + \text{missed detection} + \text{confusion})}{total} \tag{6.2}$$

que es similar a la fórmula de la actividad de conversación 6.1, exceptuando a la variable *confusión*, que es la duración en la que el algoritmo confunde interlocutores. A diferencia del pipeline anterior, aquí sí que se considera el solapamiento de interlocutores, que podría dar malos resultados si no existiese un módulo previo diseñado para tal tarea.

La pureza y *coverage* son dos métricas que proporcionan más detalles del comportamiento del algoritmo, expresadas de la forma 6.3 y 6.4.

$$Pureza = \frac{\sum_{cluster} max_{speaker} |cluster \bigcap speaker|}{\sum_{cluster} |cluster|} \tag{6.3}$$

$$Coverage = \frac{\sum_{speaker} max_{cluster} |speaker \bigcap cluster|}{\sum_{speaker} |speaker|} \tag{6.4}$$

Los denominadores *cluster* y *speaker* representan la duración del *cluster* hipotetizado por el algoritmo, y del interlocutor referenciado, respectivamente; y las expresiones dentro de cada sumatorio en los numeradores son la intersección de esas duraciones anteriores. En el caso de que existan resultados demasiado agrupados (*clusterizados*), la pureza aumenta, pero *coverage* disminuye, y viceversa en el que caso de los resultados tengan un menor agrupamiento.

A continuación, se muestra un análisis del *pipeline* pre-entrenado, y los porcentajes de precisión con un audio perteneciente al set de test en la Figura 6.12. El estudio comienza cargando la referencia que se tiene del audio: "subtítulos", con los tiempos de conversación e IDs.

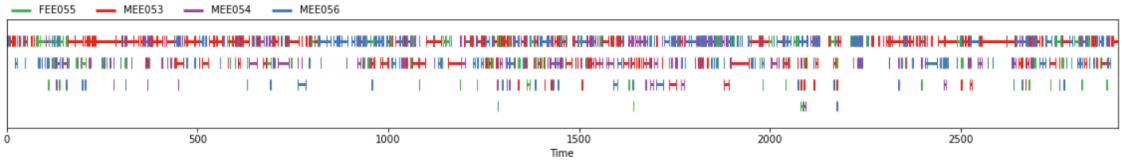


Figura 6.12: Referencia original de audio de test

Al ser tan complejo, se recorta el segmento y el trozo del audio deseado y se consigue más detalle en cuanto a las interacciones que tienen los interlocutores, y mayor facilidad a la hora del análisis visual, como aparece reflejado en la Figura 6.13.

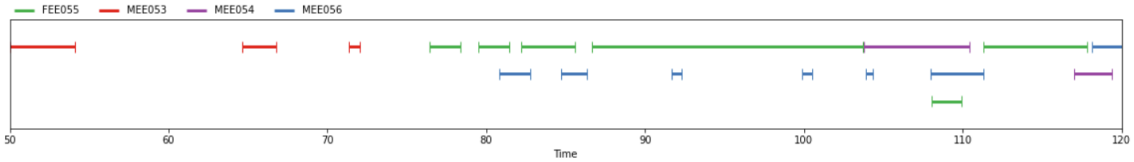


Figura 6.13: Segmento de referencia de audio de test

El resultado y la comparación con la referencia están plasmadas en la Figura 6.14. La hipótesis muestra un total de tres interlocutores (teniendo en cuenta los segmentos más grandes) y tres líneas de color rosa, azul y rojo que son confundidas con ruido de fondo. Sin entrar en detalles, a simple vista, la detección y distinción del programa de cada interlocutor son óptimas, aunque mejorables. Si se realiza una comparación más profunda (mediante *pyannote.metrics.diarization*), sacamos un 30.7 % de error de diarización. Es posible optar por una representación mejorada, y etiquetada, pero esta tendrá en cuenta todas las etiquetas de la hipótesis, incluso si son ruidos etiquetados como conversación por error, como puede ver en la tercera parte de la Figura, con las etiquetas "202", "353ƨ "280". Para terminar con este análisis, se ha añadido una representación de las zonas donde el modelo indica que hay solapamiento, siendo poco preciso, y solamente acertando en una sección (alrededor del segundo 50).

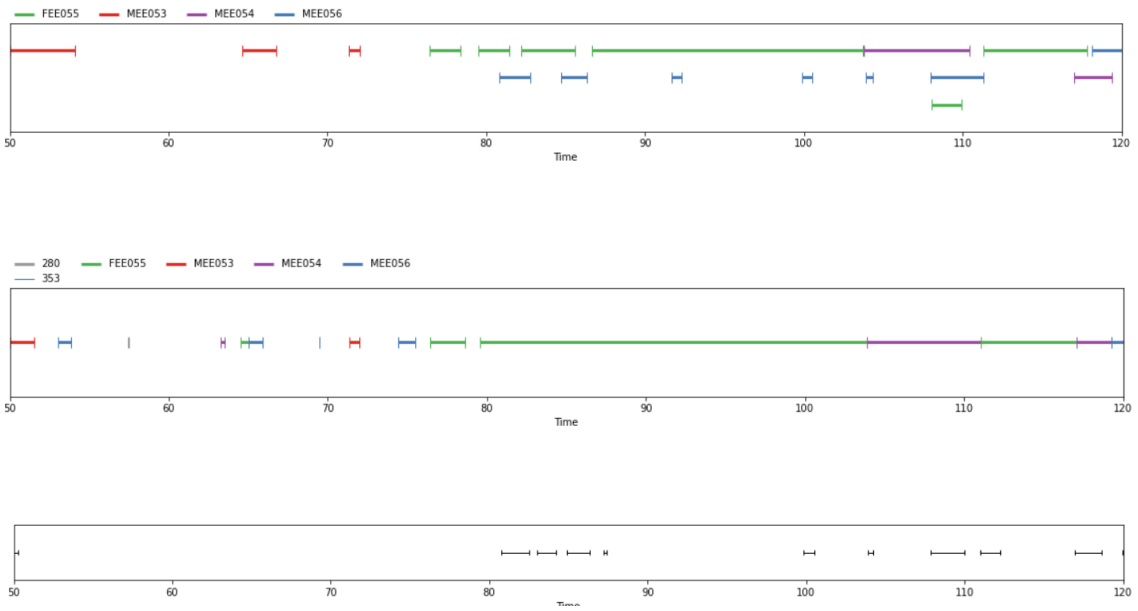


Figura 6.14: Resultados de Diarización de un Audio de la Base de Datos.

A su vez, se han comprobado la precisión de cada modelo y del pipeline final de *Speaker Diarization* con un audio aleatorio de dos personas hablando en inglés. La duración del audio es de trece segundos, y los interlocutores no se solapan nunca, sino que hablan de forma continua, dejando unas décimas de segundo de por medio. Para tener una hipótesis sobre la que apoyarse, se ha creado una referencia a mano, como se muestra en los campos referencia.[en] la Figura 6.15.

Inicialmente, el interlocutor A habla brevemente (menos de un segundo), seguido de una intervención del segundo interlocutor. La siguiente sección vuelve a hacerla el primer interlocutor, dejando menos de un segundo de silencio tras su parte, y finalizar el segundo hablante. Internamente, el audio presenta ruidos de fondo poco influyentes, y algo de música al final. *Speech Activity Detection* detecta la mayor parte de la conversación, mejor plasmada en su primer gráfico, donde las ondas sufren dos disminuciones cuando se escucha la música y varios ruidos mientras la conversación se lleva a cabo. Su precisión no es demasiado alta, como se mostró en la Tabla **??**, tras los entrenamientos. Para la siguiente parte, *Speaker Change Detection* el modelo presenta una imprecisión cuando el segundo interlocutor comienza a hablar por primera vez, dando a entender que hay cuatro cambios en esos dos segundos, cuando solamente hay un interlocutor, por ende, un intercambio. En la siguiente detección, precisa respecto a la cantidad (del interlocutor B al A), está desplazada ligeramente a la izquierda, pudiendo ser por la música y ruidos de fondo. Por último, *Overlap Speech Detection* es el menos preciso de todos, mostrando como resultados múltiples solapamientos que no existen, y que son errores que bajan considerablemente la precisión para este último.

En general, los resultados son los esperados, siendo comparados con los que se pueden obtener si se analiza un audio del dataset original. Con 6-7 horas de entrenamiento por modelo, la precisión obtenida de media a simple vista está entre el 60-65 %.

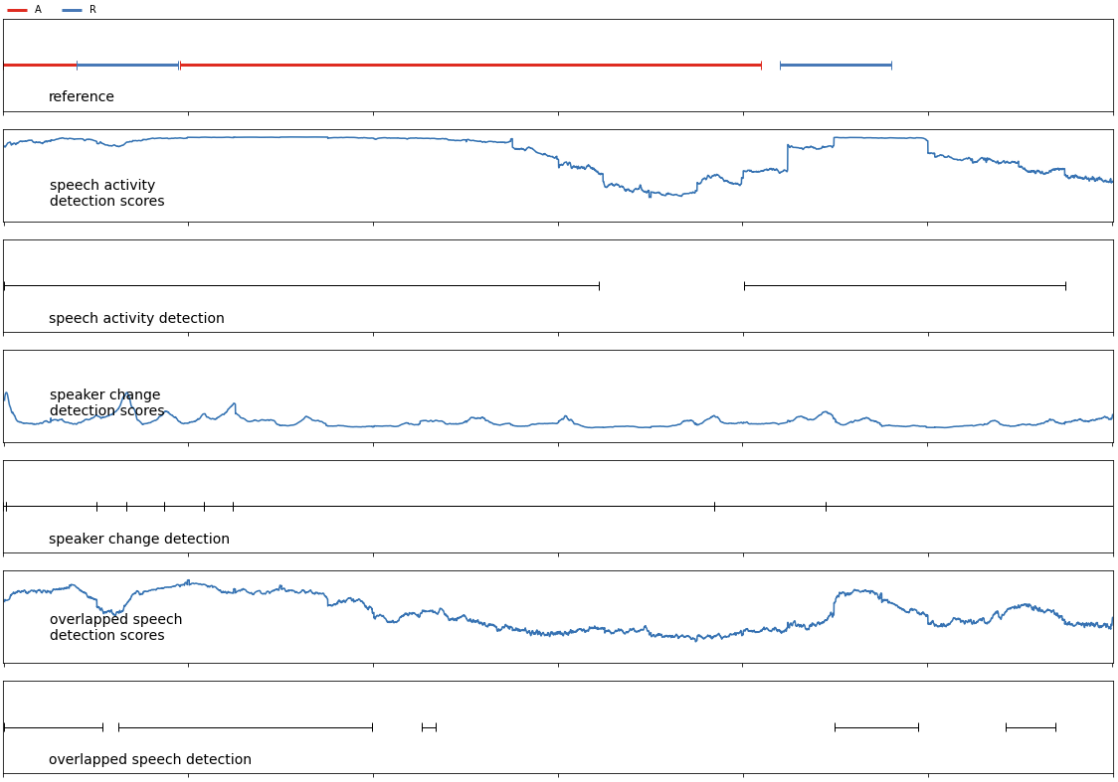


Figura 6.15: Análisis de modelos sobre un audio externo

El *pipeline* con este mismo audio ha reaccionado de forma similar, plasmado en la Figura 6.16. El resultado de *Diarization Error Rate* (DER) es de 37.1 %, debido a que no ha sido capaz de detectar a los dos interlocutores, sino más bien dos pausas, ligeramente desfasadas, en comparación con la referencia.

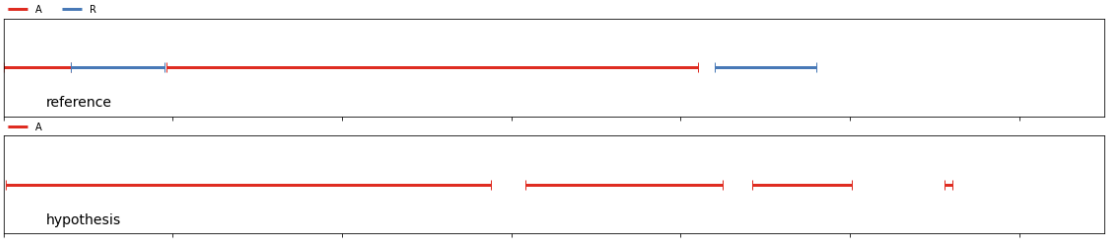


Figura 6.16: Pipeline sobre un audio externo

Capítulo 7

Conclusiones y Trabajo Futuro

Este capítulo consta de dos Secciones, 7.1 con las conclusiones de cada experimento, y 7.2, donde se redacta el trabajo que no se ha conseguido desarrollar y queda pendiente para el futuro.

7.1. Conclusiones

Este proyecto ha servido para estudiar detalladamente los procesos más usados a la hora de analizar y extraer las emociones de un audio con ayuda de técnicas de aprendizaje automático. Con este fin se ha diseñado un sistema híbrido que busca procesar el propio audio y su texto transcrito para obtener una mejor idea de las emociones y sentimientos expresados en un discurso.

Al comenzar con los campos que serían la base de este trabajo hubo múltiples dificultades a la hora de aplicar conceptos nuevos y herramientas nunca exploradas anteriormente. Con un conocimiento básico en Python se comenzó con pruebas sencillas y resultados con poco peso, para poder construir progresivamente los experimentos y modelos finales explicados en las secciones de Sistema Propuesto y Experimentos.

La primera de las conclusiones es que el reconocimiento de emociones y de información subjetiva dentro de un discurso hablado es un problema de gran complejidad, que incluso a veces los seres humanos no somos capaces de resolver, ya que la forma de expresión y de entendimiento al otro depende del contexto cultural y de la propia persona. Por ello es utópico pensar en una máquina capaz de clasificar la totalidad de muestras correctamente, aunque con el uso de las técnicas adecuadas es posible entrenar un modelo fiable y robusto que sea comparable a la comprensión de un humano, facilitando así la comunicación entre las dos partes.

La parte de Segmentación de Interlocutores se empezó con la lectura de *Speaker Diarization*, y las herramientas pertinentes. Las redes neuronales, junto con MFCCs han sido los pilares de esta investigación, usados dentro de *pyannote.audio*. Este *toolkit* es con el que más resultados se han obtenido, desechando otros programas como pyAudioAnalysis [Gia15], o VBDiarization [Pro].

El análisis y segmentación de interlocutores es complicado hasta para un programa entrenado, dejando muy lejos la perfección del 100 % de todos la herramienta elegida.

Teniendo en cuenta que en ocasiones es difícil para un oyente diferenciar a cada persona en una conversación acalorada, o con mucho ruido de fondo, para un programa lo es más. Aún así, con la información extraída de la red, y los diferentes modelos, hemos podido ver cómo hay una evolución entre el pre-entrenamiento y el entrenamiento, mejorando el análisis si el audio pertenece al *dataset*, y mostrando resultados aceptables si introducimos un audio aleatorio y desconocido al entrenamiento. Por separado, los modelos proporcionan una mejor información que el pipeline por sí solo. Ver las tres características por separado en un mismo gráfico ayuda mejor al usuario a suponer qué ocurre en el audio sin tener que escucharlo, obviamente con imperfecciones, ya que la máquina no es capaz de devolver un resultado completamente óptimo. El ratio de errores ha sido clave para poder monitorizar el avance de cada modelo, dando como media final de error un 31 % en los modelos, que dista de los porcentajes mostrados en otros artículos como [BTHVF08] o [BYC+20].

El pipeline de *Speaker Diarization* ha dado una media de un ratio de 41.22 % sin restricciones, y 27.57 % haciendo la clasificación más permisiva, con los audios del *dataset*. Estos resultados, aunque no siendo los mejores, tampoco distan excesivamente de otros como los que obtuvieron en este artículo: [TR06], con un 22.5 % los peores, y un 13 % los mejores ratios de error.

Un objetivo que no ha podido cumplirse ha sido la implementación y fusión de todas las partes, para conseguir una herramienta final que pueda devolver un análisis completo de un audio y un texto, debido a la dificultad y complejidad de la herramienta *Pyannote.audio* [BYC+20].

En cuanto al experimento de reconocimiento de emociones en audio podemos decir que en líneas generales se han obtenido unos resultados muy satisfactorios, a la altura de trabajo previo revisado. La base de datos RAVDESS nos ha aportado un buen conjunto con el que entrenar y probar nuestros modelos, y no ha sido necesaria ninguna modificación previa a los datos. Aunque nuestra clasificación constaba de ocho emociones diferentes se ha conseguido una precisión muy considerable en la mayoría de clases.

Es comprensible el extendido uso de MFCCs como método de extracción de características del audio. Podría ser recomendable usar esta técnica en experimentos con multitud de finalidades, sobre todo los que estudian cómo los humanos comprendemos el sonido. El número de vectores extraídos puede hacer variar el comportamiento de los clasificadores ya que, como observamos en el experimento 6.1, el clasificador KNN fue mejorando su rendimiento con el aumento en el número de características (72,19 % de precisión media con 50 mfccs), mientras que Random Forest cayó ligeramente en el sobreajuste (máxima precisión media de 70,52 % con 40 mfccs). Por su parte, el clasificador SGD no logró estar a la altura de los dos anteriormente mencionados, dejando claro que las máquinas de vectores de soporte no son la mejor opción en este tipo de experimentos. El empleo de la técnica de optimización de hiper-parámetros ha sido todo un acierto, ya que encontrar los parámetros a modificar y la combinación de valores óptimos para un determinado clasificador y conjunto de datos es una tarea costosa, pero puede suponer un aumento sustancial en el desempeño de los modelos.

El experimento de análisis de sentimientos ha sido más complejo de diseñar y perfeccionar, debido a la cantidad de técnicas de preprocesamiento que se pueden aplicar

al conjunto de datos. El *dataset Stanford Sentiment Treebank* (SST) con el que se ha trabajado fue diseñado para estudiar su estructura en árbol y un sistema de bolsa de palabras no es capaz de tener en cuenta algunas negaciones más complejas. Han sido las CNN las que han aprovechado mejor el potencial de este corpus en trabajo previo. Filtrar el conjunto de documentos con técnicas de PLN no ha sido especialmente determinante, ya que no se ha visto un impacto al variar de clasificador o de número de categorías. Esto probablemente se debe a que las palabras más comunes (*stopwords*) no obtienen peso en al considerar la frecuencia inversa de documento y no solo la frecuencia de término. Por lo tanto la diferencia entre los pesos de las matrices de conjunto filtrado o no filtrado debe ser mínima, dando lugar a resultados equiparables.

Reducir el número de clases ayuda, como es lógico, a aumentar la efectividad de los modelos, que realizan una buena función reconociendo si el mensaje es positivo o negativo pero no acaban de distinguir la clase neutra debido a su ambigüedad. Este análisis en tres categorías podría llegar a ser más útil en una aplicación real, además de que la tarea de distinguir entre diferentes niveles de positividad o negatividad es bastante subjetiva. Los clasificadores en este experimento se han comportado de forma similar aunque en todos los casos el mejor posicionado fue la máquina de vectores de soporte SGD. Sus tiempos de ejecución en la etapa de entrenamiento fueron considerablemente menores comparados con los de las otras dos opciones, por lo que sería la mejor elección en ese aspecto también.

7.2. Trabajo Futuro

Las principales líneas de investigación futuras que se desprenden de este trabajo son:

- Incorporar la segmentación de interlocutores a los otros dos sistemas, para así dotarles de las partes donde participa el hablante deseado como datos de entrada.
- Mejorar el rendimiento del sistema de Análisis de Sentimiento en texto para aprovechar todo el potencial que ofrece el dataset SST. Para ello sería necesario explotar la estructura en árbol que presenta en los textos a través de algoritmos de *Deep Learning* como redes neuronales convolucionales (CNN).
- Probar otros ejemplos de bases de datos de audio preparadas para el reconocimiento de emociones, preferiblemente en español, para comprobar si el cambio de idioma tiene algún impacto en el rendimiento del modelo diseñado.
- El principal aspecto a trabajar en el futuro de este proyecto es el ensamblado de los componentes ya diseñados y entrenados. De esta manera se consigue un sistema unificado para obtener una valoración general del sentimiento reflejado en un audio, considerando lo que se dice y cómo se dice. Esto sería posible gracias a una regresión logística múltiple que combinase los resultados de las secciones de reconocimiento de emociones en audio y análisis de sentimiento en texto.
- Una vez ensamblado el sistema híbrido sería necesario probar su efectividad con audios de llamadas reales y así poder dar una última valoración de las posibles aplicaciones del proyecto en un entorno real.

Capítulo 8

Contribuciones Individuales

En este capítulo, los tres integrantes del grupo explican sus conocimientos iniciales y adquiridos a lo largo del proceso de desarrollo, y las aportaciones, ya sea en cuanto a la memoria, o al código y experimentos realizados.

8.1. Mario Sánchez de Paz

Inicialmente, el conocimiento que tenía sobre los campos principales explicados en este trabajo eran escasos, conociendo las definiciones de ML y AI, junto con algunos clasificadores. *Speaker Diarization* ha sido un tema totalmente nuevo y desconocido previamente. Junto a este, *Python*, un lenguaje poco usado en la carrera, ha supuesto un desafío, con el entendimiento de la sintaxis y el uso de funciones y herramientas aportadas por librerías auxiliares. Combinar esto con el entorno de programación *Anaconda* y *Jupyter Notebook* han ayudado a conseguir los resultados requeridos.

Aportaciones

En la primera fase del proyecto, se ha encargado de investigar *Speaker Diarization*, las herramientas que lo usan y trabajos previos para entender cada concepto. Tras varias semanas, comencé con los experimentos del *toolkit pyannote.audio*, el cual se divide en modelos y pipelines (con posibilidad de entrenar o usar los que ya se proporcionan). Inicialmente, se usaron los pre-entrenados, para analizar la precisión que proporcionaban ante un audio externo aleatorio. Los resultados no fueron tan óptimos como se esperaban, ya que todo estaba acotado con la base de datos que se descargó.

Se revisaron estos modelos con mucha más atención, y la relación entre la información y las etiquetas relacionadas con esta. Después de no encontrar ninguna solución instantánea para este problema, se comenzó un entrenamiento por partes de todo el *toolkit*. Debido al coste de los entrenamientos, no hubo un avance sustancial hasta casi llegada la tercera fase del trabajo, donde se comenzó a escribir la memoria por parte de cada integrante del grupo. Un primer intento de entender cómo entrenar la aplicación, se usó la base de datos *Test*, que no serviría para los resultados oficiales, pero si para ver la evolución de los procesos.

El procesamiento con la información útil tardaba horas, y esto, junto con las limitaciones del *hardware* hizo que tardase varias semanas, para acabar obteniendo unas precisiones aceptables, que no serían tan altas como se esperaban. Tras varios intentos, se consiguieron los *epoch* suficientes para documentar el proceso y los resultados, en teoría, finales. Sin ser malos porcentajes, se quiso mejorar la media, y volvió a empezar el proceso, con la bases ya establecidas para no perder tiempo. Los modelos tenían una mejor aproximación que el *pipeline*, así que se llegó al consenso de mostrar todas las versiones de este último, para observar cómo, cambiando la tolerancia de fallos, la precisión podría mejorar significativamente.

En esta tercera fase, he añadido información en cada sección de la memoria a la vez que conseguía los porcentajes mejorados necesarios para la parte de experimentos, y análisis de interlocutores. Mi investigación me permitió también escribir el capítulo relacionado con su tema del trabajo: Segmentación de interlocutores, su historia, y evolución tecnológica, y finalmente, los algoritmos y herramientas más usadas en la actualidad. Junto con este trabajo principal, también he aportado en el resto de experimentos, colaborando a la hora de introducir la información, con la construcción de tablas pertenecientes a otras secciones, y decisiones sobre cómo organizar varios capítulos ajenos a mi tarea inicial. Dentro de los experimentos de Reconocimiento de Emociones y Análisis de Sentimientos, cuyo desarrollador ha sido Francisco Quiñones, he ayudado en la compilación y modificación del código durante la segunda y tercera fase.

Finalmente, en las últimas semanas, se me asignaron varias tareas para perfeccionar el documento, y poder cumplir con los estándares establecidos: tablas según unas pautas determinadas, código de las imágenes con *width* como parámetro interno, y corrección de títulos y uso de mayúsculas en este y otros casos, como en los pies de imágenes e incio de capítulos y sub-secciones. Múltiples imágenes tuvieron que ser hechas de nuevo en Español, además de cambiar las etiquetas y referencias. También he modificado partes del código para reducir los errores, y mejorar el estilo del documento final, referencias de la bibliografía, que no cumplían con los estilos correctos de *LAtex*. Como última tarea asignada, me he encargado de traducir completamente los capítulos: *Abstract, Introduction* and *Conclusions and Future Work*.

8.2. Francisco Quiñones Bellón

Mi interés sobre Inteligencia Artificial y Aprendizaje Automático siempre ha estado presente desde mi entrada en la facultad, pero apenas un par de asignaturas de las que he cursado han profundizado sobre estos temas. Es por ello que mi conocimiento sobre técnicas de *Machine Learning* antes de comenzar el proyecto era limitado, aunque conocía conceptos del aprendizaje automático, supervisado y no supervisado, así como algunas técnicas de procesamiento de lenguaje natural útiles para el análisis de sentimientos. Mi experiencia previa con Python era escasa, habiendo practicado solamente con la librería ScikitLearn y algunas de sus funciones más representativas. Sin embargo, ha resultado sencillo comprender el funcionamiento del lenguaje y la mayoría de librerías necesarias para el proyecto debido a la gran cantidad de documentación y tutoriales disponibles en

la red.

Aportaciones

En la fase de investigación comencé explorando el marco teórico y los experimentos más relevantes en el campo del reconocimiento de emociones en audio, así como las principales técnicas y bases de datos empleadas. Profundicé en los conceptos del aprendizaje automático y el funcionamiento de los algoritmos más usados en trabajos similares. Poco a poco comencé a implementar algunas pruebas basadas en los artículos revisados y a trabajar en el diseño de los experimentos. Tanto el experimento de reconocimiento de emociones en audio como el de análisis de sentimiento en texto siguieron un plan de desarrollo similar, en el que se fueron implementando técnicas de preprocesamiento de datos, selección de características y de optimización de parámetros para ir comparando y mejorando los resultados obtenidos.

En el caso del experimento de reconocimiento de emociones en audio se usaron técnicas generales de Machine Learning, sin ser necesaria una etapa de preprocesamiento de los datos. La librería Sklearn fue la encargada de aportar la práctica totalidad de métodos necesarios a lo largo del proceso de la información, así como la representación de los resultados. La herramienta de extracción de vectores MFCC de Librosa fue una perfecta elección para obtener las características de los audios, ya que estos podían ser introducidos directamente a los clasificadores tras partir los datos en los conjuntos de entrenamiento y test.

Al igual que en el experimento anterior, la mayoría de utilidades relacionadas con el aprendizaje automático han sido proporcionadas por Sklearn, siendo la mayor diferencia la necesidad de aplicar un prepocesamiento a los datos en crudo para poder trabajar de manera efectiva con ellos. La librería que consideré para cubrir el filtrado de las palabras es la llamada *NLTK*, un kit consistente en herramientas para el procesamiento de lenguaje natural.

Las secciones que redacté dentro de esta memoria de forma íntegra son las relacionadas con los experimentos que fueron realizados por mi persona: su diseño, explicado en el sistema propuesto, el procedimiento realizado, sus respectivos resultados y las conclusiones obtenidas del comportamiento de los modelos. Para comprender el funcionamiento de los algoritmos que estaba utilizando tuve que profundizar sobre los conceptos de la AI y el AA, reflejando lo aprendido en el marco teórico. También investigué sobre secciones más específicas de cada experimento, como el proceso de extracción de los coeficientes de Mel paso a paso, los diferentes métodos de filtrado de palabras o la información que aportan la frecuencia de término y la frecuencia inversa de documento.

También he colaborado junto a mis compañeros en la redacción del resumen, la introducción, conclusiones generales y trabajo futuro, así como sus respectivas traducciones al inglés. En el apartado de estado del arte me centré en investigar acerca de anteriores experimentos de reconocimiento de emociones en audio y de las posibles aproximaciones para la combinación con el experimento de análisis de sentimientos en texto en un sistema híbrido unificado.

Además, he contribuido a dar forma a la estructura del documento, separando los

contenidos entre los diferentes capítulos y secciones, intentando mantener la coherencia y cohesión de las ideas expresadas. Además de esta labor he colaborado en el experimento de *Speaker Diarization* para intentar solventar los problemas de los altos costes de ejecución aportando mi hardware.

Tras el desarrollo de la mayor parte del contenido mi función pasó a ser la corrección de los errores que nos indicaban nuestros tutores, para adecuarnos a los requisitos oficiales y dar una mejor presentación a nuestro trabajo. En esta fase fue necesario cambiar bloques de texto a una sección más adecuada, revisar la formulación y estructura de algunas frases, referenciar apropiadamente tablas y figuras, elaborar nuestros propios esquemas en español para reemplazar las imágenes colocadas provisionalmente, entre otras tareas de perfeccionamiento formal.

8.3. Emilio Perera Fernández de Pedro

Antes de comenzar este TFG no había indagado en exceso en el campo del análisis de llamadas y es por ello por lo que me alegro de haberlo escogido, ya que ha resultado ser un tema muy interesante y de una gran profundidad y futuro. A pesar de que Python es uno de los lenguajes más conocidos y que se hace uso de él en varias optativas de la carrera, al haber cursado otras diferentes, no lo había utilizado nunca y fue para este trabajo cuando comencé a usarlo.

Aportaciones

Al principio, me encargué de recopilar y hacer resúmenes de toda la información posible acerca del PLN y, aunque en esta primera fase la búsqueda fuera más general e iba guardando todo documento que pudiera llegar a resultar de interés para el TFG, me centré sobre todo en aquella que estuviera preparada para el español, ya que, a pesar de que cada vez hay más, la gran mayoría sigue estando en inglés. Estos son, entre otros, algunos de los principales temas sobre los que busqué y recopilé información del PLN:

- **Historia:** Comienzos y avances del PLN hasta la actualidad.
- **Herramientas:** En este apartado busqué y revisé el funcionamiento de herramientas PLN gratuitas que estuvieran disponibles para su uso en español y para Python, ya que este era el lenguaje de programación que estábamos utilizando para este TFG. Y finalmente el análisis lo realicé de: Freeling, NLTK, Pattern, SPACY y Stanza. También probé algunas versiones online de estas herramientas como la web TextAnalysisOnline, donde se encuentran Spacy, Pattern y NLTK con utilidades como tokenización, lematización, etiquetado o segmentación de frases entre otras (varían para cada herramienta). En el caso de esta web, solo detectan bien las frases en inglés y hay ciertas utilidades de alguna de las herramientas que se encuentran caídas. Otra versión online de estas es la demo de Freeling ofrecida por la *Universidad Politécnica de Cataluña*, ya que fue el grupo de investigación de PLN de esta universidad la que se encargó de desarrollarla. Esta sí se encuentra disponible en

varios idiomas entre los que se encuentra el español. Estas versiones online sirven más bien de demos, como bien se indica en la de Freeling, ya que su uso se encuentra restringido, y aunque sí se pueden probar largos textos, el uso de grandes bases de datos no está permitido.

- **Servicios:** Los servicios PLN son cada vez más demandados y, por lo tanto, muchas de las grandes compañías cuentan con los suyos propios. Por lo que en esta sección busqué algunos de los más utilizados que son los de: Google, Amazon, Microsoft e IBM, todos ellos de pago, pero con versiones gratuitas. También indagué acerca de CMU Sphinx, que es totalmente gratuito al tratarse de un proyecto público que se encuentra en GitHub.

- **Técnicas de preprocesamiento de los textos.** Técnicas más habituales para facilitar el análisis de los textos.

- **Tipos de análisis de textos:** Información sobre los análisis léxico, sintáctico y semántico de los textos y acerca de las fases de un compilador.

- **Adaptar a español programas preparados para otros idiomas:** Como, por ejemplo, SimpleNLG, que al ser muy utilizada y no encontrarse disponible en castellano, hay bastantes documentos acerca de cómo prepararla para su uso en esta lengua.

- **Ámbitos en los que se utiliza:** Cada vez más sectores utilizan herramientas PLN, entre los que se encuentran:

  - *Sector médico:* Por ejemplo, para detectar síntomas de depresión en llamadas de pacientes o detectar signos de dolor o angustia para priorizar las más urgentes.
  - *Call centers:* Se hace uso de estas sobre todo para el control de calidad de las llamadas atendidas tanto por humanos como por máquinas.
  - *Campo musical:* Para optimizar la recomendación automática de música.
  - *Ámbito policial y judicial español:* Uno de los sectores donde cada vez se utilizan más herramientas PLN, además de que en este campo su uso es muy delicado, ya que por ejemplo un error en una transcripción dónde se atribuyen frases al interlocutor que no corresponde puede acarrear graves consecuencias legales. Sin embargo, utilizar estas herramientas facilita muchos procesos de este ámbito. Además, el enfoque en este campo no está tan centrado en la investigación como sí ocurre en otros, si no que en este se centra totalmente en la practicidad.

El ver cómo se adapta el uso del PLN en cada uno de estos sectores me resultó muy interesante y demuestra la infinidad de utilidades que tiene y la cantidad de distintos sectores en los que se puede emplear, por lo que queda apuntado para posible trabajo futuro el hacer más hincapié en estos temas.

Más adelante, comencé a seleccionar y redactar la información más relevante para el trabajo a medida que este iba tomando forma, colaborando en la parte de teoría de otros

apartados, pero centrándome sobre todo en el PLN e intentando profundizar en todo aquello que se encontrara disponible para su uso en castellano.

También realicé distintas tablas resumen y de ejemplos para sintetizar y clarificar algunas partes.

Resumen del Trabajo en Inglés

Capítulo 9

Introduction

For the communication, human beings use the voice, and with it, we add non-verbal information that is feeling-dependant on what we would like to transmit to the rest of speakers. The emotions that we feel condition our live and relationships. It is possible to say that emotions control humans, due to the fact that they have an influence on our decisions, actions, and perception itself of the reality. For that, throughout history an amount of attempts have aimed to classify and categorize those emotions, based on multiple aspects. In 1983, Paul Ekman, psychologist, [Ekm83], identified, after studying human facial expression, six basic emotion types: happiness, sadness, fear, ire, displeasure and suprise. This classification was used as the ground by other psychologists that thought that there were more emotions and, that some of them appear as the result of the combination of different intensities and basic emotions.

That is the case of Robert Plutchik, that created and explained the Emotions Wheel on his articles [Plu80] y [Plu91] since 1980. It is, probably, the most known and used emotions classification by specialists nowadays. In its taxonomy, there are four main opposite emotion pairs, which combined (with diverse intensities), give secondary ones, making the next Figure 9.1.

Another model, proposed by Orthony, Clore y Collins[GLO13], on which it is asserted: "the intensity of a emotions and what originates it are aspects that Theory of Emotions has to examine". In spite of this, because of the classification of this model, with 22 emotions, its used inside preprocessing phase has been avoided. The computational cost was excessive (more emotions to classify, more information needed, and more time to process precisely all results).

Human beings are capable of feeling, thanks to empathy, emotions suggested by the speech entonation of another speaker. It is something that we do naturally, without thinking about sound-wave physical features. Although, a study about voice changes when transmiting different emotions affirms that they can be useful, as it is explained on the next table 9.1

Human speech contains a lot of linguistic information, that is explicitly expressed, and para-linguistic that is implicitly given. Linguistic information identifies cualitative patterns that the speaker has articulated (words and syllabels), while para-linguistic information is measured by cuantitive features that describe transmitted physical sound wave.

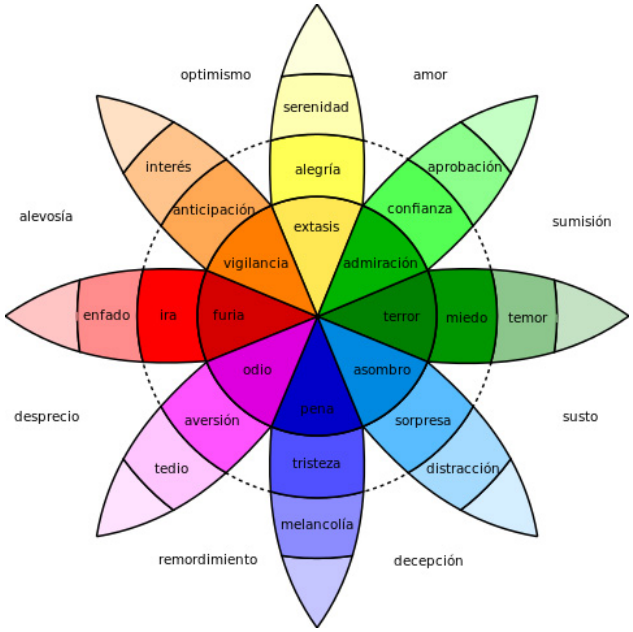


Figura 9.1: Emotions Wheel Scheme, Robert Plutchik. [Plu91]

Tabla 9.1: Emotion Features explained with the Voice.

	Ire	**Happiness**	**Sadness**	**Fear**	**displeasure**
Ratio (Speed)	Slightly faster	Fast or slow	Slightly slower	Very fast	Much faster
Medium Tone	Much higher	Very high	Slightly lower	Much higher	Much lower
Tone Range	Very wide	Very wide	Slightly narrower	Very wide	Slightly more wide
Intensity	High	High	Low	Normal	Low
Voice Quality	From the chest, with breathing	With breathings strident	Resonant	Irregular Pronunciation	From the chest, moaning
Tone Changes	Abrupt	Soft, upwards inflexions	Downwards inflexions	Normal	Wide, Downwards inflexions
Articulation	Tense	Normal	Jammed	Precise	Normal

Besides analysing non-verbal transmitted information by the voice, it is possible to make a machine to process text content to estimate its sentiment and classify documents based on the positive or negative connotation written on them. This field is called Sentiment Analysis, and for extracting subjective document information, it makes use of Natural Language Processing tools. To perform a complete audio speech processing, a model has to take into account the next seven principles, proposed on [TA19]:

- Pronunciation phonetic level.
- Words morphological level, that contain a meaning with prefixes and sufixes.
- Lexic meaning of words and sentence structures.
- Syntatic gramatic level on sentences.
- Semantic level of sentence meaning.

- Speech level, centered on different text parts.
- Pragmatic level to analyse the external document knowledge.

Communication between machine and human is complex, and until now emotions have not played an important role, because nowadays machines can recognise "what it is said.or "who says it", as we can observe on multiple voice interfaces that can ease our lives. If a machine is also capable of understading "how it is said", we can make the human-machine interaction much nicer and more natural. This could be specialy useful for scenarios such psychiatric diagnosis, helping diverse functional people, or lie detection, and more other fields. On this project, a complete audio analysis has been carried out, including sentiment analysis. from NLP techniques, Speaker Diarization, and Emotion Recognition. We think that the group of this three classifications will be able to provide a better emotions and feelings point of view, expressed by a person in some audio. In the Figure 9.2, it is shown the emotion recognition process.

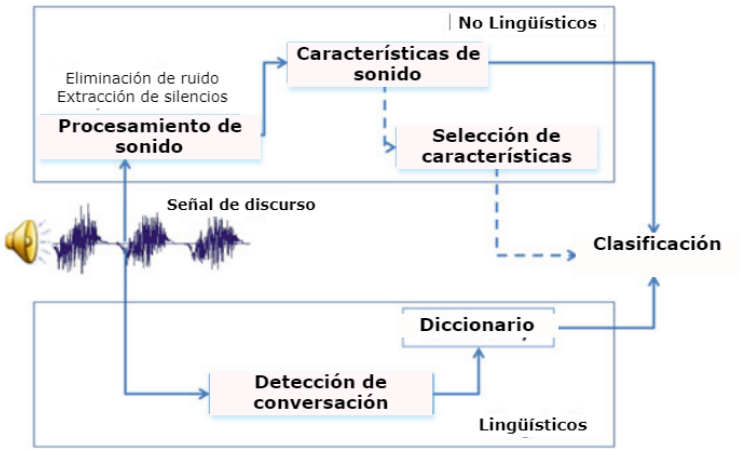


Figura 9.2: Emotion Recognition Process [Ana12]

Emotions expressed on Social Network, audio, or conversations are processed unconsciously, or consciously on the human case, but machines, even being incredibly faster than us, require, previously, more information and training before getting any kind of conclusion. The aim of this analysis is detect emotions inside any sphere, with multiple purposes, like marketing or similar services. Social Networks reach every corner of the world, and inside of them, every user publish their thoughts, experiences and opinions, making possible the sentiment analysis. The techniques used by Natural Language Processing are based on two models:

1. **Linguistic Models**, based on grammar and linguistic resources.

2. **Probabilistic models**, based on noted data and Machine Learning. At the same time, they can be divided in **Lexic Analysis** (certain words connotation and weight) and **more complex analysis**, with which it is given importance to structures that can express emotions.

Inside this technology field, there are two sub-fields when categorizing sentiments to improve the algorithm predictions. The categoric model associates data with labels, and it is thanks to this that emotions can be more tanginble for a computer, but as inconvenience, it gives a major labels lack that cover the wide emotions amount that can be categorized. In addition, more label amoutn, more need there is for a bigger database and computational cost. The dimensional model represent the data, based on the cuantitative measures group, and it is able to detect more subtle changes inside emotions that would have gone unnoticed for the categorical model.

A similar investigation related to a part of this project was done by Chuen-Hsien Wu [CHW06], where it postulates another point of view applicable to Emotion Recognition, by text content. He divided the study in two phases: training and test, and at the same time, he analysed the Corpus Emotions connected with the domain component (dependant and independant). The independant domain component is the semantic part .ºbtain", while the dependant component is the "something that is benefitial"semantic part.

In recent years, customer services have increased, with the improvement of companies based on the Internet. Until very recently, more importance was given to the product quality and administration, than to customer service. In agreement with an investigation made by advisory *Lee Resources*, 80 % of companies confirm that they offer good quality custormer service, while only a 8 % of customers agree with that number. SalesForce, a software company, that has products with Aritificial Intelligence, marketing, and communities paperwork, published the next information:

- 70 % of online services are based on the treat that the client received, beyond the product cost.
- If during the attendance process, it is possible to solve a problem, 70 % of the clients will buy a product again.
- It is and estimation, that 90 % of internet bussiness will use Social Networks will afford a better cusatomer service.
- Clients spend between a 20 and 40 % more when companies are more involved with their problems and answer through Social Networks.

Recently a call investigation has been made ([Seh18]), by means of Automatic Sentiment Analysis, following the structure of the Figure 9.3.

The objective of this investigation is to detect the different techniques applicable to this field to observe its precision and accuracy. We have used dual tone adn multi-frequency signals, Automatic Speech Recognition Systems combined with IVR, various classifiers (Bayes, Entropy and Boosted Trees), and datasets from virtual customer service, and cloud computation. 100 people have been interviewed, and the statistics provided a 60 % of users were happy about the ASR service, while a 25 % rejected it. At the same time, a huge majority felt a communication gap with the IVR system.

To reach this point, information was processed by a multinomial Bayes words-bag. Agter obtaining the information, and changing it to text, tokenized and grouped by weights, similiar to Natural Language Processing processes. From here, the system learns

to classify with a better accuracy, knowing if the customer is or not satisfied, because there are disregarded parts by text, because of the sentiment weight that it might have.

Después de analizar los trabajos citados, se descubrió la inexistencia de un artículo, o experimento que introdujese las múltiples tecnologías que han sido asignadas para este, y es por eso que desarrollar cada parte ha sido un desafío desde el principio, pero con resultados tangibles en cada campo.

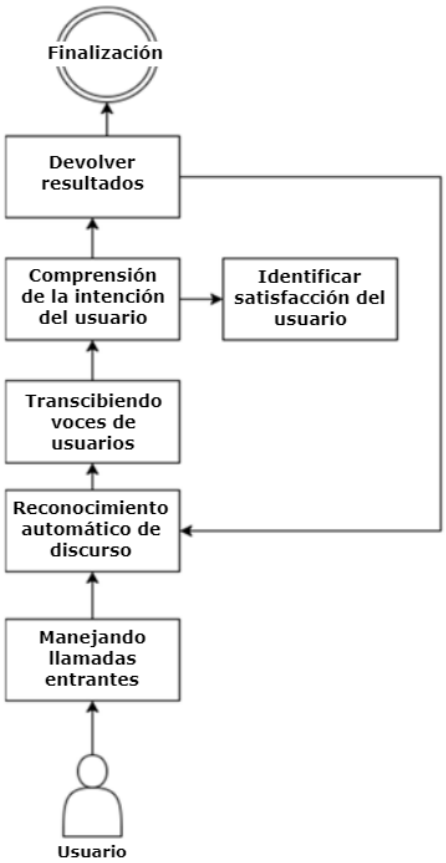


Figura 9.3: ASR Structure Combined With IVR System.[Seh18]

9.1. Motivación

As it was mentioned before, the main goal of this project is to train and improve a set of models that detect emotions, and who produces them inside an audio. The majority of previous works that has been read, is only focused on Emotion Recognition on audio through the extraction of MFC, but we think that bringing up a hybrid system that intregrates it with sentiments on text can improve significantly its reliability. For that, we need to design and train various models that work on each phase.

The first model to design is *Speaker Diarization*, which is the part of this system that checks audios to detect and separate speakers inside of it. The purpose is to çut.[a]nd transmit interesting audio parts to the other two processes to achieve a complete analysis.

Emotion Recognition phase will receive those segmented audios, from where features will be extracted to create a model capable of predict emotions from a random audio.

Concurrently, segmented audios are introduced into the audio-to-text transcription tool, that will give back the input data for the Sentiment Analysis system. This part will process transcripted texts from audios, creating a model whose function will be predicting the message positivity.

The mix of both parts could show up another innovative point of view for the Emotion Recognition, as well as more reliability and robustness. We think that other knowledge fields, such as medicine, psychology, or forensic science could find useful applications from our experiments results.

9.2. Objectives

Since the beginning, small guidelines have helped the progress, which was slow but progressive. Initially, the group was organized, to investigate each field that this project is based on, being the first goal: learn the basics and continue from there. After multiple weeks of reading and recopiling data, tools that were going to be used were exposed and studied, for learning how to obtain results for each section. We can conclude the objectives with the next list:

1. Investigate and learn the basics of *Machine Learning* and Artificial Intelligence, with its multiple techniques to extract features, pre-processing and classification.
2. Decide which tools and programmes would be necessary for the models.
3. Improve models' performance with its parameters' optimization.
4. Optimize results and performance again of each model.
5. Extract conclusions about created systems and its possible combination on future work.

9.3. Memory Structure

This project starts with and explanation of our emotions classification and its voice expression, to describe technologies responsible for its analysis afterwards, checking the history and explaining its tools. Once that the theory framework was finished, designs, results and conclusions of developed experiments will be shown. After this introduction, coming Chapters will present following the next structure:
In Chapter 2 we make an approach to a more theoretical framework related with Artificial Intellingence where main concepts are explained, along with the particularities inside Speaker Diarization and Emotions Recognition.
In the next Chapter, number 3, Audio Sentiment Analysis experiment and Text Sentiment Analysis are explained with much more detail, describing the Natural Processing Language techniques, offering and comparing other open-source tools.

In Chapter 4 State of Art is reviewed, with the previous revised work and the more relevant aproximantions results, donde on each experiment. Datasets, classifiers and features extractors are also reviewed.
Chapter refCapitulo5 presents our contributions, based on developed experimetns, that will give as a result the searched models after training. Here we exposed the decisions and considerations that were taken during the design process of each phase, along with the used tools. Experiments are explained with more detailed in Chapter 6, were the followed process is clarified, with used data by classifiers, to train and obtain the results. Those are captured on tables and images for its comparation and analysis that the reader can make.
Extracted conclusions from the investigation and improvement of each experiment correspond to Chapter 7, where we also explain the non-achieved goals, and that are pending for the future.
Chapter 8 gathers individual work from eahc member of the group.
Finally, Chapters 9 and 10 present Introduction and Conclusions in English.

Capítulo 10

Conclusions and Future Work

This chapter is compounded by two parts, 10.1 with each experiment conclusion, and 10.2, where it is explained the part of this project that could not be achieved and it is pending as future work.

10.1. Conclusions

This project has been useful to study in detail the most used process for analysing and extracting audio emotions with tecnniques such as ML. With this purpose, a hybrid system has been designed, which aims to proccess that audio, and the transcripted text to obtain a better idea about the emotions and feelings emboided inside the speech.

At the beginning, there were some difficulties related to the new scientific terms and unknown tools, never before explored. Along the basic Python knowledge, intial tests started, collecting simple results, with the only purpose of being able to progresively build experiments and final models, that are explained on 5 and 6.

The first conclusion is connected with emotions and subjective information recognition: inside the speech, they generate a complex problem, that sometimes, even the humans are not able to solve, due to our way to express and understand others, that depends on cultural context and the person himself. Because of that, it is utopic to think that a machine has the ability to classify the data correctly, although it is possible to train a model with adecuate models, approaching it to the human comprenhension, easing communication between both parts.

The Speaker Diarization part started with reads about the term itself, and tools that are based on it. Neural Networks, along with MFCCs have been the main support on this investigation, used inside pyannote.audio. This toolkit has provided more results than others, which were discarded: VBDiarization [Pro].

Speaker Analysis and its Segmentation is complex, even for a trained program, leaving back the perfect score of 100 % from the selected tool. Taking into account that in some occasions is difficult for human hearing to diferentitate each speaker inside a heated conversation, for a machine is much more than just listening and dividing. Despite of that, with the extracted information from the network, and multiple models, we have been able to see how the evolve, from the pre-trained models to the training ones, improving the

analysis from an database audio, and showing optimal results when the audio is external.

Separated, the models provide more detailed information that the pipeline by itself. Seeing the three features, separated, inside of a graphic, helps the user to guess what happens inside the audio, without having to hear it, obviously, with imperfections, because of the machine results, that will not be perfect. The error rate has been another key to monitor the improvements of each model, giving a final average of 31 %, not far away from articles like [BTHVF08] or [BYC+20].

Speaker Diarization pipeline has given an error average of 41.22 %, without restrictions, and 27.57 % applying a more permissive classification, with the dataset audios. These results, despite of not being the best, are not that far away from the ones obtained on this article: [TR06], with a 22.5 % the worst, and a 13 % the best error ratios.

A goal, that has not been achieved, has been the implementation and combination of all parts, to Un objetivo que no ha podido cumplirse ha sido la implementación y fusión de todas las partes, para conseguir una herramienta final que pueda devolver un análisis completo de un audio y un texto, debido a la dificultad y complejidad de la herramienta *Pyannote.audio* [BYC+20].

With the emotions recognition, we can conclude that the results have been very satisfactory, at the same level of previous read articles. RAVDESS Database has provided a good amount of information for training and test our models, and no modification has been needed beforehand. Even though our classification was composed by eight emotions, a high accuracy was reached on the classes majority. It is understadable the extensive use of MFCCs as audio features extraction method. It would be recommended to use this technique in experiments with multiple objectives, specifically those that study how humans understand sound. Extracted vector number can vary the classificatiors behaviour. We can see that on the experiment 6.1, and KNN improved its performance with the feature increase (72.19 % as average with 50 MFFCs), while RF slightly decreased in over-fitting (maximun accuracy of 70.52 % with 40 MFCCs). Finally, SGD did not reach the same level as previous both did, concluding that Support Vector Machines are not the best choice for this type of experiments.

The usage of the optimization technique of hyper-parameters was the right answer, because finding parameters to modify and the optimal value combination for a determined classifier and dataser, is a expensive task, that can mean a substantial increase on the models tasks.

Sentiment Analysis experiment has had more complexity when designing and perfectioning, because of the big amount of pre-processing techniques that can be applied to the data. The dataset SST that we have worked with, was designed for studying its tree structure and a words bag system is not capable of having into account more complex negations. CNN were able to get more out of this corpus.

Screening the amount of PLN documents was not determinant, due to that there was not a substancial difference when varying the categories ammount, and classifiers. This is probably because the most common words or stop-words do not have the same weight when considering the inverse frequence of the document, and not only the term frequency. Therefore, the difference between matrix filtrated weights and none-filtrated weights has

to be minimun, giving as a result comparable numbers. Reducing the amount of classes helps to increase the models effectivity, that work correctly, recognising if the message is positive or negative, but in the end, they dont differ the neutral class, because of the ambiguity. This three categories analysis could be more useful on a real application, besides the task of distinguishing different positivity or negativity levels, which is very subjective. Classifiers on this experiment have had a similar behaviour, although in all cases, the best positioned was the Support Vector Machine SGD. Its execution times on the training phase, they were considerably smaller compared with the other two options, so it would be the best choice on that aspect too.

10.2. Future Work

The main tasks that remain unfinished after this project and establish the future work are:

- Incorporate the speaking dearization system to the others, which will take the segments where the desired speaker participates as their input data.
- Improve the performance of the Text Sentiment Analysis to take advantage of all the potencial that SST has to offer. To do this it would be necessary to exploit the tree-based structure the documents present, using Deep Learning algorithms like convolutional neural networks.
- Test other audio datasets designed for emotion recognition, preferably in spanish language, to make sure the language change does not make an impact on the trained model's performance.
- The main goal is to assemble all the component already designed and trained. This way, we could get a unified system that would be used for obtaining a general valuation from the sentiment on the audio, considering what it is said, and how it is said. This would be possible thanks to a multiple logitic regression that combines Emotion Recognition section, and Sentiment Text Analysis section.
- Once our hybrid system is assembled it would be necessary to test its effectivity with real call audios to develope a definitive assessment about the possible applications of this proyect in a plausible environment.

Bibliografía

[ABB+39] A. Averbuch, L. Bahl, R. Bakis, P. Brown, A. Cole, G. Daggett, S. Das, K. Davies, S. DeGennaro, P. de Souza, E. Epstein, D. Fraleigh, F. Jelinek, S. Katz, B. Lewis, R. Mercer, A. Nadas, D. Nahamoo, M. Picheny, G. Shichman, and P. Spinelli. n ibm pc based large-vocabulary isolated-utterance speech recognizer. *Journal of the Franklin Institute*, 227(6):739–764, 1939.

[Ana12] Iliou T. Giannoukos Anagnostopoulos, CN. Features and classifiers for emotion recognition from speech: a survey from 2000 to 2011. *Springer Science+Business Media Dordrech*, pages 155–177, Noviembre 2012.

[AR52] M. M. Astrahan and N. Rochester. The logical organization of the new ibm scientific calculator. In *Proceedings of the 1952 ACM National Meeting (Pittsburgh)*, ACM '52, page 79–83, New York, NY, USA, 1952. Association for Computing Machinery.

[AZS17] DASC Alu, Elteto Zoltan, and Ioan Cristian Stoica. Voice based emotion recognition with convolutional neural networks for companion robots. *Science and Technology*, 20(3):222–240, 2017.

[BBL+08] Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette N Chang, Sungbok Lee, and Shrikanth S Narayanan. Iemocap: Interactive emotional dyadic motion capture database. *Language resources and evaluation*, 42(4):335–359, 2008.

[BKL09] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. .°'Reilly Media, Inc.", 2009.

[BSN15] Jasmine Bhaskar, K Sruthi, and Prema Nedungadi. Hybrid approach for emotion classification of audio conversation based on text and speech mining. *Procedia Computer Science*, 46:635–643, 2015.

[BTHVF08] Kofi Boakye, Beatriz Trueba-Hornero, Oriol Vinyals, and Gerald Friedland. Overlapped speech detection for improved speaker diarization in multiparty meetings. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4353–4356, 2008.

[BYC+20] 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, IEEE International Conference on Acoustics, Speech, and Signal Processing*, Barcelona, Spain, May 2020.

[Can87] Walter B. Cannon. *The James-Lange Theory of Emotions: A Critical Examination and an Alternative Theory*, volume 100. USA, 1987.

[Che16] Jim X. Chen. The evolution of computing: Alphago. *Computing in Science Engineering*, 18(4):4–7, 2016.

[CHW06] Ze-Jing Chuang Yu-Chung Lin Chun-Hsien Wu. Emotion recognition from text using semantic labels and separable mixture models. *ACM Transactions on Asian Language Information Processing*, June 2006.

[clf21] Machine learning quick guide tutorial, 2021.

[Col20] David Cole. The Chinese Room Argument. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, winter 2020 edition, 2020.

[Con21] Wikipedia Contributors. Recurrent neural network, 2021. [Online; accessed 10-June-2021].

[Das07] Mike Das, Sanjiv Chen. Yahoo! for amazon: Sentiment extraction from small talk on the web. *Management Science*, 53(9):1375–1388, 2007.

[dC] Universitat Oberta de Catalunya. Procesamiento del pln.

[dE15] Agencia Estatal Boletín Oficial del Estado. Ley orgánica 13/2015. 2015.

[Dom19] José Domínguez. Herramientas de ayuda para compiladores, 2019.

[EGR19] Mercedes Reigosa Elena Garayzábal, Sheila Queralt and Susana Ridao. La transcripción de registros de audio en el ámbito policial y judicial español, 2019.

[Ekm83] Levenson R. Friesen W. Ekman, P. Autonomic nervous system activity distinguishes among emotions. *Science*, pages 1208–1210, 1983.

[fAp06] European funded AMI project. AMI Corpus Database. https://groups.inf.ed.ac.uk/ami/corpus/, March 2006.

[FKH+19] Yusuke Fujita, Naoyuki Kanda, Shota Horiguchi, Yawen Xue, Kenji Nagamatsu, and Shinji Watanabe. End-to-end neural speaker diarization with self-attention. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 296–303, 2019.

[FLC17] Haytham M. Fayek, Margaret Lech, and Lawrence Cavedon. Evaluating deep learning architectures for speech emotion recognition. *Neural Networks*, 92:60–68, 2017. Advances in Cognitive Engineering Using Neural Networks.

[Gap99] W. Gappmair. Claude e. shannon: the 50th anniversary of information theory. *IEEE Communications Magazine*, 37(4):102–105, 1999.

[Gia15] Theodoros Giannakopoulos. pyaudioanalysis: An open-source python library for audio signal analysis. *PloS one*, 10(12), 2015.

[GLO13] Clore Gerald L. and Andrew Ortony. Psychological construction in the occ model of emotion. *Emotion Review*, page 335–343, October 2013.

[Gon99] Gerardo M. González. Approach for screening depression in chicanos/latinos. *Am J Psychol*, 1999.

[HAHM+17] Pavol Harar, Jesus B. Alonso-Hernandezy, Jiri Mekyska, Zoltan Galaz, Radim Burget, and Zdenek Smekal. Voice pathology detection using deep learning: a preliminary study. In *2017 International Conference and Workshop on Bioinspired Intelligence (IWOBI)*, pages 1–4, 2017.

[Has20] Farukh Hashmi. How to create hierarchical clustering in python, Sep 2020.

[Her18] Lionel Hertzog. First steps with non-linear regression in r, Jan 2018.

[Hun07] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[Hut04] W. John Hutchins. The georgetown-ibm experiment demonstrated in january 1954. In Robert E. Frederking and Kathryn B. Taylor, editors, *Machine Translation: From Real Users to Research*, pages 102–114, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[HWF+20] Zili Huang, Shinji Watanabe, Yusuke Fujita, Paola García, Yiwen Shao, Daniel Povey, and Sanjeev Khudanpur. Speaker diarization with region proposal network. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6514–6518, 2020.

[JHY+12] Jeeweon Jung, Heesoo Heo, Ilho Yang, Sunghyun Yoon, Hyejin Shim, and Hajin Yu. D-vector based speaker verification system using raw waveform cnn. In *Proceedings of the 2017 International Seminar on Artificial Intelligence, Networking and Information Technology (ANIT 2017)*, pages 126–131. Atlantis Press, 2017/12.

[JS61] K. Nakata J. Suzuki. Recognition of japanese vowels - preliminary to the recognition of speech. *Journal of the Radio Research Laboratories*, 8(37):193–212, 1961.

[Kal21] Prajeet Singh Kalchuri. Decision tree amp; random forest explained, Apr 2021.

[KHDB52] R. BIDDULPH K. H. DAVIS and S. BALASHE. Automatic recognition of spoken digits. *Bell Telephone Laboratories, Inc., Murray Hill*, 24:637–642, 1952.

[Kre10] Sylvia D. Kreibig. Autonomic nervous system activity in emotion: A review. *Biological Psychology*, 84(3):394–421, 2010. The biopsychology of emotion: Current theoretical and empirical perspectives.

[Kuh55] H. W. Kuhn. The hungarian method for the assignment problem. 2:83–97, 1955.

[Kum20] Dhairya Kumar. Demystifying support vector machines, Jun 2020.

[LR18] Steven R. Livingstone and Frank A. Russo. The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS), April 2018.

[LT19] Alejandra Cardoso Lorena Talamé, Matías Amor. Herramientas de procesamiento de textos para python, 2019.

[MAVC14] Patricia Melin, Jonathan Amezcua, Fevrier Valdez, and Oscar Castillo. A new neural network model based on the lvq algorithm for multi-class classification of arrhythmias. *Information Sciences*, 279:483–497, 2014.

[McC19] Nick McCullum. Introduction to k nearest neighbors models, 2019.

[MLM+20] Brian McFee, Vincent Lostanlen, Alexandros Metsai, Matt McVicar, Stefan Balke, Carl Thomé, Colin Raffel, Frank Zalkow, Ayoub Malek, Dana, and et al. librosa/librosa: 0.8.0. *Zenodo*, Jul 2020.

[NFDS03] Tin Lay Nwe, Say Wei Foo, and Liyanage C De Silva. Speech emotion recognition using hidden markov models. *Speech communication*, 41(4):603–623, 2003.

[OCRW19] Gijs Overgoor, Manuel Chica, William Rand, and Anthony Weishampel. Letting the computers take over: Using ai to solve marketing problems. *California Management Review*, 61(4):156–185, 2019.

[Pal20] Aratrika Pal. Gradient boosting trees for classification: A beginner's guide, Oct 2020.

[Plu80] Robert Plutchik. *Theories of Emotion*. Academic Press, 1980.

[Plu91] Robert Plutchik. *The Emotions*. University Press of America, 1991.

[Pro] Jan Profant. Vbdiarization: X-vectors diarization. https://github.com/BUTSpeechFIT/VBx. Updated: 2021-05-13.

[PVG+11] 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.

[Red04] Francisco A. González Redondo. Leonardo torres quevedo (1852-1936)1a parte. las máquinas algebricas. *La Gaceta de la RSME*, 7.3(1):787–810, 2004.

[RHL16] Jinfeng Rao, Hua He, and Jimmy Lin. Noise-contrastive estimation for answer selection with deep neural networks. New York, NY, USA, 2016.

[RSP06] Jean Y. Wu Jason Chuang Christopher D. Manning Andrew Y. Ng Richard Socher, Alex Perelygin and Christopher Potts. Stanford Sentiment Treebank Database. https://nlp.stanford.edu/sentiment/, March 2006.

[RSSL13] Fabien Ringeval, Andreas Sonderegger, Juergen Sauer, and Denis Lalanne. Introducing the recola multimodal corpus of remote collaborative and affective interactions. In *2013 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, pages 1–8, 2013.

[Seh86] Agarwal S. Rai G Sehgal, R. R. A synthetic speaker. *IBM Thomas J. Watson Research Center*, 11:53–56, 1986.

[Seh18] Agarwal S. Rai G Sehgal, R. R. Interactive voice response using sentiment analysis in automatic speech recognition systems. *International Conference on Advances in Computing and Communication Engineering (ICACCE)*, 2018.

[SGRS+18] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. X-vectors: Robust dnn embeddings for speaker recognition. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5329–5333, 2018.

[Sie18] Sands M. K. Van den Noortgate W. Condon P. Chang Y. Dy J. Quigley K. S. Barrett L. F. Siegel, E. H. Emotion fingerprints or emotion populations? a meta-analytic investigation of autonomic features of emotion categories. *Psychologycal Bulletin*, 4(144):343–393, 2018.

[Sin20] Akshay Singhal. K means clustering algorithm, Jan 2020.

[SMW93] M. Sugiyama, J. Murakami, and H. Watanabe. Speech segmentation and clustering based on speaker features. In *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 395–398 vol.2, 1993.

[spe18] Espectro señal de voz, 2018.

[sst] Papers with code - sst-5 fine-grained classification benchmark (sentiment analysis).

[SYG92] M.-H. Siu, G. Yu, and H. Gish. An unsupervised, sequential learning algorithm for the segmentation of speech waveforms with multiple speakers. In *[Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 189–192 vol.2, 1992.

[TA19] Zehra Taskin and Umut Al. Natural language processing applications in library and information science. *Online Information Review*, ahead-of-print, 01 2019.

[TCY19] Prasanna Tambe, Peter Cappelli, and Valery Yakubovich. Artificial intelligence in human resources management: Challenges and a path forward. *California Management Review*, 61(4):15–42, 2019.

[TK09] Tomer Toledo and Romina Katz. State dependence in lane-changing models. *Transportation Research Record*, 2124:81–88, 12 2009.

[TR06] S.E. Tranter and D.A. Reynolds. An overview of automatic speaker diarization systems. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1557–1565, 2006.

[TRB+16] George Trigeorgis, Fabien Ringeval, Raymond Brueckner, Erik Marchi, Mihalis A. Nicolaou, Björn Schuller, and Stefanos Zafeiriou. Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5200–5204, 2016.

[VM20] Albornoz Enrique M. Vignolo, Leandro D. and César E Martínez. Exploring feature extraction methods for infant mood classification. *CoRR*, abs/2012.01786, 2020.

[Wei66a] Joseph Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9(1):36–45, January 1966.

[Wei66b] Joseph Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9(1):36–45, January 1966.

[Yan18] Yi Yang. Convolutional neural networks with recurrent neural filters, 2018.

[YBJ18] Seunghyun Yoon, Seokhyun Byun, and Kyomin Jung. Multimodal speech emotion recognition using audio and text. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 112–118. IEEE, 2018.

[yCAR10] Humberto Pérez y Carlos Alberto Reyes. Reconocimiento de emociones a partir de voz, 2010.

[Zab95] S. L. Zabell. Alan turing and the central limit theorem. *The American Mathematical Monthly*, 102(6):483–494, 1995.

[ZMC19] Jianfeng Zhao, Xia Mao, and Lijiang Chen. Speech emotion recognition using deep 1d & 2d cnn lstm networks. *Biomedical Signal Processing and Control*, 47:312–323, 2019.

[ZS19] Qinghong Han Xiaofei Sun Yuxian Meng Fei Wu Jiwei Li Zijun Sun, Chun Fan. Self-explaining structures improve nlp models. *AI Communications*, 32:191–206, 2019.