UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA
ELÉCTRICA-PROCESAMIENTO DIGITAL DE SEÑALES

USO DE REDES NEURONALES ARTIFICIALES PARA EL RECONOCIMIENTO DE PALABRAS CLAVE PARA UN ROBOT DE SERVICIO

TESIS
QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN INGENIERÍA

PRESENTA
STEPHANY ORTUÑO CHANELO

TUTOR
DR. JESÚS SAVAGE CARMONA
FACULTAD DE INGENIERÍA. UNAM

CIUDAD UNIVERSITARIA, CD. MX. DICIEMBRE, 2021.

Jurado asignado

Presidente: Dra. Medina Gómez Lucía

Secretario: M. I. Escobar Salguero Larry

1 er. Vocal: Dr. Savage Carmona Jesús

2 do. Vocal: Dr. Negrete Villanueva Marco A.

3 er. Vocal: Dr. Presacco Alessandro

Lugar o lugares donde se realizó la tesis: Ciudad Universitaria, CD. MX.

TUTOR DE TESIS:
Dr. Savage Carmona Jesús

Agradecimientos

A la máxima casa de estudios, la Universidad Nacional Autónoma de México, por haberme brindado la oportunidad de una formación profesional.

A la unidad de posgrado de la Facultad de Ingeniería, que a lo largo de mi vida ha sido un segundo hogar.

Al posgrado de Procesamiento de Señales, que me dio todos los elementos para mi formación profesional.

A mi asesor, el Dr. Jesús Savage Carmona, por su apoyo, paciencia y contribuciones para el desarrollo de este proyecto.

A mi jurado y profesores, por su asesoría en este proyecto de tesis y a lo largo de mi trayectoria escolar.

A mis compañeros de laboratorio, Julio Cesar Cruz y Alessandro Presacco, por su guía y apoyo a lo largo de este proyecto. Y a los demás integrantes del laboratorio de Bio-Robótica, por compartir sus conocimientos y sabios consejos.

Al CONACYT por los recursos otorgados para cursar este grado de maestría.

Este proyecto de tesis se realizó con el apoyo del proyecto PAPIIT-DGAPA, UNAM IG-101721 y el proyecto PAPIIT IA106520; "visión activa y robótica basada en comportamientos en el desarrollo de vehículos autónomos".

A mis padres, **Josefina y José**, con quienes estaré eternamente agradecida por la oportunidad de vida que me dieron, por ser el motivo de inspiración y superación personal cada día. Por enseñarme con el ejemplo la fortaleza para vencer cualquier reto, por su fe sin límites y su amor infinito. Por mostrarme que el camino no es facil pero los sueños se cumplen. Por siempre alentarme a seguir y recordarme que la duda no existe en sus corazones.

A mi hermano, **Edgar**, por contagiarme su curiosidad por las maravillas del universo y ayudarme a dar mis primeros pasos hacia la ciencia.

A mis sobrinos, **Oliver y Vladimir**, por recordarme a cuestionar cada aspecto de la vida, por irradiar mi espiritu con su luz e infinito amor.

A **Oscar**, por ser luz en días oscuros. Por demostrarme que a pesar de que la vida se trata de una condición efímera, estar vivo es algo profundamente bello. Por tu constante amor, comprensión y por todas esas aventuras y momentos de felicidad que aún nos quedan por vivir.

A **Gude**, por brindarme un segundo hogar y recibirme siempre con brazos abiertos.

A **Lore**, por acompañarme en esta aventura, por contagiarme su entusiasmo, por compartir su conocimiento, incluso las frustraciones y por iluminarme cuando estaba pérdida.

Resumen

En el ámbito de la interacción humano-máquina se busca una interacción lo más natural posible, es por ello que se recurre a la voz como medio de comunicación. Sin embargo, a pesar de décadas de investigación en el área, el rendimiento de los sistemas de reconocimiento automático de voz no se acerca a las capacidades humanas. La dificultad de estos sistemas radica en la variabilidad de la señal de voz. Basado en esta premisa y haciendo uso de la técnica de comprensión del lenguaje conocida como dependencia conceptual, este proyecto se enfocó en desarrollar un sistema de reconocimiento de palabras clave utilizando redes neuronales. Como primera aproximación se realizaron pruebas de concepto que permitieron discernir el tipo de arquitectura y el extractor de características más apropiado para resolver la tarea del reconocimiento de voz. Posteriormente se realizaron pruebas sobre el conjunto de datos generado a partir del generador de comandos oficial de la @Robocup, con un modelo secuencial el cual permite realizar el reconocimiento en segmentos de voz continua. Además, se describen las métricas utilizadas para medir el desempeño del sistema en general. Obteniendo resultados del $96.12\,\%$ *para cada estado de tiempo y una tasa de error de palabras menor al* $1\,\%$ *para el reconocimiento de palabras clave. Las pruebas relacionadas a la comprensión del lenguaje natural arrojaron resultados satisfactorios para el* $85\,\%$ *de la base de datos y a partir del análisis de los resultados, fue posible describir las fortalezas del sistema y listar un par de consideraciones para los trabajos a futuro que se puedan derivar de este proyecto.*

Abstract

In the area of human-machine interaction, the goal is to achieve an interaction as natural as possible, because of this the voice is used as a linker. However, even with decades of investigation, the automatic speech recognition system's performance cannot be compared with the human ability to do this task. The complexity of these systems lies in the voice's variability. On account of these limitations and using the natural language understanding technique known as conceptual dependency, on this project a keyword recognition system using neural networks was built. As first approach, a series of proof of concepts were designed in order to determine the most efficient combination of the neural network's configuration and features extractor technique for the speech recognition task. Afterward, the sequence-to-sequence model designed in this project was tested on the dataset obtained by using the official command generator provided by the @Robocup team, which is able to recognize continuous speech. In addition, in this thesis, the metrics used to evaluate the system's performance are described. The system obtained an accuracy of $96.12\,\%$ *in the step-by-step proof and a word error rate less than* $1\,\%$ *for keyword recognition. On the other hand, in the proof related to natural language understanding, the system obtained an accuracy of approximately* $85\,\%$*. Analyzing the results obtained it was possible to assess the system's strengths, its limitations and list some considerations for future projects.*

Índice general

1. Introducción **1**

1.1. El problema del reconocimiento automático de voz 2
1.2. Antecedentes 3
1.3. Estado del arte 4
1.4. Justificación 5
1.5. Objetivos 6
1.6. Estructura de la tesis 6

2. Marco Teórico **9**

2.1. La voz humana 9
2.1.1. Representación de la señal de voz 10
2.2. Comprensión del Lenguaje Natural 12
2.2.1. Teoría de la dependencia conceptual 14
2.3. Reconocimiento automático del habla 16
2.4. Procesamiento digital de la señal 17
2.5. Extracción de características 20
2.5.1. Modelo de la voz y análisis de predicción lineal 20
2.5.2. Espectrograma 23
2.5.3. Coeficientes MEL-Cepstrales 24
2.6. Sistemas de reconocimiento de voz 26
2.6.1. Cuantización vectorial 26
2.6.2. Modelos ocultos de Markov 29
2.6.3. Redes neuronales artificiales 31

3. Redes neuronales artificiales **33**

3.1. Modelo de la neurona 33
3.1.1. Funciones de activación 35
3.2. Redes neuronales artificiales 36
3.3. Redes Feedforward 38
3.3.1. Retropropagación 39

3.4. Redes neuronales recurrentes 42
3.4.1. Retropropagación en el tiempo 43
3.4.2. Redes Long-Short Term Memory 45

4. Sistema de reconocimiento de voz 47
4.1. Corpus de entrenamiento 47
4.1.1. Generador de comandos 48
4.1.2. Aumentado de datos 49
4.1.3. Procesamiento digital de la señal 50
4.2. Etiquetado 53
4.2.1. Alineación de etiquetas 54
4.3. Arquitectura 55
4.4. Reconocedor de palabras clave 56

5. Pruebas y resultados 57
5.1. Pruebas de concepto 57
5.1.1. Redes FeedForward 58
5.1.2. Redes LSTM 59
5.2. Reconocimiento de comandos de voz 61
5.2.1. Word Error Rate 65
5.3. Prueba de CD 66
5.4. Carga computacional 67

6. Conclusiones 69
6.1. Trabajos a futuro 71

A. GPSR y Parser Stanza 75
A.1. GPSR 75
A.2. Parser Stanza 76

B. Google Text To Speech 79

C. ANN en MATLAB 81

Bibliografía. 89

Índice de figuras

2.1. Esquema básico del aparato fonador humano. Imagen modificada de: [Objetos-UNAM, 2021] 10
2.2. Evolución temporal de la señal de voz. 11
2.3. Espectro de potencia de un bloque de la señal de voz. 11
2.4. Evolución temporal de la señal a corto plazo. 12
2.5. Esquema del proceso de comprensión del lenguaje natural. 13
2.6. Ejemplo de estructura de la primitiva de la dependencia conceptual. 15
2.7. Esquema básico de la arquitectura de un ASR. 16
2.8. Diagrama del procesamiento digital de audio. 18
2.9. Curva de respuesta en frecuencia del filtro preénfasis. Imagen recuperada de: [Haykin, 2001] 19
2.10. Modelo de producción de voz: para sintetizar el habla, el interruptor sonoro / sordo cambiará a la fuente del sonido en ese momento en particular. Los parámetros del tracto vocal también deberán variar con el tiempo [Ward, 2001]. 21
2.11. Representación en tres dimensiones del espectrograma. Imagen recuperada de: [Carrión, 2017]. 23
2.12. Diagrama de bloques del proceso para el cálculos de los MFCC. . . . 24
2.13. Cuantización vectorial aplicada al reconocimiento de voz. 27
2.14. Esquema básico del algoritmo de Linde-Buzo-Gray. Imagen modificada de: [Amidi and Amidi, 2017]. 29
2.15. Diagrama de HMM. 30
2.16. Diagrama de red neuronal para el reconocimiento de voz. 32

3.1. Modelo de neurona. 34
3.2. Funciones de activación. 35
3.3. Clasificación de las ANN por su topología. 37
3.4. Arquitectura de ANN Feedforward. Imagen modificada de: [Loy, 2018]. 38
3.5. Diagrama del algoritmo de retro propagación. Imagen modificada de: [Shrivastava, 2018]. 39

3.6. Descenso por gradiente. Imagen recuperada de: [Akshay, 2019]. . . . 41
3.7. Arquitectura de una red neuronal recurrente. Imagen recuperada de [Olah, 2015]. 43
3.8. Arquitectura de una LSTM. Imagen modificada de [Afzal et al., 2010] 45

4.1. Ejemplo de las señales de voz. 48
4.2. Aumentado de datos por inyección de ruido y corrimiento de la señal. 50
4.3. Comparación de bloques originales de la señal y después de aplicar la ventana de Hamming. 51
4.4. Matriz de características: 13 coeficientes x L bloques. 52
4.5. Proceso de etiquetado de los comandos de voz. 53
4.6. Alineación de las etiquetas de los comandos de voz. 54
4.7. Esquema de tareas muchos a muchos. Imagen recuperada de: [Amidi, 2018]. 55
4.8. Diagrama básico del algoritmo de recuperación de palabras clave en un comando. 56

5.1. Definición de los subconjuntos de entrenamiento y prueba a partir del conjunto de datos original. 57
5.2. Diagrama general de la red neuronal feedforward clasificadora. . . . 58
5.3. Esquema de tareas muchos a uno. Imagen recuperada de: [Amidi, 2018]. 59
5.4. Reconocimiento de voz por estado de tiempo del comando ***Justina, please follow Stephany***. 62
5.5. Matrices de confusión para el reconocimiento de comandos de voz. Superior: representa los resultados de las primeras 22 clases. Inferior: representa los resultados de las últimas 22 clases. 64
5.6. Ejemplos de los tipos de error contenidos en los comandos resultantes. 65
5.7. Ejemplos de las primitivas obtenidas a partir de las palabras clave reconocidas por el sistema. 66
5.8. Ejemplo de errores al obtener la estructura de la primitiva. 67

6.1. Diagrama del sistema de segmentación de voz. 71

Índice de tablas

5.1. Características para la arquitectura FF. 59
5.2. Características para la arquitectura RNN. 60
5.3. Resultados de pruebas de concepto. 60
5.4. Tamaño de la base de datos para el reconocimiento de comandos de voz. 61
5.5. Características para la arquitectura LSTM. 61
5.6. Carga computacional de los procesos que componen el reconocimiento de voz. 68

CAPÍTULO 1

Introducción

La tecnología se ha convertido en una parte integral de nuestras vidas por lo que regularmente se siguen desarrollando nuevos avances en el campo de la informática y la tecnología. Actualmente, diversos grupos de investigación realizan esfuerzos conjuntos para emular la inteligencia humana y permitir la interacción entre el humano y las máquinas.

Intentos iniciales de proporcionar un medio para este tipo de comunicación condujo al desarrollo de hardware; el teclado, la pantalla táctil, etcétera. Sin embargo, ninguno de estos dispositivos proporciona la naturalidad que proporciona el habla. La voz o el lenguaje hablado es la forma común de comunicación entre los seres humanos, este proceso se realiza de manera natural y en condiciones o ambientes difíciles. Al ser la forma más directa de comunicación, es la manera idónea en que el hombre busca interactuar con los dispositivos que utiliza en la vida diaria.

En general, utilizar un sistema de voz para controlar los dispositivos es más conveniente y puede mejorar la eficiencia en la industria y de la vida diaria. Por lo anterior, el reconocimiento de voz es un punto de gran interés en la actualidad y algunas de sus aplicaciones se pueden descubrir en diversos ámbitos de la vida cotidiana, como las transcripciones y traducciones automáticas, los sistemas para vehículos, la automatización del hogar, los asistentes virtuales y la robótica. Esta última se centra en la interacción con el robot, el cual puede reconocer la voz y es capaz de responder o realizar la acción especificada, siendo esta última la tarea de mayor interés para este proyecto.

1.1. El problema del reconocimiento automático de voz

A pesar de décadas de investigación en el área, el rendimiento de los sistemas de reconocimiento automático de voz (en inglés, Automatic Speech Recognition (ASR)) no se acerca a las capacidades humanas. La dificultad en estos sistemas radica en la variabilidad de la señal de voz, a diferencia del texto, no existen límites definidos entre palabras debido al flujo natural del habla. Además, las características espectrales y temporales del habla varían dependiendo de una serie de factores, como los que se listan a continuación [Benzeguiba et al., 2010]:

- Fisiológicos: las diferentes dimensiones del tracto vocal cambian las frecuencias de la voz.
- Fenómenos acústicos ambientales: la señal grabada no solo captura la voz sino también múltiples efectos de reverberación y el habla de fondo de múltiples hablantes lo que da lugar a confusiones entre los sonidos. Además, la presencia de ruido de fondo reduce la calidad de la señal.
- Variabilidad: estos factores dependen de aspectos relacionados con el locutor como la pronunciación, la duración, etc.
- Continuidad: en el lenguaje natural no existen separadores entre las unidades, equivalentes a los espacios del lenguaje escrito.

Por otra parte, uno de los desafíos en el entrenamiento de un reconocedor de voz, es que requiere una gran cantidad de datos de entrenamiento transcritos. Los datos de entrenamiento transcritos consisten en datos de audio y las transcripciones de texto correspondientes. Existe un suministro ilimitado de datos de audio, pero generalmente no hay una transcripción exacta disponible. Sin embargo, la transcripción manual de audio requiere mucho tiempo y algunos segmentos del audio pueden estar mal alineados, contener ruido o habla simultánea de más de un orador. Estos hechos dificultan el uso de estos datos para un ASR.

1.2. Antecedentes

El reconocimiento del lenguaje natural por un sistema como una máquina, es un problema que ha experimentado una extensa evolución para lograr obtener los sistemas de reconocimiento de voz que existen actualmente y a lo largo de la historia, se han desarrollado distintas aproximaciones para abordarlo.

Los primeros intentos de diseñar sistemas para el reconocimiento automático del habla se guiaron principalmente por la teoría de la fonética acústica, que surge bajo la premisa de que existe un conjunto finito de unidades fonéticas en el lenguaje hablado y que pueden ser caracterizados por un conjunto de propiedades presentes en la señal de voz. A partir de esto, en la década de 1952, Homer Dudley de Bell Laboratorios propuso un sistema para el análisis de la voz, el cual podía reconocer unos cuantos dígitos. Una década más tarde, IBM presentó un sistema que entendía 16 palabras. Se operaba hablando por un micrófono que convertía la voz en impulsos eléctricos y un circuito que clasificaba estos impulsos según varios tipos de sonidos [IBM, 2016].

En la década de los 70, [Atal and Hanauer, 1971] e [Itakura and Saito, 1970] proponen la teoría de la codificación de predicción lineal (en inglés, Linear Predictive Coding (LPC)) lo cual permitió la estimación de la respuesta del tracto vocal a partir de formas de onda de la voz. Posteriormente, [Itakura, 1975] y [L. R. Rabiner and Wilpon, 1979] propusieron la aplicación de este tipo de codificación en el reconocimiento de voz.

En la década de los 80, surgen los modelos ocultos de Markov (en inglés, Hidden Markov Models (HMM)) un método estadístico que en lugar de comparar las señales, estimó la probabilidad de que los sonidos fueran palabras [Juang and Rabiner, 2004], este método permitió aumentar el vocabulario de reconocimiento a varios miles de palabras. Este modelo se ha convertido desde entonces en la representación predominante de unidades de voz para el reconocimiento de voz continua.

Otro aporte importante de Bell Laboratories fue la introducción de la técnica de reconocimiento de palabras clave (en inglés, Keyword Spotting (KWS)) a finales de la década de los 80, esta tenía como objetivo detectar una palabra o frase clave presente en un enunciado más complejo, en donde el resto de las palabras no eran significativas [J. G. Wilpon and Goldman, 1990]. La necesidad de esta técnica surgió a partir de que los usuarios preferían hablar en oraciones naturales en lugar de usar secuencias de comandos rígidas para solicitar servicios.

Otra tecnología que se reintrodujo en esta década fueron las redes neuronales artificiales (en inglés, Artificial Neural Network (ANN)), las cuales pretenden emular el proceso de reconocimiento humano, a partir de la obtención de datos, su análisis y una posterior clasificación. Primeros intentos de utilizar redes neuronales para el reconocimiento de voz a corto plazo demostraron buenos resultados, sin embargo, el problema del reconocimiento de voz continua implica el manejo de la variación temporal y las redes neuronales en su forma original no han demostrado ser eficientes para esta tarea.

1.3. Estado del arte

La investigación en curso se centra en el uso de HMM, la integración de modelos híbridos o en modelos de redes neuronales que integren los modelos de lenguaje y además aprovechan el manejo temporal. En años recientes, Michaely y colaboradores presentaron un novedoso sistema de reconocimiento de palabras clave que utiliza el reconocimiento automático de voz contextual [Michaely et al., 2017].

Por otra parte, las técnicas de aprendizaje automático, como las redes neuronales profundas DNN, convolucionales CNN [Gouda et al., 2020] y recurrentes RNN han demostrado ser útiles para la detección de palabras clave. Generalmente, estas redes utilizan un preprocesamiento que extrae los coeficientes MEL cepstrales (en inglés, Mel Frequency Cepstral Coefficients (MFCC)) [Davis and Mermelstein, 1980].

Zhang y colaboradores [Zhang et al., 2017] investigaron varias arquitecturas de red a pequeña escala e identificaron que la CNN de profundidad separable proporciona la mejor relación entre clasificación y precisión para los recursos computacionales. Otros trabajos han mejorado este resultado utilizando datos sintéticos [Lin et al., 2020], redes convolucionales temporales [Choi et al., 2019] y técnicas de autoatención [Andrade et al., 2018].

1.4. Justificación

Actualmente las técnicas de reconocimiento de voz se encuentran lejos de compararse con la habilidad de un humano. Por esta razón algunas técnicas están inspiradas o tratan de imitar la forma en que los humanos realizan esta tarea. La capacidad de las redes neuronales como excelentes clasificadores de patrones estáticos es innegable. Sin embargo, la voz dista mucho, de presentar una naturaleza estática, por lo que la aplicación de las redes neuronales simples al reconocimiento automático de voz plantea una serie de problemas. Debido a la naturaleza de la voz se recurre a las redes neuronales recurrentes (en inglés, Recurrent Neural Network (RNN)) dada su fortaleza en el análisis de secuencias donde es muy importante tener en cuenta el contexto pasado o futuro.

Por otra parte, un aspecto que permite valorar con más certeza los resultados de una investigación es la base de datos que se utilizó. Muchos autores construyen sus propias bases de datos y esto dificulta saber qué tan compleja es la tarea de reconocimiento, por ello se propone generar de manera automática los comandos haciendo uso del generador de comandos oficial de la RoboCup, lo que permite generar una base de datos posible de replicar para futuras pruebas y proyectos.

1.5. Objetivos

El objetivo principal para el presente proyecto de tesis es el siguiente:

Desarrollar un sistema de reconocimiento de palabras clave utilizando redes neuronales, el cual será utilizado por un robot de servicio autónomo que ejecutará tareas previamente programadas según las órdenes que reciba de manera verbal.

Los objetivos particulares derivados del anterior son los siguientes:

- Construir una base de datos a partir del generador de comandos oficial de la RoboCup.
- Procesar y analizar los datos con el objetivo de extraer las características necesarias para la detección de palabras clave.
- Diseñar la arquitectura óptima de la red neuronal que se empleará en el proceso de entrenamiento.
- Evaluar la viabilidad y rendimiento del método y la arquitectura propuesta.

1.6. Estructura de la tesis

En esta sección se presenta la estructura de la actual tesis, con lo que se pretende dar un mejor entendimiento de su contenido. Fundamentalmente el manuscrito está dividido en seis capítulos.

- En el primer capítulo se presenta la introducción de la tesis donde se mencionan los antecedentes y el estado del arte, a partir de los cuales se plantea la justificación del proyecto y de igual manera se plantean los objetivos que se desean alcanzar.

- En el segundo capítulo se describirán brevemente los fundamentos y la teoría que se utilizaron para el desarrollo del proyecto. Se describen las aproximaciones para resolver este tipo de tareas y se realiza una descripción breve de las técnicas de procesamiento digital de señales utilizadas para el acondicionamiento de la señal de voz.
- En el tercer capítulo se expone la teoría relacionada con el aprendizaje profundo. Se hace una descripción sobre las redes neuronales, los tipos de capas neuronales que existen y los parámetros para su entrenamiento.
- En el cuarto capítulo se expone el procedimiento para desarrollar el sistema automático de voz propuesto basado en los conceptos anteriores.
- En el quinto capítulo se mencionan las pruebas que se realizaron y se listan los resultados obtenidos para cada uno de estos procesos.
- En el último capítulo se presentan las conclusiones y se describen algunas consideraciones a futuro derivados de este proyecto.

CAPÍTULO 2

Marco Teórico

2.1. La voz humana

El lenguaje natural es el medio que utilizamos de manera cotidiana para establecer nuestra comunicación con las demás personas, haciendo uso de la voz para transmitir un mensaje. Por otra parte, la voz es la consecuencia de la asociación de distintos órganos del cuerpo humano que permiten producir sonidos a través de los cuales es posible la comunicación. Esta se produce por medio del aparato fonador, el cual está formado por los pulmones como fuente de energía, la laringe que contiene las cuerdas vocales, la faringe, las cavidades bucal y nasal y una serie de elementos articulatorios (los labios, los dientes, el paladar, la lengua, etcétera) un diagrama básico se muestra en la Figura 2.1.

Las cuerdas vocales son dos membranas dentro de la laringe y la abertura entre ambas cuerdas se denomina glotis. Cuando las cuerdas vocales se encuentran separadas, la glotis adopta una forma triangular, por lo que el aire pasa libremente y no genera sonido alguno. En cambio, cuando la glotis se encuentra parcialmente cerrada, el aire experimenta una turbulencia y genera un sonido del tipo exhalación. Al cerrarse más, las cuerdas vocales comienzan a vibrar produciéndose un sonido periódico. La frecuencia de este sonido depende de varios factores; el tamaño y la masa de las cuerdas vocales, la tensión que se les aplique y de la velocidad del flujo del aire proveniente de los pulmones [Miyara, 2017].

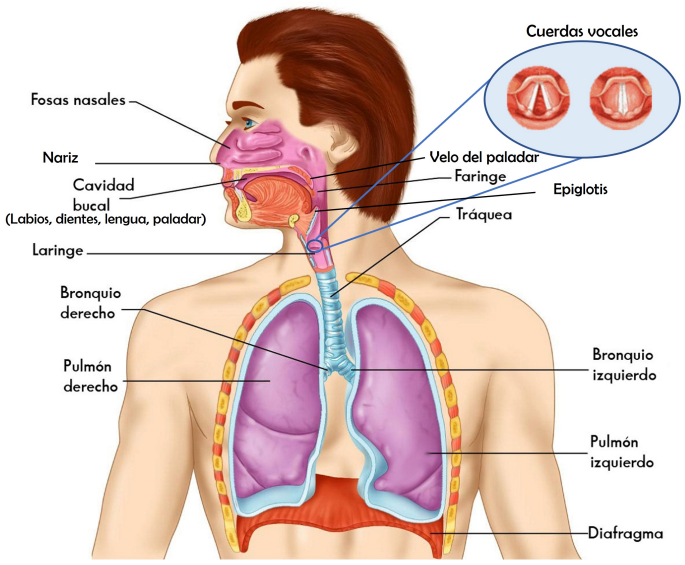


Figura 2.1: Esquema básico del aparato fonador humano. Imagen modificada de: [Objetos-UNAM, 2021]

2.1.1. Representación de la señal de voz

En la producción de voz, la información transmitida se codifica en forma de onda, la cual puede ser posteriormente manipulada dependiendo de su propósito. En la Figura 2.2 se presenta la evolución temporal de la letra "a" pronunciada en español. El eje horizontal representa el número de muestras y el vertical la amplitud de la señal. Inicialmente, antes de comenzar a hablar, la forma de onda se clasifica como silencio hasta aproximadamente las 8000 muestras. Posteriormente, se observa un abrupto incremento de energía durante aproximadamente 4000 muestras, que representa el segmento de voz correspondiente a la letra "a" y al final de la señal se observa otro segmento de silencio.

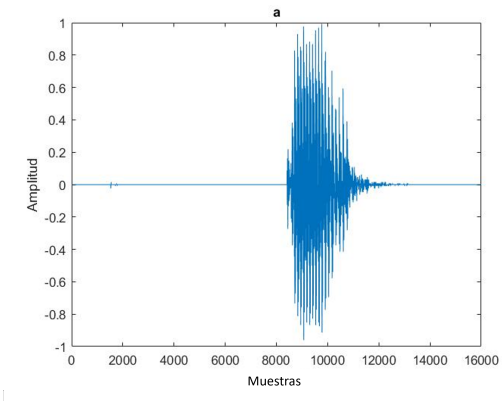


Figura 2.2: Evolución temporal de la señal de voz.

Por otra parte, el concepto de formante es de suma importancia en el análisis de la voz, pues en ellos se concentra la mayor parte de la información que permite la comprensión del mensaje, ya que éstos representan los picos de intensidad del espectro de la señal con mayor energía [Holmes et al., 1997]. En la Figura 2.3 se grafica el espectro de potencia correspondiente a un bloque de la señal de voz anterior, en la cual se muestran los formantes (F_1, F_2, ...), los cuales en general describen las frecuencias de resonancia del tracto vocal.

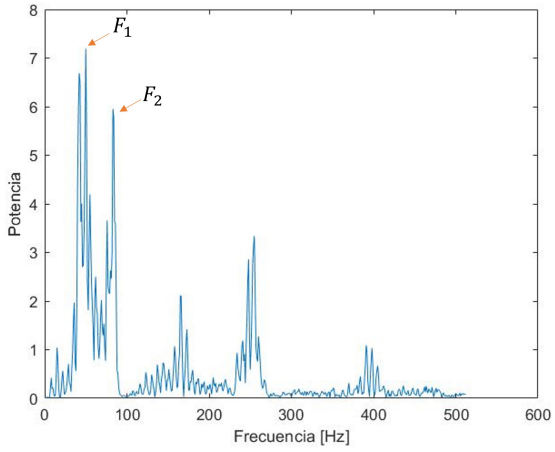


Figura 2.3: Espectro de potencia de un bloque de la señal de voz.

A partir de las gráficas anteriores se puede observar que la señal de voz tiene una estructura compleja. Sin embargo, para modelar la señal de voz, serían deseables modelos lineales e invariantes en el tiempo. Desafortunadamente, el mecanismo del habla no satisface ninguna de estas dos propiedades, ya que como puede observarse el habla es un proceso que varía continuamente con el tiempo. Además, la glotis está acoplada al tracto vocal, lo cual da lugar a características no lineales. Sin embargo, haciendo algunas suposiciones razonables, es posible desarrollar modelos en intervalos de tiempo corto, como se puede observar en la Figura 2.4 que analiza la señal por bloques y en la cual es posible observar un comportamiento aproximadamente periódico y posible de modelar.

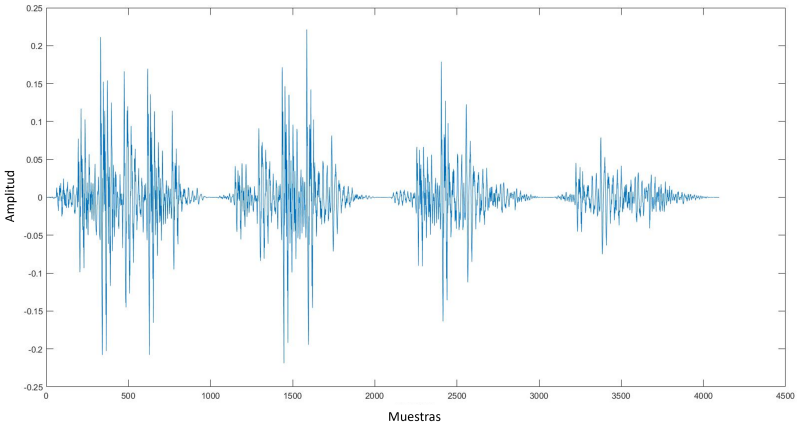


Figura 2.4: Evolución temporal de la señal a corto plazo.

2.2. Comprensión del Lenguaje Natural

Una de las tareas fundamentales de la inteligencia artificial (en inglés, Artificial Intelligence (AI)) es realizar la manipulación del lenguaje natural, permitiendo así formar un enlace entre la voz humana y la máquina, a este proceso se le conoce como procesamiento del lenguaje natural (en inglés, Natural Language Processing (NLP)).

Por otra parte, el concepto de comprensión del lenguaje natural (en inglés, Natural Language Understanding (NLU)) se refiere a poder realizar una transformación de su representación original a una donde sea posible determinar un conjunto de acciones a realizar [Rich and Knight, 1991]. El proceso de comprensión del lenguaje natural se ilustra en la Figura 2.5 y se puede descomponer en los siguientes pasos:

1. Transformación de la señal de entrada en unidades básicas; palabras, fonemas, etc.

2. El análisis sintáctico, el cual permite etiquetar gramaticalmente la secuencia de palabras.

3. El análisis semántico que permite determinar la acción principal, los participantes y los roles que desempeñan cada uno de estos. Y a partir de estas es posible realizar una transformación de la estructura que describa estas relaciones.

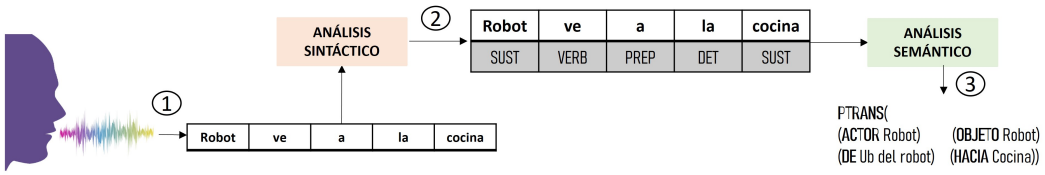


Figura 2.5: Esquema del proceso de comprensión del lenguaje natural.

Uno de los principales problemas de la NLU es la representación del significado, sin embargo, una de las representaciones de comandos de voz es a partir de la descripción de las relaciones de los objetos mencionados en la oración de entrada, lo que permite evitar las ambigüedades que existen naturalmente en cualquier lenguaje.

2.2.1. Teoría de la dependencia conceptual

La teoría de la dependencia conceptual (en inglés, Conceptual Dependency (CD)), propuesta por [Schank, 1972] propone dividir el proceso del NLU en subprocesos y tiene como premisa que una acción es la base de cualquier proposición. Es decir, el verbo clave en la oración se puede utilizar para definir la estructura y el significado de la oración.

La representación de la CD utiliza primitivas conceptuales en lugar de las palabras contenidas en la oración. Por otra parte, cada primitiva puede representar un conjunto de verbos los cuales tienen diferentes significados, sin embargo, cada una de estas contiene los siguientes componentes: **ACTOR, ACCIÓN, OBJETO, DIRECCIÓN, ESTADO y TIEMPO**.

A partir de esto, Shank propone una lista finita de primitivas que son las unidades básicas con las que ideas más complejas pueden ser entendidas [Savage et al., 2019], algunas de las cuales se listan a continuación:

1. ATRANS: transferencia de propiedad, posesión o control de un objeto (por ejemplo: dar, tomar, comprar).
2. PTRANS: transferencia de la ubicación física de un objeto (por ejemplo: ir, caminar, conducir).
3. ATTEND: concentrarse en percibir un estímulo sensorial (por ejemplo: encontrar, mirar).
4. MOVE: movimiento de una parte del cuerpo por parte de su dueño (por ejemplo: patear).
5. GRASP: agarrar un objeto por parte de un actor (por ejemplo: tomar).
6. PROPEL: aplicación de una fuerza física a un objeto (por ejemplo: tirar, patear, etc).

7. SPEAK: producción de sonidos (por ejemplo, hablar).

8. DO: cualquiera que no esté incluida en las primitivas anteriores.

Para representar una oración hablada a partir de la CD, se puede utilizar un sistema de reconocimiento de voz para determinar sus elementos constituyentes. La estructura seleccionada puede contener elementos vacíos los cuales tienen que ser completados empleando las palabras reconocidas por un ASR y la base de conocimientos general (que incluye información básica sobre la identidad y ubicación de los objetos y personas). Este proceso se muestra en la Figura 2.6, donde la primitiva inicial necesita los elementos; actor, objeto, dirección. Posterior al proceso de NLU es posible incluir los elementos faltantes.

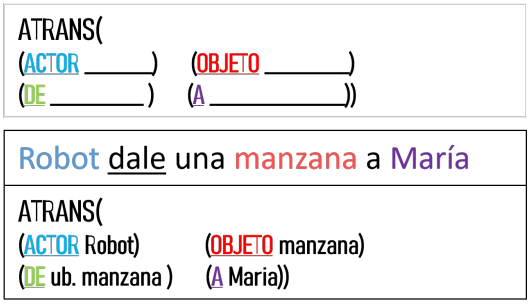


Figura 2.6: Ejemplo de estructura de la primitiva de la dependencia conceptual.

2.3. Reconocimiento automático del habla

El reconocimiento automático del habla (en inglés, Automatic Speech Recognition (ASR)) es un campo del procesamiento del lenguaje natural que se centra en la comprensión del lenguaje hablado. Este implica mapear la entrada auditiva a alguna palabra en un vocabulario del idioma, un esquema básico de este proceso se muestra en la Figura 2.7, el cual incluye un proceso de la señal de voz, un paso de extracción de características y el cual puede incluir una decodificación utilizando modelos acústicos, del lenguaje o el léxico para mejorar el reconocimiento.

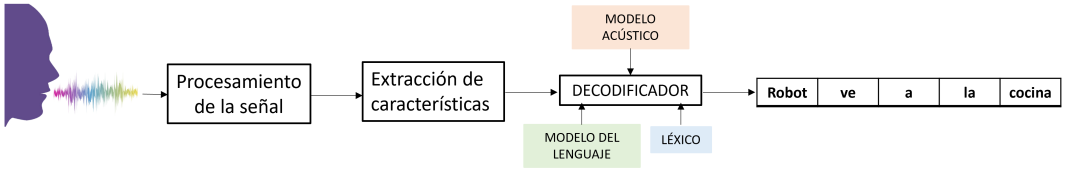


Figura 2.7: Esquema básico de la arquitectura de un ASR.

Como se mencionó anteriormente, la producción de voz es un proceso que se ve afectado por muchos factores como la entonación, el género de la persona que habla y algunos aspectos prácticos como el ruido y los fenómenos acústicos. Lo que hacen del reconocimiento de voz una tarea difícil, por lo que a lo largo de la historia se ha hecho necesario establecer estrategias que simplifiquen esta tarea, una de ellas es abordarlo por medio de la continuidad de la emisión de voz:

- Reconocedores de palabras aisladas: las entradas son palabras individuales, que contienen silencio en ambos extremos y se procesan individualmente. Esto es útil cuando se requiere que el usuario dé respuestas de una sola palabra, pero es muy poco natural para entradas de voz continua. Su principal ventaja es que es más simple de implementar porque los límites de las palabras se encuentran bien establecidos y las palabras tienden a pronunciarse claramente [Hernando, 1993].

- Reconocedores de voz continua: toma una entrada continua de voz y realiza el reconocimiento por medio de unidades inferiores a la palabras, sin necesidad de establecer silencios entre las palabras que constituyen la frase [Díaz et al., 2002]. Los sistemas de reconocimiento de voz continuo son más difíciles de crear porque deben utilizar métodos especiales para determinar los límites de las expresiones y a medida que aumenta el vocabulario, aumenta la posibilidad de confusión entre diferentes secuencias de palabras.
- Reconocedores de palabras clave: se encarga del reconocimiento de palabras de interés contenidas en un vocabulario que se encuentren en una oración emitida de forma natural. Al seleccionar las palabras clave de un enunciado, un modelo puede captar características destacadas y utilizarlas para comprender la oración en su totalidad. La principal diferencia es que el vocabulario que manejan es más reducido y no se interesan por las palabras ajenas, por lo que cualquier palabra fuera del vocabulario será descartada.

Esta última es la de mayor interés para este proyecto, ya que permitirá determinar únicamente los elementos clave del enunciado y por lo tanto permitirán construir la primitiva.

2.4. Procesamiento digital de la señal

Independientemente de cual de las aproximaciones anteriores se utilice es necesario un paso previo de procesamiento de la señal. Este conjunto de técnicas es útil para preparar la señal de voz y potenciar la extracción de características a partir de cualquier sistema, este proceso se describe en la Figura 2.8.

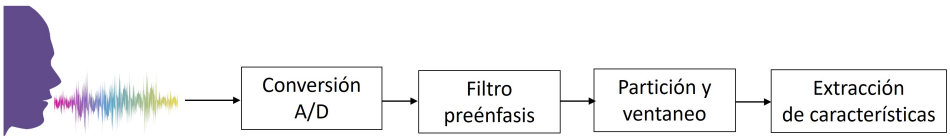


Figura 2.8: Diagrama del procesamiento digital de audio.

Primero es necesario digitalizar la señal de voz para que el ordenador la pueda comprender. Posteriormente, suele aplicarse un filtro el cual permite discriminar algunas características de la señal de entrada, donde su clasificación general es la siguiente:

- Filtro pasa bajas: deja pasar las componentes más graves (frecuencias bajas) del sonido y quita las componentes agudas (frecuencias altas) cuya frecuencia es superior a la frecuencia de corte del filtro.
- Filtro pasa altas: se atenúan los componentes de baja frecuencia pero no los de alta frecuencia.
- Filtro pasa bandas: este tipo de filtro deja pasar un determinado rango de frecuencias y atenúa el resto.
- Filtro supresor de bandas: no permite el paso de señales cuyas frecuencias se encuentran comprendidas entre las frecuencias de corte superior e inferior.

Con respecto a la voz, generalmente se aplica un filtro preénfasis, el cual realiza un incremento del nivel de altas frecuencias de audio. El objetivo es compensar la atenuación de aproximadamente 20 dB que se produce en el proceso fisiológico del mecanismo de producción del habla. Este paso es altamente recomendable para enfatizar los formantes [Rabiner and Schafer, 2007]. Un sistema simple de preénfasis cuya función de transferencia queda descrito por la siguiente ecuación:

$$H(z) = 1 - \gamma z^{-1} \tag{2.1}$$

donde γ es menor a 1. Como se muestra en la Figura 2.9 su respuesta permite que las frecuencias mayores a 2122 Hz se amplifiquen en relación con las frecuencias menores, donde m indica el factor de amplificación que sufre esta porción de frecuencias a la salida del filtro.

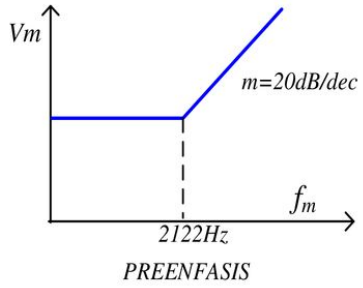


Figura 2.9: Curva de respuesta en frecuencia del filtro preénfasis. Imagen recuperada de: [Haykin, 2001]

Por otra parte, la señal de voz es un proceso aleatorio que a corto plazo es casi-estacionaria. Esto da paso a realizar un análisis por segmentos de pocos ms [Ruedo, 2011]. Esto se hace multiplicando los datos recopilados por una función llamada ventana que puede tener varias formas. La teoría de la ventana fue un tema activo de investigación por lo que en el procesado digital de señal de voz hay muchos tipos de ventanas, pero las formas más comunes se mencionan a continuación:

- Plana o rectangular: debido a su uniformidad, equivale a no efectuar modificación alguna a nivel muestras. El efecto de los datos truncados se puede apreciar como discontinuidades en la forma de ondas con ventanas.
- Hann (Hanning): tiene la forma de un ciclo de una onda cosenoidal al que se agrega 0.5 para que así siempre sea positivo. Realiza un buen trabajo, forzando los extremos hacia cero, pero también agrega distorsión a la forma de onda que se está analizando.
- Hamming: es una variación de la ventana da Hann donde la amplitud de los lóbulos laterales es prácticamente nula, lo que minimiza la dispersión espectral.

Finalmente, para mantener la continuidad de la información de la señal, es muy común aplicar un traslape entre las ventanas, de tal manera que no se pierde información en la transición entre cada una de ellas. Generalmente se aplica un desplazamiento entre ventanas aproximadamente de dos terceras partes del tamaño de la ventana original.

2.5. Extracción de características

En esta sección se describe cómo se transforma el audio en una serie de parámetros que representan de forma compacta la información del sonido. Éste es el punto de partida de cualquier sistema de reconocimiento de voz, por lo que cualquier segmento de audio que se vaya a utilizar indistintamente para entrenar o evaluar deberá ser sometido previamente a un proceso de extracción de sus características.

2.5.1. Modelo de la voz y análisis de predicción lineal

En general, un sistema de reconocimiento típico consiste en un extractor de características seguido de una técnica robusta de clasificación. La información del tracto vocal como la frecuencia de los formantes y otros valores pueden estar vinculados a la señal de voz [Afzal et al., 2010], por ello el objetivo de la etapa de extracción de características es determinar esta información.

Existe una amplia gama de posibilidades para representar paramétricamente la señal de voz que se utiliza en el proceso de ASR. Una de las técnicas mas usadas en el procesamiento de señales es la el análisis o codificación de predicción lineal (en inglés, Linear Predictive Coding (LPC)). Esta técnica ha probado ser muy eficiente debido a la posibilidad de paramétrizar la señal con un número reducido de patrones.

Su modelo establece que el tracto vocal puede modelarse mediante un filtro digital siendo los parámetros los que determinan la función de transferencia y permite aproximar una señal a partir de señales pasadas $s(n-k)$. En este caso se trata de predecir señales de voz mediante un filtro de respuesta finita al impulso (en inglés, Finite Impulse Response (FIR)) utilizando una combinación lineal de p muestras anteriores $s(n-k)$ más una señal de error $e(n)$ [Bradbury, 2020].

$$s(n) = \sum_{k=1}^{p} a_k \, s(n-k) + e(n) \tag{2.2}$$

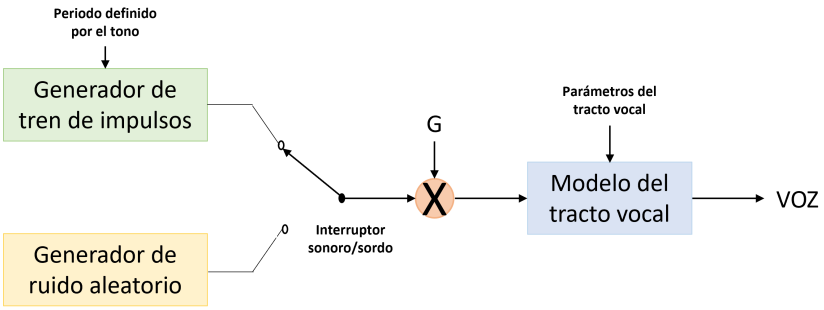


Figura 2.10: Modelo de producción de voz: para sintetizar el habla, el interruptor sonoro / sordo cambiará a la fuente del sonido en ese momento en particular. Los parámetros del tracto vocal también deberán variar con el tiempo [Ward, 2001].

El modelo describe el proceso del habla clasificando las señales en dos tipos:

- Señales sonoras: se caracterizan por tener alta energía y contenido frecuencial en el rango de los 300 Hz a 4000 Hz y además presentan cierta periodicidad.
- Señales sordas: se caracterizan por tener baja energía y componente frecuencial uniforme presentando aleatoriedad en forma de ruido blanco.

La entrada del filtro puede ser un tren de impulsos periódicos el cual producirá las señales sonoras o una fuente de ruido aleatorio que producirá los sonidos sordos (Figura 2.10) y cuya función de transferencia está descrita por la siguiente ecuación:

$$H(z) = \frac{G}{1 + \sum_{k=1}^{p} a_k \, z^{-k}} \tag{2.3}$$

Dada una señal s(n), el problema consiste en determinar los coeficientes de predicción a_k y la ganancia G, cuyo valor dependerá de la naturaleza de la señal de entrada. A partir de este proceso se obtienen los coeficientes de predicción, los que se usarán como parámetros de reconocimiento de palabras. Estos se determinan a partir de la minimización del error cuadrático medio:

$$e^{(p)} = \sum_{i=1}^{p} a_i \sum_{k=1}^{p} a_k \, \hat{r}(|k - i|) \tag{2.4}$$

De la ecuación anterior se puede obtener un sistema de ecuaciones que se es posible resolver a partir del algoritmo de Levison-Durbin [Levinson, 1947]. Una vez teniendo los coeficientes es útil calcular el error cuadrático medio, simplicifando $a_o = 1$, la expresión obtenida para los coeficientes óptimos es la distancia de Itakura-Saito, la cual es una medida de la diferencia entre un espectro original y el modelo aproximado de este, esta medida fue propuesta por Fumitada Itakura y Shuzo Saito en la década de 1960 [Itakura and Saito, 1968].

$$d_{IS} = r_a(0)\hat{r}(0) + 2 \sum_{k=1}^{p} \hat{r}(k) r_a(k) \tag{2.5}$$

donde $\hat{r}(k)$ son los coeficientes del vector de correlación y $r_a(k)$ está dado por:

$$r_a(k) = \sum_{k=0}^{p-n} a_k \cdot a_{k+n} \tag{2.6}$$

2.5.2. Espectrograma

Otra posibilidad para representar paramétricamente la señal de voz es por medio de sus características espectrales. El espectrograma es una herramienta de representación que se utiliza para el análisis de una señal en la que se visualiza el contenido frecuencial de la señal con respecto a su variación en el tiempo, como se muestra en la Figura 2.11.

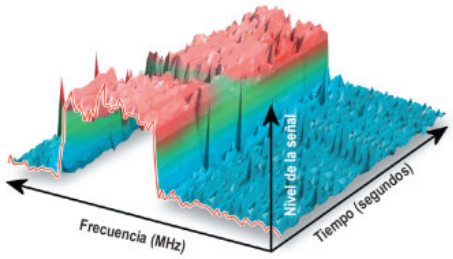


Figura 2.11: Representación en tres dimensiones del espectrograma. Imagen recuperada de: [Carrión, 2017].

El proceso para obtener el espectrograma consiste en seleccionar una determinada cantidad de muestras por medio de una ventana temporal y posteriormente se calcula su contenido espectral por medio de la transformada rápida de Fourier (en inglés, Fast Fourier Transform (FFT)) y se obtiene su espectro de potencias X_k.

$$x_k = \sum_{n=0}^{N-1} x_n e^{-j2\pi k \frac{n}{N}} \qquad k = 0, ..., N-1 \tag{2.7}$$

$$X_k = |x_k|^2 \tag{2.8}$$

Finalmente, la ventana se desplaza a lo largo la señal a analizar y se grafican los resultados en una representación de tres dimensiones. Este tipo de representación permite el análisis de la sonoridad, la duración, la estructura formante (timbre) y la amplitud de la señal.

2.5.3. Coeficientes MEL-Cepstrales

Finalmente, una alternativa a la representación espectral de la señal de voz es la representación cepstral. El nombre "*cepstrum*" es un anagrama de la palabra "*spectrum*" y se derivó invirtiendo las primeras cuatro letras. La representación MEL cepstral es una representación del espectro de potencia a corto plazo y este tipo de análisis puede usarse para separar la señal de excitación (que contiene el tono) y la función de transferencia (que contiene la calidad de la voz) [Ruedo, 2011].

Por otra parte, una extensión de esta representación son los coeficientes MEL-Cepstrales (en inglés, Mel Frequency Cepstral Coefficients (MFCC)), los cuales son una técnica popular porque se basa en la variación conocida del ancho de banda de frecuencia crítica del oído humano. Los estudios han demostrado que la percepción humana del contenido de la frecuencia del sonido para las señales del habla no sigue una escala lineal [Ramírez et al., 2019].

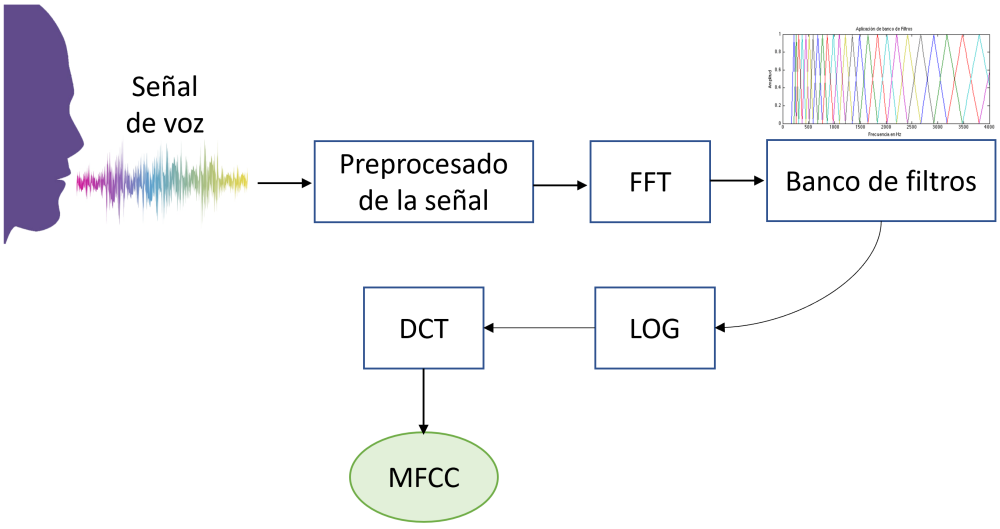


Figura 2.12: Diagrama de bloques del proceso para el cálculos de los MFCC.

Como se puede observar en la Figura 2.12, posterior a aplicar la ventana, se calcula la FFT de la señal. Debido a que la FFT asume que el audio es periódico y continúo, es necesario aplicar ventanas traslapadas en los bloques de la señal, lo cual ayudará a evitar las distorsiones en el dominio de la frecuencia.

$$x_m[k] = \sum_{n=0}^{n-1} x(n) e^{\frac{-2\pi jnk}{N}} \tag{2.9}$$

Una vez que se obtiene la potencia del espectro de la señal $X_m[k]$, esta se hace pasar por el banco de filtros en la escala MEL, esto brinda información sobre la potencia en cada banda de frecuencia. La escala MEL fue propuesta por Stevens, Volkmann y Newmann en 1937 y es una escala no lineal de tonos ordenada de acuerdo a cómo los humanos perciben el sonido. Los filtros V_r se utilizan para calcular una suma ponderada de componentes espectrales para filtrar la salida de modo que el proceso se acerque a la escala MEL [Martinez et al., 2012].

$$MF_m[r] = \sum_{k=0}^{k-1} V_r[k] X_m[k] \tag{2.10}$$

donde $V_r[k]$ indica la función del filtro MEL y r indica el número del filtro.

Finalmente, para cada ventana, m, se calcula la transformada coseno discreta II (en inglés, Discrete Cosine Transform (DCT)) del logaritmo de las salidas del filtro para obtener los MFCC [Maurya et al., 2017]. Se realiza este proceso para cambiar el espectro de MEL al dominio del tiempo.

$$mfcc_m[n] = \sum_{r=1}^{R} log(MF_m[r]) cos[\frac{\pi}{R}\left(r + \frac{1}{2}\right) n] \tag{2.11}$$

Por otra parte, dado que las características cepstrales se calculan tomando la DCT del espectro logarítmico, contienen información sobre los cambios de velocidad en las diferentes bandas del espectro. En otras palabras, en el dominio cepstral, la influencia de las cuerdas vocales (fuente) y el tracto vocal (filtro) en una señal se puede separar ya que la excitación de baja frecuencia y el filtrado de formantes del tracto vocal se localizan en diferentes regiones en el dominio cepstral [Hosom, 2003].

Por ejemplo, los coeficientes de orden inferior contienen la mayor parte de la información sobre la forma espectral general de la función de transferencia del filtro. Sin embargo, a pesar de que los coeficientes de orden superior representan niveles de detalles espectrales mayores, dependiendo de la frecuencia de muestreo y el método de estimación, entre 12 a 20 MFCC suelen ser óptimos para el análisis de voz [Rabiner, 1989]. Por el contrario, seleccionar un gran número de coeficientes cepstrales da como resultado una mayor complejidad en los modelos.

2.6. Sistemas de reconocimiento de voz

En esta sección se describirán algunos ejemplos de los sistemas utilizados para el reconocimiento de voz, haciendo énfasis en aquellos que han sido ampliamente utilizados. Entre ellos destacan la cuantización vectorial, los modelos ocultos de Markov y las redes neuronales.

2.6.1. Cuantización vectorial

La idea de aplicar la cuantización vectorial al área de reconocimiento de voz es representar a los vectores de entrenamiento, compuestos por los vectores de de características (LPC, MFCC, etcétera) de la voz previamente procesada, que pueden tomar infinitos valores en un continuo, por un conjunto finito de vectores que constituyen lo que se denomina un libro de código [Cano, 2003]. Con esta representación se logra una reducción considerable de la información.

Un cuantizador vectorial puede ser descompuesto en dos operaciones, la codificación vectorial y la decodificación vectorial, en la Figura 2.13 se muestra un diagrama de este proceso aplicado al reconocimiento de voz.

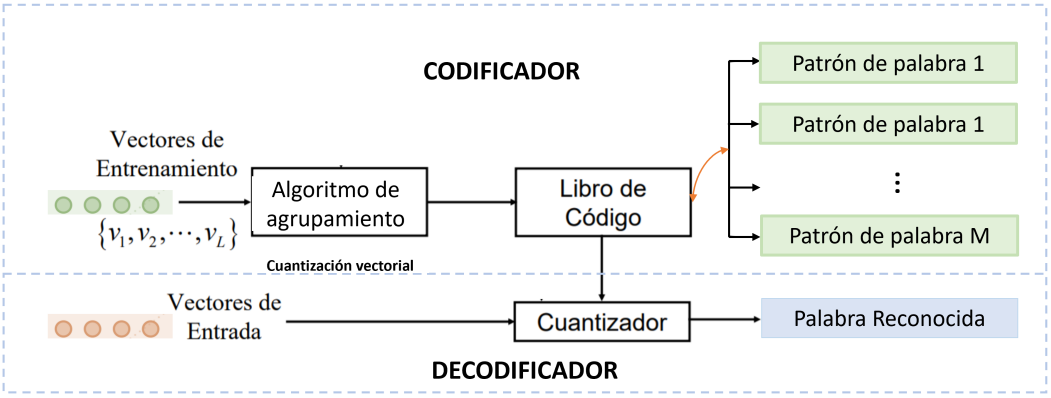


Figura 2.13: Cuantización vectorial aplicada al reconocimiento de voz.

Durante la etapa de codificación se deben generar M patrones de referencia basándose en N vectores de entrenamiento, donde $M < N$. Posteriormente, durante la etapa de decodificación, la palabra de entrada o la palabra que se desea reconocer es filtrada y se obtienen sus correspondientes vectores de características (LPC, MFCC, etcétera), los cuales se cuantizan usando cada uno de los patrones y se asigna la clase que provea una menor distorsión [Khan and Smith, 2005], es decir, se busca la mayor similitud del vector de entrada con los patrones previamente obtenidos. Si se denota con $C_n, 1 \leq n \leq N$ a los cuantizadores y con v al vector que se quiere reconocer, entonces el índice m correspondiente está dado por:

$$n = argmin\,[d(v, C_n)] \tag{2.12}$$

Los algoritmos que permiten clasificar un conjunto de N vectores de entrenamiento en M clases distintas (clusters) caracterizadas por un vector (los centroides de cada cluster) se denominan algoritmos de agrupamiento. Los algoritmos más utilizados son el algoritmo K-medias y el algoritmo de Linde-Buzo-Gray, el cual es un proceso iterativo que puede resumirse en los siguientes pasos [Savage and Rivera, 2021]:

1. Inicialización: se define el primer centroide C_i al calcular la media los N vectores.

$$C_i = \frac{1}{N} \sum_{j=1}^{N} x_j \tag{2.13}$$

2. Incremento de centroides: se generan dos nuevos centroides a partir de los centroides pasados al modificarlos por un valor ϵ.

$$C_{i+1} = C_i * \epsilon_1 \tag{2.14}$$

3. Agrupamiento: se asignan a cada cluster G_i los vectores más cercanos a cada centroide.

$$d_1 = distancia(x_j, C_{i+1}) \tag{2.15}$$

4. Actualización de centroides: se actualizan los nuevos centroides a partir de la media de los vectores asignados a cada cluster.

5. Convergencia: como se muestra en la Figura 2.14, es necesario repitir los pasos del 2 al 4 hasta que la distancia entre el centroide anterior y el nuevo sea menor que un umbral establecido. Lo que se traduce en que el nuevo centroide no difiere mucho del centroide anterior.

Finalmente, el centroide es el vector representativo de cada cluster (conjunto de vectores de entrenamiento), el cual funge como patron para el paso de reconocimiento.

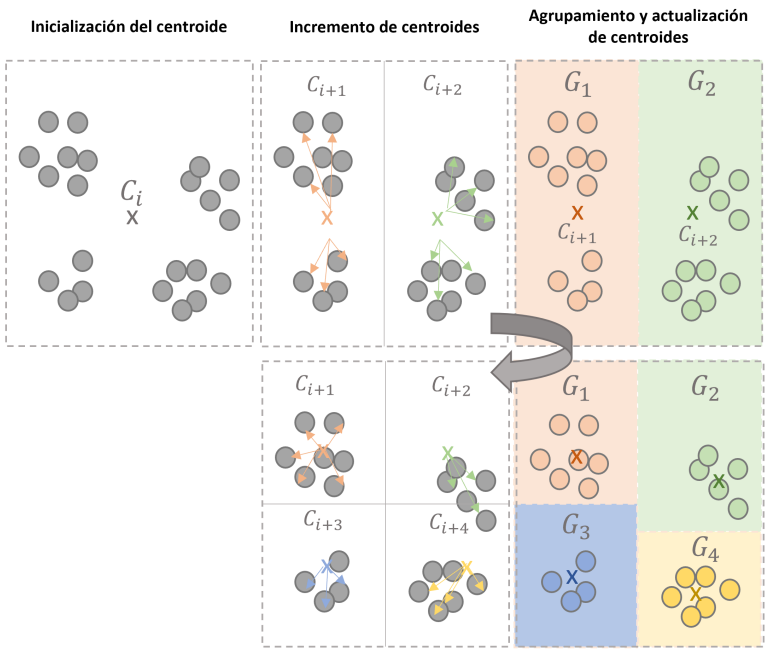


Figura 2.14: Esquema básico del algoritmo de Linde-Buzo-Gray. Imagen modificada de: [Amidi and Amidi, 2017].

2.6.2. Modelos ocultos de Markov

Los modelos ocultos de Markov (en inglés, Hidden Markov Models (HMM)) son una herramienta probabilística propuesta por Baum a principio de los años setenta, la cual contiene dos variables aleatorias; el estado S y la observación O, de las cuales, el estado S está oculto pero se tiene acceso a la observación O. Se puede analizar a un modelo λ como un conjunto de estados los cuales están conectados por probabilidades de transición entre estados (matriz A) y con una función de probabilidad de una observación en cada estado (matriz B) [Jurafsky and Martin, 2020].

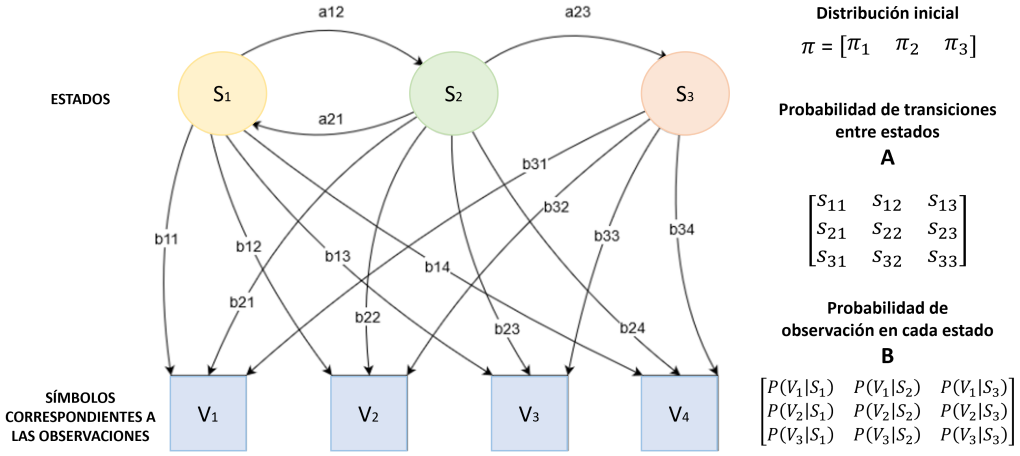


Figura 2.15: Diagrama de HMM.

Entonces el modelo completo requiere el número de estados N, el número de observaciones M, los símbolos de salida V y una especificación de las probabilidades de cambio de estado A; de símbolo observado B y la distribución de estado inicial π. Por conveniencia se usa la siguiente notación.

$$\lambda = (A, B, \pi) \tag{2.16}$$

En la Figura 2.15 se puede observar un resumen de un modelo con tres estados S_i y cuatro símbolos V_i correspondientes a las observaciones O_i y sus correspondientes matrices de transición y de observaciones. Teniendo en cuenta lo anterior existen tres problemas básicos de interés referentes al uso de los Modelos Ocultos de Markov [Tumilaar et al., 2015], los cuales son los siguientes:

1. **El problema de la evaluación:** consiste en que al tener un modelo λ y una secuencia de observaciones $O = (o_1, o_2...o_T)$; determinar de forma eficiente la probabilidad de que este modelo haya generado la secuencia de observaciones $P(O = \lambda)$. Por ejemplo, si se tiene una secuencia de observaciones es posible determinar si esta secuencia de observaciones corresponde al segmento de voz que describe el modelo.

2. **El problema de decodificación:** dado un modelo λ y una secuencia de observaciones $O = (o_1, o_2...o_T)$, se debe decidir la secuencia de estados $Q = (q_1, q_2, ...q_T)$ más probable en producir esta secuencia de observaciones. Este es útil cuando un modelo representa un segmentos de voz y y es necesario determinar cuales palabras están presentes en la secuencia de observaciones.

3. **El problema de aprendizaje:** dada una secuencia de observaciones y un modelo inicial, se debe ajustar este modelo para que la probabilidad de que este haya generado la secuencia de observaciones sea lo más alta posible. Esto permite generar modelos para utilizar en los dos problemas anteriores.

Con respecto a su aplicación en el reconocimiento de voz, los HMM fueron estudiados previamente por Rabiner [Rabiner, 1989], donde se propone remplazar el paso de cuantización vectorial para identificar una palabra por un HMM, al cual se le entrega el índice del centroide como una observación, a partir de los HMM se puede obtener el modelo más probable y por lo tanto la palabra más probable dentro de una trama.

2.6.3. Redes neuronales artificiales

Las redes neuronales artificiales (en inglés, Artificial Neural Network (ANN)) son un modelo matemático que busca simular características propias de los humanos. Con respecto al reconocimiento de voz, en la Figura 2.16 se ilustra un diagrama básico de este proceso donde las entradas a estas redes suelen ser vectores de características descriptoras de la señal, por otra parte las clases a reconocer (palabras)

son representadas por conjuntos de respuestas distribuidas sobre una red constituida por unidades de proceso (neuronas de capas ocultas en azul).

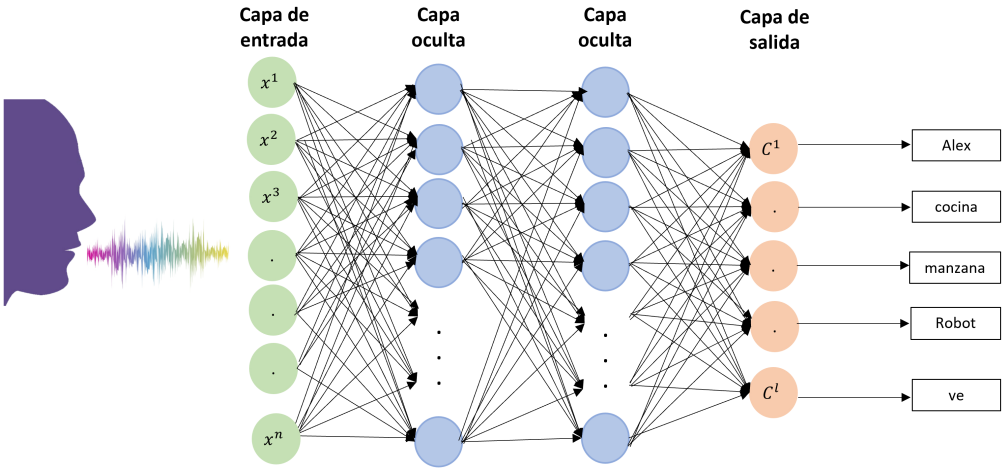


Figura 2.16: Diagrama de red neuronal para el reconocimiento de voz.

Dentro de este marco de trabajo se encuentran diversas arquitecturas que han sido utilizadas para la tarea de ASR. En general, estos sistemas están compuestos por modelos acústicos, de pronunciación y de lenguaje que se procesan de manera independiente, sin embargo, también existen los modelos secuencia a secuencia (RNN, Connectionist Temporal Classification (CTC), modelos de atención, etc) que permiten entrenar estos componentes de manera conjunta, ya que predicen fonemas o palabras directamente [Prabhavalkar et al., 2017].

En el siguiente capítulo se describe de manera más detallada la teoría detrás de las ANN. Por ahora, basta con mencionar que algunas de las redes propuestas inicialmente para el ASR han logrado mejorar los resultados de los sistemas anteriores en segmentos cortos de voz. Finalmente, es importante mencionar que también se han realizado aproximaciones híbridas en las que el procesamiento secuencial se realiza por sistemas tradicionales y el reconocimiento haciendo uso de modelos de ANN.

CAPÍTULO 3

Redes neuronales artificiales

Dentro de la inteligencia artificial (en inglés, Artificial Intelligence (AI)), una de las ramas más prometedoras es la que corresponde a las denominadas redes neuronales artificiales (en inglés, Artificial Neural Network (ANN)), las cuales son modelos computacionales que intentan emular el comportamiento del cerebro humano, caracterizado por el aprendizaje a través de la experiencia y la extracción de patrones a partir de un conjunto de datos. Estos modelos hacen uso de estructuras de procesamiento con capacidad de cálculo paralelo cuya estructura se aproxima a la red neuronal biológica [Hilera and Martínez, 1995].

3.1. Modelo de la neurona

El elemento fundamental de estos sistemas es la neurona, cuyo modelo matemático fue propuesto por [McCulloch and Pitts, 1943]. Como se muestra en la Figura 3.1, existen tres secciones de operaciones que se realizan en este bloque. La primer sección consta de la multiplicación de las entradas x_{ij} por los pesos correspondientes w_{ij}. El segundo, estos resultados son sumados, agregando además un umbral de activación o sesgo θ_j para conformar la salida de la neurona. Finalmente, esta salida se pasa a través de una función de activación σ, que produce la salida o_j.

$$o_j = \sigma(\sum_{i=1}^{N} w_{ij} \cdot x_{ij} + \theta_j) \tag{3.1}$$

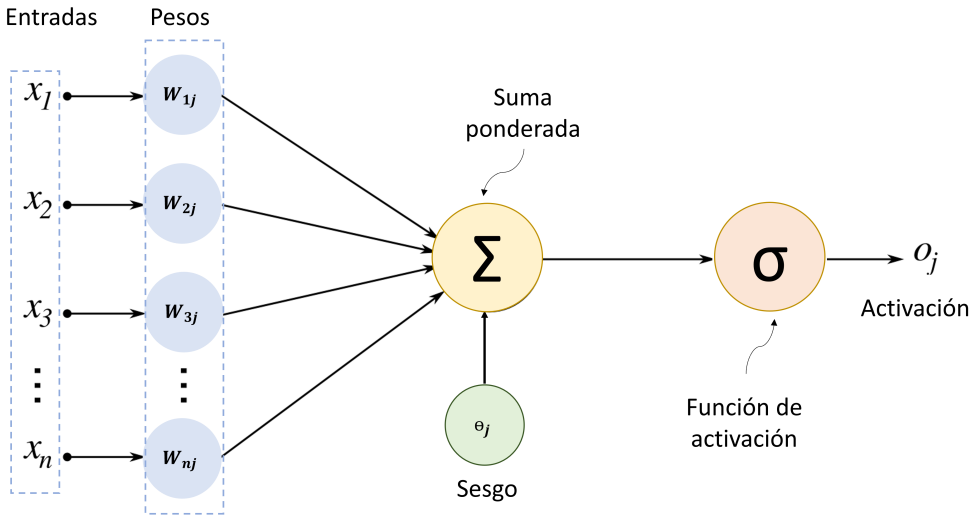


Figura 3.1: Modelo de neurona.

El peso sináptico w_{ij} define la importancia de una conexión entre dos neuronas. En caso de una entrada positiva, un peso positivo actúa como un amplificador o atenuador, un peso negativo como inhibidor y en caso de que el peso sea cero, esto implicaría que no existe comunicación entre este par de neuronas. Mediante el ajuste de estos pesos, la neurona es capaz de adaptarse y realizar la tarea para la que fue entrenada.

3.1.1. Funciones de activación

La función de activación determina el estado de una neurona en base a su valor resultante y tiene como objetivo el acotar los valores de salida de una neurona para mantenerlos dentro de ciertos rangos. La función de activación depende del tipo de red y de la tarea a realizar [Nwankpa et al., 2018]. Existen una gran variedad de funciones lineales y no lineales, en la Figura 3.2 se muestran las funciones de activación más comunes, en donde las funciones de activación se dividen en funciones lineales y no lineales (escalón, sigmoide, tangente hiperbólica, ReLU).

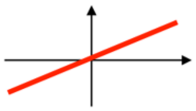
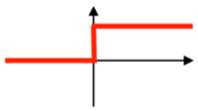
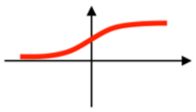
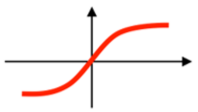
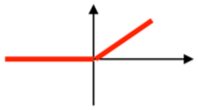
Función de activación	Ecuación	Gráfica
Lineal	$\phi(z) = z$	
Escalón unitario	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	
Sigmoide	$\phi(z) = \frac{1}{1 + e^{-z}}$	
Tangente hiperbólica	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	
ReLU	$RELU(x) = \begin{cases} 0 \text{ if } x < 0 \\ x \text{ if } x >= 0 \end{cases}$	

Figura 3.2: Funciones de activación.

Un resumen básico de las características de las funciones de activación se lista a continuación:

- Lineal: permite que los valores de la entrada sean iguales a los de salida, por lo que también es conocida como función identidad. Es utilizada si se requiere realizar un proceso de regresión lineal.
- Escalón unitario: es una función que permite clasificar los valores a partir de un valor umbral.
- Sigmoide: es una función no lineal que se encuentra en un rango de valores de salida está entre cero y uno, por ello su salida puede interpretarse como un valor de probabilidad.
- Tangente hiperbólica: tiene un rango de valores de salida entre -1 y 1. Se puede entender como un escalamiento de la función sigmoide.
- ReLU: es una función no lineal, la cual si tiene valores negativos de entrada el resultado es cero pero si tiene valores positivos el valor de entrada es igual al de salida.

3.2. Redes neuronales artificiales

A partir de este bloque básico, se pueden construir redes compuestas por millones de ellas y que están organizadas en estructuras de capas:

- Capa de entrada: es la capa en donde están ingresando los valores de entrada, que son los valores numéricos del proceso que se desea caracterizar.
- Capas ocultas: estas capas son las encargadas de la extracción de características necesarias para describir los patrones más significativos de las entradas.

- Capa de salida: es la capa final encargada de la clasificación o la interpretación de los datos de entrada.

En la actualidad existe una gran variedad de clasificaciones para las ANN, una de ellas es a partir de su topología, entre las cuales se distinguen la topología acíclica donde no hay retroalimentación de las neuronas de salida hacia alguna anterior y la topología cíclica la cual tiene ciclos de retroalimentación hacia capas anteriores.

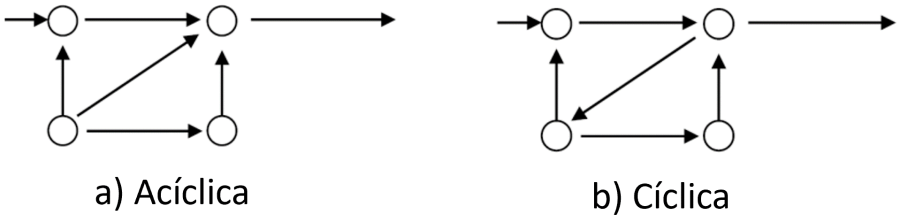


Figura 3.3: Clasificación de las ANN por su topología.

En general, una red neuronal puede ser caracterizada por el modelo de la neurona, su arquitectura, y el algoritmo de aprendizaje empleado para adaptar su función a las necesidades del problema a resolver.

3.3. Redes Feedforward

En este tipo de redes, las unidades de procesamiento están completamente conectadas a las unidades de la capa siguientes pero no a unidades en su misma capa o retroalimentando las anteriores. Sólo las salidas de las unidades en la capa presente están conectadas a las unidades de la siguiente capa [Schmidt et al., 1992], como se muestra en la Figura 3.4.

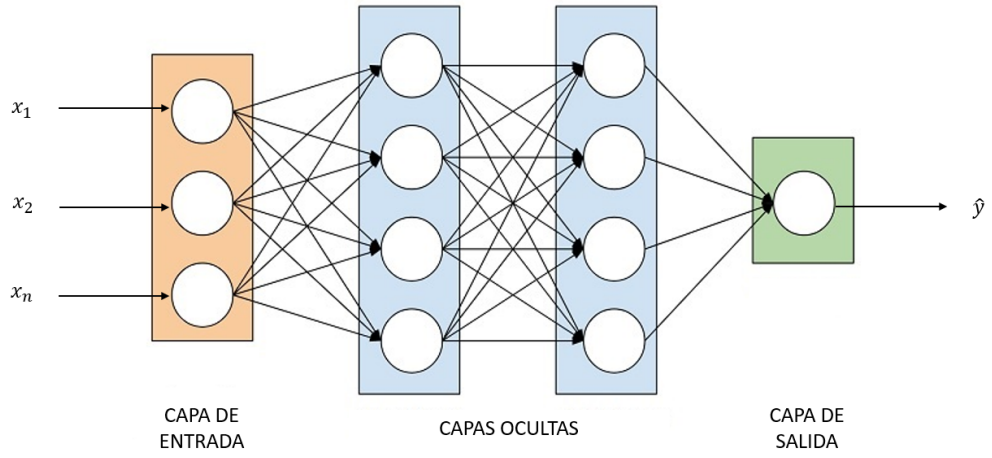


Figura 3.4: Arquitectura de ANN Feedforward. Imagen modificada de: [Loy, 2018].

La expresión para una red Feedforward multicapa queda descrita por la siguiente ecuación:

$$o_{jk}^{L} = \sigma(\sum_{i=1}^{N} w_{ij}^{L}\, o_{ij}^{L-1} + \theta_{ij}^{L}) \tag{3.2}$$

donde: la neurona j, en la capa L, para la muestra de entrenamiento k genera el valor o_{ij}^{L}, σ es la función de activación, w indica los pesos de la neurona y θ son los valores de sesgo.

Una de las fases más importantes en el aprendizaje automático es el entrenamiento de los modelos, ya que a partir de los resultados obtenidos en esta etapa dependerá el rendimiento general del modelo. Siendo en este momento donde entran en juego los métodos de optimización y los algoritmos de aprendizaje.

3.3.1. Retropropagación

La versión moderna de retropropagación del error (en inglés, Backpropagation) fue publicado en 1970 por Seppo Linnainmaa [Linnainmaa., 1976]. Posteriormente, el algoritmo de retropropagación del error fue formulado en 1986 por [Rumelhart et al., 1986], el cual es un algoritmo que ajusta de manera iterativa los pesos de las conexiones en la red para minimizar la función de costo y ajustar los parámetros de la red.

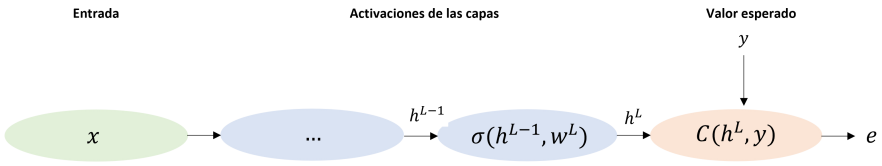


Figura 3.5: Diagrama del algoritmo de retro propagación. Imagen modificada de: [Shrivastava, 2018].

Suponiendo una red simple, con un par de capas y una sola neurona para cada capa, como el ejemplo de la Figura 3.5. Una vez aplicado un proceso de entrenamiento de la red, esta genera una salida z la cual genera un valor h una vez que pasa por la función de activación σ.

$$h^L = \sigma(w^L h^{L-1}) = \sigma(z^L) \tag{3.3}$$

Esta salida es comparada con la salida deseada y y se calcula la función de costo C para cada una de las salidas.

$$C^L = (h^L - y)^2 \tag{3.4}$$

El objetivo es minimizar la función de costo con respecto a los pesos w. Por lo tanto, para realizar las actualizaciones de los pesos es necesario encontrar el gradiente de esta función de costo con respecto a los parámetros de cada capa, para ello se utiliza la regla de la cadena [Zhang et al., 2020].

$$\frac{\partial C}{\partial w^L} = \frac{\partial z}{\partial w^L}\frac{\partial h^L}{\partial z^L}\frac{\partial C}{\partial h^L} \tag{3.5}$$

Sustituyendo, es posible observar que la activación de la capa pasada h^{L-1} tiene una influencia en la de la capa actual. Si se propaga este proceso hacia atrás con la finalidad de actualizar los pesos de cada neurona, esto permite que la red converja a un estado que le permita clasificar correctamente todos los patrones de entrenamiento.

$$\frac{\partial C}{\partial w^L} = 2(h^L - y^L)(h^{L-1})\sigma^{'} \tag{3.6}$$

Sin embargo, son necesarios algoritmos iterativos para realizar el calculo de la actualización de los pesos para cada neurona en cada capa.

Descenso por gradiente

El descenso por gradiente (en inglés, Gradient Descent) es un algoritmo de optimización utilizado para minimizar el valor de una función de costo C. Se calcula iterativamente en dirección al valor mínimo en cada paso, el cual está definido por el negativo del gradiente. Para determinar el siguiente punto a lo largo de la curva de la función de pérdida, el algoritmo de descenso de gradientes agrega alguna fracción de la magnitud del gradiente al punto de partida [Singer, 2016].

$$w_{ij}^{L}(t+1) = w_{ij}^{L}(t) + \eta \frac{\partial C}{\partial w_{ij}} \tag{3.7}$$

donde η es la tasa de aprendizaje, la cual es un hiperparámetro fundamental del algoritmo ya que es el responsable de la rapidez con la que el modelo resuelve el problema.

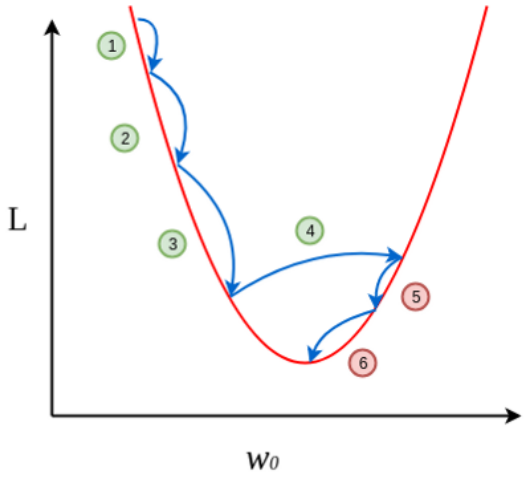


Figura 3.6: Descenso por gradiente. Imagen recuperada de: [Akshay, 2019].

Como se puede observar en la Figura 3.6, las tasas de aprendizaje más pequeñas requieren más épocas de entrenamiento (punto 1) dados los cambios más pequeños realizados en los pesos en cada actualización, mientras que las tasas de aprendizaje más grandes (punto 3) dan como resultado cambios rápidos y requieren menos épocas de entrenamiento para llegar al valor mínimo, sin embargo, debe buscarse un balance entre estos ya que si este hiperparámetro es muy grande (punto 5) podría no converger al valor mínimo. Generalmente tiene un pequeño valor positivo, a menudo en el rango entre 0 y 1.

3.4. Redes neuronales recurrentes

Las redes neuronales recurrentes (en inglés, Recurrent Neural Network (RNN)) pertenecen a la topología cíclica, las cuales son sistemas dinámicos, es decir, el cálculo de una entrada en un paso depende del paso anterior. Este tipo de redes son capaces de realizar gran variedad de tareas incluyendo: el reconocimiento de secuencias de datos, la reproducción o predicción de secuencias, la asociación temporal [Cruz et al., 2007]. A partir de las anteriores es posible utilizar este tipo de arquitecturas para el procesamiento del lenguaje natural.

Una RNN se puede considerar como copias múltiples de la misma red, cada una de las cuales pasa un mensaje a un sucesor. Generalmente los modelos de aprendizaje para sistemas dinámicos necesitan estados ocultos para almacenar la información contextual. En la Figura 3.7 se muestra un ejemplo de la arquitectura recurrente; en cada instante de tiempo t, esta neurona recurrente recibe la entrada del estado de tiempo actual x_t, así como la salida del instante de tiempo anterior h_{t-1} para generar su salida h_t, siguiendo la notación de la sección anterior, se podría expresar de la siguiente manera:

$$h_t = \sigma(x_t\, w_{xh} + h_{t-1}\, w_{hh} + b_h) \tag{3.8}$$

donde w_{hh} es la matriz de pesos que opera sobre el estado anterior h_{t-1}. Por otra parte, en el paso de tiempo t, la salida de la capa está descrita por la siguiente ecuación:

$$o_t = x_t\, w_{hq} + b_q \tag{3.9}$$

Los parámetros de la RNN incluyen los pesos w_{xh} y el sesgo b_h de la capa oculta, junto con los pesos w_{hq} y el sesgo b_q de la capa de salida. Incluso para diferentes pasos de tiempo, las RNN usan estos mismos parámetros del modelo.

Por lo tanto, el costo de parametrización de un RNN no aumenta a medida que aumenta el número de pasos de tiempo [Zhang et al., 2020].

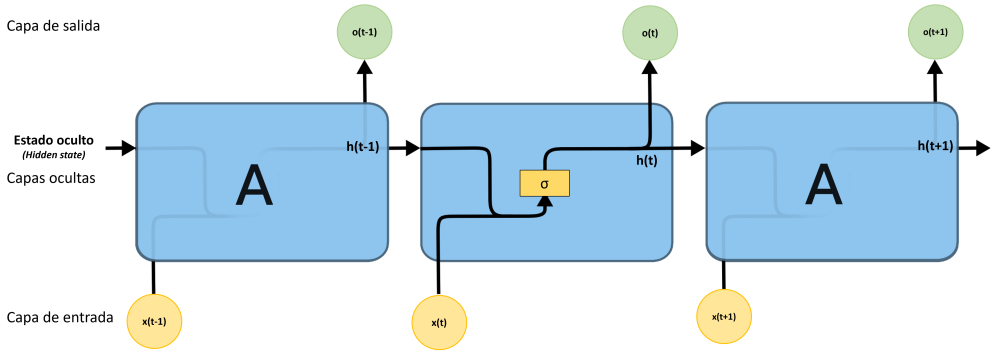


Figura 3.7: Arquitectura de una red neuronal recurrente. Imagen recuperada de [Olah, 2015].

Uno de los atractivos de las RNN es la idea de que podrían conectar la información anterior a la tarea actual. En los casos, donde el contexto que se necesita es pequeño, las RNN pueden aprender a usar la información pasada. Pero también hay casos en los que necesitamos más contexto, desafortunadamente, a medida que crece el contexto, las RNN se vuelven incapaces de aprender a conectar la información.

3.4.1. Retropropagación en el tiempo

Entrenar RNN implica alternar propagación hacia adelante y retropropagación a lo largo del tiempo. Recordando la estructura de éstas, los estados ocultos y las salidas en cada tiempo se pueden expresar de la siguiente manera:

$$h_t = f(x_t, h_{t-1}, w_h) \tag{3.10}$$

$$\hat{o}_t = f(h_t, w_o)$$

A partir de éstas ecuaciones se puede ver que estas estructuras presentan una cadena de valores que dependen de los valores anteriores $(x_t, h_t, \hat{o}_t)$. La diferencia entre la salida $\hat{o}_t$ y la etiqueta o_t es evaluada por una función de pérdida C a lo largo de los T pasos de tiempo.

$$C = \frac{1}{T} \sum_{t=1}^{T} c(o_t, \hat{o}_t) \tag{3.11}$$

Para realizar la retropropagación, se obtiene el gradiente de la siguiente manera:

$$\frac{\partial C}{\partial w} = \frac{1}{T} \sum_{t=1}^{T} \frac{\partial}{\partial w} c(o_t, \hat{o}_t) \tag{3.12}$$

Entonces, la retropropagación a lo largo del tiempo calcula y almacena el gradiente en turno. Este proceso es lento y el valor del gradiente puede desbordarse con cambios pequeños en las condiciones iniciales, afectando los valores de salida en gran medida [Zhang et al., 2020]. Los fenómenos de desvanecimiento y explosión de gradiente se encuentran a menudo en el contexto de las RNN.

La razón por la que ocurren es que es difícil capturar las dependencias a largo plazo debido al gradiente multiplicativo que puede aumentar o disminuir exponencialmente con respecto al número de capas. Como estrategia para resolver el problema de explosión del gradiente se encuentra el recorte de gradiente (en inglés, Gradient Clipping) el cual es un cambio de escala del gradiente [Qian et al., 2021].

$$g \leftarrow c \cdot \frac{g}{||g||} \tag{3.13}$$

Esta estrategia garantiza que el gradiente tenga la norma como máximo c. Esto ayuda a que el descenso del gradiente tenga un comportamiento razonable incluso si la función de pérdidas del modelo es irregular.

Aunque la explosión puede abordarse con estrategias de recorte simples, el desvanecimiento es un problema que requiere arquitecturas especiales para ser correctamente dirigido. Ya que si un valor de gradiente se vuelve extremadamente pequeño, no contribuye demasiado al aprendizaje. Un enfoque común se basa en los llamados Gated RNN, cuya idea central es introducir un mecanismo de activación para controlar mejor el flujo de información a través de los distintos pasos de tiempo.

3.4.2. Redes Long-Short Term Memory

Las redes de memoria a corto y largo plazo (en inglés, Long-Short Term Memory (LSTM)) son una extensión de las redes neuronales recurrentes, las cuales amplían su memoria y esto les permite aprender información de acuerdo a su relevancia sin importar que haya ocurrido un largo período de tiempo [Zhang et al., 2020]. Las LSTM también tienen esta estructura similar a una cadena de módulos repetidos como las RNN estándar, pero el módulo repetido tiene una estructura diferente: en lugar de tener una sola capa, tiene cuatro que interactúan entre si (F_t, I_t, $\tilde{C}_t$, O_t), como se puede observar en el diagrama de la Figura 3.8.

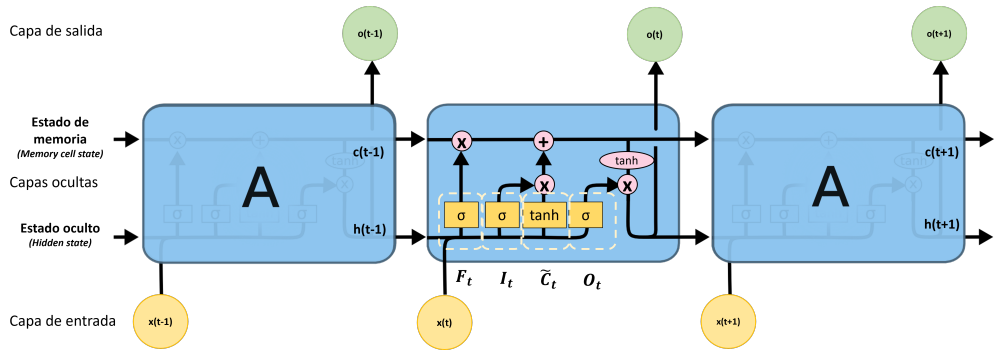


Figura 3.8: Arquitectura de una LSTM. Imagen modificada de [Afzal et al., 2010]

La clave de los LSTM es el estado de la celda, estos tienen la capacidad de eliminar o agregar información al estado de la celda del estado presente, lo cual es regulado cuidadosamente por estructuras llamadas compuertas.

Las entradas son procesadas por tres capas completamente conectadas con una función de activación sigmoide para calcular los valores de las compuertas: entrada I_t, olvido F_t y salida O_t. Como resultado, los valores de las tres puertas están en el rango de (0,1) [Zhang et al., 2020].

$$I_t = \sigma(x_t w_{xi} + h_{t-1} w_{hi} + b_i) \tag{3.14}$$

$$F_t = \sigma(x_t w_{xf} + h_{t-1} w_{hf} + b_f)$$
$$O_t = \sigma(x_t w_{xo} + h_{t-1} w_{ho} + b_o)$$

Por otra parte, el candidato de celda memoria $\tilde{C}_t$ utiliza una función $tannh$, la cual devuelve un valor en el rango de $(-1, 1)$

$$\tilde{C}_t = tanh(x_t w_{xc} + h_{t-1} w_{hc} + b_c) \tag{3.15}$$

En las LSTM se tienen dos compuertas dedicadas a regular el flujo de información; la puerta de entrada define si se toma en cuenta la nueva entrada a través del $\tilde{C}_t$ y la compuerta olvido determina el aporte de la celda de memoria del estado anterior C_{t-1} que se retendrá.

$$C_t = F_t \odot C_{t-1} + I_t \odot \tilde{C}_t \tag{3.16}$$

Este diseño se introduce para resolver el problema del desvanecimiento del gradiente y para capturar mejor las dependencias a largo plazo dentro de las secuencias. Finalmente, el estado oculto h_t se obtiene mediante la siguiente ecuación, el cual incluye la influencia de celda de estado actual y por tanto del resto de compuertas.

$$h_t = O_t \odot tanh(C_t) \tag{3.17}$$

CAPÍTULO 4

Sistema de reconocimiento de voz

4.1. Corpus de entrenamiento

La serie de pruebas para robots de servicio de propósito general (en inglés, General Purpose Service Robot (GPSR)) evalúa todas las habilidades de los robots que se requieren a lo largo de las etapas de la competencia de la @RoboCup. El robot tiene que resolver múltiples tareas que le serán asignadas por el usuario. Estos robots se distinguen por su capacidad para lograr una amplia variedad de objetivos posibles utilizando capacidades básicas como navegación, manipulación, percepción e interacción. Sin embargo, las acciones que deben realizar son elegidas aleatoriamente por los árbitros. En las competiciones de 2010 a 2013 se han utilizado variantes del generador GPSR para generar los comandos dados al robot.

En general, no existe un corpus del estilo GPSR de escala suficiente para evaluar la tarea de interés para este proyecto, por lo que se construyó un conjunto de datos sintéticos basados en el generador de comandos utilizado en la tarea Robot de servicio de propósito general de la RoboCup@Home internacional de 2018. En esta sección, se describen los detalles de la construcción de este conjunto de datos, su procesamiento y las arquitecturas que constituyen el sistema de ASR.

4.1.1. Generador de comandos

A partir del generador de comandos GPSR se crearon archivos de texto cuyo contenido son los comandos escritos. En el apéndice A se incluye una breve descripción de las modificaciones que se le aplicaron al generador para la obtención de estos archivos. Posteriormente, estos fueron ingresados al sintetizador de voz Google Text-to-Speech (gTTS), que permite generar un habla similar a la humana y genera un archivo en formatos de audio (véase apéndice B), un ejemplo de los datos que conforman el corpus se pueden observan en la Figura 4.1, el cual está compuesto por la señal de audio a traves del tiempo y su correspondiente transcripción en texto como título de la figura.

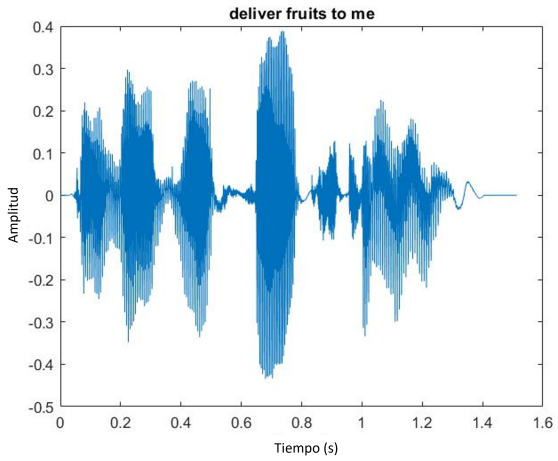


Figura 4.1: Ejemplo de las señales de voz.

Google Text-to-Speech permite crear voces sintéticas en audio reproducible a partir de texto. El primer bloque del sistema de síntesis tiene la función de convertir la entrada de texto impreso en un conjunto de sonidos que la máquina debe sintetizar. Una vez que se ha determinado la pronunciación correcta del texto, crea la secuencia de sonido adecuada para representar el mensaje. En esencia, el algoritmo de síntesis debe simular la acción del sistema del tracto vocal en la creación del habla [Google, 2021].

4.1.2. Aumentado de datos

El aumentado de datos es un método para generar datos sintéticos, es decir, crear nuevas muestras ajustando los factores de las muestras originales, lo que permite obtener una gran cantidad de datos para una sola muestra. Esto no solo ayuda a aumentar el tamaño del conjunto de datos, sino que también brinda múltiples variaciones de una sola muestra, lo que ayuda a disminuir el sobreajuste en un modelo de aprendizaje profundo.

Se utilizaron las siguientes estrategias para realizar el aumentado de datos de las señales de voz generadas por el GPSR.

- Inyección de ruido a la señal: este proceso implica la adición de ruido blanco a la muestra, para lograrlo, se generó un valor aleatorio dentro de la distribución anterior y se agregó a la muestra original. Con una relación señal a ruido (en inglés, Signal-to-Noise Ratio (SNR)) de 18.84, lo cual implica que la señal es aún claramente legible.
- Corrimiento en el tiempo: Esto moverá la señal hacia la derecha o izquierda por un factor dado a lo largo del eje del tiempo. Si se realiza un corrimiento del audio a la derecha por n muestras, las primeras n muestras se llenan con valores de 0.

El corpus de entrenamiento consta de 20 comandos generados a los cuales se les aplica una inyección de ruido blanco (para hacer una diferencia entre las señales sintéticas) y un desfasamiento con el fin de contemplar un mayor número de muestras posibles al momento de tratar la señal capturada. De estos desfasamientos se generan quince repeticiones, lo anterior da un total de 300 muestras de entrenamiento. En la Figura 4.2 se muestra la comparativa de la señal original (en la parte superior) y un ejemplo de los desfasamientos (en la parte inferior), donde se puede observar que lo único que se ve afectado es el corrimiento de la señal en el tiempo y los valores de amplitud se mantienen dentro del mismo rango entre una muestra y otra.

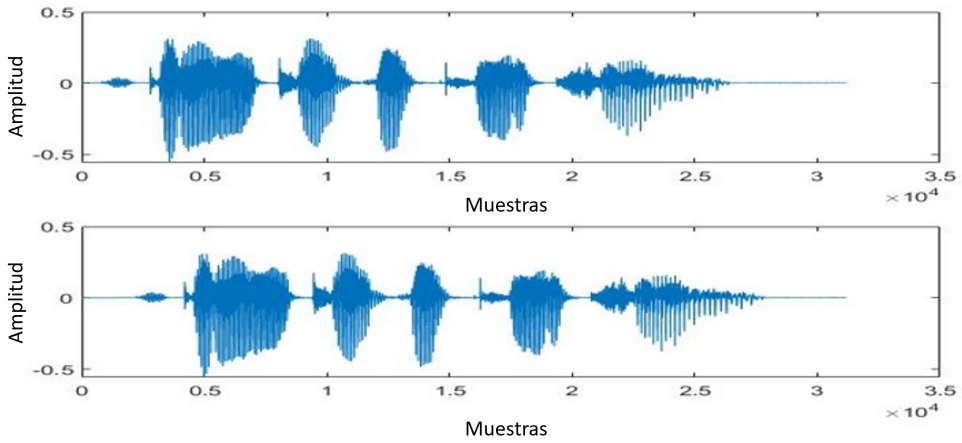


Figura 4.2: Aumentado de datos por inyección de ruido y corrimiento de la señal.

4.1.3. Procesamiento digital de la señal

Para este punto, es necesario realizar un procesamiento de la señal de voz, como el descrito en en capítulo 2. Esto se realiza a través de técnicas que permitan extraer la información acústica. Primero, se aplicó un filtro preénfasis de primer orden cuya función de transferencia es:

$$H(z) = 1 - 0.95z^{-1} \tag{4.1}$$

Posteriormente, se partió la señal en bloques y se aplicó una ventana Hamming, la cual elimina los problemas causados por los cambios rápidos de la señal en los extremos de cada trama de voz para conseguir transiciones suaves entre tramas. Una ventana Hamming se define como:

$$v(n) = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right) \tag{4.2}$$

donde N implica el tamaño de la ventana.

En la Figura 4.3 se puede observar el resultado de aplicar la ventana a tres bloques de la señal. La información original se concentra al inicio, específicamente dentro de las primeras 1000 muestras y se puede observar como al aplicar la ventana, la información se distribuye siguiendo la forma de la ventana aplicada.

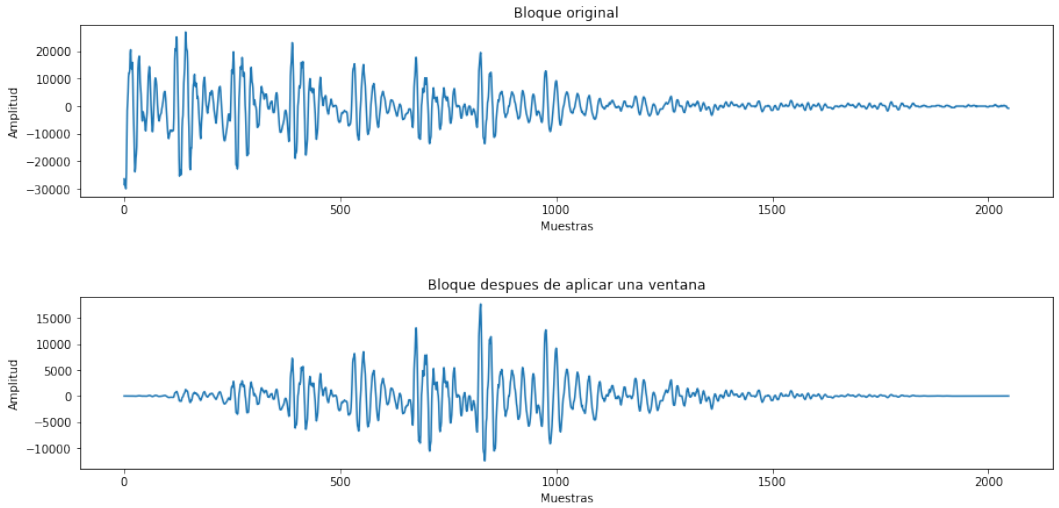


Figura 4.3: Comparación de bloques originales de la señal y después de aplicar la ventana de Hamming.

Por otra parte, el número de bloques quedó determinado por su frecuencia de muestreo f_s, en el caso de la base de datos es de 24 KHz. El tamaño de la ventana es del 3 % de la f_s y el traslape entre ventanas es del 2 % de la f_s. El número de bloques L resultantes puede ser determinado a partir de la siguiente expresión:

$$L = floor\left(\frac{T-N}{N-O}\right) + 1 \tag{4.3}$$

donde N implica el tamaño de la ventana, T el tamaño total de la señal y O el tamaño del traslape entre ventanas.

Como se mencionó en la sección 2.5.3 los sistemas de ASR tradicionales usan entre 8 y 13 coeficientes. Por lo tanto, para la extracción de características de los coeficientes LPC se ajustó el número de coeficientes obtenidos con el algoritmo Levinson-Durbin, para dejarlo establecido con 13 coeficientes de cada palabra.

De manera similar, se toman en cuenta sólo los primeros 13 coeficientes MFFC que son los que contienen mayor información acerca de la señal. Para más información sobre la documentación de la función $mfcc$ proporcionada por MATLAB visitar: https://www.mathworks.com/help/audio/ref/mfcc.html. De esta manera, independientemente si la palabra es larga en pronunciación o corta, el número de los coeficientes de los dos tipos de características será del mismo tamaño, lo que si dependerá del largo de la señal es L, un diagrama de la matriz obtenida se muestra en la Figura 4.4.

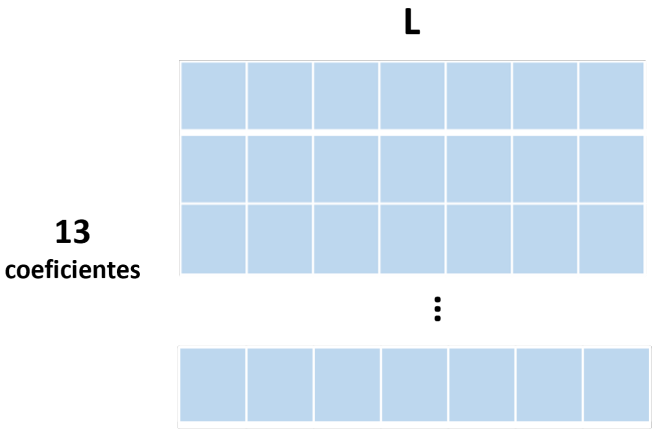


Figura 4.4: Matriz de características: 13 coeficientes x L bloques.

4.2. Etiquetado

Para realizar el etiquetado de los comandos de texto, estos fueron ingresados al módulo Parser Stanza el cual constituye una estructura de árbol de palabras a partir de la oración de entrada, las cuales representan dependencias sintácticas entre ellas [Stanford-NLP-Group, 2020]. El árbol resultante es útil ya que permite realizar el etiquetado de las palabras clave de manera automática, seleccionando sólo los verbos, pronombres, entre otros (verde) y asignándole una única etiqueta a las palabras que no son de interés para el reconocimiento (rojo), como se muestra en la Figura 4.5.

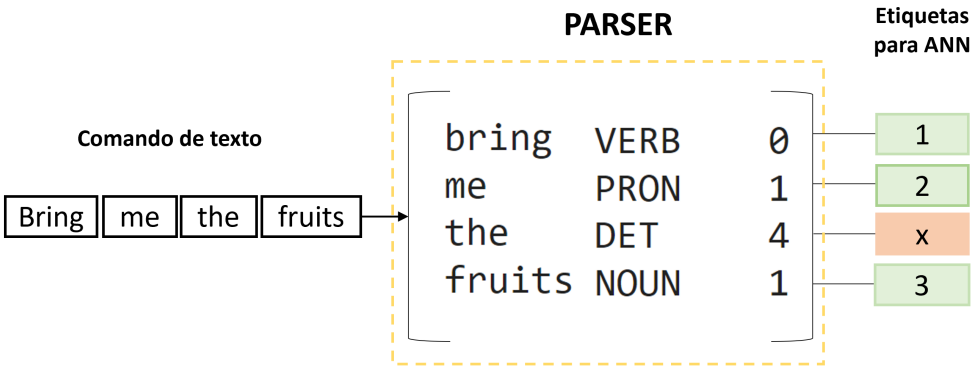


Figura 4.5: Proceso de etiquetado de los comandos de voz.

Posterior a este proceso se obtuvo un diccionario que contiene 44 palabras clave, entre verbos, pronombres, objetos, sustantivos y una clase exclusiva para las palabras desconocidas o que se encuentran fuera del vocabulario (véase apéndice A). Esto permite incluir en la matriz de confusión las etiquetas de referencia, haciendo más entendible la comparación de los resultados del sistema. Es decir, el corpus cuenta con 44 clases diferentes, ordenadas de manera alfabética.

4.2.1. Alineación de etiquetas

Por otra parte, cada una de estas las etiquetas corresponden a una porción de la señal, por lo que es necesario asignar a cada bloque de entrada una salida con la etiqueta correspondiente. Es decir, cualquier secuencia de la señal representada por los vectores de características T_x corresponde a una asignación en la secuencia de etiquetas T_y, por lo que obtendremos una salida por cada una de las entradas.

$$T_x = T_y \tag{4.4}$$

En la Figura 4.6 se observa una señal de entrenamiento, donde cada uno de los colores delimita el número de muestras que corresponden a cada una de las palabras que conforman el comando. Este proceso fue realizado de manera supervisada, indicando el inicio y fin de cada clase de manera manual para cada comando y posteriormente fue extendido para el resto de repeticiones.

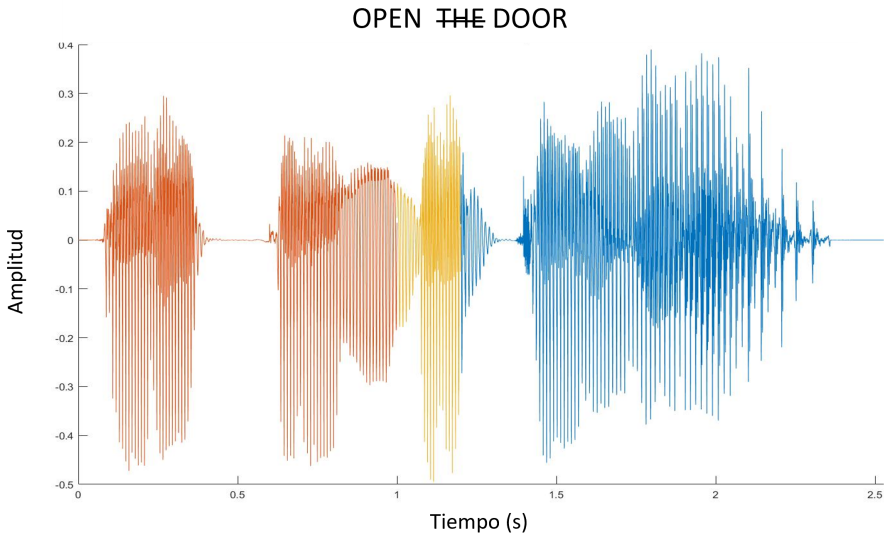


Figura 4.6: Alineación de las etiquetas de los comandos de voz.

4.3. Arquitectura

Para entrenar una red neuronal para clasificar cada paso de tiempo, se utilizó una red LSTM secuencia a secuencia. Este tipo de esquema de trabajo (muchos a muchos) permitió realizar diferentes predicciones para cada paso de tiempo indistintamente de los datos de la secuencia. Es decir la red tiene entradas nuevas T_x y su correspondiente salida T_y en cada estado de tiempo, este esquema se ilustra en la Figura 4.7.

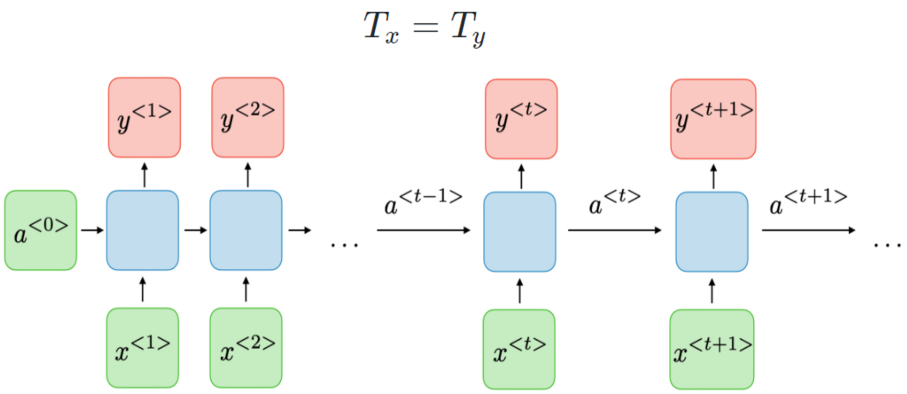


Figura 4.7: Esquema de tareas muchos a muchos. Imagen recuperada de: [Amidi, 2018].

Para este tipo de red los datos de cada bloque con 13 coeficientes fueron ingresados en cada uno de los estados de tiempo. Se tiene una capa de entrada secuencial, posteriormente se tienen capas recurrentes del tipo LSTM, una capa softmax para determinar la clase a la que pertenece la entrada y una capa de salida secuencial.

4.4. Reconocedor de palabras clave

Posteriormente, se debe realizar un postproceso a la salida de la red, ya que para una señal de voz de entrada, la secuencia de etiquetas estimadas tiene la misma longitud del número de bloques en el que se partió la señal durante su procesado. Sin embargo, esto estaría brindando más información de la requerida, ya que para el reconocimiento de palabras clave no es de vital importancia conocer el estado de tiempo exacto en el que se reconoce la palabra, sino simplemente las palabras claves predichas a lo largo de toda la señal de voz.

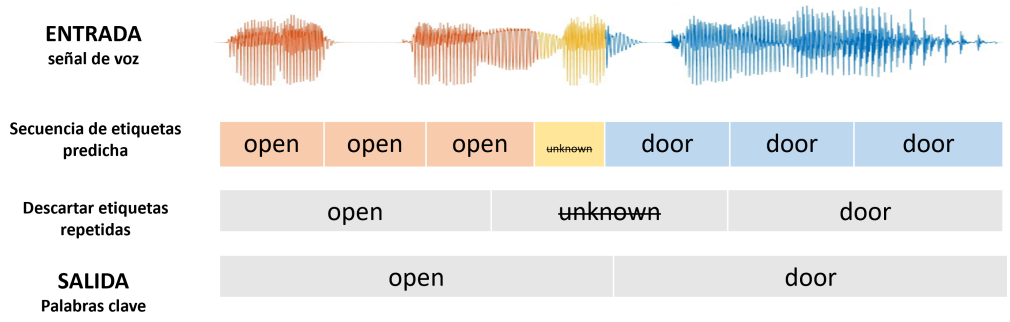


Figura 4.8: Diagrama básico del algoritmo de recuperación de palabras clave en un comando.

Por ello, a la secuencia estimada por el sistema es necesario aplicarle un proceso de descarte de las etiquetas repetidas pero respetando el orden de aparición de estas y posteriormente, eliminando las etiquetas de desconocido. Este proceso se resume en la Figura 4.8, donde puede observarse que el resultado final de aplicar este proceso será un vector que contiene únicamente las palabras claves del comando de voz de entrada.

CAPÍTULO 5

Pruebas y resultados

5.1. Pruebas de concepto

Se llevaron a cabo una serie de pruebas iniciales, las cuales consistieron en reconocer palabras aisladas por medio de distintos tipos de ANN. Cada palabra se procesa individualmente y el reconocimiento de una palabra no se ve afectado por otra. El reconocedor de palabras aisladas implica tomar una entrada de palabra en forma de archivo wav. A continuación, el archivo wav se procesa en una serie de vectores acústicos (LPC, MFCC, etcétera). El reconocedor de palabras requiere una unidad de reconocimiento que toma el vector acústico como entrada y lo asigna a la palabra correspondiente.

En específico el conjunto de datos consta de 10 palabras diferentes y 40 repeticiones para cada una. Posteriormente este conjunto se dividió en 75 % para entrenamiento y 25 % para prueba. Como se muestra en la Figura 5.1, los datos fueron mezclados aleatoriamente, utilizando una única iteración para crear los subconjuntos de entrenamiento y prueba.

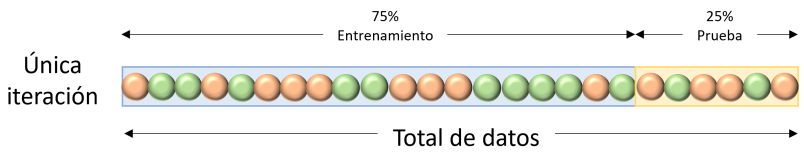


Figura 5.1: Definición de los subconjuntos de entrenamiento y prueba a partir del conjunto de datos original.

5.1.1. Redes FeedForward

Para este tipo de red los datos de cada bloque fueron concatenados en un solo vector, siendo este la entrada a la red. Donde todas las capas de la red son capas completamente conectadas, con sigmoides como funciones de activación en las capas ocultas y una función softmax en la capa de salida, la cual determinará la clase a la que los datos de entrada pertenecen. El entrenamiento se lleva a cabo por medio de retropropagación utilizando el cálculo del gradiente descendiente, un esquema muy general de esta arquitectura se ilustra en la Figura 5.2. Véase ápendice C para una mayor descripción del proceso de construcción de este tipo de red.

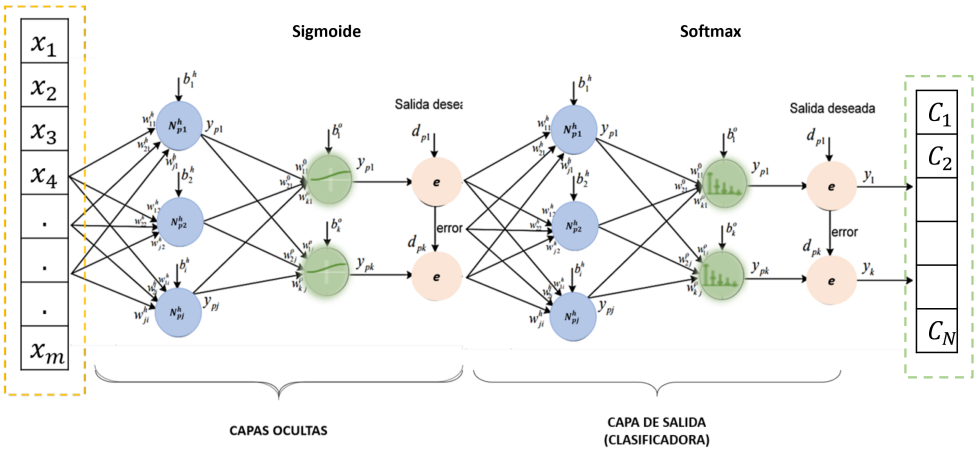


Figura 5.2: Diagrama general de la red neuronal feedforward clasificadora.

Por otra parte, en la Tabla 5.1 se resumen las características de la arquitectura utilizada para el reconocimiento de palabras aisladas, principalmente el número de neuronas definido en cada capa y sus respectivas funciones de activación. Estas características fueron determinadas al realizar pruebas con distintas cantidades de capas $[1, 2, 3, 4, 5]$ y de neuronas $[5, 10, 50, 100, 150, 200, 250, 500, 750, 1000]$, siendo la combinación de 2 capas ocultas y 250 neuronas por capa la que mostró un mayor desempeño al resolver la tarea de reconocimiento, ya que las estructuras más complejas presentaban sobreajuste en la base de datos de entrenamiento.

Capa	Cantidad de neuronas	Función de activación
Entrada	1300	
1era. capa	250	Sigmoide
2da. capa	250	Sigmoide
Salida	10	Softmax

Tabla 5.1: Características para la arquitectura FF.

5.1.2. Redes LSTM

Para este tipo de red se ingresaron los 13 coeficientes en cada estado de tiempo hasta terminar los valores de la señal, sin embargo, se obtiene una sola salida durante todo este proceso, a este esquema de trabajo se le conoce como tareas muchos a uno, la cual se ilustra en la Figura 5.3. Es decir la red tiene entradas nuevas en cada estado de tiempo T_x y una única salida T_y al terminar de procesar todos los estados.

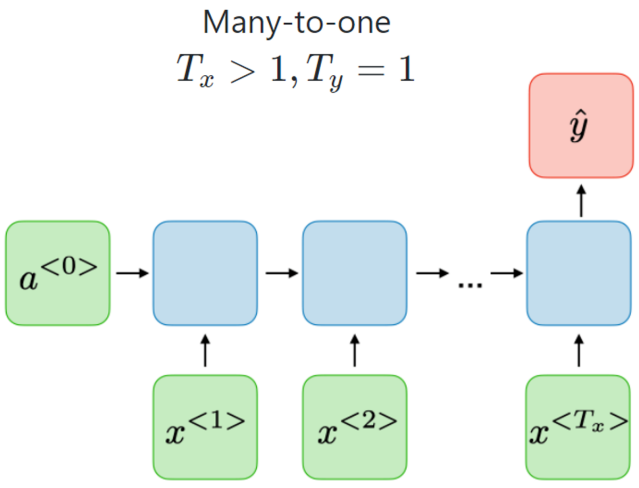


Figura 5.3: Esquema de tareas muchos a uno. Imagen recuperada de: [Amidi, 2018].

La red está conformada por una capa de entradas secuenciales, las capas ocultas están conformadas por celdas recurrentes del tipo LSTM y posteriormente los valores ingresan a una capa clasificadora con función de activación softmax la cual determinará la clase a la que los datos de entrada pertenecen. En la Tabla 5.2 se resumen las características de la arquitectura RNN. Estas características fueron determinadas al realizar pruebas con distintas cantidades de capas $[1, 2, 3, 4, 5]$ y un número de unidades ocultas de $[10, 50, 100, 150, 200, 250, 500]$, siendo la combinación de 1 capa oculta y 100 neuronas por capa la que mostró un mayor desempeño al resolver la tarea de reconocimiento.

Capa	**Cantidad de neuronas**	**Tipo de capa**
Entrada	13	
1era. capa	100	LSTM
2da. capa	10	Fully connected
Salida	10	Softmax

Tabla 5.2: Características para la arquitectura RNN.

En la Tabla 5.3 se comparan los porcentajes de exactitud obtenidos por los dos tipos de redes utilizando cada uno de los extractores de características. Como es posible distinguir, el mejor desempeño se obtuvo al combinar la arquitectura de las LSTM con los coeficientes MFCC, con los cuales se obtuvo un porcentaje de exactitud del 100 % en las pruebas de concepto. Por ello, se utilizará esta combinación de arquitectura y extractor de características para las pruebas posteriores.

	FeedForward	**LSTM**
Espectrograma	68 %	18 %
LPC	84 %	84 %
MFCC	97 %	100 %

Tabla 5.3: Resultados de pruebas de concepto.

5.2. Reconocimiento de comandos de voz

Para este tipo de pruebas se utilizó la base de datos descrita en el capítulo anterior, contienen los datos de los veinte comandos de voz sintética generados a partir del GPSR. Para crear los subconjuntos se siguió el mismo proceso que en las pruebas de concepto, es decir, el conjunto de datos se dividió en 75 % para el conjunto de entrenamiento y 25 % para el conjunto de prueba y fue mezclado aleatoriamente antes de realizar el entrenamiento de la red.

Entrenamiento	**Prueba**
300	100

Tabla 5.4: Tamaño de la base de datos para el reconocimiento de comandos de voz.

La red está conformada por una capa de entradas secuenciales, las capas ocultas están conformadas por celdas recurrentes del tipo LSTM secuencial y posteriormente los valores ingresan a una capa clasificadora con función de activación softmax la cual determinará la clase a la que los datos de entrada pertenecen. En la Tabla 5.5 se resumen las características de la arquitectura RNN.

Capa	**Cantidad de neuronas**	**Tipo de capa**
Entrada	13	
1era capa	150	LSTM
2da capa	44	Fully connected
Salida	44	Softmax

Tabla 5.5: Características para la arquitectura LSTM.

Como primeros resultados, se pueden analizar las predicciones en cada estado de tiempo mediante diagramas como el de la Figura 5.4, en donde la línea naranja representa los valores de las etiquetas reales y la línea azul representa los valores estimados por el modelo propuesto. En el eje vertical se listan las 44 clases que pueden ser reconocidas en orden alfabético y el eje horizontal representa el estado de tiempo en el que se reconoce una clase.

Si se analiza individualmente, el comando ***Justina, please follow Stephany*** (con exactitud del 98.29 %) es posible observar que aproximadamente en el estado de tiempo 150 se muestra un desfase en el reconocimiento; la etiqueta real indica que se está reconociendo una nueva palabra ***follow*** mientras que la etiqueta estimada indica que se está reconociendo aún la palabra ***unknown***, la cual representa la clase anterior.

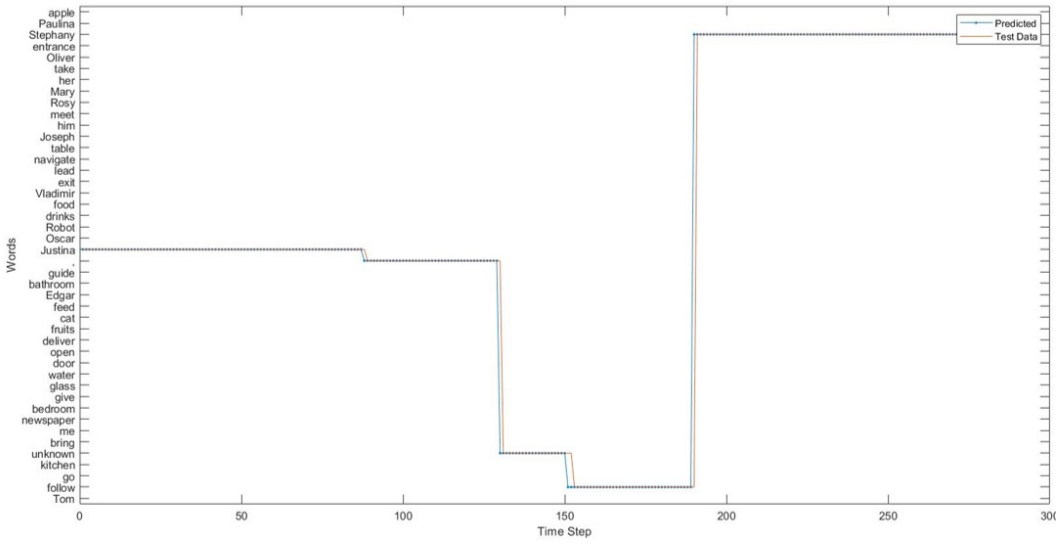


Figura 5.4: Reconocimiento de voz por estado de tiempo del comando ***Justina, please follow Stephany***.

Recordando que el corpus cuenta con 44 clases, donde las 43 de ellas representan cada una de las palabras claves que conforman los comandos y una clase más representa las palabras desconocidas, todas estas clases se ordenaron en orden alfabético para una representación más ordenada. Entonces, en la Figura 5.5 se presentan dos matrices de confusión, la matriz superior muestra los resultados obtenidos para las primeras 22 clases y la matriz inferior muestra los resultados obtenidos para el resto de las clases.

Estas matrices representan los resultados obtenidos al comparar en cada estado de tiempo las salidas estimadas del modelo y los comandos de referencia, la cual obtuvo una exactitud general de reconocimiento del 96.12 %. El valor en la diagonal (azul) indica el porcentaje de aciertos para cada clase y el resto de los valores (naranja) a lo largo del renglón de una clase indican el porcentaje de errores.

Sin embargo, este tipo de matriz ilustra un análisis básico, es decir, los errores que se muestran en la matriz ocurren porque en el mismo estado de tiempo la clase estimada es diferente a la clase verdadera, lo cual no permite distinguir si se está confundiendo con alguna otra clase o simplemente existe un desfase en el reconocimiento, como el mostrado en el el estado de tiempo 150 de la Figura 5.4.

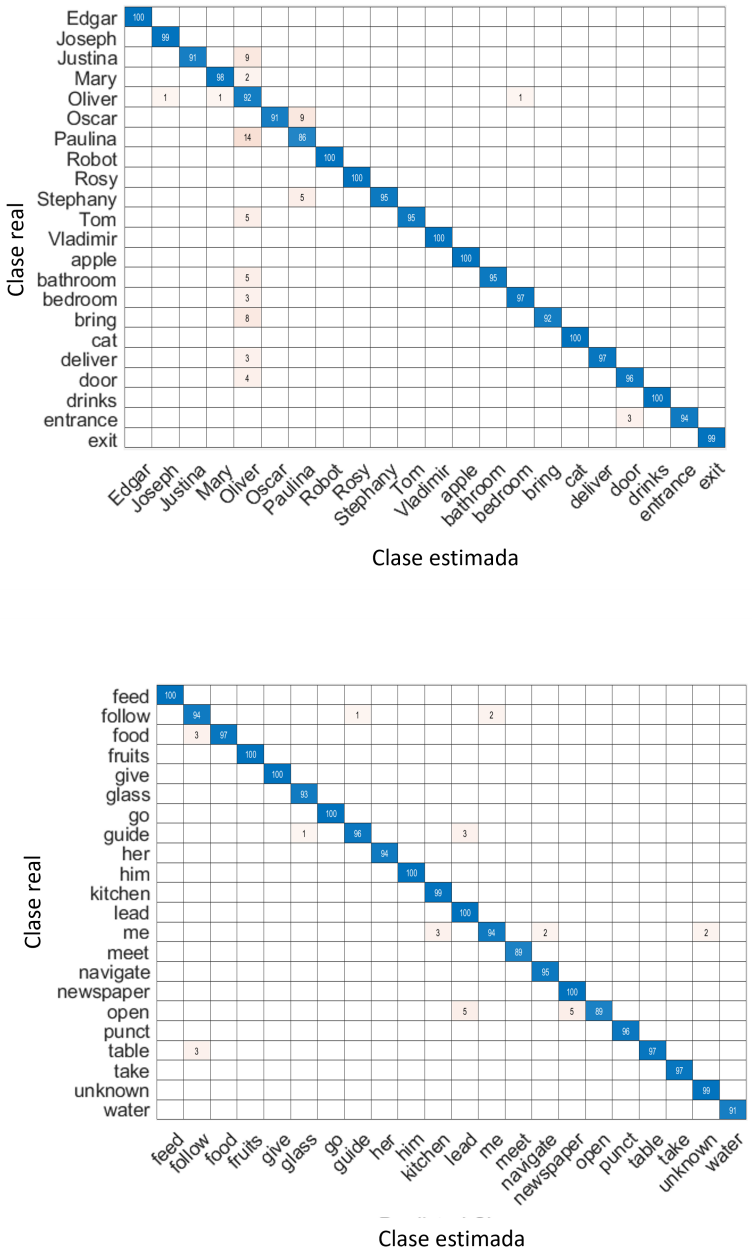


Figura 5.5: Matrices de confusión para el reconocimiento de comandos de voz. Superior: representa los resultados de las primeras 22 clases. Inferior: representa los resultados de las últimas 22 clases.

5.2.1. Word Error Rate

La exactitud del análisis anterior no es completamente representativa para el reconocimiento de palabras clave de un comando de voz. Una métrica más apropiada sería la tasa de error de palabras (en inglés, Word Error Rate (WER)), esta es una métrica común del rendimiento de un reconocimiento de voz. Calcula el número mínimo errores entre una frase generada por el sistema (con una longitud de N palabras) y una frase de referencia correcta: mide el número de inserciones I, eliminados E y sustituciones S de una palabra por otra.

$$WER = \frac{S + E + I}{N} \tag{5.1}$$

En la Figura 5.6 se muestran los dos comandos en donde ocurrieron errores del tipo S (amarillo) e I (azul) en todo el conjunto de datos. En el primer comando ***go to the kitchen*** se insertó la palabra ***meet*** al incio y se sustituyo una palabra desconocida por la palabra ***bring***. En el segundo comando ***deliver fruits to me*** se insertó la palabra ***him*** al final de la oración. El WER obtenido por el sistema es igual a 0.67 % que representa 3 errores de las 455 palabras a reconocer dentro de los 20 diferentes comandos.

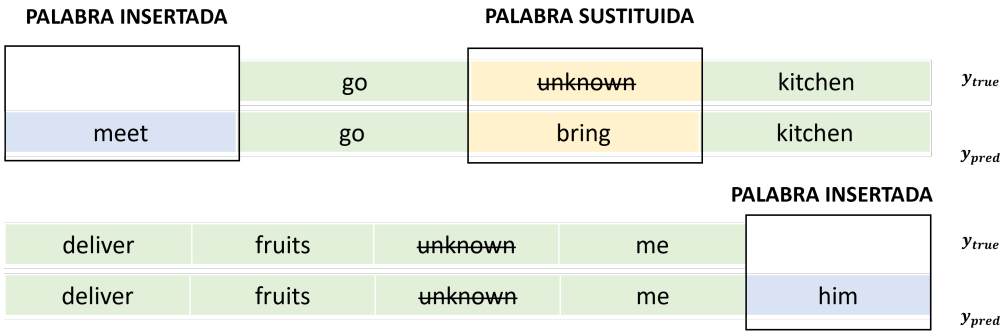


Figura 5.6: Ejemplos de los tipos de error contenidos en los comandos resultantes.

5.3. Prueba de CD

Finalmente, se realizaron pruebas para generar las estructuras de CD a partir de las palabras claves reconocidas por el sistema, utilizando estas palabras para completar los espacios de la plantilla de la primitiva con los roles de los participantes en la oración. Para ello, el primer paso fue ingresar las palabras clave al módulo Parser-Stanza para determinar las relaciones gramaticales de cada una de las palabras y por tanto distinguir entre los elementos de la primitiva.

Posteriormente, se identificó el verbo de la oración y esto permitió determinar la primitiva correspondiente. Para completar los espacios de la plantilla se buscó dentro de los tipos de palabras (sustantivos, pronombres, etc) y dentro de la base de conocimientos, esta última contiene la información de identificación y ubicación de un objeto. En la Figura 5.7 se muestran un par de ejemplos con los resultados obtenidos a partir de las palabras clave reconocidas.

```
[Sentence  BRING ME THE NEWSPAPER]
bring           bring           VERB    0       root
me              I               PRON    1       iobj
newspaper       newspaper       NOUN    1       obj
ATRANS((ACTOR Robot) (OBJECT newspaper) (FROM livingroom) (TO me))

[Sentence  GUIDE EDGAR TO THE BATHROOM]
guide           guide           VERB    0       root
Edgar           Edgar           PROPN   3       compound
bathroom        bathroom        NOUN    1       obj
PTRANS((ACTOR Robot) (OBJECT Edgar) (FROM Edgars place) (TO bathroom))

[Sentence  ROBOT, GO TO THE KITCHEN, MEET PAULINA AND DELIVER AN APPLE TO HER]
Robot           robot           NOUN    3       vocative
,               ,               PUNCT   1       punct
go              go              VERB    0       root
kitchen         kitchen         NOUN    5       obj
meet            meet            VERB    3       parataxis
Paulina         Paulina         PROPN   5       obj
deliver         deliver         VERB    5       xcomp
apple           apple           NOUN    9       compound
her             she             PRON    7       obj
PTRANS((ACTOR Robot) (OBJECT Robot) (FROM Robots place) (TO kitchen))
PTRANS((ACTOR Robot) (OBJECT Robot) (FROM Robots place) (TO Paulinas place))
ATRANS((ACTOR Robot) (OBJECT apple) (FROM kitchen) (TO Paulina))
```

Figura 5.7: Ejemplos de las primitivas obtenidas a partir de las palabras clave reconocidas por el sistema.

La oración subrayada, muestra el comando original a reconocer por el sistema, en la parte inferior se muestra el listado de palabras claves reconocidas con la clase gramatical asociada a cada una de ellas y al final se incluye la primitiva con los elementos de la plantilla completos. En el primer ejemplo se tiene el comando ***bring me the newspaper*** el cual resulta en la primitiva del tipo ***ATRANS()*** la cual está asociada a la transferencia de posesión o ubicación de un objeto.

Como se puede observar a partir de los ejemplos, en general el sistema obtuvo un porcentaje de acierto del 85 % al momento de transformar las palabras clave en estructuras de CD. El 15 % restante está representado por verbos muy específicos contenidos en la base de datos, como en el caso de ***feed the cat*** debido a que en esta oración no es posible determinar a partir de las palabras clave el objeto que se debe entregar, que correspondería a ***comida de gato*** y por tanto no es posible acceder a la base de conocimientos del objeto.

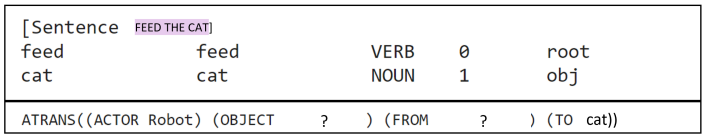


Figura 5.8: Ejemplo de errores al obtener la estructura de la primitiva.

5.4. Carga computacional

A lo largo del presente documento se han ido estableciendo las bases que constituyen este proyecto. Se ha descrito el proceso de obtención de datos, entrenamiento de la red y el postproceso para la obtención de las palabras clave, explicando sus ventajas e inconvenientes. A continuación se incluye el análisis de su rendimiento con respecto al coste computacional. En la Tabla 5.6 se indica la carga computacional (expresada en milisegundos) para cada uno de los procesos que componen este proyecto.

El entrenamiento de la red se llevó a cabo utilizando el Toolbox de Deep Learning proporcionado por MATLAB y el corpus de entrenamiento de 300 comandos. Por otra parte, los tiempos para el reconocimiento y recuperación de palabras clave KW fueron calculados al procesar cada uno de los 100 comandos que componen el corpus de prueba y obteniendo su promedio.

Proceso	**Carga computacional [ms]**
Entrenamiento de la red	188,030 ± 98,217
Reconocimiento de voz	935 ± 574
Recuperación de KW	0.193 ± 0.135
Conversión CD	878.500 ± 253

Tabla 5.6: Carga computacional de los procesos que componen el reconocimiento de voz.

Por otra lado, el término tiempo real se utiliza ampliamente en muchos contextos, tanto técnicos como convencionales. Sin embargo, el Random House Dictionary of the English Language [Random-House-Dictionary, 1987] define como tiempo real al tiempo de respuesta apropiado que requiera el proceso que se está controlando.

Específicamente en robótica, el tiempo de respuesta se mide desde el momento en que el usuario da un comando de voz hasta que el robot provee una respuesta [Laplante and Seppo, 2012]. Sin embargo, esto involucra procesos adicionales al reconocimiento de voz. Por lo tanto, en este contexto se acotará el tiempo real al tiempo de respuesta desde el reconocimiento de palabras a partir de una señal de voz hasta cuando se obtienen las palabras clave (sin ningún tipo de análisis consecuente).

Los algoritmos se desarrollaron y se probaron en una computadora portátil Lenovo Legión Y540, con procesador Intel Core ™ i7-9750H. Como se puede observar el tiempo de entrenamiento es de aproximadamente 3 minutos. Por otra parte, el proceso de reconocimiento de palabras clave consume un tiempo de aproximadamente 935 milisegundos.

CAPÍTULO 6

Conclusiones

El proceso de obtener de manera automática los comandos a partir del generador de comandos GPSR de la @*Robocup* permite obtener una base de datos posible de replicar para futuros proyectos. Otra ventaja es que podemos obtener una base de datos de buen tamaño y de calidad en un tiempo menor al que requeriría grabar la misma cantidad de muestras.

Con base en las pruebas realizadas se observó una mejora del desempeño del sistema proporcional al tamaño y calidad de los datos. Es decir, para realizar un sistema utilizando estrategias de inteligencia artificial no basta sólo con tener una gran cantidad de datos, sino que es aún más importante el poder determinar las características relevantes para la tarea a resolver, aplicar las técnicas adecuadas de procesamiento de la señal y realizar un proceso de etiquetado pertinente.

A partir de las pruebas, fue posible notar que para datos sintéticos es necesario aplicar un proceso de aumentado de datos, con el fin de darle generalidad al sistema de reconocimiento. De lo contrario, el sistema estaría aprendiendo patrones muy específicos siendo las señales de voz similares.

Durante las pruebas de concepto, se exploraron varias alternativas con respecto a la extracción de características. También se realizó un análisis entre el tipo de redes FeedForward y recurrentes y a partir de este, se aplicaron las ventajas de cada uno de estos para intentar solucionar el reconocimiento de voz continua que representaba una tarea más complicada. Se determinó que la combinación entre RNN y MFCC resulta en un mejor desempeño respecto al resto de propuestas.

Una de las ventajas principales que presenta la RNN con respecto a las redes FeedForward es que tiene en cuenta información histórica, es decir nos permite incluir el contexto de las muestras anteriores. Por otra parte, las redes FeedForward pierden el contexto al realizar el aplanado de la matriz de características. Otra ventaja de las RNN es que permiten procesar secuencias de diferente longitud para cada señal (dado por el número de bloques), siempre y cuando se respete el tamaño de entrada en cada estado de tiempo (el cual corresponde a 13).

El modelo final presenta un ventaja con respecto a las pruebas de concepto porque permite reconocer palabras clave en una señal de voz continua. Obteniendo resultados del 96.12 % para cada estado de tiempo y un WER menor al 1 % para el reconocimiento de palabras clave.

Las pruebas de CD mostraron que es posible utilizar las palabras clave reconocidas por un ASR para completar los elementos de una primitiva. Lo que permite construir un puente entre el lenguaje natural del humano y el lenguaje basado en reglas utilizado en un robot de servicio.

El sistema desarrollado no depende del uso de un servidor externo para realizar el reconocimiento, ni requiere estar conectado a la red, lo cual supone una ventaja al poder realizar el entrenamiento de manera local. Por otra parte, al comparar con sistemas de reconocimiento de voz enfocados en robótica [Choi and Kim, 2006] cuyo tiempo de respuesta es menor a 1 s, se concluye que el reconocimiento de voz tiene un tiempo de respuesta corto, que podría ser considerado tiempo real en aproximadamente el 80 % de las pruebas, pero el paso subsecuente de generar las estructuras CD implica un tiempo de respuesta largo que no entra dentro de este definición.

6.1. Trabajos a futuro

1. Como se mencionó en la sección de conclusiones, se observó una mejora del desempeño del sistema proporcional al número de datos contenido en la base de datos y si además se toman en cuenta las limitaciones que presenta la voz sintética, en especifico su poca variabilidad. Se sugiere como trabajo a futuro una mejora en el contenido del corpus; aumentar la cantidad de datos, incluir nuevos hablantes, incluir nuevas técnicas de aumentado de datos, lo cual permitirá aumentar la robustez del sistema de reconocimiento.

2. Por otra parte, una de las tareas más laboriosas durante la realización del proyecto consistió en el etiquetado de la señal en cada estado de tiempo, este es un proceso realizado de manera manual por lo que requiere una gran cantidad de tiempo. Posteriormente, se planteó un sistema de etiquetado automático, ilustrado en la Figura 6.1, el cual consiste en analizar las propiedades de los coeficientes MFCC en cada bloque en el que fue partida la señal a partir del método de agrupación K-medias; el cual permite distinguir entre bloques que contienen voz y silencio.

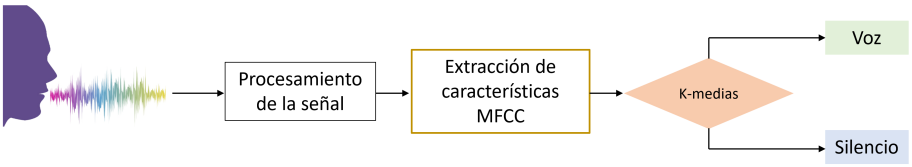


Figura 6.1: Diagrama del sistema de segmentación de voz.

A partir de este algoritmo fue posible etiquetar algunas señales de manera aceptable, sin embargo, el sistema no logró etiquetar todas las señales que le fueron asignadas, debido a las características propias de la voz; no existen silencios marcados entre palabras y algunos segmentos de voz sordos son confundidos con silencios. Sin embargo, se considera un punto de partida para optimizar el proceso de etiquetado, el cual podría ser abordado en proyectos futuros.

3. Otra alternativa para reducir la carga de trabajo que implica la segmentación previa de la señal, sería abordar la tarea del reconocimiento de voz utilizando una arquitectura del tipo CTC, la cual no requiere un pre-etiquetado de la señal y tampoco un post-procesado de las salidas, sin embargo, es importante considerar que al ser una estructura más compleja aumentará el tiempo y el trabajo de cómputo.

Glosario

AI Artificial Intelligence. 12, 33

ANN Artificial Neural Network. 4, 31–33, 37

ASR Automatic Speech Recognition. 2, 15, 16, 20, 52, 70

CD Conceptual Dependency. 14, 15, 66, 67, 70

CNN Convolutional Neural Network. 4

CTC Connectionist Temporal Classification. 32

DCT Discrete Cosine Transform. 25

DNN Deep Neural Network. 4

FFT Fast Fourier Transform. 23

FIR Finite Impulse Response. 21

GPSR General Purpose Service Robot. 47, 75

gTTS Google Text-to-Speech. 48, 79

HMM Hidden Markov Models. 3, 29

KW Keyword. 68

KWS Keyword Spotting. 4

LPC Linear Predictive Coding. 3, 20, 57

LSTM Long-Short Term Memory. 45, 60

MFCC Mel Frequency Cepstral Coefficients. 4, 24, 26, 57, 60

NLP Natural Language Processing. 12

NLU Natural Language Understanding. 13–15

RNN Recurrent Neural Network. 4, 5, 42, 43, 45

SNR Signal-to-Noise Ratio. 49

WER Word Error Rate. 65

APÉNDICE A

GPSR y Parser Stanza

A.1. GPSR

En este proyecto se utilizó el generador de comandos oficial utilizado en las competiciones de RoboCup@Home para la prueba de robot de servicio de propósito general (en inglés, General Purpose Service Robot (GPSR)). Para obtener el repositorio del generador consultar el siguiente enlace: https://github.com/RoboCupAtHome/gpsr_command_generator.

Una vez instalado el repositorio, para generar comandos basta con correr los siguientes comandos.

```
cd GPSRCmdGen
make gpsr
```

Sin embargo, el usuario necesita ingresar un "*enter*" para generar cada uno de los comandos, lo cual no es eficiente para generar una gran cantidad de comandos, como los que se podrían requerir para entrenar una red neuronal. Para ello se realizó una modificación en el código ***Program.cs*** ubicado en la misma carpeta, en donde se incluyó un loop que genera un nuevo comando, por medio de la función ***gen.GenerateTask(DifficultyDegree.High)*** y lo guarda en la dirección indicada utilizando el índice como nombre del archivo.

```
1                   private Task GetTask()
2                   {
3                   for (int i=0; i<10000; i++){
4                       string path=Convert.ToString(i)+".txt";
5                       Console.WriteLine(path);
6                       var fs=File.Create(path);
7                       fs.Dispose();
8
9                       TextWriter tw = new StreamWriter(path);
10                      tw.WriteLine(gen.GenerateTask(
                            DifficultyDegree.High));
11                      tw.Close();
12                  }
13                  return gen.GenerateTask(DifficultyDegree.High);
14                  }
```

A.2. Parser Stanza

El módulo Parser Stanza es un conjunto de herramientas Python NLP que admite más de 60 lenguajes humanos. El módulo de análisis de dependencias construye una estructura de árbol de palabras a partir de la oración de entrada, que representa las relaciones de dependencia sintáctica entre palabras. Las representaciones de árbol resultantes y los cuales son útiles en muchas aplicaciones.

Para procesar un fragmento de texto, primero se debe construir un pipeline con diferentes unidades de procesador. Cada una es específica del idioma, por lo que nuevamente este deberá ser especificado. La canalización de Stanza es compatible con CUDA, lo que significa que se usará un dispositivo CUDA siempre que esté disponible, de lo contrario, se usarán CPU.

```
# Build an English pipeline, with all processors by default
en_nlp = stanza.Pipeline('en')
```

Una vez que se ha construido correctamente el paso anterior, se pueden obtener anotaciones de un fragmento de texto simplemente pasando la cadena al objeto creado, lo cual devolverá un objeto documento, que se puede utilizar para acceder a anotaciones más detalladas.

```
def text_to_NLP(index):
  PATH="/Comandos/"+str(index)+".txt"
  file1 = open(PATH, "r")         #Open the "txt" files
  FileContent = file1.read()         #Read the content
  en_doc1 = en_nlp(FileContent)      # Processing English text
  return en_doc1
```

El objeto documento contiene una lista de oraciones y una oración contiene una lista de símbolos y palabras. Un ejemplo de la salida de este objeto se muestra a continuación.

```
[Sentence 1]
Robot           robot           NOUN    3       vocative
give            give            VERB    0       root
me              I               PRON    3       iobj
the             the             DET     6       det
food            food            NOUN    3       obj
```

A partir de esta lista de dependencias es posible realizar muchas aplicaciones posteriores, en este proyecto en específico se utilizó para poder determinar las palabras clave en cada comando y poder obtener las etiquetas correspondientes a cada uno de estos. Como es posible observar, las palabras clave se adjuntaron sin cambio alguno, sin embargo, a las palabras ajenas (conectores o preposiciones) se les colocó una única etiqueta demonimada ***unknown***.

```
def desm(en_doc1):
  command=[]
  for word in sent.words:
    if word.pos=="VERB":
      command.append(word.text)
    elif word.pos=="PRON":
      command.append(word.text)
    elif word.pos=="PROPN":
      command.append(word.text)
    elif word.pos=="NOUN":
      command.append(word.text)
    else:
      command.append("unknown")
  return command
```

Para obtener más referencias de cómo fue utilizado el modulo Parser Stanza es posible consultar el siguiente enlace: https://colab.research.google.com/drive/1PFYHLzaWJX1IAxaGyWGKSkC2C2Zd-tJN?usp=sharing

APÉNDICE B

Google Text To Speech

La API de Google Text-to-Speech (gTTS) permite generar un audio similar al humano. La API convierte texto en formatos de audio como WAV, MP3, entre otros. gTTS es una biblioteca de Python que permite interactuar con la API de texto a voz de Google Translate, la cual admite varios idiomas.

La lista de idiomas admitidos es la siguiente, en la que se puede observar 49 idiomas y sus correspondientes variantes. Además de una selección de voces múltiples en diferentes géneros y calidades, hay varios acentos disponibles: inglés australiano, británico, indio y americano. Esta lista no es fija y crece a medida que se incorporen nuevas voces.

```
---------------------- Languages: 49 -----------------------
      af-ZA      ar-XA      bg-BG      bn-IN      ca-ES
     cmn-CN     cmn-TW      cs-CZ      da-DK      de-DE
      el-GR      en-AU      en-GB      en-IN      en-US
      es-ES      es-US      fi-FI     fil-PH      fr-CA
      fr-FR      gu-IN      hi-IN      hu-HU      id-ID
      is-IS      it-IT      ja-JP      kn-IN      ko-KR
      lv-LV      ml-IN      nb-NO      nl-NL      pl-PL
      pt-BR      pt-PT      ro-RO      ru-RU      sk-SK
      sr-RS      sv-SE      ta-IN      te-IN      th-TH
      tr-TR      uk-UA      vi-VN     yue-HK
```

A continuación se muestra la función utilizada para este proyecto, la cual recibe como argumentos; la dirección del comando en texto, el índice asignado a este comando y la dirección en donde se desea guardar el archivo de audio. Durante las primeras 3 líneas, encuentra y lee el comando de texto que ingresa al sintetizador. Posteriormente, crea un objeto gTTS y como primer argumento se ingresa el texto que se desea convertir. Adicionalmente acepta otros parámetros, entre los que destacan ***slow*** que es un booleano que indica si la pronunciación debería ser lenta o a velocidad normal, y ***lang***, que es una cadena que indica el idioma y acento en la que el texto será pronunciado. Finalmente, para guardar el texto convertido a audio en un archivo existe el método ***save***, el cual recibe el nombre del archivo de salida.

```
def text_to_speech(PATH_command, index, save_wav):
  example=PATH_command+str(index)+".txt"
  file1 = open(example, "r")          #Open the "txt" files
  FileContent = file1.read()          #Read the content
  tts = gTTS(FileContent, lang='en')
  tts.save(save_wav+str(index)+".mp3")   #Save the string
  sound_file=str(index)+".mp3"    #Provide the files name
  return sound_file
```

Para obtener más referencias de como fue utilizado el modulo es posible consultar el siguiente enlace: https://colab.research.google.com/drive/1fHRejz-N8XI5bsEGuElAUTYkyk2dSMP1?usp=sharing

APÉNDICE C

ANN en MATLAB

En esta sección se incluye parte del código utilizado para las redes del tipo FeedForward, utilizadas en las pruebas de concepto. El flujo de trabajo para el diseño de la red neuronal en MATLAB tiene los siguientes pasos principales.

- Crear un objeto de red neuronal (***net***) para almacenar la información que define una red neuronal.

```
net = feedforwardnet;
```

- Configurar entradas y salidas de redes neuronales. La configuración implica organizar la red para que sea compatible con el problema que desea resolver, según lo definido por los datos de muestra. Dentro de esta configuración se puede especificar el tamaño de las entradas, el número de capas ocultas, el número de neuronas en cada capa, las conexiones entre ellas y el tipo de funciones de activación de cada capa.

```
net.numInputs = 1;
net.numlayers = 3;
net.layers{1}.size = 100;
net.biasConnect = ones(3,1);
net.inputConnect = [1 ; 0 ; 0];
net.layerConnect = [0 0 0;1 0 0 ;0 1 0];
net.outputConnect = [0 0 1];
net.layers{1}.transferFcn = 'logsig';
net.layers{2}.transferFcn = 'logsig';
net.layers{3}.transferFcn = 'softmax';
```

- Entrenar y validar la red: Una vez configurada la red, es necesario ajustar los parámetros de red ajustables (denominados pesos y sesgos) para optimizar el rendimiento de la red. En esta sección es posible especificar el algoritmo de entrenamiento, el número de épocas, entre otros. La función ***trainNetwork*** recibe como entradas, el objeto ***net*** que contiene la configuración antes especificada, las entradas y las salidas del conjunto de entrenamiento.

```
net.trainFcn ='trainscg';
net.performFcn = 'crossentropy';
net.trainParam.epochs = 5;
net.trainParam.goal = 0.0001;
net.trainParam.min_grad = 1e-3;

[net,tr] = train(net,data_train,targets);
```

Para obtener más referencias consultar la documentación de MATLAB: https://www.mathworks.com/help/deeplearning/ug/workflow-for-neural-network-design.html

De forma predeterminada, la función ***trainNetwork*** usa la CPU si no hay una GPU disponible. Sin embargo es posible especificar si se desea utilizar la GPU mediante la siguiente línea de comando. Además es posible aprovechar los múltiples trabajadores especificando el entorno de ejecución con la función ***trainingOptions***. Si se cuenta con más de una GPU, se puede especificar ***'multi-gpu'***, de lo contrario, es necesario especificar con ***'parallel'***.

```
[net,tr] = train(net,data_train,targets, 'useParallel', 'yes',
    'useGPU', 'yes');
```

Para obtener más referencias consultar el siguiente ejemplo: https://www.mathworks.com/help/deeplearning/ug/neural-networks-with-parallel-and-gpu-computing.html

Bibliografía

[Afzal et al., 2010] Afzal, H., Sheeraz, M., and Mark, A. (2010). A novel approach for mfcc feature extraction. In *Conference: Signal Processing and Communication Systems (ICSPCS), 2010 4th International Conference on.*

[Akshay, 2019] Akshay, L. C. (2019). Learning parameters, part 2: Momentum-based and nesterov accelerated gradient descent. https://towardsdatascience.com/learning-parameters-part-2-a190bef2d12.

[Amidi, 2018] Amidi, A. (Noviembre, 2018). Vip cheatsheet: Recurrent neural networks. https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks.

[Amidi and Amidi, 2017] Amidi, A. and Amidi, S. (2017). Aprendizaje automático. https://stanford.edu/~shervine/l/es/teaching/cs-229/hoja-referencia-aprendizaje-no-supervisado.

[Andrade et al., 2018] Andrade, D., Leo, S., Viana, M. L. D. S., and Bernkopf, C. (2018). A neural attention model for speech command recognition. arXiv preprint arXiv:1808.08929.

[Atal and Hanauer, 1971] Atal, B. S. and Hanauer, S. L. (1971). Speech analysis and synthesis by linear prediction of the speech wave. *The Journal of the Acoustical Society of America*, 50(2).

[Benzeguiba et al., 2010] Benzeguiba, M., de Mori, R., Deroo, O., Dupon, S., Erbes, T., L., Fissore, P., Laface, A., and Mertins, C. (2010). Automatic speech recognition and speech variability: a review. *Speech Communication, Elsevier :*, (49):763.

[Bradbury, 2020] Bradbury, J. S. (2020). *Linear Predictive Coding.* OntarioTech University, Ontario, Canada.

[Cano, 2003] Cano, A. (2003). Codificación vectorial de voz en español. Master's thesis, Instituto Politécnico Nacional.

[Carrión, 2017] Carrión, J. B. (2017). Diseño de una interfaz para la captura de patrones de vibración hápticos. Master's thesis, Universidad Politécnica de Madrid.

[Choi and Kim, 2006] Choi, I. and Kim, T. (2006). A study on asr/tts server architecture for network robot system. *In Proceedings of the Third International Conference on Informatics in Control, Automation and Robotics*, pages 277–282.

[Choi et al., 2019] Choi, S., Seo, S., Shin, B., Byun, H., Kersner, M., Kim, B., Kim, D., and Ha, S. (2019). Training keyword spotters with limited and synthesized speech data. *Temporal Convolution for Real-Time Keyword Spotting on Mobile Devices," Proc. Interspeech 2019.*

[Cruz et al., 2007] Cruz, I., Salazar, S., Rodríguez, A., Grau, R., and García, M. (2007). Redes neuronales recurrentes para el análisis de secuencias. *Revista cubana de ciencias informáticas*, 1(4):48–57.

[Davis and Mermelstein, 1980] Davis, S. and Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE transactions on acoustics, speech, and signal processing*, 28(4).

[Díaz et al., 2002] Díaz, J. E., Segura, J., and A.J., R. (2002). *Reconocimiento de voz continua. Aproximaciones basadas en HMM y redes neuronales recurrentes.* Universidad de Granada, Granada.

[Google, 2021] Google (2021). Text-to-speech. https://cloud.google.com/text-to-speech.

[Gouda et al., 2020] Gouda, S. K., Harrison, V., Kanetkar, S., and Warmuth, M. (2020). Speech recognition: Key word spotting through image recognition. arXiv:1803.03759v2.

[Haykin, 2001] Haykin, S. (2001). *Communication systems.* John Wiley and Sons, USA.

[Hernando, 1993] Hernando, F. (1993). *Técnicas de procesado y representación de la señal de voz para el reconocimiento del habla en ambientes ruidosos*. Universidad Politécnica de Cataluña, Barcelona.

[Hilera and Martínez, 1995] Hilera, J. and Martínez, V. (1995). *Redes neuronales artificiales: fundamentos, modelos y aplicaciones*. RA-MA, Madrid.

[Holmes et al., 1997] Holmes, J., Holmes, W., and Garner, P. (1997). Using formant frequencies in speech recognition.

[Hosom, 2003] Hosom, J.-P. (2003). Speech recognition. In Bidgoli, H., editor, *Encyclopedia of Information Systems*, pages 155–169. Elsevier, New York.

[IBM, 2016] IBM (2016). Ibm shoebox. https://www.ibm.com/ibm/history/exhibits/specialprod1/specialprod1_7.html.

[Itakura, 1975] Itakura, F. (1975). Minimum prediction residual principle applied to speech recognition. *IEEE Trans. Acoustics, Speech and Signal Proc*, ASSP-23:57–72.

[Itakura and Saito, 1968] Itakura, F. and Saito, S. (1968). Analysis synthesis telephony based on the maximum likelihood method. *In Proc. 6th of the International Congress on Acoustics*, page C–17–C–20.

[Itakura and Saito, 1970] Itakura, F. and Saito, S. (1970). A statistical method for estimation of speech spectral density and formant frequencies. *Electronics and Communications in Japan*, 53A:36–43.

[J. G. Wilpon and Goldman, 1990] J. G. Wilpon, L. R. Rabiner, C. H. L. and Goldman, E. R. (1990). Automatic recognition of keywords in unconstrained speech using hidden markov models. *IEEE Trans. Acoustics, Speech and Signal Proc*, 38(11):1870–1878.

[Juang and Rabiner, 2004] Juang, B. and Rabiner, L. (2004). Automatic speech recognition: A brief history of the technology development. https://web.ece.ucsb.edu/Faculty/Rabiner/ece259/Reprints/354_LALI-ASRHistory-final-10-8.pdf.

[Jurafsky and Martin, 2020] Jurafsky, D. and Martin, J. (2020). Speech and language processing. https://web.stanford.edu/~jurafsky/slp3/A.pdf.

[Khan and Smith, 2005] Khan, M. A. and Smith, M. J. (2005). 5.3 - fundamentals of vector quantization. In BOVIK, A., editor, *Handbook of Image and Video Processing (Second Edition)*, Communications, Networking and Multimedia, pages 673–688. Academic Press, Burlington, second edition edition.

[L. R. Rabiner and Wilpon, 1979] L. R. Rabiner, S. E. Levinson, A. E. R. and Wilpon, J. G. (1979). Speaker independent recognition of isolated words using clustering techniques. *IEEE Trans. Acoustics, Speech and Signal Proc*, ASSP-27:336–349.

[Laplante and Seppo, 2012] Laplante, P. A. and Seppo, J. O. (2012). *REAL-TIME SYSTEMS DESIGN AND ANALYSIS*. A JOHN WILEY and SONS, INC., PUBLICATION, New Jersey.

[Levinson, 1947] Levinson, N. (1947). The Wiener rms (root mean square) error criterion in filter design and prediction. *Journal of Mathematics and Physics*, 25.

[Lin et al., 2020] Lin, L., Kilgour, K., Roblek, D., and Sharifi, M. (2020). Training keyword spotters with limited and synthesized speech data. *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. arXiv preprint arXiv:1711.07128.

[Linnainmaa., 1976] Linnainmaa., S. (1976). The representation of the cumulative rounding error of an algorithm as a taylor expansion of the local rounding errors. Master's thesis, Univ. Helsinki.

[Loy, 2018] Loy, J. (2018). How to build your own ANN from scratch in python. https://towardsdatascience.com/how-to-build-your-own-neural-network-from-scratch-in-python-68998a08e4f6.

[Martinez et al., 2012] Martinez, J., Perez, H., Escamilla, E., and Suzuki, M. (2012). Speaker recognition using mel frequency cepstral coefficients (mfcc) and vector

quantization (vq) techniques. *CONIELECOMP 2012, 22nd International Conference on Electrical Communications and Computers*. doi: 10.1109/CONIELECOMP.2012.6189918.

[Maurya et al., 2017] Maurya, A., Kumar, D., and Agarwal, R. (2017). Speaker recognition for hindi speech signal using mfcc-gmm approach. *6th International Conference on Smart Computing and Communications, ICSCC 2017, 7-8 December 2017, Kurukshetra, India*.

[McCulloch and Pitts, 1943] McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133.

[Michaely et al., 2017] Michaely, A. H., Zhang, X., Simko, G., Parada, C., and Aleksi, P. (2017). Keyword spotting for google assistant using contextual speech recognition. *Google Inc.* https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/46554.pdf.

[Miyara, 2017] Miyara, F. (Agosto, 2017). La voz humana. https://www.fceia.unr.edu.ar/prodivoz/fonatorio.pdf.

[Nwankpa et al., 2018] Nwankpa, C., Ijomah, J., Gachagan, A., and Marshall, S. (2018). *Activation Functions: Comparison of Trends in Practice and Research for Deep Learning*. https://arxiv.org/pdf/1811.03378.pdf.

[Objetos-UNAM, 2021] Objetos-UNAM (2021). Sistemas y aparatos. http://objetos.unam.mx/etimologias/terminologiaMedica.

[Olah, 2015] Olah, C. (2015). Understanding lstm networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs.

[Prabhavalkar et al., 2017] Prabhavalkar, R., Roa, K., Sainath, T., Li, B., Johnson, L., and Jaitly, N. (2017). A comparison of sequence-to-sequence models for speech recognition. https://www.cs.toronto.edu/~graves/icml_2006.pdf.

[Qian et al., 2021] Qian, J., Yuren, W., Bojin, Z., Shaojun, W., and Jing, X. (2021). Understanding gradient clipping in incremental gradient methods.

[Rabiner, 1989] Rabiner, L. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proc. IEEE 77*, page 257–286.

[Rabiner and Schafer, 2007] Rabiner, L. R. and Schafer, R. W. (2007). *Introduction to Digital Signal Processing*. NOW, USA.

[Ramírez et al., 2019] Ramírez, J. M., Montalvo, A., and Calvo, J. (2019). Evaluación de rasgos acústicos para el reconocimiento automático del habla en escenarios ruidosos usando kaldi. *Ingeniería electrónica, automática y comunicaciones*, 40(3).

[Random-House-Dictionary, 1987] Random-House-Dictionary (1987). *Random House Dictrionary of the English Language*. 2nd edition.

[Rich and Knight, 1991] Rich, E. and Knight, K. (1991). *Artificial intelligence*. McGraw-Hill, Inc., NJ, USA, second edition.

[Ruedo, 2011] Ruedo, L. (2011). *Mejoras en reconocimiento del habla basadas en mejoras en la parametrización de la voz*. Universidad Autónoma de Madrid, Madrid.

[Rumelhart et al., 1986] Rumelhart, D., Geoffrey, E., and Ronald, J. (1986). Learning representations by back-propagating errors. *Nature*, 323.

[Savage and Rivera, 2021] Savage, J. and Rivera, C. (2021). Cuantización vectorial. In *Reconocimiento de Patrones, Posgrado en Ingeniería Eléctrica, UNAM*.

[Savage et al., 2019] Savage, J., Rosenblueth, D., Matamoros, M., Negrete, M., Contreras, L., Cruz, J., Martell, R., Estrada, H., and Okada, H. (2019). Semantic reasoning in service robots using expert systems. *Robotics and autonomous systems*, (114). https://doi.org/10.1016/j.robot.2019.01.007.

[Schank, 1972] Schank, R. (1972). Conceptual dependency: A theory of natural language understanding. *Cognitive Psychology*, 3:552–631.

[Schmidt et al., 1992] Schmidt, W., Kraaijveld, M. A., and Duin, R. (1992). Feedforward neural networks with random weights. *Delf University of Technology*.

[Shrivastava, 2018] Shrivastava, A. (2018). Introduction to deep learning. https://www.cs.umd.edu/~abhinav/.

[Singer, 2016] Singer, Y. (2016). *Advanced Optimization. The Gradient Descent Algorithm.* Harvard Univesity, USA.

[Stanford-NLP-Group, 2020] Stanford-NLP-Group (2020). Dependency parsing. https://stanfordnlp.github.io/stanza/depparse.html.

[Tumilaar et al., 2015] Tumilaar, K., Langi, Y., and Rindengan, A. (2015). Hidden markov model. *d'CARTESIAN*, 4:86.

[Ward, 2001] Ward, W. (2001). *Speech Recognition and Production by Machines.* International Encyclopedia of the Social and Behavioral Sciences.

[Zhang et al., 2020] Zhang, A., Lipton, Z., Li, M., and Smola, A. (2020). *Dive into Deep Learning.* https://d2l.ai/index.html.

[Zhang et al., 2017] Zhang, Y., Suda, N., Lai, L., and Chandra, V. (2017). Hello edge: Keyword spotting on microcontrollers. *IEEE transactions on acoustics, speech, and signal processing.* arXiv preprint arXiv:1711.07128.