SISTEMA INTERACTIVO DE RECONOCIMIENTO DE FONEMAS PARA LA INTERPRETACIÓN DE VOZ Y TRADUCCIÓN A LENGUA DE SEÑAS

DEIBY ERASMO OBANDO PORTILLA
GUILLERMO DANIEL ORTEGA GALEANO

SAN JUAN DE PASTO
UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
PROGRAMA INGENIERÍA ELECTRÓNICA
2011

SISTEMA INTERACTIVO DE RECONOCIMIENTO DE FONEMAS PARA LA INTERPRETACIÓN DE VOZ Y TRADUCCIÓN A LENGUA DE SEÑAS

DEIBY ERASMO OBANDO PORTILLA
GUILLERMO DANIEL ORTEGA GALEANO

DIRECTOR
ING. MSC. DARIO FERNANDO FAJARDO FAJARDO

SAN JUAN DE PASTO
UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
PROGRAMA INGENIERÍA ELECTRÓNICA
2011

“LAS IDEAS Y CONCLUSIONES APORTADAS EN EL TRABAJO DE GRADO, SON DE RESPONSABILIDAD EXCLUSIVA DE LOS AUTORES”

ARTÍCULO 1 DEL ACUERDO No. 324 DE OCTUBRE 11 DE 1966, EMANADO DEL HONORABLE CONSEJO DIRECTIVO DE LA UNIVERSIDAD DE NARIÑO

Nota de aceptación:

_____
_____
_____
_____
_____
_____

_____
Presidente

_____
Jurado

_____
Jurado

San Juan de Pasto, Febrero de 2011

DEDICATORIA

“A mi familia por su apoyo y confianza durante mi vida académica, a los docentes del programa de ingeniería electrónica por sus enseñanzas, a mis amigos a quienes les debo su motivación y consejos para superar las situaciones difíciles, y a Guillermo Ortega, con quien tuve el privilegio de desarrollar este proyecto, a quien le agradezco su amistad y de quien aprendí el valor de la perseverancia y dedicación para conseguir los objetivos en la vida.”

Deiby Erasmo Obando Portilla

“A Dios quien me dio las herramientas necesarias y la capacidad para realizar este trabajo, a mis padres Jaime y Lorena por su apoyo incondicional, a mi familia por su confianza”

Guillermo Daniel Ortega Galeano

RESUMEN

En este trabajo de investigación se da a conocer el diseño y la implementación de un sistema experimental, que basado en técnicas de reconocimiento de voz, permite traducir la voz reconocida a lengua de señas colombiana (LSC). El sistema, que es diseñado como herramienta de apoyo docente, es además independiente del hablante y busca generar procesos de inclusión de los estudiantes con discapacidad auditiva al entorno educativo de la Universidad de Nariño.

Asimismo, se enmarcan los conceptos claves dentro del reconocimiento automático de voz ASR (Automatic Speech Recognition), y se desarrollan diferentes técnicas para realizar este reconocimiento, técnicas como el DTW (*Dynamic Time Warping*), las RNA (Redes Neuronales Artificiales) y los HMM (*Hidden Markov model*) que permiten realizar un reconocimiento a mayor o menor escala dependiendo de la aplicación.

El sistema implementado, en etapas iníciales es dependiente del usuario y con un vocabulario reducido. En seguida, se desarrolla un sistema multiusuario utilizando las RNA y el DTW en donde se logra crear un sistema capaz de reconocer a dos personas diferentes. Finalmente se desarrolla un sistema independiente del usuario que es aplicado a cinco personas con porcentajes de reconocimiento cercanos al 98%.

ABSTRACT

This research explains the design and implementation of an experimental system, which based on speech recognition techniques, allows translating the speech recognized to Colombian Sign Language (CSL). The system, which is designed as an educational support tool, is also speaker independent and aims to generate processes of inclusion of deaf students to the educational environment of the University of Nariño.

In addition, key concepts are embedded within the automatic speech recognition ASR (Automatic Speech Recognition), and develop different techniques to accomplish this recognition techniques such as DTW (Dynamic Time Warping), ANN (Artificial Neural Networks) and HMM (Hidden Markov model) that allow speech recognition of a greater or lesser extent depending on the application.

System implemented in early stages is speaker dependent and with a reduced vocabulary. Then, it is developed a multiuser system using DTW and ANN where it is create a system capable of recognizing two different people. Finally an independent speaker system is applied to five people with recognition rates close to 98%.

CONTENIDO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE TABLAS

Pág.

LISTA DE ANEXOS

Pág.

GLOSARIO

.NET Framework: conjunto de nuevas tecnologías, incluidas en los sistemas operativos Microsoft Windows, que provee soluciones pre-codificadas para requerimientos comunes de los programas y que gestiona la ejecución de los mismos.

ASR: (*Automatic Speech Recognition*) Tecnología que permite a un computador identificar las palabras que una persona pronuncia al micrófono. El ASR, en español se conoce como Reconocimiento Automático de Voz o Reconocimiento Automático del Habla.

DTW: (*Dynamic Time Warping*) Algoritmo para medir la similitud entre dos secuencias que pueden variar en tiempo y velocidad. El DTW se conoce en español como Alineamiento Temporal Dinámico.

Fonema: Cada uno de los sonidos simples del lenguaje hablado.

HMM: (*Hidden Markov Model*) Modelos estadísticos en los que se asume que el sistema a modelar es un proceso de Markov de parámetros desconocidos. Los HMM se conocen como Modelos Ocultos De Markov.

LENGUA DE SEÑAS: Lengua natural de expresión y configuración gesto-espacial y percepción visual, gracias a la cual las personas sordas pueden establecer un canal de comunicación en su entorno social, ya sea con otros individuos sordos o cualquier persona que conozca la lengua de señas empleada.

LPC: (*Linear Predictive Coding*) herramienta para la representación de la envolvente espectral de una señal digital de la voz en forma comprimida, usando la información de un modelo de predicción lineal. Los LPC se conocen en español como Coeficientes de Predicción Lineal.

MFCC: (*Mel Frecuency Cepstral Coefficients*) coeficientes para la representación de una señal de voz basados en la percepción auditiva humana. En español se conocen como Coeficientes Cepstrales en Frecuencias de Mel.

RNA: (*Redes Neuronales Artificiales*) Sistemas de procesamiento de información cuya arquitectura y funcionamiento están inspirados en las redes neuronales biológicas y en la forma en que funciona el sistema nervioso.

SISTEMAS NEURO-DIFUSOS: sistemas híbridos que hacen uso de la lógica difusa y redes neuronales y tienen como objetivo unir las ventajas de procesamiento lingüístico de un sistema difuso con la capacidad de adaptación y aprendizaje de las redes neuronales.

INTRODUCCIÓN

El reconocimiento de patrones como una parte del procesamiento digital de señales ha sido aplicado en diferentes campos como la astronomía, la instrumentación electrónica, la domótica, la robótica, la seguridad etc. ayudando al desarrollo de nuevas herramientas que ofrecen una mayor eficiencia en cuanto al uso de los recursos tecnológicos y una mayor interactividad con el usuario. Algunos de estos sistemas, están orientados a mejorar la calidad de vida de personas que presentan algún tipo de discapacidad física, ofreciendo diferentes maneras para establecer comunicación entre el hombre y la máquina

Tal es el caso del reconocimiento de patrones de voz, que aplicado a la enseñanza de lengua de señas, ayudaría a las personas que presentan algún tipo de discapacidad auditiva o sordera; definida como la dificultad o la imposibilidad de usar el sentido del oído debido a una pérdida parcial o total. Así pues, una persona sorda no solo estará limitada por su discapacidad, sino que también podría verse excluida en ciertos ámbitos de la sociedad.

Por otra parte y aunque existen medios para la enseñanza de lengua de señas como cursos, herramientas en internet e instituciones especializadas, estas resultan poco efectivas, pues aun no están presentes en nuestro entorno y además no son de fácil acceso, lo cual genera discriminación e intolerancia.

En consecuencia, el desarrollo de un sistema que haciendo uso del reconocimiento de patrones de voz (fonemas), identifique las palabras y las traduzca a lengua de señas, será de gran ayuda para personas con discapacidad auditiva dentro de ambientes educativos, y en especial las instituciones de educación superior como la Universidad de Nariño, por ende su implementación facilitará la integración, el acceso y, en definitiva, la inclusión a la educación de estas personas.

1. DESCRIPCIÓN DEL PROBLEMA

1.1 PLANTEAMIENTO DEL PROBLEMA

En el ambiente en el que estamos inmersos identificamos a muchas personas que presentan algunas discapacidades físicas, entre las cuales se encuentran las relacionadas con la parte auditiva, lo cual dificulta la comunicación. Además esta clase de discapacidades, ya sean congénitas u ocasionadas por enfermedad o accidente, ocasionan un sin número de problemas, generando una disminución en el rendimiento y la oportunidad de desarrollo de estas personas, y aunque existen lugares de enseñanza y herramientas virtuales a nivel nacional y regional para capacitar a personas con discapacidades auditivas en lo referente a la lengua de señas, estas herramientas resultan poco eficientes, para personas con discapacidad auditiva, pues aun no han sido aplicadas en instituciones educativas de la región debido a las características fisiológicas y del habla propias de los habitantes de nuestro entorno.

Por otra parte, las técnicas de comunicación mediante señas no son conocidas por la comunidad en general y resultan inadecuadas en circunstancias particulares como por ejemplo, para la comunicación entre un emisor con capacidad del habla, que desconoce la lengua de señas, y un receptor con discapacidad auditiva, obstruyéndose la comunicación y limitándose el número de ambientes a los cuales las personas con discapacidad auditiva pueden acceder. Tal es el caso de las instituciones educativas nacionales y regionales, que no ofrecen una educación orientada a personas con discapacidad auditiva, pues los docentes no conocen la lengua de señas, y por lo tanto no existe la posibilidad de que estas personas puedan acceder a la educación, especialmente la superior.

Por tal motivo, con el desarrollo del proyecto se logra crear una herramienta didáctica, que a partir del reconocimiento de patrones de voz, posibilite entablar una comunicación, entre un docente y un estudiante no oyente, dentro de las instituciones educativas como lo es la Universidad de Nariño, lo cual constituye un paso más para la integración de las personas con discapacidad auditiva en ambientes educativos.

1.2 FORMULACIÓN DEL PROBLEMA

¿Es posible desarrollar un sistema interactivo que, haciendo uso del reconocimiento de fonemas, pueda identificar palabras y traducirlas a lengua de señas?

2. OBJETIVOS

2.1 OBJETIVO GENERAL

Desarrollar un sistema interactivo para ser aplicado en la enseñanza que, a partir del reconocimiento de fonemas, sea capaz de interpretar y traducir las palabras a lengua de señas mejorando la comunicación con individuos que presenten deficiencia auditiva.

2.2 OBJETIVOS ESPECÍFICOS

Identificar, recolectar y analizar información referente a las diferentes técnicas utilizadas en el reconocimiento de fonemas.

Diseñar un módulo de reconocimiento de fonemas.

Implementar un sistema, que basado en diversas técnicas, clasifique los fonemas y reconozca las palabras pronunciadas por un hablante.

Desarrollar una interfaz gráfica que permita la visualización de las palabras identificadas en lengua de señas.

Realizar pruebas respectivas, para validar y enriquecer el funcionamiento del sistema.

3. JUSTIFICACIÓN

En la actualidad nacional y regional existen muchos ambientes, como instituciones educativas, entidades públicas y privadas entre otros, a los cuales no es posible de acceder debido a discapacidades auditivas, ambientes en donde este proyecto pretende convertirse en una herramienta útil, tanto para personas con discapacidad auditiva, como para personas que deseen aprender la lengua de señas, pues mediante el sistema desarrollado se puede realizar la traducción del lenguaje oral a lengua de señas, pudiendo mejorar la comunicación con personas que presenten discapacidad auditiva.

Además cada individuo que utilice el sistema de reconocimiento de fonemas, puede emplearlo de manera cómoda y segura. Por lo tanto, su implementación permitirá a individuos con discapacidad auditiva integrarse, a las diferentes actividades de desarrollo personal y social de nuestro entorno.

El sistema desarrollado, se caracteriza por su facilidad de utilización, esencialmente, porque el lenguaje utilizado para el proceso de identificación es el oral, el cual es traducido a lengua de señas; en caso de desconocer la lengua de señas el sistema también sirve para capacitar a las personas.

Por todas las utilidades proporcionadas, este proyecto llega a concebirse en el momento justo, porque las crecientes necesidades de comunicación hacen que las personas con discapacidad auditiva queden fuera de contexto en un mundo cambiante. En consecuencia el desarrollo del sistema brinda la oportunidad para que personas con discapacidad auditiva se integren al proceso de construcción social y tecnológico.

4. REVISIÓN SOBRE EL RECONOCIMIENTO DE VOZ

En este capítulo se explora los conceptos fundamentales del reconocimiento de voz y se muestra el desarrollo del tema en la actualidad, luego se describe la estructura general de un sistema de reconocimiento de voz y se explican, las partes que lo componen, finalmente se describen dos algoritmos muy usados como reconocedores: el alineamiento de tiempo dinámico y los modelos ocultos de Markov como posibles soluciones para el reconocimiento de voz.

4.1 RECONOCIMIENTO DE VOZ

El habla es el principal medio de comunicación entre las personas. Por razones que van desde la curiosidad tecnológica sobre los mecanismos para la realización de la voz humana, hasta el deseo de automatizar tareas simples que requieren intrínsecamente interacciones hombre-máquina, la investigación en el **Reconocimiento Automático del Habla** o **Reconocimiento Automático de Voz** (ASR por sus siglas en inglés, *Automatic Speech Recognition*) ha atraído una gran atención en las últimas cinco décadas.

El objetivo del ASR es permitir la comunicación hablada entre seres humanos y computadores. El reconocimiento de voz convierte las palabras habladas (señal acústica) en texto. Las palabras reconocidas pueden servir de entrada a otros sistemas que los requieran para realizar alguna acción. Es decir, que un sistema de reconocimiento de voz es capaz de procesar la señal de voz emitida por el ser humano y reconocer la información contenida en ésta, convirtiéndola en texto o emitiendo órdenes que actúan sobre un determinado proceso.

En otras palabras el reconocimiento de voz (ASR) es fundamentalmente una tarea de clasificación de patrones, que toma un patrón de entrada, en este caso es la señal de voz, y la clasifica como una palabra dentro del vocabulario implementado para el sistema. Los patrones de entrada, señal de voz, pueden ser tratados como fonemas, sílabas o palabras. Si estos patrones fuesen invariantes el problema sería trivial, ya que simplemente se compararía la secuencia de patrones de entrada con patrones almacenados y se encontraría el más similar. Sin embargo esto no es así, ya que la señal de voz es muy variable, debido a la gran variedad de locutores (hombres, mujeres, niños), diferentes velocidades a la hora de hablar, diferentes ambientes, distintas condiciones acústicas, e incluso el estado anímico del locutor.

El rendimiento de los sistemas de reconocimiento de voz se suele especificar en términos de precisión y velocidad. La precisión es generalmente clasificada con la tasa de error de palabra (WER por sus siglas en ingles, Word Error Rate), mientras que la velocidad se mide con el factor de tiempo real . Otras medidas de precisión son Single Word Error Rate (SWER) y Command Success Rate (CSR).

Las aplicaciones basadas en el reconocimiento de voz están presentes en servicios financieros, asistencia de directorio, llamadas por cobrar (operadora automática), transferencia de llamadas telefónicas, consultas de información (clima, tráfico, reservaciones). Las ventajas que presentan este tipo de aplicaciones son que al interactuar el usuario utiliza la eficiencia del habla (rápida, flexible y natural) y está libre de movimientos de las manos en caso de que las tenga ocupadas. Otras aplicaciones son el dictado automático, desarrollo de inventarios, control de robots, etc.

Las actuales investigaciones sobre el ASR buscan la finalidad de desarrollar una máquina capaz de reconocer voz de manera continua espontanea, independiente de cualquier locutor y sin restricciones de vocabulario, pues la capacidad de los seres humanos supera con creces a los reconocedores actuales en cuanto a la habilidad para distinguir nuevas palabras y sonidos ambientales no vocálicos de palabras correctas.

4.2 TÉRMINOS Y DEFINICIONES EN EL RECONOCIMIENTO DE VOZ

Bajo la denominación de Reconocimiento Automático del Habla se consideran en realidad tareas de diferente complejidad, como el reconocimiento de palabras aisladas o palabras conectadas, identificación de palabras clave en discurso continuo, reconocimiento de discurso continuo, etc. Además, las características propias del habla (continuidad temporal, variabilidad, redundancia informativa, etc.), afectan al proceso de reconocimiento y obligan a imponer ciertas limitaciones como acotar el nivel de ruido en la señal de entrada, establecer el tamaño del vocabulario (reducido, medio o grande), la capacidad de adaptación a nuevos locutores, etc. Por esta razón esta sección pretende dar a conocer algunos de los parámetros que definen e influyen en el comportamiento y funcionamiento del reconocimiento de voz.

4.2.1 Dependencia del Hablante: Es quizás el aspecto que introduce mayor variabilidad. La dependencia del hablante determina si el sistema debe entrenarse para cada usuario o es independiente de él. Los sistemas dependientes del hablante requieren muestras específicas de la voz del usuario. Estos sistemas deben tener una muestra de la manera en que el usuario pronunciara cada elemento del vocabulario antes de que el sistema pueda reconocer alguna palabra. Un sistema dependiente del hablante tendrá un porcentaje de reconocimiento mayor, ya que solo se reconoce a una persona en particular.

Los sistemas independientes del hablante pueden reconocer la voz de cualquier usuario potencial, sin importar que esta no haya sido incluida en el conjunto de entrenamiento. Esta clase de sistema tiene cierta dificultad, ya que las representaciones paramétricas de la voz son altamente dependientes del

hablante. Los sistemas independientes del hablante resultan útiles cuando múltiples usuarios pueden acceder a la aplicación.

Los sistemas de reconocimiento de voz que empiezan como sistemas independientes del hablante, pero se modifican dependiendo de la voz de usuarios individuales a medida que el sistema recibe muestras de voz se definen como sistemas adaptativos dependientes del usuario. Estos tipos de sistemas se diseñan para usarse por un conjunto estable de operadores que accederán al sistema repetidamente durante un tiempo determinado. Para estos sistemas la precisión del reconocimiento es relativamente alta, ya que las muestras de las características de voz para cada elemento del vocabulario son actualizadas continuamente en cada utilización del sistema.

4.2.2 Variabilidad. La voz presenta un amplio margen de variabilidad, dependiendo de aspectos relacionados con el hablante. Se pueden distinguir dos tipos de variabilidad:

Variabilidad intra-hablante: relacionada con las modificaciones introducidas por un mismo hablante sobre diferentes pronunciaciones de los mismos fonemas o palabras. Incluso en idénticas condiciones, cada pronunciación presentará diferencias con las restantes debido a la diferente duración temporal.

Variabilidad inter-hablante: debida a aspectos relacionados con el locutor y el entorno, ya que la señal obtenida dependerá de los dispositivos utilizados en su captación, del entorno donde se obtiene y, principalmente, del hablante, ya que cada sujeto presenta características diferentes, debidas por ejemplo a las diferencias anatómicas del aparato fonador. [1]

4.2.3 La forma de hablar. Otro parámetro que está presente en el reconocimiento de voz es la manera en la cual un usuario tiene que hablar al sistema. De acuerdo con esto aparecen distintos tipos de complicaciones en la tarea de reconocimiento que, en orden más o menos ascendente, corresponderían a los siguientes tipos:

Reconocedores de Palabras Aisladas: El reconocimiento se realiza sobre palabras completas emitidas de forma aislada entre sí.

Reconocedores de Palabras Conectadas: en este caso se utilizan también las palabras como unidades de reconocimiento, si bien estas son emitidas secuencialmente con pausas entre ellas.

[1] GUSTAVO, Romero. Sistema de reconocimiento automático del habla. Argentina: Universidad Nacional de Entre Ríos 2001

Reconocedores de Voz o Discurso Continuo: el reconocimiento se realiza atendiendo, generalmente, a unidades inferiores a la palabra (fonemas) sobre frases emitidas de forma natural, aunque cuidadosa, sin necesidad de establecer silencios entre las palabras que las constituyen.

Reconocedores de Palabras Clave: aquí la locución suele ser cuidadosa, aunque no es una condición. Su finalidad es el reconocimiento de las palabras clave contenidas en el vocabulario que se encuentren en frases emitidas de forma natural.

Reconocedores de Habla Espontanea: este es el caso más complicado, ya que aquí suelen no respetarse algunas reglas del lenguaje, e inclusive es muy probable que en la señal aparezcan eventos acústicos diferentes al habla, como tos, estornudo, hipo, bostezos, pausas repentinas, repetición de términos, etc[2].

4.2.4 Tamaño del vocabulario. Un cuarto parámetro presente en los sistemas de reconocimiento de voz es el tamaño del vocabulario del sistema. En un sistema de reconocimiento de voz, el tamaño del vocabulario afecta la complejidad, los requerimientos de proceso y la precisión del sistema. Aunque no hay definiciones establecidas, en general, se suele hablar de dos tipos de vocabularios: limitados (vocabularios que van de unas cuantas a miles de palabras) e ilimitados (vocabularios de decenas de miles de palabras). Los sistemas de vocabulario limitado realizan un apareamiento de los patrones acústicos, entre las palabras pronunciadas por el usuario y las palabras almacenadas en los diccionarios. Los sistemas de vocabulario ilimitado no requieren un diccionario de palabras, en lugar de ello, utiliza algoritmos que analizan las palabras pronunciadas en segmentos fonéticos. Una vez se tienen los segmentos fonéticos el sistema intenta determinar cual palabra fue realmente pronunciada y genera una respuesta apropiada.

4.3 ETAPAS DE UN SISTEMA DE ASR

La estructura general de los sistemas de reconocimiento de voz está conformada esencialmente por tres etapas, la primera de ellas es de pre-procesamiento, la segunda es de extracción de características y la tercera es de algoritmo de reconocimiento.

Figura 1. Etapas de un sistema de reconocimiento de voz.

[2] GUSTAVO, Romero. Sistema de reconocimiento automático del habla. Argentina: Universidad Nacional de Entre Ríos 2001

La etapa de extracción de características convierte la señal de voz en un conjunto de parámetros obteniendo información relevante de la señal, para su posterior análisis. En esta etapa se realiza un análisis en tiempo y en frecuencia de segmentos sucesivos de señal de voz, generando un conjunto de parámetros acústicos que representan de forma compacta y eficiente la señal de voz. Debido a que un sistema de reconocimiento de voz depende directamente de la calidad de extracción de parámetros, se han desarrollado algoritmos como los *coeficientes cepstrales de frecuencia de Mel* y los *modelos de predicción lineal*, que garantizan una entrega de datos significativos.

Por otro lado, el algoritmo de reconocimiento se encarga de clasificar e identificar el conjunto de parámetros, para reconocer las palabras diferenciándolas del resto que tiene almacenadas en su vocabulario. Dentro de los algoritmos de reconocimiento se encuentran el *Dynamic Time Warping*, los *Modelos Ocultos de Markov*, las *Redes Neuronales* y las *Técnicas Hibridas*, que usan combinaciones de algunos de los anteriores algoritmos.

4.4 COEFICIENTES CEPSTRALES EN LAS FRECUENCIAS DE MEL

Los coeficientes Cepstrales en las frecuencias de Mel (MFCC por sus siglas en inglés, *Mel Frecuency Cepstral Coefficients*) son coeficientes para la representación de una señal de voz basados en la percepción auditiva humana. Los MFCC se derivan de la Transformada de Fourier (FT) y de la Transformada Discreta del Coseno (DCT). En los MFCC las bandas de frecuencia se encuentran espaciadas logarítmicamente, según la escala de Mel, con el fin de modelar la respuesta auditiva humana de forma más adecuada que las bandas espaciadas linealmente de la FT. Esto permite un procesado de datos mucho más eficiente en la compresión de audio.

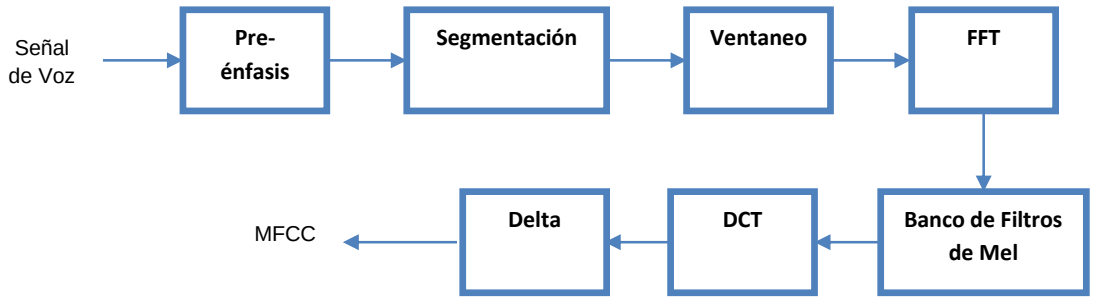


Figura 2. Proceso de extracción de los Coeficientes Cepstrales en las frecuencias de Mel.

En la figura 2, se indica el proceso para la obtención de los MFCC para una señal de voz el cual se detalla a continuación.

Pre-énfasis: La señal de voz $s(n)$ se envía a un filtro pasa-altas:

$$s_1(n) = s(n) + a * s(n-1) \qquad (1)$$

Donde $s_1(n)$ es la señal de salida y el valor de a suele estar entre 0,9 y 1,0. La transformada Z del filtro es:

$$H(z) = 1 - a * z^{-1} \qquad (2)$$

El objetivo del pre-énfasis es compensar a la parte de alta frecuencia que fue suprimido durante el mecanismo de producción de sonido de los seres humanos. Por otra parte, también puede amplificar la importancia de los formantes de alta frecuencia.

Segmentación: La señal de entrada se segmenta en tramas de 10 a 30 ms con la opción de superposición del 20-50% del tamaño de la trama (este proceso es conocido *Overlap-Add* o solapamiento aditivo), con el fin de conservar la mayor cantidad de información contenida en la señal.

Ventaneo: Cada trama tiene que ser multiplicada por una ventana para mantener la continuidad de los primeros y los últimos puntos en la trama. La ventana más común para el reconocimiento, es la ventana de Hamming. Ahora, si la señal en una trama se denota por $s(n)$ donde $n = 0, \dots, N-1$, entonces la señal después de ser ventaneada, a través de la ventana de Hamming, es $s(n) * w(n)$, donde $w(n)$ es la ventana de Hamming definida por:

$$w(n) = 0.54 - 0{,}46 \cos\left(\frac{2\pi n}{N-1}\right), \qquad 0 \leq n \leq N-1 \qquad (3)$$

Análisis espectral: Para realizar el análisis espectral se puede escoger entre varios procedimientos, los más utilizados en el reconocimiento de voz son la FFT y los LPC (que se describen en la sección 4.4). El análisis espectral mediante la FFT muestra que diferentes timbres en la señal de voz corresponden a diferentes distribuciones de energía a través de frecuencias. Sin embargo, como se está más interesado en el desarrollo de la respuesta de frecuencia, en lugar de la respuesta de frecuencia como tal, se usa los filtros pasa-banda triangulares para extraer las características de la señal de voz.

Banco de Filtros de Mel: este proceso consiste en aplicar una serie de filtros pasa-banda triangular diseñados para simular lo que se cree ocurre en el oído humano. Es decir, una serie de filtros pasa-banda con ancho de banda constante e igualmente espaciados en la escala de frecuencia de Mel. La cantidad de filtros

empleados para sistemas de reconocimiento de voz varían entre 10 y 25 filtros. La siguiente función transforma las frecuencias lineales a frecuencias de Mel:

$$mel(f) = 2595 * \log\left(1 + \frac{f}{700}\right) \qquad (4)$$

Los filtros pasa-banda se utilizan para suavizar la magnitud del espectro, de tal manera que los armónicos sean aplanados y así obtener un desarrollo del espectro con armónicos, reduciendo el tamaño de las funciones relacionadas.

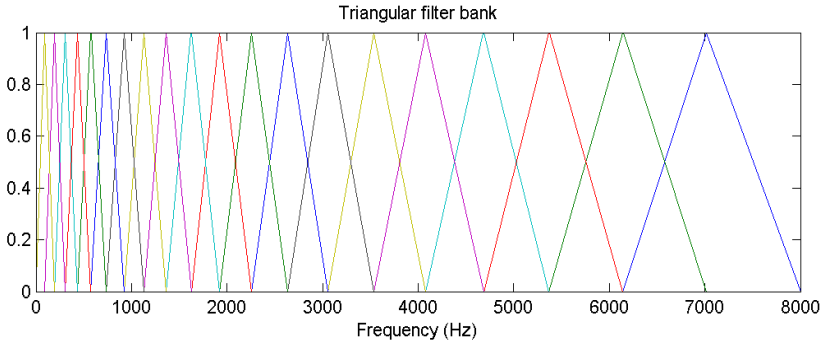


Figura 3. Banco de Filtros triangular[3].

Si consideramos $S(k)$ como la salida del análisis espectral realizado, $M_l(k)$ l -ésimo filtro del banco de filtros, L la cantidad de filtros que conforman al banco de filtros, y N la cantidad de datos obtenida del análisis espectral, entonces se puede obtener una aproximación del espectro en la escala de Mel mediante la ecuación:

[3] Speech Features [En Linea] [Citado: febrero de 2010]. Disponible en Internet: http://neural.cs.nthu.edu.tw/jang/books/audioSignalProcessing/speechFeatureMfcc.asp?title=12-2%20MFCC

$$S_f(l) = \sum_{k=0}^{N/2} S(k) M_l(k), \qquad 0 \leq l < L - 1 \qquad (5)$$

Transformada de Coseno Discreta: Este proceso convierte el logaritmo del espectro de Mel, al dominio del tiempo mediante Transformada de Coseno Discreta (DCT). El resultado de la conversión es lo que se denomina los coeficientes Cepstrales en las frecuencias de Mel. Los C coeficientes se encuentran a partir de la fórmula:

$$c(i) = \sqrt{\frac{2}{L}} \log [S_f(m)] \cos\left[(m - 0.5)\frac{\pi i}{L}\right], \qquad 0 \leq i < C \qquad (6)$$

Donde C es el número de coeficientes cepstrales en escala de Mel. Sin embargo, para un mejor rendimiento, se puede agregar el logaritmo de la energía y realizar la operación delta, como se explica en los dos pasos siguientes.

Logaritmo de la Energía: La energía dentro de una trama es también una característica importante que puede obtenerse fácilmente. De ahí que se suele añadir el logaritmo de la energía como la característica 13 al MFCC.

Delta MFCC y delta-delta MFCC: Como la señal de voz cambia, es necesario añadir características relacionadas con el cambio de la señal a lo largo del tiempo, con el fin de mejorar el sistema de reconocimiento. Por esta razón se calcula la derivada en tiempo de los coeficientes cepstrales. Lo cual introduce nuevas características, como la velocidad y la aceleración. La ecuación para calcular la derivada está dada por:

$$\Delta c(m) = \left[\sum_{k=-K}^{K} k c_{t-k}(m)\right] G, \qquad 1 \leq m \leq C \qquad (7)$$

Como resultado se obtienen los MFCC-Delta ($\Delta c(m)$, coeficientes de velocidad), los cuales dan los cambios por tiempo, y los MFCC-Delta-Delta ($\Delta^2 c(m)$, coeficientes de aceleración), que miden la velocidad de cambio de los MFCC-Delta. El vector definitivo de características de cada trama analizada, corresponde a concatenar los datos de c , Δc y $\Delta^2 c$ en un vector $Q = [c, \Delta c, \Delta^2 c]$.

4.5 COEFICIENTES DE PREDICCIÓN LINEAL

La Codificación por Predicción Lineal (LPC por sus siglas en Inglés, *Linear Predictive Coding*) es una de las técnicas más útiles en el análisis, codificación y reconocimiento de voz, porque representa la envolvente espectral de una señal de voz digitalizada en forma comprimida con una buena aproximación de la

envolvente espectral del tracto vocal, siendo más efectiva en regiones vocales y menos efectiva en regiones no vocálicas. El proceso de extracción de los coeficientes LPC es similar al descrito en la sección 4.4, con una fase de pre-énfasis, una de segmentación y una de ventaneo, antes de obtener los Coeficientes de Predicción Lineal.

El modelo matemático se obtiene asumiendo que cada muestra de la señal de voz está estrechamente relacionada con las muestras anteriores, de manera que el valor presente de la señal se puede obtener como una combinación lineal de p muestras anteriores y la excitación actual escalada:

$$s(n) \approx -\sum_{k=1}^{p} a_k s(n-k) + Gu(n) \qquad (8)$$

Donde: a_k son los coeficientes de predicción lineal (LPC), G la ganancia de excitación y $u(n)$ es la excitación normalizada.

El modelo LPC se puede derivar discretizando un modelo continuo de transmisión acústica, basado en la concatenación de tubos acústicos sin pérdidas.

En el dominio Z, la ecuación (8) es:

$$S(z) = -\sum_{k=1}^{p} a_k z^{-k} S(z) + GU(z) \qquad (9)$$

De donde se obtiene la función de transferencia:

$$H(z) = \frac{S(z)}{GU(z)} = \frac{1}{1 + \sum_{k=1}^{p} a_k z^{-k}} = \frac{1}{A(z)} \qquad (10)$$

Que corresponde a la función de transferencia de un filtro todo-polos.

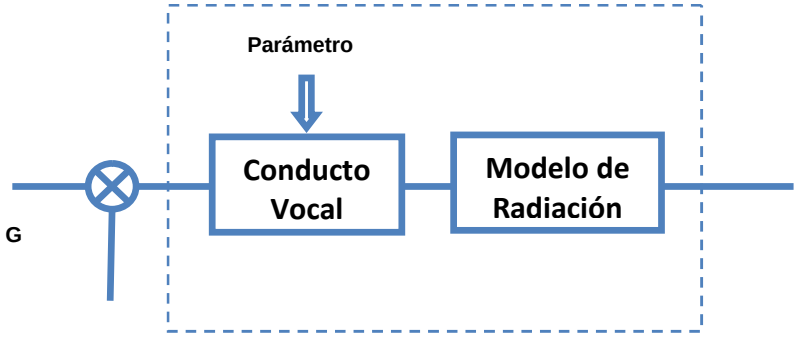


Figura 4. Modelo de predicción lineal de producción de voz.

$H(z) = \frac{1}{A(z)}$, representa la función de transferencia de un modelo lineal del conducto vocal + radiación. Los parámetros del filtro digital $H(z)$, son controlados por la señal de voz que está siendo producida $u(n)$, la cual es escalada por una ganancia G , para producir la señal de voz $s(n)$.

4.5.1 Estimación de los LPC. Una predicción de $s(n)$ basada en p muestras anteriores, se puede definir como sigue:

$$\tilde{s}(n) = -\sum_{k=1}^{p} a_k s(n-k) \qquad (11)$$

Y el error de la predicción como $\varepsilon(n) = s(n) - \tilde{s}(n)$. De donde se obtiene:

$$\varepsilon(n) = s(n) + \sum_{k=1}^{p} a_k s(n-k) \qquad (12)$$

Entonces, cuando la señal $s(n)$ es generada por un sistema lineal como el de la figura 4, el error de estimación $\varepsilon(n)$ deberá ser igual al término de excitación $Gu(n)$. Por lo general los LPC se obtienen minimizando un criterio cuadrático de los errores de predicción $\varepsilon(n)$, para cada ventana en que es dividida la señal de voz.

Suponiendo que en cada ventana hay $m+1 \gg p$ muestras. La ecuación (11) se puede escribir como:

$$\begin{bmatrix} s(n) \\ s(n+1) \\ \vdots \\ s(n+m) \end{bmatrix} = \begin{bmatrix} -s(n-1) & \cdots & -s(n-p) \\ -s(n) & \cdots & -s(n-p+1) \\ \vdots & \vdots & \vdots \\ -s(n+m-1) & \cdots & -s(n+m-1-p) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_p \end{bmatrix} \qquad (13)$$

O equivalentemente:

$$S_m(n) = \Phi^T(n)\alpha \qquad (14)$$

Donde $\Phi(n)$ es la matriz de regresión y α es el vector de coeficientes de predicción lineal.

La estimación de los coeficientes α de predicción lineal ($\tilde{\alpha}$), que minimiza un criterio cuadrático en los errores de predicción, se define por:

$$\vec{\alpha} = \left(\mathbf{\Phi}\mathbf{\Phi}^{\mathbf{T}}\right)^{-1} \mathbf{\Phi} S_m(\square) \qquad (15)$$

Que depende únicamente de las muestras de la señal de voz.

4.6 ALINEAMIENTO TEMPORAL DINÁMICO

El alineamiento temporal dinámico (DTW por sus siglas en inglés, *Dynamic Time Warping*) es una técnica de comparación de patrones. Este método permite alinear temporalmente las características de una palabra, que se desea reconocer, con las características de una palabra de referencia, obtenida en una fase de entrenamiento previa, mediante programación dinámica; esta técnica se utiliza debido a que la duración temporal de los fonemas en dos pronunciaciones diferentes de una misma palabra no siempre son iguales, incluso la duración total de las pronunciaciones pueden diferir bastante, dependiendo de diferentes factores: personalidad del hablante, estado de ánimo, la tranquilidad, etc. Estos factores implican la probabilidad de que si la comparación se realiza entre segmentos de análisis simultáneos de la señal de voz, sus vectores de análisis no coincidan por corresponder a fonemas diferentes.

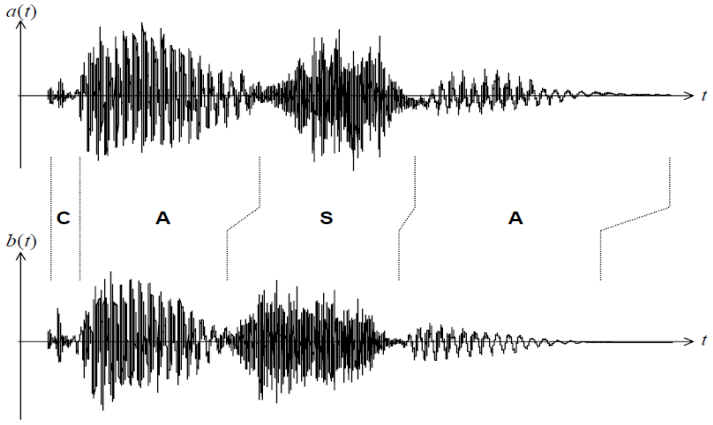


Figura 5. Pronunciación de la palabra CASA en dos momentos diferentes.[4]

La solución a este problema consiste en llevar a cabo un proceso de *alineamiento temporal (DTW)* de los segmentos sucesivos de las dos señales de voz, para comparar segmentos fonéticamente homólogos, es decir segmentos que correspondan a la misma posición fonética en las dos palabras como se muestra en la figura 5.

Para llevar a cabo este proceso, inicialmente se divide la palabra de referencia en segmentos y se les aplica el análisis de extracción de características con el fin de modelar cada palabra como una secuencia de vectores de parámetros, que son

[4] MIYARA, Federico. Alineación Temporal Para El Reconocimiento De Palabras. 2002

obtenidos mediante LPC o MFFC (ver secciones 4.4 y 4.5). Este procedimiento aplicado a la palabra de prueba, y que corresponde a la etapa de extracción de características, se realiza en condiciones similares a las realizadas con las palabras de referencia en la etapa de entrenamiento. Luego, se procede a hacer coincidir la secuencia de los vectores de parámetros de la palabra de prueba con la secuencia de vectores de la palabra de referencia, de manera que las secuencias estén alineadas en tiempo lo mejor posible.

4.6.1 Descripción del alineamiento temporal dinámico. Como se mencionó anteriormente, el objetivo del DTW es alinear de manera óptima la secuencia de vectores de parámetros de entrada con la secuencia del modelo de referencia, para así determinar la similitud o diferencia entre los dos vectores. Para lograr esto se utiliza la distancia Euclídea, no obstante, esta distancia tiene una limitación, pues en el caso de tener dos vectores de características idénticas pero ligeramente desplazados en el eje del tiempo, se establecería que los dos vectores son diferentes, porque la comparación se realiza punto a punto. Sin embargo el algoritmo de alineamiento temporal dinámico (DTW) permite superar esta limitación.

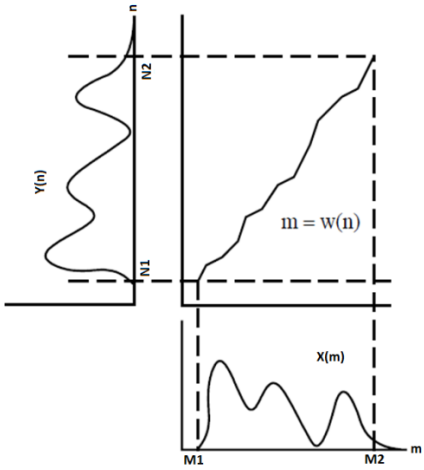


Figura 6. Ilustración del alineamiento temporal.

Sean dos vectores de características:

$$X = [x_1, x_2, \dots, x_i, \dots, x_M] \quad (16)$$

$$Y = [y_1, y_2, \dots, y_j, \dots y_N] \quad (17)$$

Donde X es el vector de referencia, Y el vector de prueba, y M y N las longitudes de los vectores generalmente diferentes por la variabilidad de la duración en la pronunciación.

Para alinear las secuencias usando el DTW, se debe construir una matriz de dimensiones N por M, donde cada elemento de la matriz (i, j) contiene la

distancia $d(x_i, y_j)$ o el alineamiento entre los puntos x_i e y_j. Esta distancia puede ser la distancia Euclídea:

$$D^* = min\left\{\sum_{n=1}^{N} d[y_n, x_{W(n)}]\right\} \quad (18)$$

Donde $d[y_n, x_{W(n)}]$ es la distancia entre el instante n de la secuencia de entrada y el instante $W(n)$ de la plantilla. Además, se necesita una función que relacione las N muestras de la secuencia de prueba y las M de la secuencia de referencia, minimizando la distorsión entre las dos. Esta función es de la forma:

$$m = W(n) \quad (19)$$

Que corresponde al camino de alineación y es un conjunto continuo de elementos de la matriz que definen una trayectoria entre X y Y. El k -esimo elemento de W está definido como $W_k = (i, j)$, por lo cual se tiene:

$$W = [w_1, w_2, \dots, w_K] \quad (20)$$

Donde:

$$\max(M, N) \leq K < M + N - 1 \quad (21)$$

El camino de alineación debe cumplir las siguientes restricciones:

- Debe comenzar en el inicio de cada vector en $W_1 = (1,1)$ y terminar en el final de los vectores en $W_K = (M, N)$. Esto exige que el camino de alineación empiece y termine de forma diagonal entre las dos esquinas opuestas de la matriz.
- Dado el elemento $W_k = (a, b)$ entonces $W_{k-1} = (a', b')$ donde $a - a' \leq 1$ y $b - b' \leq 1$, esto restringe los pasos permitidos a las celdas adyacentes para el camino de alineación.
- Dado el elemento $W_k = (a, b)$ entonces $W_{k-1} = (a', b')$ donde $a - a' \geq 0$ y $b - b' \geq 0$, esto obliga a que los puntos en W estén monotamente espaciados en el tiempo.

Los caminos de alineación que cumplen las anteriores condiciones son muchos, sin embargo, el camino de interés es el que minimiza el costo del alineamiento y puede ser encontrado por medio de programación dinámica, la cual permite evaluar la siguiente iteración que define la distancia acumulada $\gamma(i,j)$ como la distancia $d(i,j)$ encontrada en la celda actual y la mínima de las distancias acumuladas para los elementos adyacentes:

$$\gamma(i,j) = d(x_i, y_j) + \min[\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1)] \quad (22)$$

Si los vectores de características X y Y son idénticos, el camino de alineación al final será una línea recta.

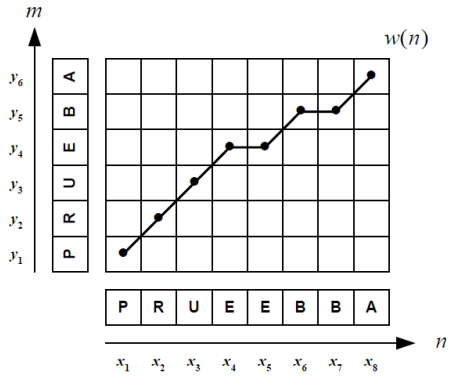


Figura 7. Ejemplo de alineación de patrones.

En resumen el algoritmo de alineamiento temporal dinámico (DTW) requiere de las siguientes condiciones, que optimizan su funcionamiento:

- **Condición monótona:** el camino no se sobrescribirá sobre sí mismo, ambos índices i y j deben permanecer iguales o aumentar, pero nunca disminuir.
- **Condición de continuidad:** el camino avanza un paso a la vez. a lo largo del camino.
- **Condición de contorno:** el camino se inicia en la parte inferior izquierda y termina en la parte superior derecha.
- **Condición de ajuste de ventana:** un buen camino es poco probable que ronde lejos de la diagonal.
- **Condición de limitación de pendiente:** el camino no debe ser demasiado pendiente o demasiado superficial. Esto evita que las secuencias muy cortas se igualen con las secuencias muy largas. Esta condición se expresa como una razón n/m donde m es el número de pasos en la dirección x y n es el número de pasos en la dirección y. Después de los m pasos de x debe dar un paso en y, y viceversa.

Mediante la aplicación de estas condiciones se consigue restringir los movimientos que se pueden realizar desde cualquier punto del camino y así limitar el número de caminos que necesitan ser considerados.

4.7 MODELOS OCULTOS DE MARKOV

En esta sección se describe un método para entrenar y reconocer expresiones de voz (fonemas, palabras u oraciones) a partir de observaciones dadas, $\mathbf{O}_t \in R^D$, donde t es un índice de tiempo, $\mathbf{D}$ es la dimensión del vector y una secuencia de observaciones completa, que se utiliza para describir una expresión, se denota como $\mathbf{O} = [\mathbf{O}_1, \mathbf{O}_2, \dots, \mathbf{O}_T]$. Este método se conoce como Modelo Oculto de Markov o HMM (por sus siglas en inglés, Hidden Markov Model).

El HMM es un método estocástico (aleatorio) que modela el problema dado como un "proceso doblemente estocástico" en donde los datos observados son el resultado de pasar desde un primer proceso “oculto” a un segundo proceso “observado”. No obstante, ambos procesos son representados únicamente por el proceso que pudo ser observado, lo cual ocasiona un problema, pues no se conoce nada acerca de las cadenas de Markov que forman el habla. Además, el número de estados en el modelo es desconocido, las funciones probabilísticas son desconocidas y no se puede decir desde que estado una observación fue producida. Estas propiedades están ocultas y de ahí su nombre Modelo Oculto de Markov. En las siguientes secciones se detallan algunas características que permiten comprender a fondo el funcionamiento de los Modelos Ocultos de Markov.

4.7.1 Modelo de Markov en tiempo discreto. Una cadena de Markov es un proceso estocástico de Markov discreto. En las cadenas de Markov la parte oculta se encuentra descubierta, por lo tanto este sistema es lo que se podría denominar como un Modelo Observable de Markov. Considere que el sistema que puede ser descrito en cualquier momento por uno de un conjunto de N estados distintos indicados por $1,2,\dots,N$. En tiempos discretos espaciados regularmente, el sistema sufre un cambio de estado (posiblemente permanezca en el mismo estado) de acuerdo a un conjunto de probabilidades asociadas al estado. Los instantes de tiempo para un cambio de estado se denotan t y el estado actual en el tiempo t como q_t. En caso de una cadena de Markov de primer orden, las probabilidades de transición de estados no dependen de todo el proceso, sólo del estado anterior y del actual. Esto, que se denomina la propiedad de Markov, se define así:

$$P(\mathbf{q_t = j | q_{t-1} = i, q_{t-2} = k, \dots}) = P(\mathbf{q_t = j | q_{t-1} = i}) \qquad (23)$$

También considere que la parte derecha de la ecuación (23) es independiente del tiempo, lo cual conduce a un conjunto de probabilidades de transición de un estado a otro a_{ij}, de la forma:

$$a_{ij} = P(\mathbf{q_t = j | q_{t-1} = i}), \qquad \mathbf{1 \leq} i, j \leq N \tag{24}$$

Estas probabilidades de estado a_{ij}, tienen las siguientes propiedades:

$$a_{ij} \geq 0 \qquad \forall\, j, i \tag{25}$$

$$\sum_{\mathbf{j=1}}^{\mathbf{N}} a_{ij} = \mathbf{1} \qquad \forall\, i \tag{26}$$

Las probabilidades de transición de estado, para todos los estados en un modelo, puede ser descrito por una matriz de probabilidad de transiciones:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \ldots & a_{NN} \end{bmatrix} NxN \tag{27}$$

Lo único que queda para describir el sistema, es el vector de distribución inicial de estado (la probabilidad de iniciar en un estado u otro). Este vector es descrito por:

$$\boldsymbol{\pi} = [\pi_1 = P(q_1 = 1) \quad \pi_2 = P(q_2 = 2) \quad \ldots \quad \pi_N = P(q_N = N)] \tag{28}$$

La propiedad estocástica para la distribución del estado inicial es:

$$\sum_{i=1}^{N} \pi_i = \mathbf{1} \tag{29}$$

Donde π_i se define como:

$$\pi_i = P(q_1 = i), \qquad 1 \leq i \leq N \tag{30}$$

Estas son las propiedades y las ecuaciones para describir un proceso de Markov "observable". El modelo de Markov puede ser descrito por $\mathbf{A}$ y $\boldsymbol{\pi}$.

4.7.2 Modelo de Markov del clima. Aquí se presenta un ejemplo de un proceso de Markov en tiempo discreto para establecer las ideas sobre las cadenas de Markov. Considere un modelo de Markov de cuatro estados, correspondientes al clima, véase figura 8.

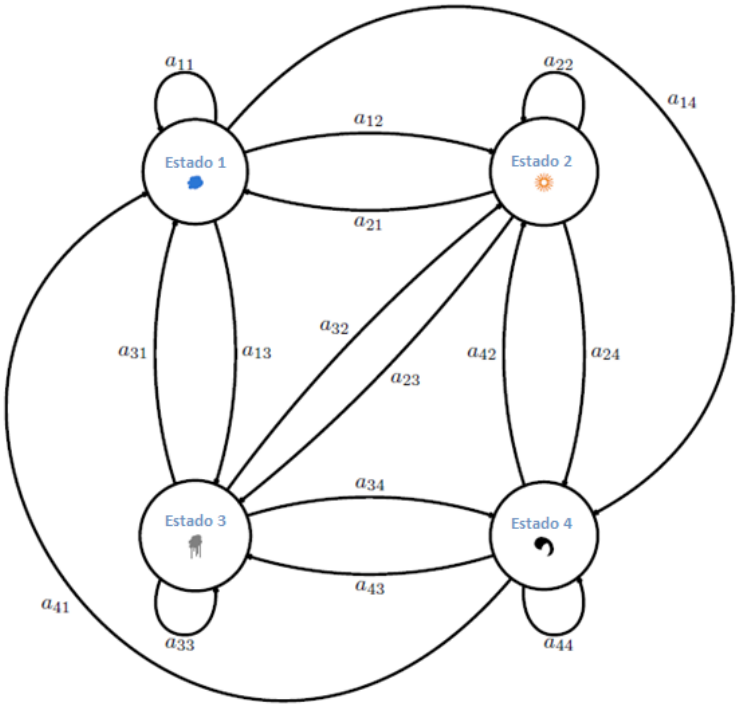


Figura 8. Modelo de Markov del clima.

Suponga que una vez al día (por ejemplo, en la mañana), el clima es observado como uno de los siguientes estados:

- Estado 1: nublado
- Estado 2: soleado
- Estado 3: lluvioso
- Estado 4: tempestuoso

Dado el modelo de la figura 8, ahora es posible responder a varias preguntas de interés sobre los patrones climáticos en el tiempo. Por ejemplo, ¿cuál es la probabilidad de obtener la secuencia "soleado, lluvioso, soleado, tempestuoso, nublado, nublado" en seis días consecutivos? Lo primero que se debe hacer es definir la secuencia de estado $\mathbf{O}$, como:

$$\mathbf{O} = (\mathbf{soleado}, \mathbf{lluvioso}, \mathbf{soleado}, \mathbf{tempestuoso}, \mathbf{nublado}, \mathbf{nublado}) = (2,3,2,4,1,1) \quad (31)$$

Ahora con la secuencia y el modelo de la figura 8, el cálculo de la probabilidad de la secuencia de observación $P(\mathbf{O}\,|\mathbf{A}, \boldsymbol{\pi})$, puede ser directamente determinada así:

$$\begin{aligned} P(\mathbf{O}\,|\mathbf{A}, \boldsymbol{\pi}) &= P(2,3,2,4,1,\mathbf{1}|\mathbf{A}, \boldsymbol{\pi}) \\ &= P(\mathbf{2})P(\mathbf{3}|\mathbf{2})P(\mathbf{2}|\mathbf{3})P(\mathbf{4}|\mathbf{2})P(\mathbf{1}|\mathbf{4})P(\mathbf{1}|\mathbf{1}) \end{aligned}$$

$$= \pi_2 \cdot a_{23} \cdot a_{32} \cdot a_{24} \cdot a_{41} \cdot a_{11} \qquad (32)$$

De manera más general, este cálculo de probabilidad para una secuencia de estado $\mathbf{q} = (q_1, q_2, \ldots, q_T)$ será:

$$P(\mathbf{O} \mid \mathbf{A}, \boldsymbol{\pi}) = \pi_{q1} \cdot a_{q1q2} \cdot a_{q2q3} \cdot \ldots \cdot a_{qT-1qT} \qquad (33)$$

Otro asunto de importancia, es encontrar la probabilidad de que el sistema permanezca en el mismo estado, ya que el sistema se encuentra en un estado conocido, exactamente el día d. Esto corresponde a la siguiente secuencia de observación:

$\mathbf{O} = (i, i, i, \ldots, i, j \neq i)$

$$dia \quad 1\ 2\ 3 \quad d\ \ d+1 \qquad (34)$$

Y a la siguiente probabilidad:

$$\begin{aligned} P(\mathbf{O} \mid \mathbf{A}, \boldsymbol{\pi}, q_1 = i) &= P(\mathbf{O}, q_1 = i \mid \mathbf{A}, \boldsymbol{\pi}) \Big/ P(q_1 = i) \\ &= \pi_i (a_{ii})^{d-1} (1 - a_{ii}) \Big/ \pi_i \\ &= (a_{ii})^{d-1} (1 - a_{ii}) \\ &= p_i(d) \end{aligned} \qquad (35)$$

La probabilidad $p_i(d)$ puede ser vista como la duración en el estado. En base a $p_i(d)$ también es posible encontrar el número esperado de observaciones (duración) de un estado, con la condición de que comiencen en ese estado como:

$$\begin{aligned} E[d_i] &= \sum_{d=1}^{\infty} d \cdot p_i(d) \\ &= \sum_{d=1}^{\infty} d \cdot (a_{ii})^{d-1} (1 - a_{ii}) \\ &= (1 - a_{ii}) \frac{\partial}{\partial a_{ii}} \left(\sum_{d=1}^{\infty} (a_{ii})^d \right) \\ &= (1 - a_{ii}) \frac{\partial}{\partial a_{ii}} \left(\frac{a_{ii}}{1 - a_{ii}} \right) \\ &= \frac{1}{1 - a_{ii}} \end{aligned} \qquad (36)$$

4.7.3 Modelo oculto de Markov en tiempo discreto. El Modelo de Markov en tiempo discreto, descrito en la sección anterior, es demasiado restrictivo para ser aplicable a muchos problemas de interés. Por lo tanto, se extenderá al Modelo Oculto de Markov en tiempo discreto. La extensión hace que todos los estados ya no sean deterministas (por ejemplo, soleado), sino que sean probabilísticos. Esto

significa que cada estado genera una serie de observaciones $\mathbf{O}_t$, de acuerdo con alguna función probabilística. La obtención de la observación dentro de este enfoque estocástico se caracteriza por un conjunto de medidas de probabilidad de observación $B = \{b_j(\mathbf{O}_t)\}_{\mathbf{j=1}}^{\mathbf{N}}$, donde la función de probabilidad para cada estado $\mathbf{j}$ es:

$$b_j(\mathbf{O}_t) = P(\mathbf{O}_t \mid q_t = j) \qquad (37)$$

Para entender cómo trabaja un Modelo Oculto de Markov, a continuación se presenta el denominado modelo de la urna y la bola.

4.7.4 El modelo de la urna y la bola. Supóngase que en un salón se encuentra un número N muy grande de urnas de vidrio. Dentro de cada urna hay una cantidad de bolas de colores. Se asume que hay M distintas bolas de colores. Como se trata de un ejemplo, considere un conjunto de N urnas que contienen bolas con $M = 6$ colores diferentes (R = rojo, O=naranja, B = negro, G=verde, B = azul, P = púrpura), véase la figura 9.

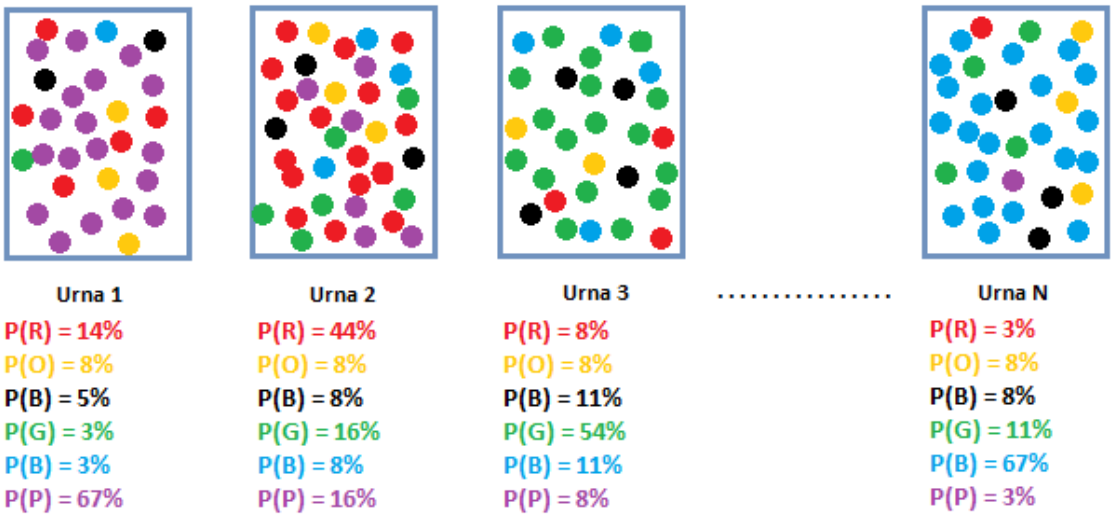


Figura 9. Ejemplo de la urna y la bola

Los pasos para generar una secuencia de observación, para el ejemplo de la urna y la bola, son:

1. Elija un estado inicial (aquí estado igual a urna) $q_1 = i$, de acuerdo a la distribución del estado inicial $\boldsymbol{\pi}$.
2. Establezca $t = 1$ (reloj, $t = 1,2,\ldots,T$).
3. Escoja una bola de la urna seleccionada de acuerdo a la distribución de probabilidades de observación de un símbolo en el estado i, es decir $b_j(\mathbf{O}_t)$

(por ejemplo la probabilidad para una canica de color morado en la primera urna 0.67, véase figura 9). Esta bola de color representa la observación $\mathbf{O}_t$. Poner la bola nuevamente en urna.

4. Cambie a un nuevo estado (urna) $q_{t+1} = j$ de acuerdo con la distribución de probabilidad de transición de estados para el estado i , es decir a_{ij}.
5. Establezca $t = t + 1$; regrese al paso 3 si $t < T$, de lo contrario, de por terminado el procedimiento.

Estos pasos describen como un modelo oculto de Markov funciona cuando se está generando la secuencia de observación. Cabe señalar que los colores de las bolas en cada urna pueden ser los mismos, y la distinción entre las diferentes urnas es la manera en que las bolas de colores están combinadas. Por lo tanto, una observación aislada de una canica de un color en particular no dice inmediatamente de que urna se extrae[5].

4.7.5 Densidades de observación discreta. El ejemplo de la urna y la bola, descrito en la sección anterior, es un ejemplo de un HMM de densidad de observación discreta. Esto debido a que hay M distintos colores. En general, los HMMs de densidad de observación discreta están basados en la partición de la función de densidad de probabilidad (pdf)[6] de la observación en un conjunto discreto de pequeñas celdas y símbolos $V_1, V_2, \dots V_M$ donde cada símbolo representa una celda. Esta partición se conoce generalmente como *vector de cuantización* y hay varios métodos para realizarlo. Inmediatamente se consigue el vector de cuantización, se crea una base de datos o plantilla con el promedio de los vectores para cada grupo (o cluster).

El símbolo correspondiente para la observación se determina por la regla del vecino más cercano, es decir, se selecciona el símbolo de la celda de acuerdo al vector más cercano a la plantilla. Haciendo un paralelo con el modelo de urna y la bola, significa que si una bola de color gris oscuro es observada, esta observación será probablemente más cercana al color negro. En este caso los símbolos $V_1, V_2, \dots V_M$ representan un color cada uno (por ejemplo $V_1 = \text{roja}$). La distribución de probabilidades de observación de un símbolo $B = \{b_j(\mathbf{O}_t)\}_{j=1}^{N}$, tendrá ahora la distribución de símbolo en el estado $\mathbf{j}$, $b_j(\mathbf{O}_t)$, definida como:

$$b_j(\mathbf{O}_t) = b_j(k) = P(\mathbf{O}_t = V_k \mid q_t = j), \qquad 1 \le k \le M \qquad (38)$$

La estimación de las probabilidades $b_j(k)$ se realiza normalmente en dos etapas, primero la determinación de la plantilla y luego la estimación de los conjuntos de probabilidades de observación para cada vector de la plantilla en cada estado. El

[5] NILSSON, Mikael y EJNARSSON, Marcus. Speech Recognition using Hidden Markov Model.
[6] Las siglas *pdf* viene del inglés, probability density function

problema principal con la probabilidad de salida discreta es que la operación del vector cuantización divide el espacio continuo acústico en regiones separadas que destruyen la estructura de señal original. A medida que las distribuciones de salida de diferentes estados se caracterizan por una gran superposición la partición introduce pequeños errores. La falta de fiabilidad de los parámetros estimados debido a la duración de los datos de entrenamiento impide la presentación de plantillas muy grandes. Esto introduce errores de cuantificación que puede causar la degradación en el rendimiento de HMM discretos cuando los vectores de características observados son intermedios entre dos símbolos de la plantilla. Por tal motivo aparecen las densidades de observación continua que permiten aumentar el nivel de reconocimiento aunque esto implique más cálculos.

4.7.6 Densidades de observación continua. Para crear los HMMs de densidades de observación continua $b_j(\mathbf{O}_t)$, se crean algunas funciones de densidad de probabilidad paramétrica o mezclas de ellas. Para poder re-estimar los parámetros de la función de densidad de probabilidad (pdf) se implanta una restricción. Esta restricción limita la pdf a ser una densidad logarítmicamente cóncava o elípticamente simétrica. La representación más general de la pdf, para la que un proceso de re-estimación ha sido formulado, es una mezcla de la forma:

$$b_j(\mathbf{O}_t) = \sum_{\mathbf{k=1}}^{\mathbf{M}} \mathbf{c_{jk}} b_{jk}(\mathbf{O}_t), \qquad j = \mathbf{1}, \mathbf{2}, \dots, N \tag{39}$$

Donde M es el número de mezclas y las restricciones estocásticas para los pesos de las mezclas $\mathbf{c_{jk}}$, son:

$$\sum_{\mathbf{k=1}}^{\mathbf{M}} \mathbf{c_{jk}} = \mathbf{1} \qquad j = \mathbf{1}, \mathbf{2}, \dots, N$$

$$\mathbf{c_{jk}} \geq \mathbf{0} \qquad j = \mathbf{1}, \mathbf{2}, \dots, N, \qquad k = \mathbf{1}, \mathbf{2}, \dots, M$$

Además $b_{jk}(\mathbf{O}_t)$ es una densidad D-dimensional logarítmicamente cóncava o elípticamente simétrica con vector promedio $\boldsymbol{\mu}_{jk}$ y matriz de covarianza $\sum jk$:

$$b_{jk}(\mathbf{O}_t) = \mathcal{N}\left(\mathbf{O}_t, \boldsymbol{\mu}_{jk}, \sum jk\right) \tag{40}$$

La densidad D-dimensional más utilizada, ya sea logarítmicamente cóncava o elípticamente simétrica, es la densidad de Gauss. La densidad de Gauss se puede encontrar mediante:

$$b_{jk}(\mathbf{O}_t) = \mathcal{N}\left(\mathbf{O}_t, \boldsymbol{\mu}_{jk}, \sum jk\right) = \frac{\mathbf{1}}{(2\pi)^{\mathrm{D}/2}\left|\sum jk\right|^{1/2}} \mathbf{e}^{-\frac{1}{2}\left(\mathbf{O}_t-\mu_{jk}\right)^{\mathrm{T}} \Sigma_{jk}^{-1}\left(\mathbf{O}_t-\mu_{jk}\right)} \quad (41)$$

Para aproximar las fuentes simples de observación, las mezclas gaussianas proporcionan una manera fácil de obtener una precisión considerable debido a la flexibilidad y la estimación práctica de la pdf. Si la fuente de la observación genera una pdf complicada de alta dimensión, la mezcla gaussiana puede ser difícil de tratar computacionalmente, debido al número excesivo de parámetros y a las grandes matrices de covarianza.

Cuando se estima un gran número de parámetros, el problema práctico es la disponibilidad de una cantidad necesaria de datos de entrenamiento que sean representativos. Con datos de entrenamiento insuficientes algunos parámetros tendrán valores más o menos arbitrarios y en especial las matrices de covarianza, lo que puede generar problemas.

A medida que la longitud de los vectores de características incrementa, el tamaño de las matrices de covarianza también aumenta en proporción cuadrada a la dimensión del vector. Si se tiene en cuenta que los vectores de características son diseñados para evitar determinados componentes redundantes y que los elementos de la diagonal de las matrices de covarianza son generalmente pequeños, se podría aproximar la covarianza mediante matrices diagonales. Además, el uso de matrices diagonales también proporciona una implementación más sencilla y rápida para el cálculo de probabilidad reduciéndola a:

$$b_{jk}(\mathbf{O}_t) = \mathcal{N}\left(\mathbf{O}_t, \boldsymbol{\mu}_{jk}, \sum jk\right) = \frac{\mathbf{1}}{(2\pi)^{\mathrm{D}/2}\left|\sum jk\right|^{1/2}} \mathbf{e}^{-\frac{1}{2}\left(\mathbf{O}_t-\mu_{jk}\right)^{\mathrm{T}} \Sigma_{jk}^{-1}\left(\mathbf{O}_t-\mu_{jk}\right)}$$

$$= \frac{\mathbf{1}}{(2\pi)^{\mathrm{D}/2}\left(\prod_{\mathrm{l}=1}^{\mathrm{D}} \boldsymbol{\sigma}_{\mathbf{jkl}}\right)^{1/2}} \mathbf{e}^{-\sum_{\mathrm{l}=1}^{\mathrm{D}} \frac{\left(\mathbf{O}_t-\mu_{jk}\right)^2}{2\boldsymbol{\sigma}^2_{\mathbf{jkl}}}} \quad (42)$$

Donde los términos de varianza $\sigma_{\mathbf{jk1}}, \boldsymbol{\sigma}_{\mathbf{jk2}}, \ldots, \boldsymbol{\sigma}_{\mathbf{jkD}}$ son los elementos de la diagonal de la matriz de covarianza.

4.7.7 Tipos de HMMs. Existen diferentes tipos de arquitecturas de HMMs que se pueden utilizar. La arquitectura en los HMMs es definida por la matriz de transición $\mathbf{A}$. Dentro de las diferentes arquitecturas la más general es el HMM ergódico o totalmente conectado. En este modelo, cada estado puede ser alcanzado desde cualquier otro estado del modelo. Un ejemplo de esta arquitectura se muestra en la figura 8, para un modelo de $N = 4$ estados, este modelo tiene la propiedad $0 < a_{ij} < 1$ (el cero y el uno tienen que ser excluidos, de lo contrario la propiedad

ergódica no se cumple). La matriz de transición de estados $\mathbf{A}$, para un modelo ergódico, puede ser descrita por:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}_{4x4}$$

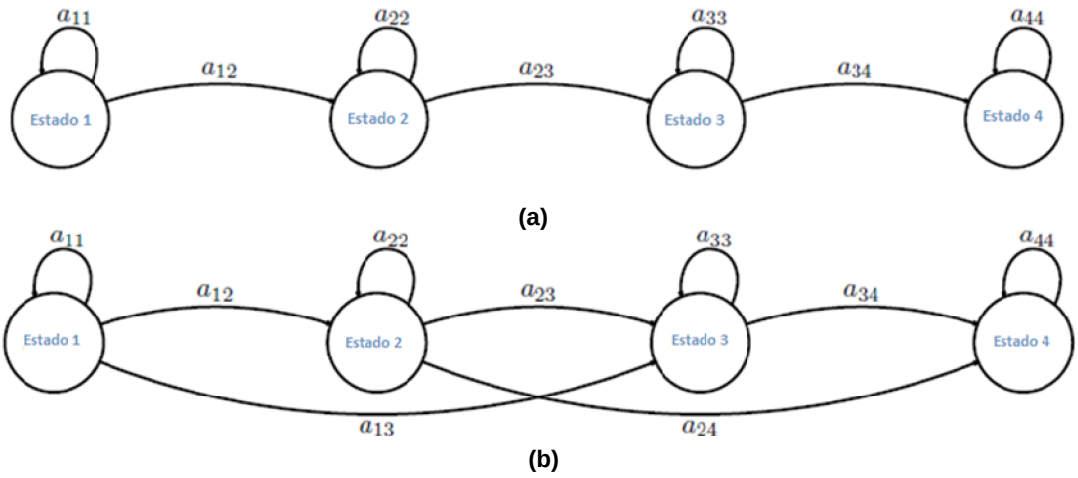


Figura 10. Diferentes arquitecturas izquierda-derecha de HMMs

Sin embargo, en el reconocimiento de voz es conveniente utilizar un modelo que modele las observaciones de una manera sucesiva, ya que esta es la característica de la voz. Los modelos que cumplen con esta técnica de modelado, es el modelo de izquierda-derecha o modelo de Bakis, véase figura 10. La propiedad de un modelo de izquierda-derecha es:

$$a_{ij} = 0, \qquad j < i$$

La cual significa que no se puede saltar a estados anteriores. Las longitudes de las transiciones normalmente se limitan a alguna longitud máxima, típica de dos o tres:

$$a_{ij} = 0, \qquad j > i + \mathbf{\Delta}$$

Tenga en cuenta que, para un modelo de izquierda-derecha, los coeficientes de las transiciones de estado para el último estado tienen la siguiente propiedad:

$a_{NN} = \mathbf{1}$

$$a_{Nj} = 0, \qquad j < N$$

En la figura 10.a $\mathbf{\Delta = 1}$ y la matriz de transición de estado $\mathbf{A}$, es:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & a_{22} & a_{23} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & a_{33} & a_{34} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & a_{44} \end{bmatrix}_{4x4}$$

Y en la figura 10.b $\boldsymbol{\Delta = 2}$ la matriz de transición de estado $\mathbf{A}$, será:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \mathbf{0} \\ \mathbf{0} & a_{22} & a_{23} & a_{24} \\ \mathbf{0} & \mathbf{0} & a_{33} & a_{34} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & a_{44} \end{bmatrix}_{4x4}$$

La escogencia de un modelo de estructura limitada como un modelo de izquierda-derecha, no genera ninguna modificación en los algoritmos de entrenamiento. Esto debido a que cualquier probabilidad de estado del sistema a cero seguirá siendo cero en los algoritmos de entrenamiento.

4.7.8 Resumen de elementos para un Modelo oculto de Markov. Los elementos de un Modelo Oculto de Markov en tiempo discreto se resumen en esta sección. Estos elementos se utilizan más adelante para realizar un sistema de reconocimiento.

1. Número de estados N del modelo. Aunque los estados están ocultos, para muchas aplicaciones prácticas a menudo hay algún significado físico pertinente a los estados del modelo. Por ejemplo, en el modelo de urna y la bola, los estados se corresponden a las urnas. Las etiquetas de los estados son $\{1,2,\dots,N\}$, y el estado en el tiempo t se denota q_t.

2. Parámetros M del modelo o número de símbolos de observación posibles M para cada estado. Si se utiliza densidades de observación discreta, el parámetro M es el número de clases o celdas que se utiliza, por ejemplo M es igual al número de colores en el ejemplo de la urna y la bola. Si se utiliza la densidad de la observación continua, M es representada por el número de mezclas en cada estado. En otras palabras, M corresponde a las salidas físicas del sistema a modelar.

3. Distribución de estado inicial o distribución inicial de probabilidades de cada estado. Se denota como $\boldsymbol{\pi = \{\pi_i\}_{i=1}^{N}}$, donde:

$$\pi_i = P(q_1 = i) \qquad (43)$$

4. Distribución de probabilidades de transición de estado. Indica la probabilidad de que estando el sistema en un estado, en un instante determinado, el mismo

evolucione a otro estado o permanezca en el mismo estado, en el instante subsiguiente. Se denota como $A = [a_{ij}]$, donde:

$$a_{ij} = P(\mathbf{q_{t+1}} = j | \mathbf{q_t} = i), \qquad \mathbf{1} \leq i, j \leq N \qquad (44)$$

5. Distribución de probabilidades de observación de un símbolo. $B = \{b_j(\mathbf{O}_t)\}_{j=1}^{N}$, en el cual la función probabilística para cada estado, j , es:

$$b_j(\mathbf{O}_t) = \mathbf{P}(\mathbf{O}_t | q_t = j) \qquad (45)$$

   El cálculo de $b_j(\mathbf{O}_t)$ puede ser encontrado con las densidades de observación continua o discreta.

Ahora debería quedar claro que una especificación completa de un HMM requiere dos parámetros N y M . Además, es necesario especificar los tres conjuntos de medidas de probabilidad $\boldsymbol{\pi}$, $\mathbf{A}$ y $\mathbf{B}$. Para mayor comodidad estas medidas de probabilidad utilizan la notación λ :

$$\lambda = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi}) \qquad (46)$$

4.7.9 Tres problemas básicos para los modelos de Markov ocultos. En el desarrollo de la metodología de HMM, existen tres problemas básicos que deben ser resueltos para ser aplicados en problemas reales, como el reconocimiento de voz. Estos problemas son:

Problema 1. En primer lugar, dada la secuencia de observación $\mathbf{O} = [O_1, O_2, \dots, O_T]$ y un modelo $\lambda = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$, ¿cómo se calcula la probabilidad de la secuencia de observación, dado el modelo? Es decir, ¿cómo se calcula $P(\mathbf{O}|\lambda)$ de manera eficiente? (Problema de Evaluación)

Problema 2. En segundo lugar, dada la secuencia de observación $\mathbf{O} = [O_1, O_2, \dots, O_T]$, y un modelo $\lambda = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$, ¿cómo encontrar la secuencia de estados $\mathbf{q} = (q_1, q_2, \dots, q_T)$, mas optima (es decir, la que mejor explica la observación)? (Problema de Decodificación)

Problema 3. Y por último y más complicado de los tres problemas, ¿cómo se deben ajustar los parámetros del modelo $\lambda = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$ para maximizar la probabilidad de la secuencia de observación dado el modelo $P(\mathbf{O}|\lambda)$? (Problema de Entrenamiento)

En el primer problema, dado un modelo y una secuencia de observaciones, se busca calcular la probabilidad de que la secuencia observada fue producida por el

modelo. No obstante, y enmarcados en el reconocimiento de voz, este problema también puede ser visto como un problema de reconocimiento, en donde se califica la probabilidad del modelo ante una secuencia de observaciones. Esto resulta muy ventajoso en un reconocedor de palabras aisladas, pues se puede escoger la palabra que se ha dicho escogiendo aquella que tenga la calificación más alta.

En el segundo problema se trata de revelar la parte oculta del modelo, para encontrar una secuencia "correcta" de estados. Debe quedar claro que para todos, excepto para el caso de modelos degenerado, no existe una secuencia "correcta" de estados a ser encontrada. Por lo tanto para situaciones prácticas, se usa un criterio de optimización para solucionar el problema de la mejor manera posible. Infortunadamente, hay muchos criterios de optimización que pueden dar solución al problema, entonces la elección del criterio depende fuertemente del uso que se le vaya a dar a la solución.

En el tercer problema, denominado problema de entrenamiento, se trata de optimizar los parámetros del modelo para describir de forma óptima la manera en que ocurre una determinada secuencia de observación. Aquí, la secuencia de observación usada para ajustar los parámetros del modelo se denomina secuencia de entrenamiento, pues es usada para entrenar al HMM. El problema de entrenamiento es crucial para la mayoría de aplicaciones, ya que permite modificar los parámetros de los modelos para que se ajusten a los datos de observación, es decir, se van a crear modelos que mejor se ajusten al fenómeno.

Para resolver el primer problema se emplea dos algoritmos: el algoritmo de avance (forward algorithm) y el algoritmo de retroceso (backward algorithm). El algoritmo de avance, permite calcular la probabilidad de la secuencia de observación $\mathbf{O} = [\mathbf{O}_1, \mathbf{O}_2, \dots, \mathbf{O}_T]$, dado el modelo λ , es decir, $P(\mathbf{O}\,|\lambda)$ de una manera eficiente. El algoritmo de retroceso, que es la versión inversa del algoritmo de avance, esencialmente propaga las probabilidades en forma inversa para el aprendizaje del HMM.

Para resolver el segundo problema existen muchos criterios, sin embargo el criterio más extendido se denomina algoritmo de Viterbi (Viterbi algorithm) que está basado en programación dinámica. Este algoritmo, trata es de encontrar la mejor secuencia de estados, es decir, maximizar la probabilidad $P(\mathbf{O}, \mathbf{q}\,|\lambda)$.

El principal inconveniente del tercer problema es que no existe ningún método analítico conocido para encontrar los parámetros del modelo, que maximicen la probabilidad de la secuencia de observaciones. No obstante, este problema se puede resolver utilizando un procedimiento iterativo que se conoce como el algoritmo de Baum-Welch. El procedimiento del algoritmo consiste en actualizar

los pesos de forma iterativa para poder explicar mejor las secuencias de entrenamiento observadas.

5. REVISIÓN SOBRE SISTEMAS NEURO-DIFUSOS

En el capítulo anterior se hace una introducción de la temática referente al reconocimiento de voz, ahora se va a presentar los sistemas neuro-difusos como una posible solución a algunos de los problema de reconocimiento de patrones, especialmente al reconocimiento de fonemas en el campo del ASR. Se empieza con una descripción de las generalidades de las redes neuronales artificiales. Luego se realiza una descripción general de la lógica difusa y se finaliza con una explicación a grandes rasgos de los sistemas neuro-difusos.

5.1 REDES NEURONALES ARTIFICIALES

Las redes neuronales artificiales (denominadas habitualmente como RNA o en inglés ANN) son sistemas de procesamiento de información cuya arquitectura y funcionamiento están inspirados en las redes neuronales biológicas y en la forma en que funciona el sistema nervioso. Los tres conceptos clave de los sistemas nerviosos, que se pretende emular en los artificiales, son: el paralelismo de cálculo (manipulación y procesamiento de gran cantidad de información), la memoria distribuida (tolerancia a fallos) y adaptabilidad (modifican su sinapsis dependiendo al entorno y aprenden de la experiencia). De esta manera, podemos hablar de las redes neuronales como sistemas paralelos, distribuidos y adaptativos, que pueden presentar un cierto comportamiento “inteligente”.

La idea que subyace en los sistemas neuronales artificiales es que, para abordar el tipo de problemas que el cerebro resuelve con eficiencia, puede resultar conveniente construir sistemas que copien en cierta medida la estructura de las redes neuronales biológicas con el fin de alcanzar una funcionalidad similar.[7]

Las características de las redes neuronales artificiales son:

- *Aprender*: Pueden adquirir el conocimiento de una cosa por medio del estudio, ejercicio o experiencia, por lo cual pueden cambiar su comportamiento de acuerdo al entorno. Se le muestra un conjunto de entradas y las neuronas de la red se ajustan para producir salidas consistentes.
- *Generalizar*: extender o ampliar una cosa, las redes neuronales generalizan automáticamente debido a su propia estructura y naturaleza. Pueden

[7] MARTIN DEL BRIO, Bonifacio y MOLINA, Alfredo. Redes Neuronales y Sistemas Borrosos. 3ra edición 2007

ofrecer dentro de un margen, respuestas correctas a entradas que presentan pequeñas variaciones debido a los efectos del ruido.

- *Abstraer*: aislar mentalmente o considerar por separado las cualidades de un objeto. Algunas redes son capaces de abstraer la esencia de un conjunto de entradas que aparentemente no presentan aspectos comunes.

En consecuencia las RNA aprenden de la experiencia, generalizan de ejemplos previos a ejemplos nuevos y abstraen las características principales de una serie de datos.

5.1.1 Estructura básica de una red neuronal. Los elementos básicos de un sistema neuronal biológico son las neuronas, que se agrupan en conjuntos compuestos por millones de ellas organizadas en capas, constituyendo un sistema con funcionalidad propia. Un conjunto de estos subsistemas da lugar a un sistema global. En la realización de un sistema neuronal artificial puede establecerse una estructura jerárquica similar. El elemento esencial de partida será la neurona artificial, que se organizara en capas; varias constituirán una red neuronal; y por último, una red neuronal (o conjunto de ellas), junto con las interfaces de entrada y salida, mas los módulos convencionales adicionales necesarios, constituirán el sistema global de proceso.[8] Un sistema neuronal, está compuesto por los siguientes elementos:

- Un conjunto de procesadores elementales o neuronas artificiales.
- Un patrón de conectividad o arquitectura.
- Una dinámica de activaciones.
- Una regla dinámica de aprendizaje.
- El entorno donde opera.

5.1.2 Modelo general de una Neurona artificial. En este punto se describe la estructura genérica de una neurona artificial, que es un dispositivo simple de cálculo que, a partir de un vector de entrada procedente del exterior o de otras neuronas, proporciona una única respuesta o salida. Los elementos que constituyen la neurona de etiqueta i son los siguientes (véase la Figura 11):

[8] MARTIN DEL BRIO, Bonifacio y MOLINA, Alfredo. Redes Neuronales y Sistemas Borrosos. 3ra edición 2007

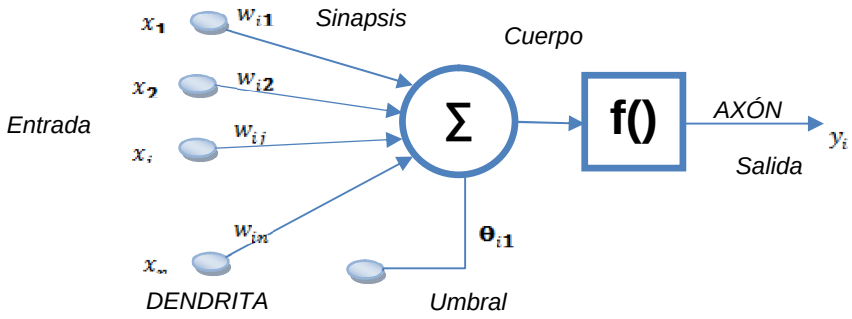


Figura 11. Modelo de neurona estándar.

- Conjunto de ***entradas***, $x_j(t)$, que constituyen el vector de entrada
- Conjunto de ***pesos sinápticos*** w_{ij} que representa la intensidad de interacción entre cada neurona presináptica j y la neurona postsináptica i .
- Una ***regla de propagación*** $\sigma(w_{ij}, x_j(t))$, que proporciona el valor del potencial postsinaptico $h_i(t) = \sigma(w_{ij}, x_j(t))$, de la neurona i en función de sus pesos y entradas.
- Una ***función de activación*** $f_i(a_i(t-1), h_i(t))$, que proporciona el estado de activación actual $a_i(t) = f_i(a_i(t-1), h_i(t))$ de la neurona i , en función del estado anterior y su potencial postsináptico actual. Esta función que puede ser lineal o no lineal, se emplea para limitar la amplitud de la salida de la neurona.

  Dentro de las funciones de activación se destacan (véase la Figura 12):
  - la función de activación lineal: $y = x$
  - la función de activación de mantenimiento: $y = \begin{cases} 0 \; si \; x \leq 0 \\ 1 \; si \; x > 0 \end{cases}$
  - la función de activación sigmoidea: $y = \dfrac{1}{1 + e^{-x}}$

  La más utilizada en el reconocimiento de patrones es la función sigmoidea, porque presenta ventajas de no linealidad, continuidad y diferenciabilidad, lo que permite a una red neuronal multicapa computar cualquier valor real, además de soportar diferentes tipos de algoritmos de entrenamiento.

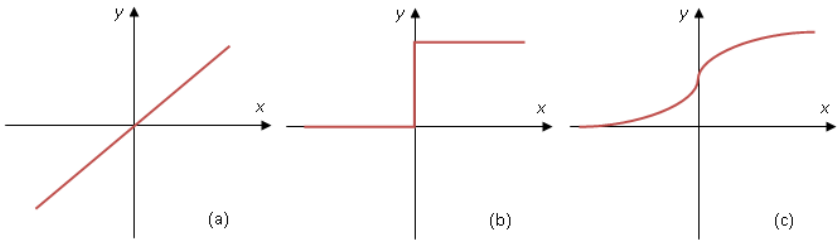


Figura 12. Funciones de activación (a) lineal, (b) mantenimiento (c) sigmoidea.

- Una ***función de salida*** $F_i(a_i(t))$, que proporciona la salida actual $y_i(t) = F_i(a_i(t))$ de la neurona en función de su estado de activación.

De este modo, la operación de la neurona i puede expresarse como:

$$y_i(t) = F_i\left(f_i\left[a_i(t-1), \sigma_i\left(w_{ij}, x_j(t)\right)\right]\right) \qquad (47)$$

Este modelo de neurona formal se inspira en la operación de la neurona biológica, en el sentido de integrar una serie de entradas y proporcionar cierta respuesta, que se propaga por el axón.

5.1.3 Arquitecturas neuronales. Los elementos básicos citados anteriormente se pueden conectar entre sí para dar lugar a las estructuras neuronales, las cuales se pueden clasificar de diferentes formas según el criterio que se utilice.

Según el número de capas: se encuentran las *redes neuronales monocapas*, que son las más simples, ya que solo tienen una capa de neuronas que proyectan las entradas a una capa de neuronas de salida donde se realizan diferentes cálculos, y las *redes neuronales multicapa*, que son una generalización de las redes monocapa, en las cuales existen un conjunto de capas intermedias entre la entrada y la salida (capas ocultas).

Según el tipo de conexiones: se encuentran las *redes neuronales no recurrentes*; donde la propagación de las señales se produce en un solo sentido, no existe la posibilidad de realimentación y no tienen memoria; y las *redes neuronales recurrentes*; que se caracterizan por la existencia de lazos de realimentación entre neuronas de diferentes capas neuronales, neuronas de la misma capa o entre una misma neurona.

Según el grado de conexión: donde se encuentran las *redes neuronales totalmente conectadas*; donde las neuronas de una capa se encuentran conectadas con las neuronas de la capa siguiente (redes no recurrentes) o con las de la capa anterior (redes recurrentes); y las *redes parcialmente conectadas*; donde la conexión no es total entre neuronas de diferentes capas.

5.1.4 Modos de operación de las redes neuronales. Clásicamente se distinguen dos modos de operación en los sistemas neuronales: el modo de recuerdo o ejecución, y el modo de aprendizaje o entrenamiento. Este último es de particular interés, pues una característica fundamental de las RNA es que se trata de sistemas entrenables, capaces de realizar un determinado tipo de procesamiento o computo aprendiéndolo a partir de un conjunto de patrones de aprendizaje o ejemplos.

Fase de Entrenamiento. El objetivo del entrenamiento de una red neuronal es conseguir que en una aplicación determinada, un conjunto de entradas produzca un conjunto de salidas deseadas o mínimamente consistentes. Por eso, el proceso de entrenamiento consiste en la aplicación secuencial de diferentes conjuntos o vectores de entrada, con el fin de ajustar los pesos de las interconexiones según un procedimiento, por medio del cual las conexiones varíen y se produzcan la salida deseada (algoritmo de aprendizaje)

$$\Delta w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t) \qquad (48)$$

Los algoritmos de entrenamiento o los procedimientos de ajuste de los valores de las conexiones de las redes neuronales se pueden clasificar en dos categorías: entrenamiento supervisado y entrenamiento no supervisado.

En el *entrenamiento supervisado*, se presenta a la red las salidas que debe proporcionar ante las señales de entrada que se le suministra, es decir requieren del emparejamiento de cada vector de entrada con su correspondiente vector de salida. Este tipo de entrenamiento consiste en presentar un vector de entrada a la red, calcular la salida de la red, compararla con la salida deseada y el error o diferencia resultante se utiliza para realimentar la red y así modificar los pesos de acuerdo con un algoritmo que tiende a minimizar el error. El entrenamiento supervisado admite dos variantes:

- *Aprendizaje por refuerzo*: solo se conoce si la salida de la red se corresponde o no con la señal deseada. La información es de tipo booleana (verdadero o falso).
- *Aprendizaje por corrección*: se conoce la magnitud del error y se determina la magnitud en el cambio de los pesos.

En el *Entrenamiento no supervisado*, no se conoce la señal que debe dar la red neuronal (señal deseada), por ende no se realizan comparaciones entre las salidas reales y las esperadas. Aquí la misma red se organiza agrupando según sus características las diferentes señales de entrada. En otras palabras, se modifica los pesos de la red de forma que produzca vectores de salida consistentes. En este proceso de entrenamiento se extrae las propiedades estadísticas del conjunto de vectores de entrenamiento y se agrupan en clases los vectores similares. Estos sistemas proporcionan un método de clasificación de las diferentes entradas mediante técnicas de agrupamiento o clustering.

Fase de Ejecución (Estabilidad). Una vez el sistema neuronal ha sido entrenado, se desconecta. Por lo cual los pesos y estructuras quedan fijos, esto en las redes unidireccionales significa que ante un patrón de entrada, las neuronas responden proporcionando directamente la salida del sistema. Al no existir bucles de

realimentación no existen problemas en relación con su estabilidad, lo contario ocurre con las redes con realimentación, sistemas dinámicos no lineales, los cuales requieren de ciertas condiciones para que su respuesta termine convergiendo a un estado estable o fijo, el método más utilizado en estos casos es el *método de Lyapunov*.

Básicamente este método establece que si un sistema dinámico de variables de entrada $(x_1, x_2, \dots, x_n)$ y descrito por el sistema de ecuaciones diferenciales:

$$\dot{x_i} = \frac{dx_i}{dt} = F(t, x_1, x_2, \dots, x_n) \qquad (49)$$

Se cumplen las condiciones:

a) El sistema está en reposo únicamente en el reposo.
b) Existen las derivadas de las ecuaciones que lo describen en todo el dominio.
c) Las variables están acotadas.

Y se puede encontrar una función de Lyapunov V de las variables x_i, $V: R^n \to R$, tal que:

$$\dot{V} = \sum_{i=1}^{n} \frac{\partial V}{\partial x_i} \leq 0, \qquad \forall \dot{x_i} \qquad (50)$$

Entonces el sistema converge para todas las posibles entradas y es globalmente estable.

5.2 LÓGICA DIFUSA

Revisados algunos conceptos basados sobre las redes neuronales (RNA), que emulan de forma simplificada la estructura del cerebro para reproducir sus capacidades, en esta sección se hablara sobre la lógica difusa, que mas bien emulan la manera en que el cerebro razona o piensa.

La denominada lógica difusa (o en inglés, Fuzzy Logic) permite tratar información imprecisa, como *estatura media*, *temperatura baja* o *mucha fuerza*, en términos de conjuntos borrosos o difusos (imprecisos). Estos conjuntos difusos se combinan en reglas para definir acciones, como ejemplo *Si* la temperatura es alta *entonces* enfría mucho. De esta manera, los sistemas de control basados en lógica difusa combinan unas variables de entrada (definidas en términos de conjuntos difusos), por medio de grupos de reglas que producen uno o varios valores de salida.

Los sistemas basados en lógica difusa pueden ser aplicados a similares problemas que las redes neuronales, de modo que resultaran especialmente

interesantes para problemas no lineales o no bien definidos. De la misma manera, los sistemas difusos permiten modelar cualquier proceso no lineal y aprender de los datos haciendo uso de determinados algoritmos de aprendizaje (a veces tomados de otros campos, como las propias redes neuronales). No obstante, a diferencia de los sistemas neuronales, los basados en lógica difusa permiten utilizar fácilmente el conocimiento de los expertos en un tema para una optimización automática. Además, gracias a la simplicidad de los cálculos necesarios, normalmente pueden realizarse en sistemas baratos y rápidos.[9]

5.2.1 Teoría de conjuntos difusos. En la teoría clásica de conjuntos un elemento cualquiera que pertenece a un universo X, pertenece o no pertenece a un subconjunto A incluido en X, sin que exista otra posibilidad al margen de esas dos. La pertenencia o no de un elemento arbitrario a un subconjunto A, viene dada en la mayoría de los casos por la verificación o no de un predicado que caracteriza a A y da lugar a una bipartición del universo del discurso X.

La lógica de conjuntos difusos, trabaja con conjuntos que no tienen límites perfectamente definidos, en otras palabras, la transición entre la pertenencia o no de una variable a un conjunto es gradual. Caracterizada por las ***funciones de inclusión o pertenencia*** (en inglés, *membership function*), las cuales dan flexibilidad a la modelización utilizando expresiones lingüísticas como: mucho, poco, leve, severo, escaso, suficiente, caliente, frio, joven, viejo, etc. La función de pertenencia representa numéricamente la pertenencia o no de un elemento a un conjunto A, esto se expresa por la siguiente expresión:

$$A = \{(x, \mu_A(x)) \,|\, x \epsilon X\} \qquad (51)$$

Donde $\mu_A(x)$, es la función de pertenencia al conjunto difuso.

La *Función de pertenencia* juega un papel clave en el estudio de la teoría de conjuntos difusos, porque esta función da a cada elemento de X un grado de pertenencia al conjunto A. El valor de esta función esta en el intervalo de [0,1], siendo 1 el valor para la máxima pertenencia. Si el valor de esta función se restringiera únicamente a los valores 0 y 1, se tendría un conjunto clásico (no difuso). Normalmente esta función se define de forma arbitraria, adquiriendo cualquier forma gráfica. Las más frecuentes son la función de tipo trapezoidal, singleton, triangular, S, exponencial y tipo π (forma de campana) que se muestran en la figura 13.

[9] MARTIN DEL BRIO, Bonifacio y MOLINA, Alfredo. Redes Neuronales y Sistemas Borrosos. 3ra edición 2007

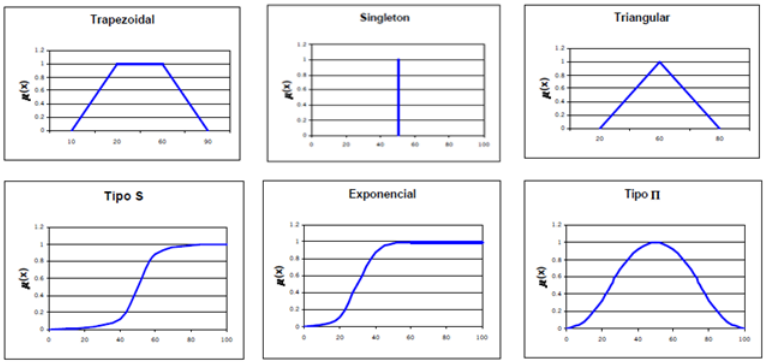


Figura 13. Funciones de pertenencia.[10]

Los conjuntos difusos presentan operaciones típicas: intersección, unión y complemento. De esta forma se pueden realizar operaciones del mismo modo que se realizan en los conjuntos clásicos.

La lógica difusa presenta un razonamiento aproximado, es decir, el razonamiento trata con conceptos poco precisos desde el punto de vista de la lógica clásica. De esta forma es posible inferir un resultado aunque el antecedente no verifique la regla completamente. Entonces el resultado también será un concepto difuso con su correspondiente función de pertenencia.

5.2.2 Estructura de un sistema basado en lógica difusa. Uno de los aspectos más interesantes que presentan los diseños basados en lógica difusa es su sencillez, porque no es necesario partir de un modelo matemático del sistema a controlar, solo basta con tener una idea general de cómo funciona el mismo. Para esto se definen los rangos de las variables de entrada y salida, las funciones de pertenencia asociadas a cada una de ellas y el conjunto de reglas que definen el sistema.

Un sistema basado en razonamiento difuso funciona como un sistema de control que recibe información en valores numéricos de conjuntos clásicos o valores numéricos y los transforma en lógica de variables lingüísticas haciendo uso de funciones de pertenencia contenidas en cada nivel difuso del sistema.

[10] Una descripción detallada de las funciones en: MARTIN DEL BRIO, Bonifacio y MOLINA, Alfredo. Redes Neuronales y Sistemas Borrosos. 3ra edición 2007 págs. 251-256

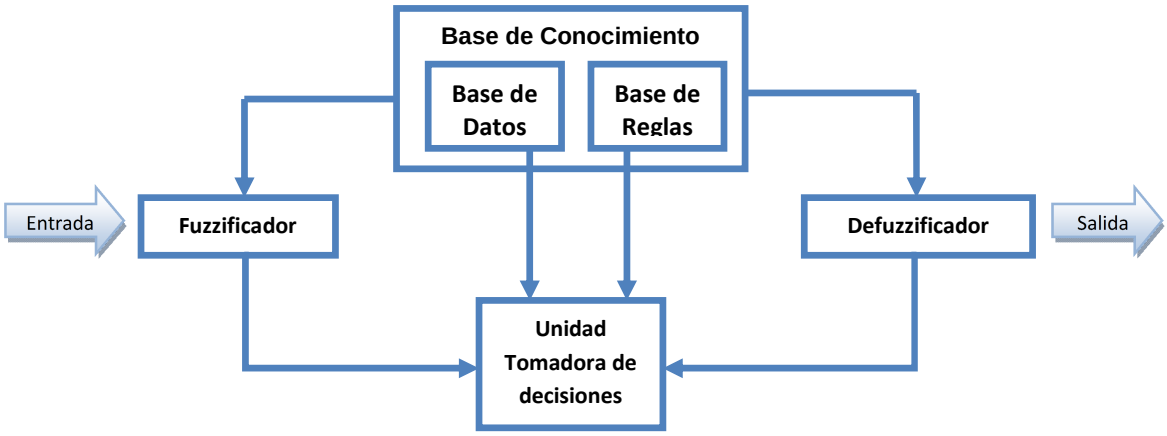


Figura 14. Estructura de un sistema basado en lógica difusa.

La primera fase del sistema, denominada *capa de fuzzyficación*, utiliza un ***fuzzificador*** (en inglés, *Fuzzyfier*) para realizar la conversión de valores discretos a términos borrosos o difusos. Esta fase opera como rótulos lingüísticos y se encarga de asignar a las variables numéricas de entrada los grados de pertenecía a diferentes clases o conjuntos difusos.

La segunda fase del sistema aplica un conjunto de reglas del tipo IF-THEN a las variables de entrada. Los resultados de las diferentes reglas se agrupan para obtener la salida difusa (variable lingüística). Este procesamiento se fundamenta en una base de datos y reglas que representan el conocimiento del especialista que elaboro el conocimiento a ser aplicado por el sistema.

La tercera fase del sistema, denominada *capa de defuzzyficación*, utiliza un ***defuzzificador*** (en inglés, *Defuzzyfier*), para convertir al conjunto difuso de salida en un valor numérico (conjunto clásico), necesario para el mecanismo de control. Para ello se utilizan diferentes métodos, entre los que se destacan: el método de máxima pertenencia y el método de centro de gravedad.

5.3 SISTEMAS NEURO-DIFUSOS

Los sistemas basados en lógica difusa son aplicados a un gran número de problemas en ingeniería, pero los métodos difusos puros no son utilizados en sistemas dinámicos, o sea, sistemas donde se torna esencial un entrenamiento o ajuste de los parámetros para interactuar mejor con el ambiente en el cual el sistema está inmerso. Esto se debe a que un sistema difuso básicamente está proyectado para trabajar con variables de entrada y salida difusas muy bien definidas, por lo tanto no hay posibilidad de ajustes en sus variables o correcciones en sus parámetros.

Por otro lado, las redes neuronales poseen una gran capacidad de adaptación al ambiente, a través de los ajustes aplicados a sus parámetros. Esa adaptación es ocasionada por su capacidad de aprendizaje y generalización. Una red neuronal pasa por una etapa de entrenamiento; en esta etapa, aprende (con ajuste de parámetros) la mejor manera para interactuar con las variables del ambiente, consiguiendo así una actuación precisa y optimizada para la solución de un problema. Sin embargo, trabajar con sistemas de redes neuronales no proporciona facilidad para el procesamiento de variables lingüísticas dominantes en el mundo real.

Los sistemas híbridos son sistemas que hacen uso de por lo menos dos paradigmas de procesamiento de información sobre un único sistema. En el caso de sistemas híbridos que hacen uso de la lógica difusa y redes neuronales, denominados sistemas neuro-difusos (en inglés, *fuzzy neural networks*), el gran objetivo es unir las ventajas de procesamiento lingüístico de un sistema difuso con la capacidad de adaptación y aprendizaje de las redes neuronales. Estos sistemas desarrollan las propiedades y ventajas propias de cada tecnología en beneficio de la otra, complementándola, obteniendo así una mejora importante en el comportamiento global del sistema.[11]
La cooperación de las dos tecnologías se realiza en los siguientes pasos:

a) A partir del conocimiento que tienen los expertos del sistema a controlar, se infieren las funciones de pertenencia y reglas difusas que definen el modelo difuso del sistema.
b) Se establecen las conexiones y el valor inicial de los pesos de la red neuronal de acuerdo al modelo difuso.
c) Se aplica la red neuronal al sistema objeto de estudio.
d) La red neuronal se entrena con datos obtenidos del sistema objeto de estudio, con el fin de mejorar su precisión.
e) Luego del entrenamiento, las conexiones y pesos de la red neuronal son interpretados como funciones de pertenencia y reglas difusas.

Así queda de forma explícita el conocimiento adquirido por la red neuronal y el modelo difuso del sistema que representa con una mayor precisión al sistema real.

Básicamente existen tres formas de acoplamiento de modelos neuro-difusos: recurrentes, cooperativos e incorporados.

El **Modelo recurrente (o secuencial)**, se caracteriza por presentar un subsistema diseñado con un modelo que actúa sobre los datos de entrada del sistema y otro subsistema con otro modelo que actúa sobre la salida del primero. De esta forma,

[11] SANDMANN, H. Predição não-Linear de séries temporais usando sistemas de arquitectura neuro-fuzzy. Tesis de Maestría. Universidad de São Paulo. 2006

el primero procesa y prepara los datos de entrada para una adquisición secuencial del segundo subsistema, el cual posee un modelo de tratamiento de la información diferente del primero.

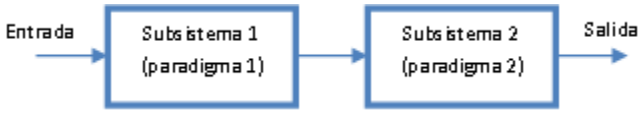


Figura 15. Modelo Neuro-difuso recurrente (Híbrido secuencial).

El **Modelo cooperativo (o auxiliar)**, tiene como característica el uso de un subsistema auxiliar diseñado con un modelo diferente del subsistema principal. Generalmente, se utilizan para la optimización de un proceso en el cual el prototipo principal es deficiente, o en una situación en el cual el subsistema auxiliar sea más óptimo que el subsistema principal.

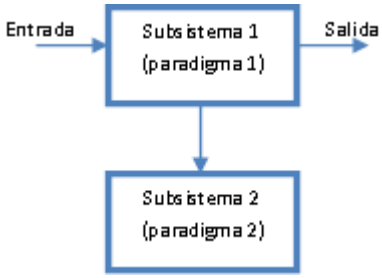


Figura 16. Modelo Neuro-difuso Cooperativo (Híbrido auxiliar).

El **Modelo incorporado,** está representado por sistemas que poseen un alto grado de acoplamiento entre los modelos utilizados, permitiendo distinguir con certeza donde comienza y termina el acoplamiento. Este tipo de modelo se utiliza para diseñar sistemas que suplan las fallas de un modelo con las cualidades de otros.[12]

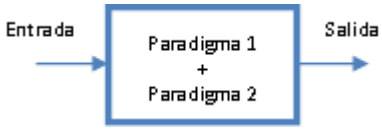


Figura 17. Modelo Neuro-difuso Incorporado.

[12] SANDMANN, H. Predição não-Linear de séries temporais usando sistemas de arquitectura neuro-fuzzy. Tesis de Maestría. Universidad de São Paulo. 2006

6. REVISIÓN SOBRE .NET FRAMEWORK Y LA LENGUA DE SEÑAS

En el capítulo anterior se realizo una introducción a la temática de las redes neuro-difusas para ser aplicadas al reconocimiento de voz. Ahora, el presente capítulo se divide en dos secciones, en la primera se presenta el .NET framework, como un conjunto de tecnologías que permiten desarrollar una plataforma sencilla y potente a través de soluciones pre-codificadas, y en la segunda una definición general de la lengua de señas y del porqué su uso dentro de la comunidad sorda.

6.1 .NET FRAMEWORK

El *Microsoft .NET Framework* es el conjunto de nuevas tecnologías en las que Microsoft ha estado trabajando durante los últimos años con el objetivo de obtener una plataforma sencilla y potente para distribuir el software en forma de servicios que puedan ser suministrados remotamente y que puedan comunicarse y combinarse unos con otros de manera totalmente independiente de la plataforma, lenguaje de programación y modelo de componentes con los que hayan sido desarrollados. Es decir que .NET Framework aparte de ser compatible con múltiples lenguajes de programación, permite interoperabilidad entre dichos lenguajes, en donde cada lenguaje puede utilizar el código escrito en otros lenguajes.

El .NET Framework, incluido en los sistemas operativos Microsoft Windows, provee soluciones pre-codificadas para requerimientos comunes de los programas y gestiona la ejecución de programas escritos específicamente para este framework. Las soluciones pre-codificadas, que forman la librería .NET, cubren un

amplio rango de necesidades de programación, y son utilizadas por los programadores que la combinan con su propio código para producir nuevas aplicaciones. El framework incluye soluciones en áreas como: la interfaz de usuario, acceso a datos, conectividad a bases de datos, criptografía, desarrollo de aplicaciones web, algoritmos numéricos y comunicación de redes.

6.1.1 Principales características de diseño.

Interoperabilidad: Como los sistemas de computación con frecuencia requieren la interacción entre aplicaciones nuevas y mayores, .NET Framework proporciona los medios para acceder a esta funcionalidad que se implementa en los programas que se ejecutan fuera del entorno .NET. El acceso a componentes COM se proporciona en el System.EnterpriseServices y el espacio de nombres System.Runtime.InteropServices del framework, el acceso a otras funciones se proporciona con la función P/Invoke.

Common Runtime Engine: El Common Language Runtime (CLR) es el motor de ejecución del .NET Framework. Todos los programas .NET se ejecutan bajo la supervisión del CLR, garantizando determinadas propiedades y comportamientos en las áreas de manejo de memoria, seguridad y manejo de excepciones.

Independencia del lenguaje: El .NET Framework introduce un Common Type System (CTS) o Sistema de Tipo Común. EL CTS define todos los posibles tipos de datos, programaciones soportadas por el CLR y la forma en que pueden o no pueden interactuar entre sí de acuerdo a la especificación Common Language Infrastructure (CLI) o Lengua de Infraestructura Común. Debido a esta característica, el .NET Framework admite el intercambio de tipos y objetos entre las librerías y las aplicaciones escritas utilizando cualquier ajuste al lenguaje .NET.

Base Class Library: La Base Class Library (BCL) o Librería de clase base, que forma parte del Framework Class Library (FCL), es una librería incluida en el .NET Framework formada por cientos de tipos de datos que permiten acceder a los servicios ofrecidos por el CLR y a las funcionalidades más frecuentemente usadas a la hora de escribir programas. Además, a partir de estas clases prefabricadas el programador puede crear nuevas clases que mediante herencia extiendan su funcionalidad y se integren a la perfección con el resto de clases de la BCL.

Dado la amplitud de la BCL, ha sido necesario organizar las clases en ella incluida en Espacios de Nombres que agrupen clases con funcionalidades similares.

Implementación Simplificada: .NET Framework incluye características de diseño y herramientas que ayudan a administrar la instalación de software para asegurarse de que no interfiera con el software instalado previamente y que se ajuste a los requisitos de seguridad.

Seguridad: El diseño está destinado a abordar algunas de las vulnerabilidades, tales como desbordamientos de búfer, que han sido explotadas por software malicioso. Además, .NET proporciona un modelo de seguridad común para todas las aplicaciones.

Portabilidad: El diseño del .NET Framework lo hace una multiplataforma compatible. Es decir, que un programa escrito para utilizar el framework debe funcionar sin cambios en cualquier tipo de sistema para el cual el framework es implementado. Microsoft presentó el pliego de condiciones para el *Common Language Infrastructure* (que incluye las librerías de clase base, el *Common Type System*, y el *Common Intermediate Language*), el lenguaje C#, y el lenguaje C++/CLI tanto a ECMA como a ISO, haciéndolas disponibles como normas abiertas. Esto hace posible que terceras partes puedan crear implementaciones compatibles al framework y a sus lenguajes en otras plataformas.

6.1.2 Arquitectura

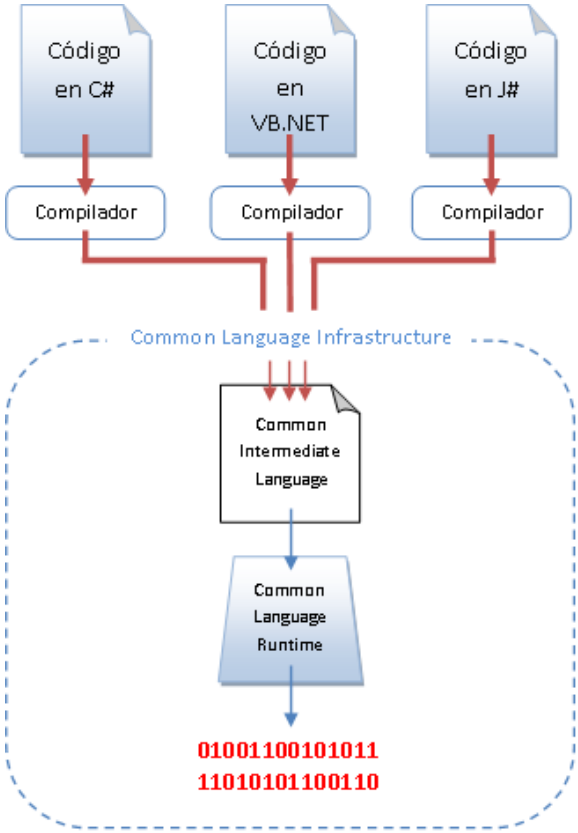


Figura 18. Vista general del Common Language Infrastructure (CLI).

Common Language Infrastructure (CLI): El propósito del *Common Language Infrastructure* (CLI), es proporcionar una plataforma independiente del lenguaje de programación para el desarrollo y ejecución de aplicaciones, incluyendo funciones para el manejo de excepciones, recolección de basura, seguridad e interoperabilidad. Mediante la implementación de los aspectos centrales de *.NET Framework* en el ámbito del CLR, esta funcionalidad no estará atada a un solo lenguaje, sino que estará disponible a través de muchos lenguajes soportados por el framework. La implementación de Microsoft del CLI se llama *Common Language Runtime* o CLR.

Ensamblados: Un *ensamblado* es una agrupación lógica de uno o más módulos o ficheros de recursos (ficheros .GIF, .HTML, etc.) que se engloban bajo un nombre común. Un programa puede acceder a información o código almacenado en un ensamblado sin tener porqué saber cuál es el fichero en concreto donde se encuentran, por lo que los ensamblados nos permiten abstraernos de la ubicación física del código que ejecutemos o de los recursos que usemos. Por ejemplo, podemos incluir todos los tipos de una aplicación en un mismo ensamblado pero colocando los más frecuentemente usados en un cierto módulo y los menos usados en otro, de modo que sólo se descarguen de Internet los últimos si es que se van a usar.

Todo ensamblado contiene un manifiesto, que son metadatos con información sobre las características del ensamblado. Este manifiesto puede almacenarse cualquiera de los módulos que formen el ensamblado o en uno específicamente creado para ello, caso éste último necesario cuando es un ensamblado satélite (sólo contiene recursos)

Metadatos: En la plataforma .NET se distinguen dos tipos de módulos de código compilado: ejecutables (extensión ***.exe***) y *librerías de enlace dinámico* (extensión **.dll** generalmente). Ambos son ficheros que contienen definiciones de tipos de datos, y la diferencia entre ellos es que sólo los primeros disponen de un método especial que sirve de punto de entrada a partir del que es posible ejecutar el código que contienen haciendo una llamada desde la línea de comandos del sistema operativo. A ambos tipos de módulos se les suele llamar *ejecutables portables* (PE), ya que su código puede ejecutarse en cualquiera de los diferentes sistemas operativos de la familia Windows para los que existe alguna versión del CLR.

Librería de Clase: .NET Framework incluye un conjunto estándar de librerías de clase. La librería de clase se organiza en una jerarquía de *Espacios de Nombres*. La mayoría de los construidos en las APIs son parte de los espacios de nombres `System.*` o `Microsoft.*`. Estas librerías de clase implementan un gran número de funciones comunes, tales como archivo de lectura y escritura, representación gráfica, interacción de bases de datos, y manipulación de documentos XML, entre

otros. La librería de Clase del .NET Framework se divide en dos partes: la Librería de Clase Base (Base Class Library) y la Librería de Clase Framework (Framework Class Library).

La Librería de Clase Base (BCL) es una librería formada por cientos de tipos de datos que permiten acceder a los servicios ofrecidos por el CLR y a las funcionalidades más frecuentemente usadas a la hora de escribir programas. Dado la amplitud de la BCL, ha sido necesario organizar las clases en ella incluida en *espacios de nombres* que agrupen clases con funcionalidades similares.

La Librería de Clase Framework (FCL) es una librería estándar y uno de los dos componentes principales de Microsoft .NET Framework. La FCL es una colección de miles de clases reutilizables dentro de cientos de espacios de nombres e interfaces. La librería de case base es una parte del FCL y proporciona la funcionalidad más fundamentales. La FCL es mucho mayor en alcance que las librerías estándar para lenguajes como C++, y comparable con el alcance de las librerías estándar de Java.

6.2 LENGUA DE SEÑAS

La lengua de señas es una lengua natural de expresión y configuración gesto-espacial y percepción visual, gracias a la cual las personas sordas pueden establecer un canal de comunicación en su entorno social, ya sea con otros individuos sordos o cualquier persona que conozca la lengua de señas empleada. A diferencia del lenguaje oral en el cual la comunicación se establece en un canal vocal-auditivo, en el cual sólo se puede emitir o recibir un sonido a la vez, el lenguaje de señas lo hace por un canal gesto-viso-espacial, porque explotan únicamente los disparos del medio visual, pudiéndose referir a un espacio entero al mismo tiempo, por ende la información puede fluir mediante varios "canales" y expresarse simultáneamente: otra diferencia con las orales es la dificultad de ser escrita, pues se trata de una lengua tradicionalmente ágrafa.

Las unidades simbólicas primarias o fonemas de la mayoría de lenguas de señas pueden analizarse en términos de siete parámetros formativos básicos:

- **Configuración**: forma que adquiere la mano al realizar una seña.
- **Orientación de la mano**: palma hacia arriba, hacia abajo, hacia el signante.
- **Lugar de articulación**: lugar del cuerpo donde se realiza la seña: boca, frente, pecho, hombro.
- **Movimiento**: movimiento de las manos al realizar una seña: giratorio, recto, vaivén, quebrado.
- **Punto de contacto**: Parte de la mano dominante que toca otra parte del cuerpo: yemas de los dedos, palma de la mano, dorso de los dedos.

- **Plano**: es donde se realiza el signo, según la distancia que lo separa del cuerpo, siendo el Plano 1 en contacto con el cuerpo, y el Plano 4 el lugar más alejado (los brazos estirados hacia delante).
- **Componente no manual:** Es la información que se transmite a través del cuerpo: Expresión facial, componentes hablados y componentes orales, movimientos del tronco y hombros.[13]

6.2.1 Las personas sordas: La definición de sordera difiere del enfoque con el cual se observa a la persona que tiene esta condición:

Desde el punto de vista Clínico Terapéutico, se concibe la sordera como una patología, describiéndola desde los grados de audición y la etiología (u origen de la enfermedad) que ocasionó la pérdida, planteando objetivos rehabilitadores con el apoyo de tecnología: audífonos, sistemas de amplificación FM, implante coclear, lo que ocasiona que la educación se dé en contextos clínicos, con currículos estructurados para oyentes sin ninguna adaptación y con unas exigencias pedagógicas mínimas en cuanto a desarrollo cognitivo se refiere.

Desde la concepción socio-antropológica se reconoce a las personas sordas como seres humanos que viven una diferencia desde el plano lingüístico, que gozan de todas sus potencialidades, que poseen una Lengua de Señas, producto construido histórica y socialmente por la Comunidad Sorda, y que si el entorno les brinda oportunidades respetando su condición, pueden alcanzar el desarrollo pleno de su personalidad aportando al fortalecimiento de la sociedad. La educación desde esta perspectiva se concibe como bilingüe y bicultural para los sordos e implica pensar en alternativas pedagógicas diferentes a las que hasta ese momento se había implementado como medio para la formación académica y humana de los sordos.

6.2.2 La Lengua de Señas Colombiana: La Lengua de Señas Colombiana es una modalidad no vocal del lenguaje humano, caracterizada por ser visocorporal, se expresa con el cuerpo en el espacio y se percibe a través de la vista, surgida naturalmente al interior de la Comunidad Sorda colombiana por interacción de sus integrantes, como respuesta a la necesidad de comunicación. Como cualquier lengua, permite el acceso a todas las funciones lingüísticas y cognitivas, posee dialectos y variables individuales y evoluciona constantemente al interior de la Comunidad Sorda; comparte universales lingüísticos con otras lenguas de señas, pero posee su propio vocabulario y sistema de reglas morfosintácticas y pragmáticas.

[13] WIKIPEDIA.Lengua de Señas. [En Línea] [Citado: Agosto de 2010]**.** Disponible en Internet: http://es.wikipedia.org/wiki/Lengua_de_se%C3%B1as

Según la Ley 982 de agosto 2 de 2005 se la define como “la lengua natural de una Comunidad de Sordos, la cual forma parte de su patrimonio cultural y es tan rica y compleja en gramática y vocabulario como cualquier lengua oral”.

La Lengua de Señas Colombiana es una lengua porque comparte las siguientes características de las lenguas orales:

- Se adquiere de forma natural cuando se está expuesto a ambientes ricos en estimulación.
- Permite el desarrollo social, cognitivo y la expresión de sentimientos.
- A través de ella se fortalece la transmisión de la cultura.
- Con ella se puede hacer referencia a hechos o lugares que suceden en el tiempo y en el espacio.
- Al igual que las palabras, no existe relación entre el significado de las señas y su forma.
- La lengua de señas responde a las necesidades sociales y académicas de los usuarios utilizando formas particulares para producir y comprender nuevas señas

La Lengua de Señas se adquiere porque se la asume como un proceso que se da en un ambiente lingüístico natural donde el niño percibe los usos que su lengua tiene y los va incorporando a su repertorio de lenguaje para ser empleados eficazmente en situaciones comunicativas que sean necesarias. Por el contrario se considera que una persona oyente aprende la Lengua de Señas, porque se entiende el aprendizaje como un acto consciente y deliberado de análisis de la gramática de la lengua en donde se incorporan sus normas y se pone en práctica su uso.

6.2.3 ¿Es la Lengua de Señas universal? De igual forma como en el mundo hay muchas lenguas orales, existen gran variedad de Lenguas de Señas y difieren de un país a otro, pues al surgir de una necesidad de comunicación entre las personas sordas de cualquier lugar del mundo, están sujetas a todos los procesos y transformaciones que las lenguas orales sufren en sus etapas de desarrollo, siendo afectadas por rasgos socioculturales, filosóficos, topográficos y formas de vida de cada lugar.

Por lo tanto se puede decir que las Lenguas de Señas no son prefabricadas, entonces no se puede hablar de una Lengua de Señas universal para la Comunidad sorda.

En Colombia, con la Ley 324 de 1996, se reconoció a la Lengua de Señas como idioma de natural adquisición por parte de la Comunidad Sorda. Ello ha permitido a las personas sordas que el legado lingüístico que sus antecesores le dejaron, se posicione dentro de la sociedad y reclame los derechos que como lengua tiene.

6.2.4 Educación para sordos. Con el fin de enseñar la lengua oral a los niños sordos, se han creado a lo largo de la historia diferentes métodos de enseñanza, tales como los canales sensoriales que se estimulan para la recepción y la expresión del habla; la aceptación o no del empleo de signos manuales; los recursos tecnológicos utilizados; el enfoque que se ha adoptado para la enseñanza del lenguaje, etc.

Estas prácticas para la enseñanza formal de la lengua oral se hacen necesarias con los niños sordos, debido a que la pérdida auditiva interrumpe el canal de acceso de la información lingüística e interrumpe de manera determinante el proceso de adquisición natural del lenguaje como facultad humana, proceso que se basa en la plena exposición y participación en interacciones comunicativas significativas y fluidas con usuarios competentes de una lengua en particular.[14]

7. DESARROLLO DEL PROYECTO

Habiendo realizado una revisión de las temáticas que involucra el reconocimiento de voz, el .NET Framework y la lengua de señas, como ejes fundamentales del proyecto, el presente capítulo aborda el desarrollo de los objetivos específicos, que dan lugar a una descripción detallada de las pruebas experimentales realizadas para obtener el reconocedor, y que permiten alcanzar el objetivo general del trabajo.

La metodología aplicada para el desarrollo del proyecto, así como el proceso de selección de las palabras que el sistema es capaz de reconocer, consta de tres etapas. La primera corresponde a la adquisición de las palabras que el sistema será capaz de reconocer; la selección de las palabras se realizó teniendo en cuenta el diccionario básico de la lengua de señas Colombiana y se opto como base de datos las palabras más comunes dentro de de la comunidad sorda. La segunda etapa, consiste en la implementación de un modulo para el reconocimiento de los parámetros adquiridos utilizando diferentes algoritmos, de los cuales se escogió el mas optimo y el que asegura la mayor tasa de reconocimiento. Y la tercera etapa, consiste en el desarrollo de la interfaz grafica que permite visualizar las señas correspondientes a las palabras reconocidas por el sistema.

[14] COLOMBIA APRENDE. Lengua de señas colombiana. [En Línea] [Citado: Julio de 2010]. Disponible en Internet: http://mail.colombiaaprende.edu.co:8080/recursos/lengua_senas/

Todo el proceso desarrollado hace uso de varias rutinas en el programa MATLAB, que permiten almacenar, representar y manipular las señales, con las cuales se realizaron los entrenamientos y las pruebas respectivas del sistema.

7.1 DISEÑO DE UN MÓDULO DE ADQUISICIÓN Y PROCESAMIENTO DE SEÑALES DE VOZ PARA EL RECONOCIMIENTO DE FONEMAS

En esta etapa del proyecto se desarrollaron varios algoritmos que permiten: adquirir información de audio, por medio de la tarjeta de sonido del computador; analizar y pre-procesar las señales adquiridas, estableciendo la base de datos del sistema; y procesar la información adquirida, mediante diferentes métodos de extracción de características.

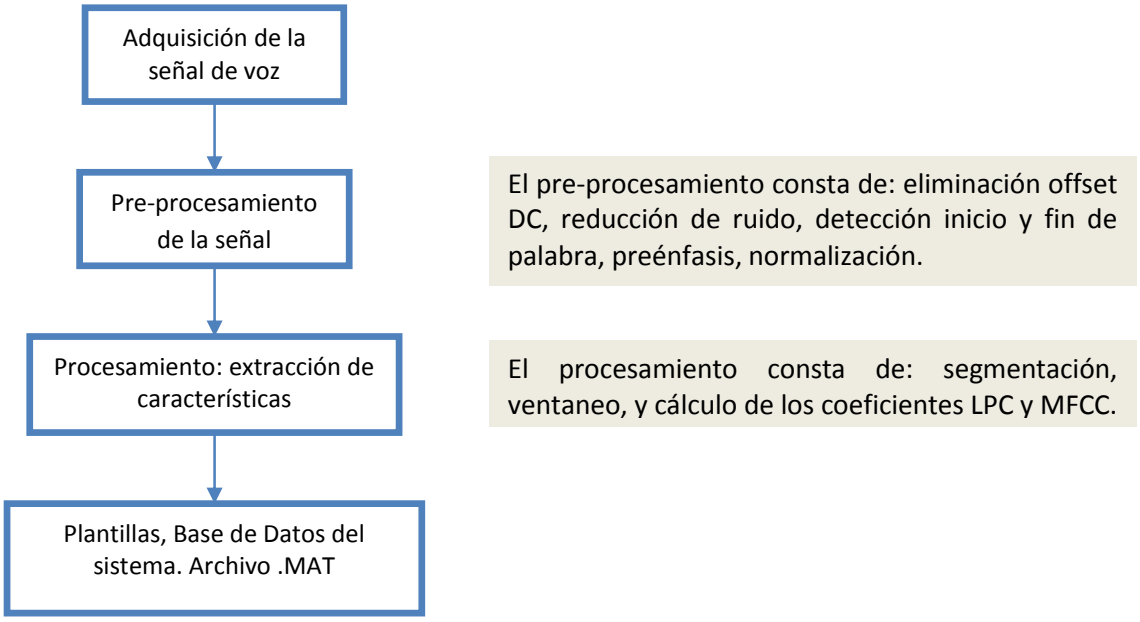


Figura 19. Diagrama de flujo correspondiente al modulo de adquisición y procesamiento de la señal de voz.

7.1.1 Adquisición de la señal de voz. MATLAB posee librerías y comandos necesarios para trabajar con señales de audio, lo que facilita la configuración de la tarjeta de sonido del computador para la adquisición de la señal de voz. La configuración permite establecer la frecuencia de muestreo, la cuantificación de los datos (número de bits) y el tipo de canal (mono o estéreo), que influyen directamente sobre la calidad del sonido. Para el desarrollo del proyecto se estableció una frecuencia de muestreo (Fs) de 16KHz, una cuantificación de 16 bits y un canal mono-estéreo.

Una vez configurada la tarjeta de sonido, se crea la base de datos (diccionario de referencia) del sistema. La base de datos se obtiene realizando varias capturas de

cada una de las palabras escogidas. Las palabras, que se encuentran dentro del diccionario básico de la lengua de señas colombiana, son pronunciadas 20 veces por un hablante, pues inicialmente se pretendía desarrollar un sistema dependiente de locutor. Las 20 palabras escogidas se muestran en la tabla 1.

No de palabra	Palabra
1	Cero
2	Uno
3	Dos
4	Tres
5	Cuatro
6	Cinco
7	Seis
8	Siete
9	Ocho
10	Nueve
11	Cantar
12	Vaca
13	Toro
14	Día
15	Bien
16	Odiar
17	Noche
18	Nube
19	Coco
20	Toser

Tabla 1. Base de palabras del sistema.

Parte del código desarrollado en MATLAB para la adquisición de las señales de voz se observa en la figura 21 y uno de los resultados obtenidos con el código en la figura 20.

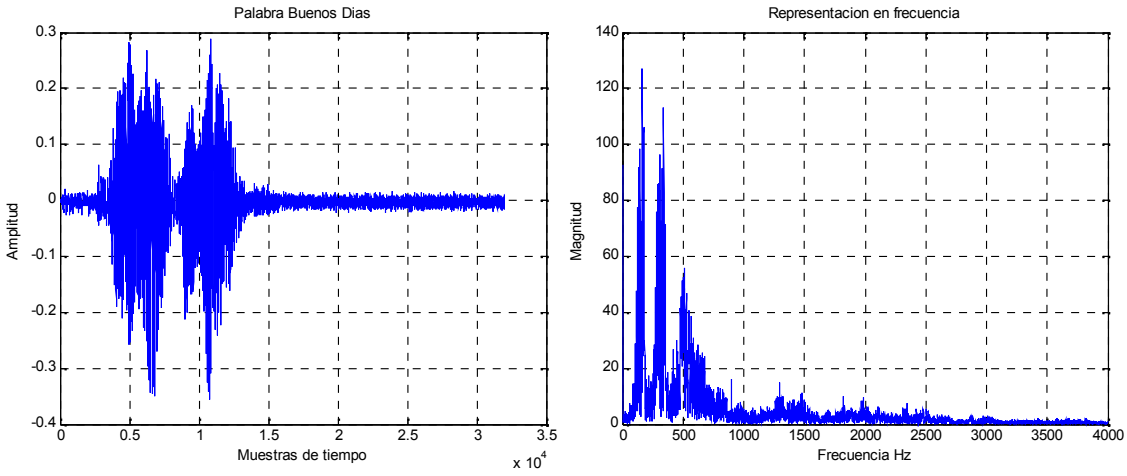


Figura 20. Representación de la palabra *Buenos Días* en el dominio del tiempo y la frecuencia.

```
%Configuración de la tarjeta de sonido para la adquisición de audio
AI = analoginput('winsound');
chan=addchannel(AI,1);          %Canal mono-stereo
duration=2;                     %Tiempo duración de grabación
set(AI,'samplerate',16000)      %Fs=16000Hz
actualrate=get(AI,'samplerate');
Fs=actualrate;
set(AI,'SamplesPerTrigger',duration*actualrate);

%--------------Adquisición de datos y base de datos--------------%
n_pal=20;                       %número palabras a pronunciar
n_rep=5;                        %número de repeticiones
words=cell(1,n_pal);            %Matriz de palabras

%--------------------------Grabar datos--------------------------%
for s=1:n_pal
    ...
    <MATRIZ DE PALABRAS SEGÚN SE VAN ADQUIRIENDO>
    ...
    for n=1:n_rep
        start(AI);      %Inicio de adquisición
      ...
  <PROCESADO DE LOS DATOS>
      ...
   %Guardar archivo
        archivo=['D:\pruebas_tesis\Base0\grupo' ...
            num2str(n) '\5_' num2str(s)];
        wavwrite(data,Fs,archivo)
    end
end
stop(AI); delete(AI); clear AI
```

Figura 21. Código en MATLAB para la adquisición de señales utilizando la tarjeta de sonido.

7.1.2 Pre-procesamiento de la señal de voz. El objetivo del pre-procesamiento es acondicionar la señal de voz adquirida conservando la mayor cantidad de sus características. Este proceso se realiza para que la información pueda ser procesada adecuadamente por los algoritmos de reconocimiento y no presenten problemas de inestabilidad en las etapas de entrenamiento, aprendizaje y reconocimiento. La figura 22 muestra los pasos necesarios para el pre-procesamiento de la señal de voz.

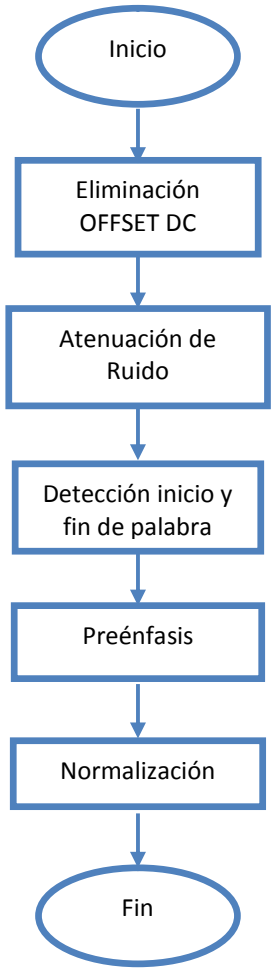


Figura 22. Diagrama del pre-procesamiento de la señal de voz.

El pre-procesamiento es aplicado tanto a la base de datos, como a cualquier otra señal que ingrese al sistema. Cada uno de los pasos necesarios para el pre-procesamiento de la señal de voz se describe a continuación.

Eliminación del OFFSET DC: Inicialmente, se elimina el factor DC de la señal, ya que es un factor indeseable en el procesamiento y análisis de la señal de voz. La eliminación se obtiene restando la amplitud media de la señal a la señal original, la ecuación utilizada para realizar este proceso es:

$$x(n) = x(n) - mean(x(n)) \qquad (52)$$

Atenuación de ruido: Luego, se reducen las componentes de frecuencia indeseadas en la señal de voz. El método utilizado para la atenuación de ruido es el método de Sustracción Espectral, que se basa en lo siguiente:

- El ruido contiene energía en todo el espectro de frecuencias, por lo cual es aleatorio en el dominio del tiempo (ruido de banda ancha).
- El espectro de las señales de audio está compuesto de elementos que corresponden a un tono fundamental y a los parciales de las notas que se van interpretando.
- La señal de audio original $s(n)$ está contaminada por un ruido aditivo $v(n)$ no correlacionado con $s(n)$, obteniéndose una señal degradada $x(n)$.
- El ruido se considera estacionario en los bloques en que se subdivide la señal para su tratamiento, por tanto es posible estimar su densidad espectral de potencia o cantidades directamente relacionadas con ella.

El algoritmo para la reducción del ruido de fondo, que se describe esquemáticamente en la figura 23, se puede resumir en las siguientes etapas:

- Estimación del espectro de la señal mediante una Transformada de Fourier.
- Estimación del espectro del ruido.
- Sustracción espectral.
- Reducción del ruido residual.
- Supresión del ruido en segmentos de inactividad de la señal subyacente.
- Síntesis de la señal limpia.

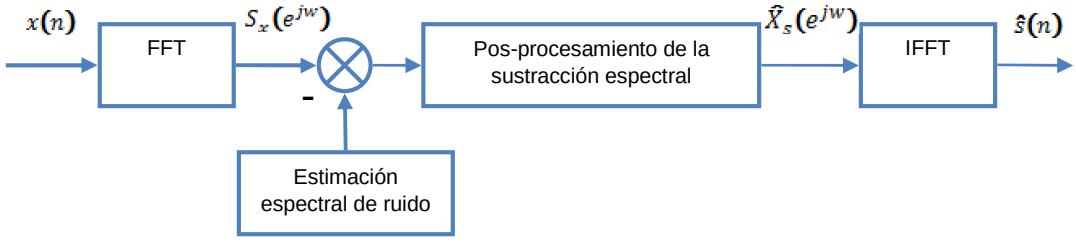


Figura 23. Diagrama esquemático de la Sustracción Espectral.

El modelo aquí adoptado es:

$$x(n) = s(n) + v(n) \qquad (53)$$

Donde $s(n)$ es la señal de voz limpia, $v(n)$ el ruido contaminante y $x(n)$ la señal de voz ruidosa. Como las señales son localmente estacionarias, el proceso de

análisis se realiza de forma localizada, segmentando la señal en tramas o bloques de 25ms y multiplicando cada una de ellas por una ventana Hamming $w(m)$, de modo que la señal de voz ruidosa localizada sea de la forma:

$$x(n;m) = x(n)w(m-n) \qquad (54)$$

Donde m indica el número de la trama de análisis. En consecuencia, la señal ruidosa en términos de análisis localizado será:

$$x(n;m) = s(n;m) + v(n;m) \qquad (55)$$

y su función de autocorrelación localizada estará dada por:

$$r_x(k;m) = r_s(k;m) + r_v(k;m) \qquad (56)$$

En términos de las densidades espectrales de potencia, la densidad espectral de potencia de la señal de voz ruidosa está dada por:

$$S_x(e^{jw};m) = S_s(e^{jw};m) + S_v(e^{jw};m) \qquad (57)$$

Como las densidades espectrales de potencia no se conocen, se deben estimar por algún método de estimación espectral.

La sustracción espectral busca obtener la estimación de la densidad espectral de potencia de la señal limpia a partir de la densidad espectral de potencia de la señal ruidosa y una estimación de la densidad espectral de potencia del ruido, estimado en intervalos de silencio:

$$\hat{S}_s(e^{jw};m) = \hat{S}_x(e^{jw};m) - \hat{S}_v(e^{jw};m) \qquad (58)$$

Donde $\hat{S}_x(e^{jw};m)$ es la estimación de la densidad espectral de potencia de la señal de voz ruidosa en la trama m -ésima, $\hat{S}_s(e^{jw};m)$ es la estimación de la densidad espectral de potencia de la señal limpia en la trama m -ésima y $\hat{S}_v(e^{jw};m)$ es la estimación de la densidad espectral de potencia del ruido.

Este proceso permite estimar la magnitud del espectro de la señal de voz limpia para la trama m -ésima. Sin embargo falta estimar la fase para poder reconstruir la señal. Afortunadamente, el oído no es especialmente sensible a la fase de la señal por lo cual se puede utilizar la fase del espectro de la señal ruidosa como estimación de la fase para reconstruir la señal limpia, es decir:

$$\hat{\varphi}_s(e^{jw};m) = \hat{\varphi}_x(e^{jw};m) \qquad (59)$$

Finalmente, la estimación de la señal de voz limpia para la trama m -ésima se obtiene a partir de la transformada de Fourier inversa del espectro estimado de la señal limpia:

$$\hat{X}_s(e^{jw};m) = \left[\hat{S}_x(e^{jw};m) - \hat{S}_v(e^{jw};m)\right]^{\frac{1}{2}} e^{j\hat{\varphi}_s(e^{jw};m)} \qquad (60)$$

De manera que:

$$\hat{s}(n;m) = IFFT\left(\hat{X}_x(e^{jw};m)\right) \qquad (61)$$

y aplicando la técnica del solapamiento aditivo se reconstruye la señal $\hat{s}(n)$. Para el presente proyecto ser trabajo con un solapamiento del 40% de la trama. En la figura 25 se muestra parte del código utilizado para la atenuación del ruido y en la figura 24 los resultados obtenidos al aplicar el método de atenuación espectral a la señal de voz.

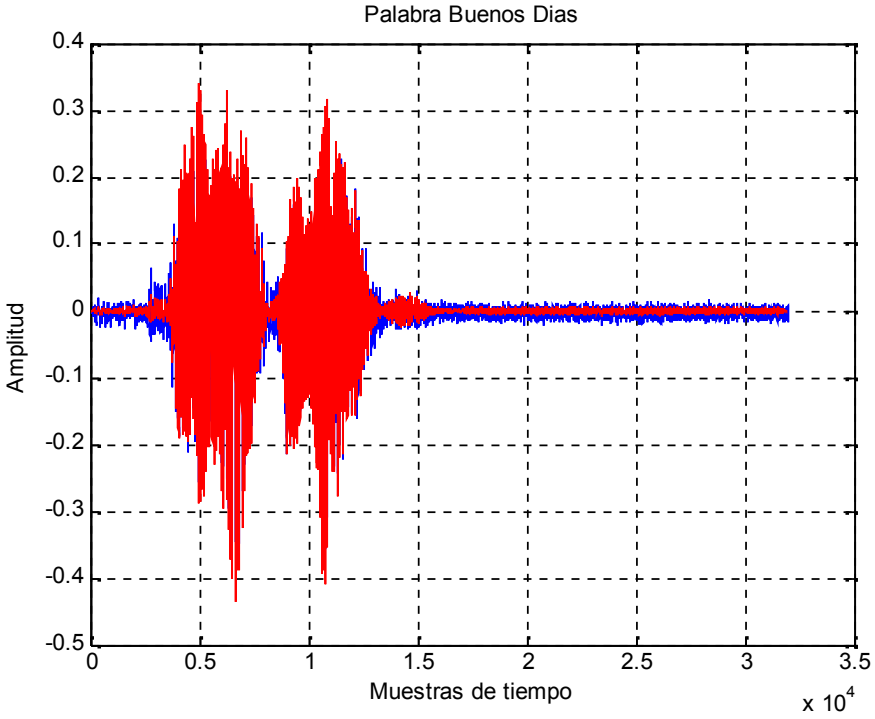


Figura 24. Palabra *Buenos Días* con ruido espectral y OFFSET DC (color azul).Palabra *Buenos Días* sin ruido espectral y sin OFFSET DC (color rojo).

En la figura 24, se indica en color azul la señal original y en rojo la señal con reducción de ruido aplicando la sustracción espectral.

```
%<Definicion de Variables de Segmentacion de la señal>
...
t_ini=0.150;    %Tiempo de suposicion de silencio
t_seg=0.025;    %Duracion de la trama de 25ms
...
win=hamming(m_seg); %Ventana Hamming
...
<Segmentacion de la señal de voz>
...
%-------------Transformada de Fourier de cada segmento-------------%
X=fft(M_seg,m_seg,2);
mitad=fix(m_seg/2)+1;
Alfa=1
Xmodulo=(abs(X(:,1:mitad))).^Alfa;
Xfase=angle(X(:,1:mitad));
%-----------------------Estimación de Ruido----------------------%
M_ruido=Xmodulo(1:N_sil,:);
Rprom=mean(M_ruido);
MRR=zeros(size(Rprom));
%------------------------Sustracción Espectral--------------------%
ContRuido=0;    %Contador para ruido
C=9;            %Factor de suavizado para el ruido
Beta=0.3;
Y=zeros(size(Xmodulo));
for k=1:N_seg
    [Ruido, Voz, ContRuido, Dist]=VAD(Xmodulo(k,:).^(1/Alfa),...
        Rprom.^(1/Alfa),ContRuido);
    if Voz==0
        Rprom=(C*Rprom+Xmodulo(k,:))/(C+1);
        MRR=max(MRR,Xprom(k,:)-Rprom);
        Y(k,:)=Beta*Xmodulo(k,:);
    else
        D=Xprom(k,:)-Rprom;
        if k>1 && k<N_seg
            for j=1:length(D)
                if D(j)<MRR(j)
                    D(j)=min([D(j) Xprom(k-1,j)-Rprom(j)...
                        Xprom(k+1,j)-Rprom(j)]);
                end
            end
        end
        Y(k,:)=max(D,0);
    end
end
%----------------------solapamiento aditivo-----------------------%
%Señal con reduccion espectral de ruido en el dominio del timepo
Salida=OverlapAdd(Y.^(1/Alfa),Xfase,m_seg,m_sol);
```

Figura 25. Código en MATLAB para la reducción de ruido en la señal de voz.

Detección Inicio-Fin Palabra: Una vez eliminado el ruido de la señal de voz, se realiza la detección del inicio y el fin de la palabra con el fin de eliminar información redundante de la señal de voz, es decir, eliminar muestras de tiempo

que contienen silencios, pues estas zonas de la señal no contienen ninguna información para el reconocimiento.

Este procedimiento se llevo a cabo mediante el uso de dos métodos, el método por evaluación de energía y el método por evaluación de cruces por cero, que se utilizan como medidas complementarias para la detección de actividad vocal.

El *Método por evaluación de energía*, permite evaluar las zonas donde se encuentran segmentos sordos y sonoros de la señal de voz, ya que los valores de amplitud de la señal de voz varían a lo largo del tiempo. En este método se divide la señal en varios segmentos, calculando la energía para cada uno de ellos, y se calcula la energía promedio de toda la señal. Luego se define un valor umbral, proporcional a la energía promedio máxima de la señal, entonces cada segmento cuya energía promedio sea menor que el valor umbral se descarta eliminando así los silencios en la señal de voz.

Para el desarrollo de este proyecto se utilizaron segmentos de 5 ms y un valor de umbral de 20% de la energía máxima. Este valor se obtiene de forma experimental pues se observa que un aumento o disminución en el porcentaje de la energía ocasiona pérdida de segmentos sonoros o adicción de segmentos sordos, respectivamente. Este problema enmarcado en el sistema de reconocimiento de voz significa mayor tiempo de cómputo y problemas en la identificación de las palabras. Las ecuaciones (62) y (63) se utilizaron para la energía promedio de la señal de voz y la energía promedio de cada segmento respectivamente.

$$E_{avg} = \frac{1}{N}\sum_{k=1}^{N}|x(k)|^2 \qquad (62)$$

$$E_n = \sum_{k=1}^{w_n}|x(k)|^2\, w(n-k) \qquad (63)$$

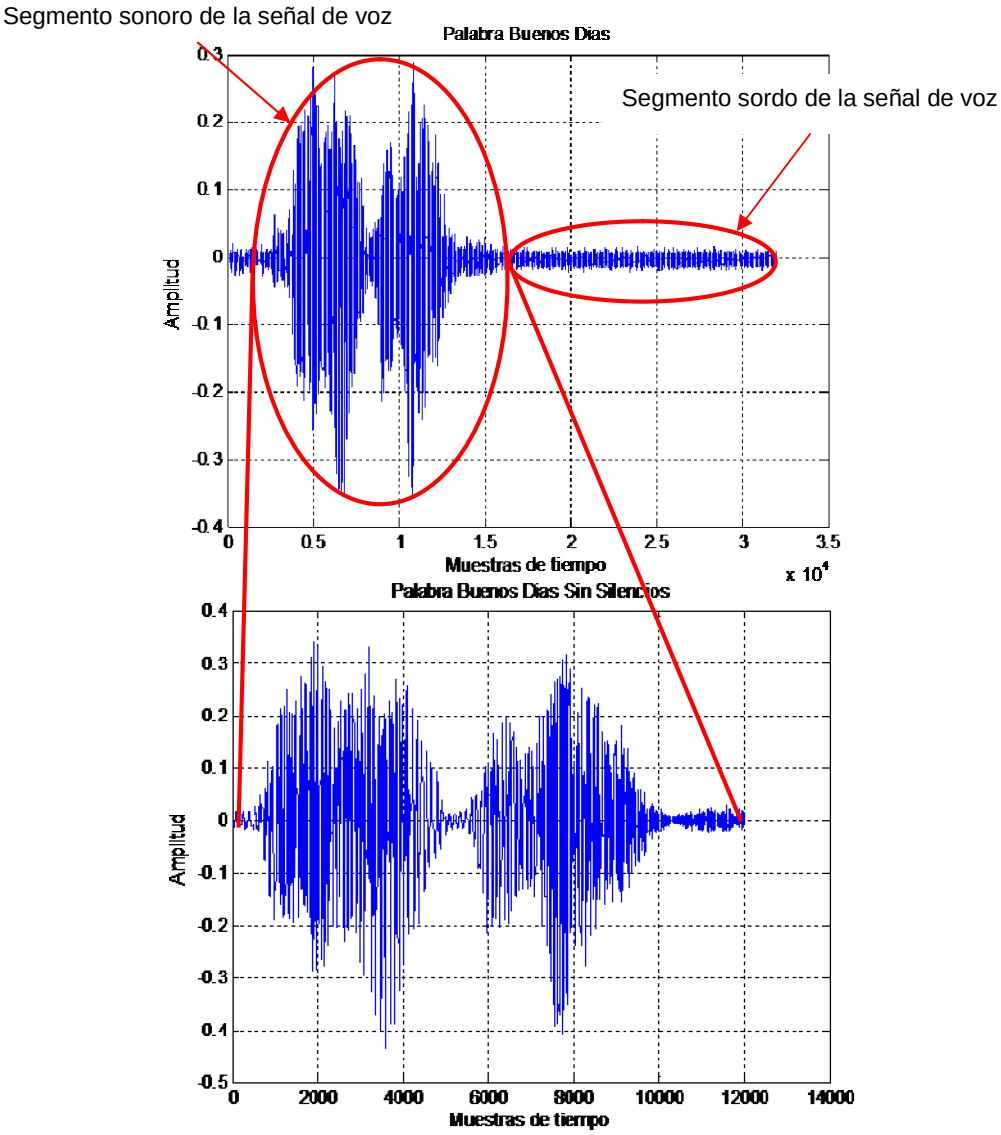


Figura 26. Eliminación de los segmentos sordos de la señal de voz.

El *método por evaluación de cruces por cero,* evalúa el número de veces que la señal de voz toma el valor de cero. Para la señal de voz digital un cruce por cero ocurre cuando dos muestras consecutivas difieren de signo o una muestra toma el valor de cero. Dependiendo de los diferentes sonidos que conforman las palabras pronunciadas se tiene que en los segmentos donde el sonido es “sordo” existe un predominio de las altas frecuencias, por lo cual se presenta un valor mayor de cruces por cero y el valor energético del segmento disminuye; mientras que en los segmentos donde el sonido es sonoro, el valor de cruces por cero disminuye y su valor energético se incrementa.

La densidad de cruces por cero para señales digitales obedece a la siguiente expresión:

$$Z_z(m) = {}^{1}/_{N} \sum_{n=m-N+1}^{m} \frac{|sgn\{s(n)\} - sgn\{s(n-1)\}|}{2} . w(n-m) \qquad (64)$$

Donde la función signo (sgn) toma los valores:

$$sgn\{s(n)\} = \begin{cases} +1, & s(n) \geq 0 \\ -1, & s(n) < 0 \end{cases}$$

A continuación se muestra parte del código utilizado para la detección de inicio y fin de palabra, utilizando el método de energía y el método de evaluación de cruces por cero y los resultados al aplicar dicho código a la señal de voz.

```
%----------------------------Variables--------------------------%
t_ini=0.1;                  %Duracion inicial - silencio 100ms
t_seg=0.01;                 %Ancho del segmento o trama 10ms
t_sol=0.005;                %Solapamiento entre trama 5ms (50%)
t_ava=t_seg-t_sol;          %Avance entre tramas 5ms
...
    <DEFINICION DE VARIABLES>
...
win=hamming(m_seg);         %Ventana Hamming

%Calculo de Energía Por segmento
...
    <Segmentacion de señal y Calculo de energía de cada segmento>
...
%----------------- Tasa de Cruces por Cero (ZCR) ----------------%
v_ZCR=zeros(1,seg_v);       %Vector de ZCR.
for t=1:seg_v
    indice=(t-1)*m_sol+1;
...
    <Numero de cruces por cero para cada trama>
...
end
umbral=40;                              %ZCR para comparacion de umbral
ZCR_prom=mean(v_ZCR(1:seg_s));          %Promedio de ZCR para silencio
des_est=std(v_ZCR(1:seg_s));            %Desviacion estandar para ZCR
umZCR=min(umbral,ZCR_prom+2*des_est);%Umbral ZCR
busquedaN=min(inicio,25);

%deteccion de punto inicial y final.
...
    busquedaF=min(length(v_energia)-final,25);
```

Figura 27. Código en MATLAB para el recorte de la señal de voz con el método de cruces por cero.

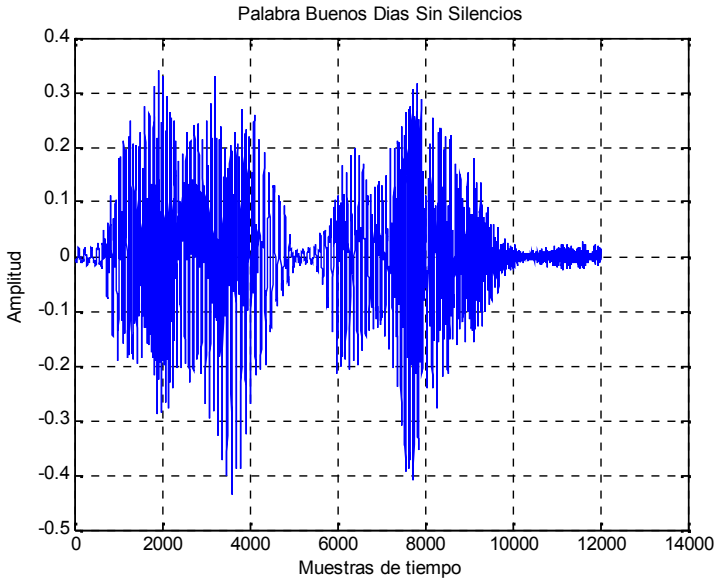


Figura 28. Detección de Inicio y fin de la palabra *Buenos Días*.

Preénfasis: Inmediatamente después, mediante el preénfasis se acentúa las altas frecuencias de la señal de voz, haciéndola menos susceptible a interpretaciones erróneas en posteriores análisis. El preénfasis obedece a la ecuación:

$$H(z) = 1 - az^{-1} \quad \therefore \quad 0.9 \leq a \leq 1 \qquad (65)$$

Para efectos de este proyecto de utilizo $a = 0.97$. En la siguiente figura se muestra el efecto del preénfasis sobre la señal de voz.

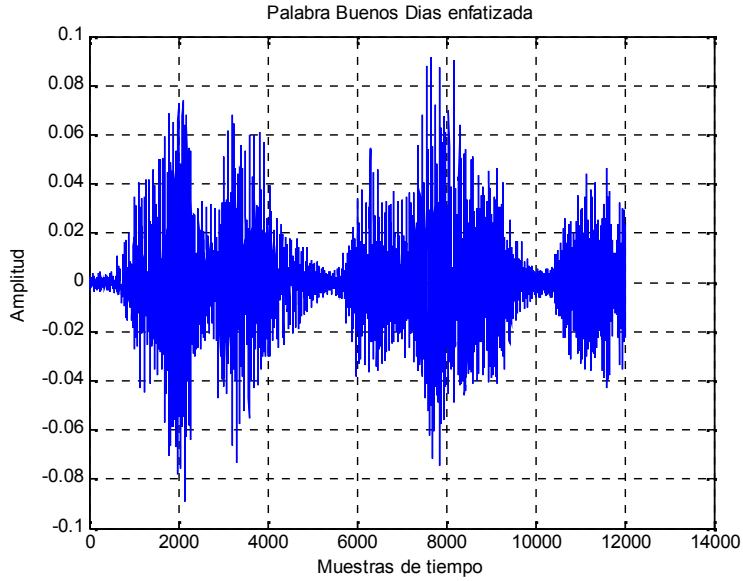


Figura 29. Preénfasis de la Palabra *Buenos Días.*

Normalización: El pre-procesamiento de la señal de voz concluye con la normalización, que ajusta todos los valores a una sola escala. Aquí se referencia todos los valores de la señal al máximo valor de la amplitud de la misma, para que al momento de ser utilizados por los algoritmos de reconocimiento no causen problemas de estabilidad.

En la figura 30 se muestra el código utilizado para este fin y en la figura 31 el resultado obtenido. Este resultado pasa a una etapa de procesamiento, en donde se extraen las características de la señal de voz.

```
%-------------------------Normalización-------------------------%
m=length(data);              %Longitud del vector
data=data./max(abs(data));   %Vector normalizado
X =fft(data,m);              %Espectro de la señal normalizada
mag_X=(abs(X)).^2;           %Magnitud del espectro al cuadrado
f=(0:m/4)*Fs/m;              %vector de frecuencias hasta 4KHz
```

Figura 30. Código en MATLAB para la normalización de la señal de voz.

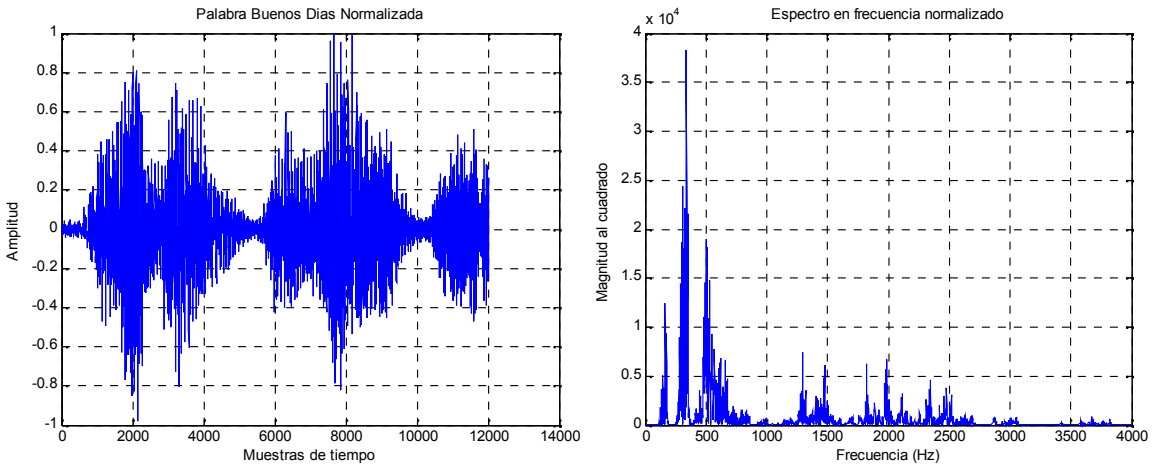


Figura 31. Amplitud y espectro normalizado de la Palabra *Buenos Días.*

Con el fin de demostrar la unicidad de la transformada de Fourier de la señal mostrada en la figura 31, se aplica el teorema de la energía de Rayleigh o teorema de Parseval, el cual, para el caso de señales en tiempo discreto, se define matemáticamente como:

$$\sum_{n=0}^{N-1} |x[n]|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X[k]|^2 \qquad (66)$$

El teorema plantea, que la energía total contenida en una forma de onda $x[n]$ sumada a través de todo el tiempo n, es igual a la energía total de la forma de

onda de su transformada de Fourier $X[k]$ sumada a través de todas sus componentes frecuenciales f .

Los resultados obtenidos, mediante la ecuación 55 (cuyo código se muestra en la figura 32), son de $2.82336 * 10^2$ para la señal en el dominio del tiempo y de $2.82336 * 10^2$ para la señal en el dominio de la frecuencia, demostrándose así el teorema.

```
%-------------------Teorema de Energía de Rayleigh-----------------%
% Energía de la señal en el dominio del tiempo
xn=(abs(data)).^2;
E_xn=sum(xn);

% Energía de la señal en el dominio de la frecuencia
m=length(data);
Xk=fft(data,m);
Xk=(abs(Xk)).^2;
E_Xk=sum(Xk)/m;
```

Figura 32. Código en MATLAB para el Teorema de Energia de Rayleigh.

7.1.3 Extracción de características de la señal de voz. Una vez la señal de voz es pre-procesada, en esta etapa se obtienen las características de la señal de voz. Para el desarrollo del presente proyecto se seleccionaron los Coeficientes de Predicción Lineal (LPC) y los Coeficientes Cepstrales de Frecuencia de Mel (MFCC), como algoritmos para extraer las características de la señal digitalizada de voz, pues ambos destacan las características que componen la voz humana y facilitan el trabajo que debe realizar cualquier algoritmo de reconocimiento.

El procesamiento, inicia con la segmentación y ventaneo de la señal de voz y luego pasa al algoritmo de extracción de características, que puede estar basado en LPC o MFCC. Estos procesos son descritos detalladamente a continuación.

Segmentación y ventaneo. La señal de voz es considerada casi estacionaria en intervalos cortos de tiempo (de 10 a 40ms) razón por la cual es segmentada en tramas de 30ms con una superposición del 50% del tamaño de la trama, con el fin de conservar la mayor cantidad de información contenida en la señal[15]. Luego, cada trama es multiplicada por una ventana Hamming (figura 34.a), del mismo tamaño, para mantener la continuidad de los primeros y los últimos puntos, y para evitar cortes abruptos que ocasionen componentes de alta frecuencia indeseados.

[15] El porcentaje de superposición puede ser inferior (de 20% a 49%), dependiendo de la aplicación y de la capacidad de procesamiento del sistema, sin embargo, una superposición menor ocasiona pérdida de información que afectaría de alguna manera la etapa de reconocimiento.

En las figuras 33 y 34 se muestra parte del código utilizado para la segmentación y ventaneo de la señal de voz y el resultado obtenido en una trama después de aplicado el código.

```
%----------------------------Variables--------------------------%
long_y=length(data);        %Longitud del vector
tseg=0.03;                  %Tiempo del segmento
N=fix(Fs*tseg)+1;           %Número de muestras en el segmento
tavance=0.015;              %Tiempo traslape de trama
L=fix(Fs*tavance)+1;        %Número de muestras en el segmento
segm=floor(long_y/L);       %Numero de segmentos redondeado
...
    <Redondear vector de datos>
...
mat_segm=zeros(N,segm);     %Matriz de segmentos
vent=hamming(N);            %Ventana Hamming de 30 ms
%--------------------Segmentación y Ventaneo--------------------%
for i=1:segm
    avance=i*L-(L-1);
    % y: vector redondeado
    mat_segm(1:N,i)=y(avance:avance+(N-1)).*vent;
end
```

Figura 33. Código en MATLAB para la segmentación y ventaneo de la señal de voz pre-procesada.

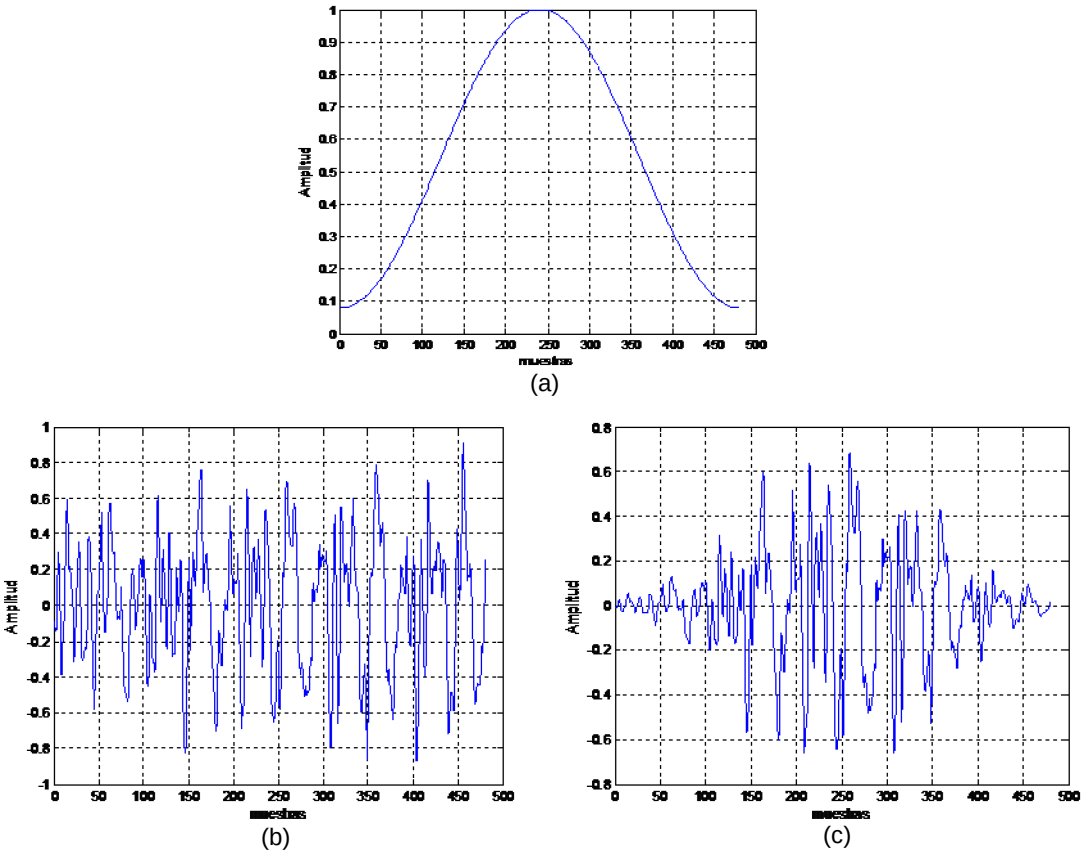


Figura 34. La figura muestra (a) la ventana Hamming, y la trama de la señal de voz de 30ms (b) antes y (c) después de la ventana Hamming.

Calculo coeficientes de predicción lineal (LPC). Los LPC son utilizados para la representación de la envolvente espectral de una señal digital de voz en forma comprimida, usando la información de un modelo de predicción lineal. Esta técnica es una de las más poderosas para el análisis de voz, pues permite codificar la señal de voz con una buena calidad y una baja tasa de bits, lo que proporciona estimaciones muy precisas de los parámetros de la señal.

Para el desarrollo del proyecto se implemento una rutina en Matlab, por medio de la cual es posible realizar el cálculo de los coeficientes de predicción lineal LPC para cada una de las palabras adquiridas. El número de coeficientes LPC utilizados fueron 12 para cada trama y las ecuaciones para su cálculo se describen en la sección 4.5.

En la figura 35, se muestra la distribución de las magnitudes de cada uno de los coeficientes de predicción lineal (LPC) obtenidos para cada segmento de la señal de voz. El eje vertical representa la magnitud de cada coeficiente, el eje horizontal representa cada uno de los coeficientes LPC y las líneas representan los LPC correspondientes a cada segmento de la señal de voz, en este caso el fonema a.

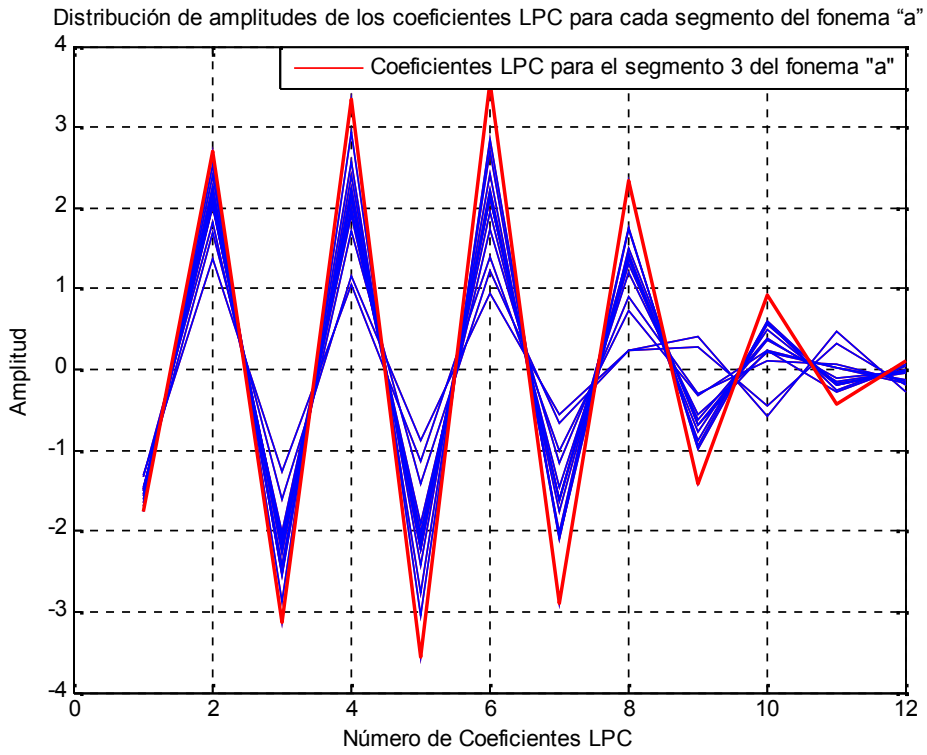


Figura 35. Distribución de amplitudes de los coeficientes LPC para cada segmento del fonema “*a”.*

Cálculo coeficientes Cepstrales en la escala de Mel (MFCC). Los MFCC, como se menciona en la sección 4.4, son coeficientes para la representación del habla basados en la percepción auditiva humana. Esta técnica modela la respuesta auditiva humana, haciendo uso de la transformada de Fourier y la Transformada del coseno discreta. Aquí las bandas de frecuencia están situadas logarítmicamente según la escala de Mel, lo que permite un procesado de datos más eficiente que si estuvieran situadas linealmente.

Para el proyecto se realizo un algoritmo en Matlab con el cual es posible realizar el cálculo de los coeficientes Cepstrales en la escala de Mel para la señal de voz digitalizada. El número de coeficientes MFCC utilizados varían de 13 a 36, pues como vio en la sección 4.4 se puede obtener Delta-MFCC y Delta-Delta-MFCC.

```
M_coef=mfcc(num,s,Fs)
Ncoef=numero de coeficientes;
palabraIn=palabra de entrada;
Fs=frecuencia de muestreo
N_fil=numero de filtros Mel
...
    <Definición de variables>
...

%Segmentación de la señal y cálculo de los MFCC

for i=1:No_seg-2
    segmento=s((i-1)*p_fr+1:(i-1)*p_fr+N);
    ...
        <Cálculo de la energía de cada segmento>
        <Definición del banco de filtros Mel para los segmentos>
        <Filtrado en escala de Mel>
        <Cálculo de la DCT para cambio al dominio Ceptral>
        <Obtención de los coeficientes>
    ...
    M_coef(i,:)=No_coef';
end
```

Figura 36. Código en MATLAB para el cálculo de los MFCC.

En la figura 37, se muestra los 13 MFCC obtenidos para cada segmento de la señal de voz, y al igual que en la figura 35, el eje vertical representa la magnitud de cada coeficiente, el eje horizontal representa cada uno de los coeficientes 13 MFCC y las líneas representan los MFCC correspondientes a cada segmento de la señal de voz, en este caso el fonema a.

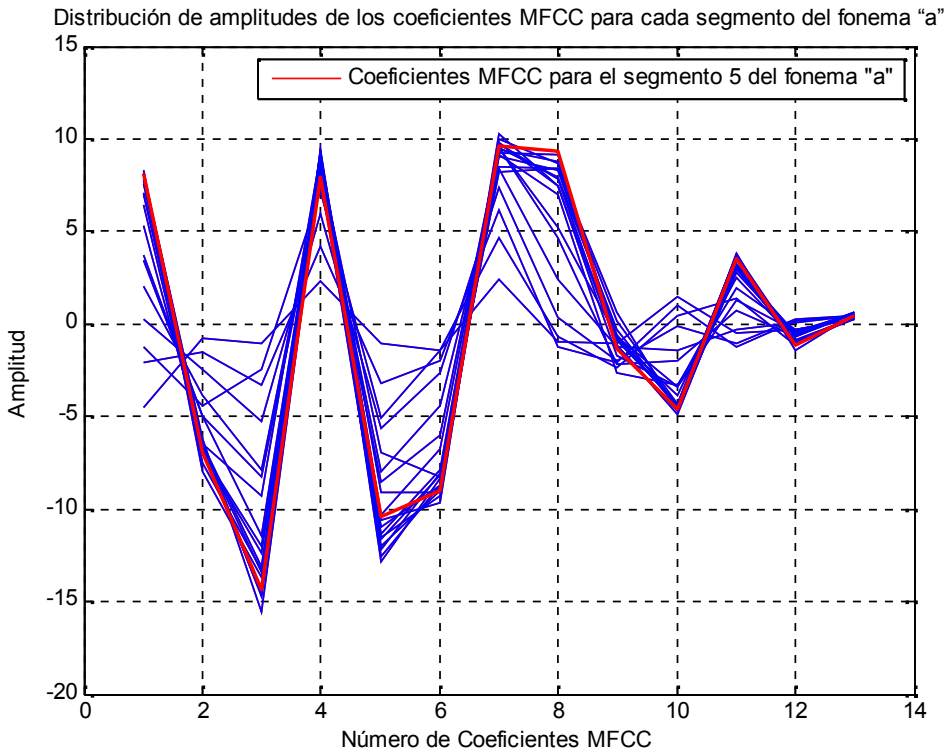


Figura 37. Distribución de amplitudes de los MFCC para cada segmento del fonema “*a”.*

7.2 IMPLEMENTACIÓN DE ALGORITMOS PARA EL RECONOCIMIENTO

En la segunda etapa general del sistema se implementa un algoritmo de reconocimiento. Al finalizar esta etapa, el sistema debe ser capaz de discriminar entre los diferentes sonidos de entrada y realizar el reconocimiento de cada uno de estos con respecto a la base de datos previamente establecida.

Para realizar el algoritmo de reconocimiento, se usaron diferentes técnicas que permitieron realizar un pequeño estudio comparativo y que dieron lugar al sistema de reconocimiento que se implemento finalmente. La figura 38 muestra el esquema general reconocedor.

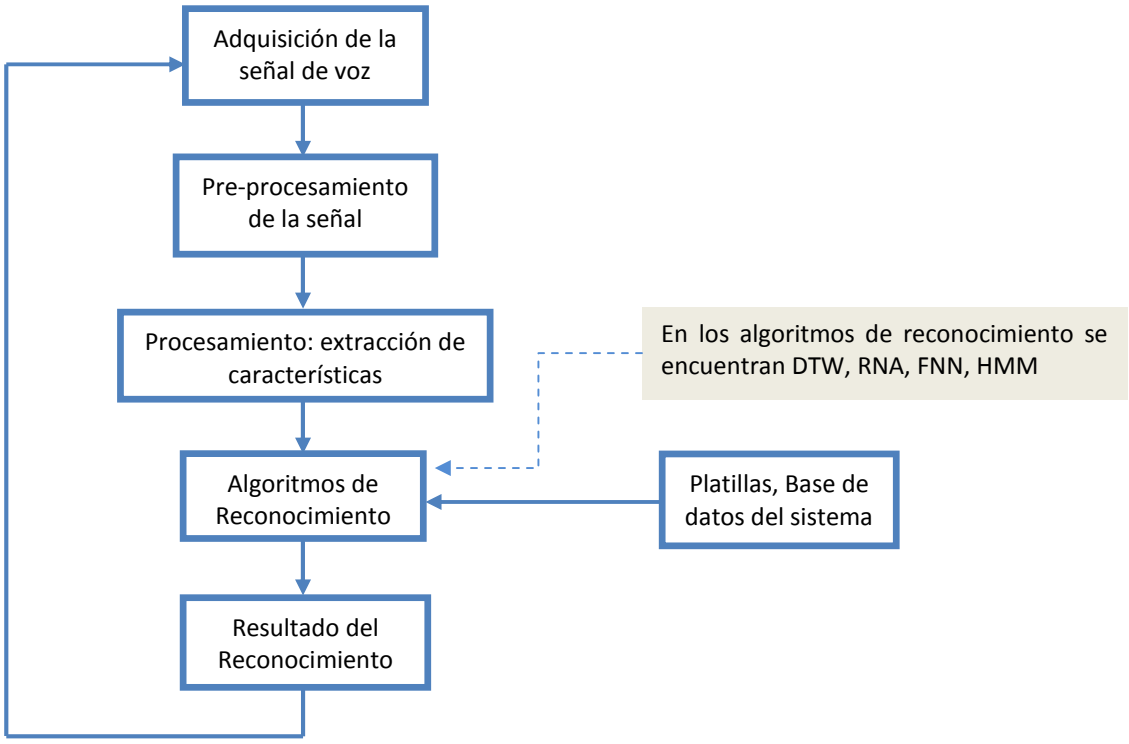


Figura 38. Diagrama de flujo del algoritmo de reconocimiento.

7.2.1 Algoritmo de reconocimiento mediante alineamiento temporal dinámico (DTW). El DTW (véase sección 4.5) es un algoritmo para medir la similitud entre dos secuencias que pueden variar en tiempo o velocidad. En general, este método permite encontrar una coincidencia óptima entre dos secuencias dadas, con ciertas restricciones. Las secuencias son "deformadas" de manera no lineal en la dimensión de tiempo para determinar una medida de su similitud independiente de ciertas variaciones no lineales en la dimensión de tiempo.

El algoritmo diseñado para el reconocimiento basado en DTW, adquiere una señal de voz por medio de la tarjeta de sonido del PC, a la cual le realiza un procesamiento similar al que se realizó en la etapa de adquisición de las muestras de voz y de extracción de características del sistema. El procesamiento consta de las etapas de eliminación Offset DC, detección inicio-fin de palabra, preénfasis, normalización, segmentación y ventaneo y el cálculo de parámetros LPC o MFCC. Luego del procesamiento, la señal representada por parámetros, pasa a través del algoritmo DTW (figura 39) que busca encontrar el camino más óptimo sobre la matriz de distancias locales. Este camino, se logra mediante la comparación de la secuencia de vectores que conforman la matriz de parámetros de la señal de entrada, con la secuencia de vectores que conforman la matriz de parámetros de la señal de referencia (parámetros de la base de datos). Finalmente la distancia

más óptima que minimiza el costo, es la que permite obtener el reconocimiento de la palabra.

```
C=Matriz_C;                          %Matriz señal de prueba
R=Matriz_R;                          %Matriz señal de referencia
[r1,V1]=size(C);                     %Dimensión matriz de prueba
[r2,V2]=size(R);                     %Dimensión matriz de referencia
distancia_local=zeros(r1,r2);        %Matriz para distancias locales
%------------Obtencion de la matriz de distancia local-----------%
for n=1:r1
    for m=1:r2
        FR=C(n,:)-R(m,:);
        FR=FR.^2;
        distancia_local(n,m)=sqrt(sum(FR));
    end
end
%-----------------------------------------------------------------%
distancia=zeros(r1+1,r2+1);     %Matriz de ceros - distancia local
distancia(1,:)=inf;
distancia(:,1)=inf;
distancia(1,1)=0;
distancia(2:(r1+1),2:(r2+1))=distancia_local;

%-------------------Obtener mínima distancia global-------------%
for i=1:r1;
    for j=1:r2;

[d_min]=min([distancia(i,j),distancia(i,j+1),distancia(i+1,j)]);
        distancia(i+1,j+1)=distancia(i+1,j+1)+d_min;
    end
end
%---------------Se obtiene el Puntaje global mínimo-------------%
DTWF=distancia(r1+1,r2+1);
%---------------Se obtiene el Puntaje global mínimo-------------%
DTWF=distancia(r1+1,r2+1);
```

Figura 39. Código en MATLAB para la obtención de la mínima distancia global con DTW.

El algoritmo del DTW, toma como muestra de referencia la palabra *Buenos Días* y como muestra de prueba una nueva pronunciación de la misma palabra, consiguiéndose el resultado mostrado en la figura 40.

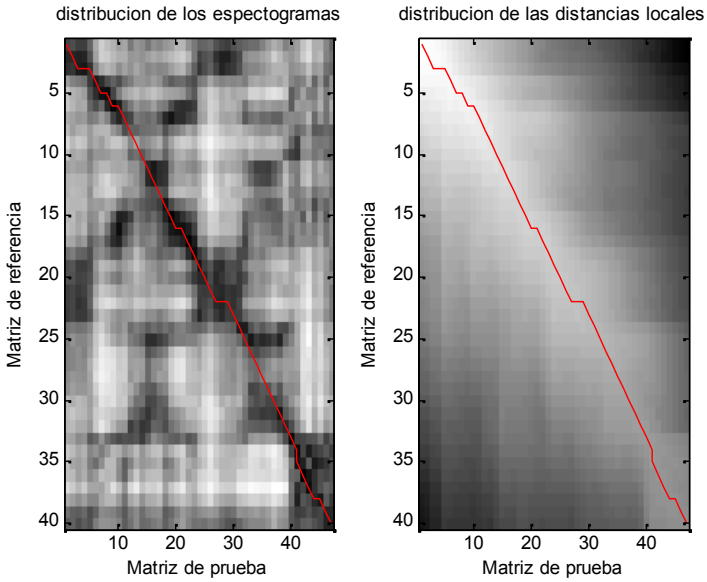


Figura 40. Distribución de los espectrogramas y las distancias locales tomando como referencia la palabra *Buenos Días* y como prueba una nueva versión de la misma palabra.

Ahora, si se toma como muestra de referencia la palabra *Buenos Días* y como muestra de prueba la palabra *Cinco*, se obtiene el resultado mostrado en la figura 41.

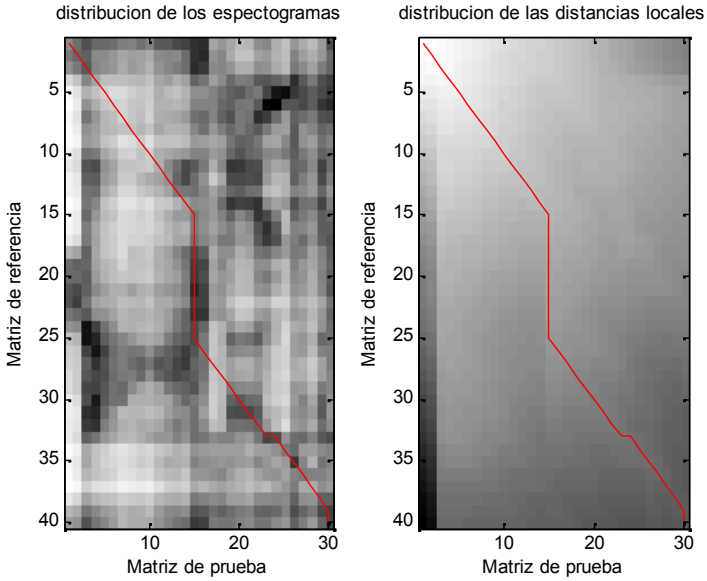


Figura 41. Distribución de los espectrogramas y las distancias locales tomando como referencia la palabra *Buenos Días* y como prueba la palabra *Cinco.*

En las anteriores figuras, la línea roja es el camino más óptimo y de mínimo costo desde el principio hasta el final de las señales. Las figuras de la izquierda muestran el camino DTW en una matriz de similitud, lo que indica la semejanza de

las dos señales. El camino que se tiende a elegir son los bloques más oscuros, ya que va a maximizar el rendimiento. Si se tiene en cuenta que reducir al mínimo la distancia es lo mismo que maximizar la similitud, se concluye que el mínimo camino es el presentado en la figura 40 y por lo tanto la palabra que se pretende reconocer corresponde a *Buenos Días* y no *Cinco*, pues el camino es mayor en la figura 41 lo que implica una menor similitud.

7.2.2 Algoritmo de reconocimiento mediante los híbridos RNA/DTW y RNA/Lógica Difusa. Con el fin de mejorar la taza de reconocimiento, se realizaron algunos sistemas híbridos, basados en la mezcla del DTW, las redes neuronales artificiales y la lógica difusa. Estos híbridos se desarrollaron con el fin de investigar el rendimiento de las redes neuronales y la lógica difusa dentro del reconocimiento de voz.

7.2.2.1 Híbrido RNA/DTW. La primera aplicación que se realiza es un sistema híbrido basado en redes neuronales y el DTW, en donde la red neuronal opera como clasificador de fonemas y el DTW permite manejar el problema que se presenta con las diferentes duraciones en tiempo de las palabras pronunciadas en distintas instancias.

La red neuronal clasifica en fonemas las salidas representadas en forma de Coeficientes de Predicción Lineal (LPC) o en Coeficientes Cepstrales en la escala de Mel (MFCC), obtenidos en la etapa de extracción de características. Por lo tanto, los doce LPC o trece MFCC, que representan las características de la voz, son las entradas a la red neuronal artificial. Como el reconocimiento de las palabras mostradas en la tabla 1 involucra trece fonemas diferentes, estos trece fonemas son las salidas de la red neuronal (ver tabla 2).

No del fonema	Sonido
1	/a/
2	/e/
3	/i/
4	/o/
5	/u/
6	/s/
7	/r/
8	/n/
9	/D/
10	/t/
11	/k/
12	/ č/ (ch)
13	/B/

Tabla 2. Fonemas involucrados en la primera aplicación.

No obstante, para hacer que la red neuronal sea capaz de clasificar los LPC o MFCC en fonemas es necesario añadir una capa oculta que procese la información. De esta manera se dispuso una red neuronal con 12 entradas, conectadas a 35 neuronas en la capa oculta con función de activación sigmoidal y estas a su vez conectadas a 13 neuronas en la capa de salida. El número de neuronas en la capa oculta se obtuvo por pruebas de ensayo y error, pues aumentar o disminuir el número de neuronas en la capa oculta afecta el porcentaje de acierto en la clasificación de fonemas. El número de entradas de la red neuronal se establece en 12 pues se obtuvieron mejores resultados con los LPC. La figura 42 muestra el código en MATLAB para la red neuronal y sus parámetros.

```
n_salidas=13;     %Numero de neuronas de salida
n_ocultas=35;     %Numero de neuronas en capa oculta

%Matriz de coeficientes, datos de entrada a la red
fo_ent=[fo1,fo2,fo3,fo4,fo5];

%Matriz de codificación u objetivos del entrenamiento para la red
fo_tar=[fot1,fot2,fot3,fot4,fot5];

%Creacion de la red neuronal
net=newff(fo_ent,fo_tar,[n_ocutas,n_salidas],{'tansig'
'tansig'},'trainbr');

%Inicializacion de los pesos sinápticos de la red
net=init(net);

%Entrenamiento de la red neuronal
[net,tr]=train(net,fo_ent,fo_tar);

%Guardado de la neurona y parametros
save NEURONA.mat net tr
```

Figura 42. Código en MATLAB para la Red Neuronal.

Como se menciono en la anterior sección el algoritmo DTW trabaja con una matriz de referencia por cada palabra a reconocer. Las matrices de referencia son creadas a partir de nuevos datos que se obtienen de la red neuronal, previamente entrenada. El resultado obtenido por cada fonema, es posteriormente agrupado, dependiendo de las palabras involucradas en el reconocimiento, conformando la base de datos del sistema.

7.2.2.2 Híbrido RNA/Lógica Difusa. Pensando en mejorar los resultados obtenidos en la primera aplicación mediante el uso de redes neuronales, la segunda aplicación trabaja con redes neuronales modulares, pues la modularidad nos permite dividir el problema del reconocimiento en sub-problemas más simples que pueden ser más fáciles de resolver. Además se utiliza un sistema de

inferencia difuso (FIS) Mamdani para modelar la incertidumbre en los resultados obtenidos por las redes neuronales a partir de los datos de entrenamiento. En esta aplicación, el sistema neuro-difuso está compuesto por las redes neuronales modulares y el sistema de inferencia difuso, conformándose un sistema secuencial o recurrente (véase sección 5.3).

El bloque difuso se diseño utilizando la herramienta FIS Editor del programa MATLAB, el cual permite diseñar sistemas difusos, estableciendo el número de entradas, el número de salidas, el tipo de funciones de pertenecía y las reglas que rigen el funcionamiento del sistema.

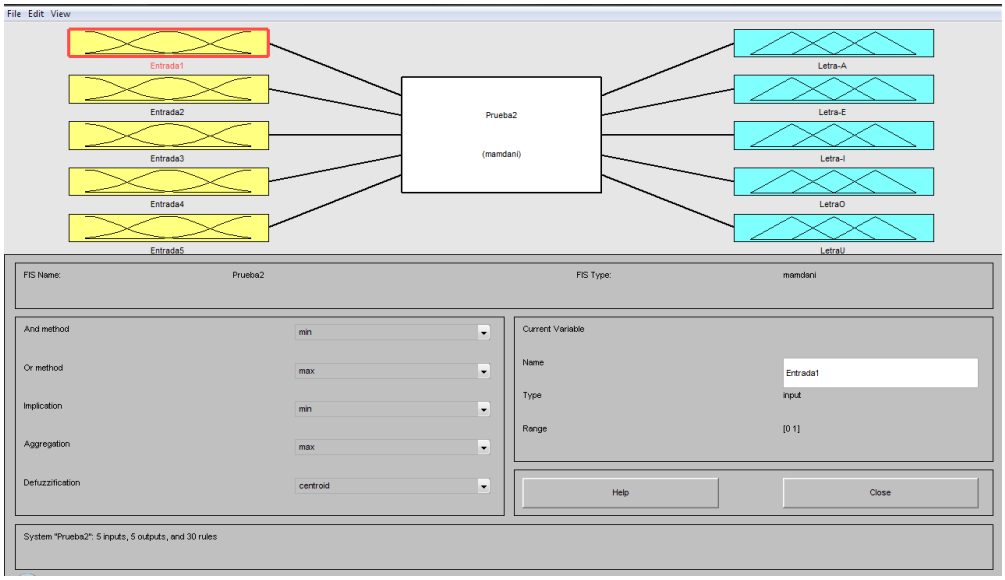


Figura 43. FIS Editor GUI del programa MATLAB.

A continuación, se muestra un ejemplo para ilustrar el enfoque híbrido del sistema. Se usan dos módulos con una red neuronal en cada uno de ellos, constituyendo la arquitectura modular. Cada módulo es entrenado con los mismos datos, pero los resultados son algo diferentes debido a la incertidumbre en el proceso de aprendizaje. En todos los casos, se usa redes neuronales con 35 neuronas en la capa oculta y "trainrp" como algoritmo de aprendizaje. La diferencia en los resultados se utiliza para crear un intervalo difuso que representa la incertidumbre en la clasificación de la palabra. La aplicación, trabajó con fonemas que corresponden a las vocales (tabla 3).

No del fonema	Sonido
1	/a/
2	/e/
3	/i/
4	/o/

5	/u/

Tabla 3. Fonemas involucrados en la segunda aplicación.

Teniendo en cuenta por el momento sólo 5 fonemas en el entrenamiento, con la primera red neuronal se obtiene los siguientes resultados:

Salida = [0,9895 -0,0007 0.0000 0,0059 0,0054]

La salida puede ser interpretada como el valor que representa a cada uno de los 5 fonemas en la base de datos. En este caso, se aprecia que el valor de 0,9895 es el valor que representa al fonema "a", que está muy cercano a 1. Pero, si ahora se entrena una segunda red neuronal con la misma arquitectura, debido a la inicialización aleatoria de los pesos, los resultados serán diferentes. Se tiene entonces los resultados de la segunda red neuronal.

Salida = [0,9709 -0,0102 0,0128 0,0552 -0,0287]

Ahora se nota que el valor que representa al fonema "a" es de 0.9709. Con estos dos valores de representación para el fonema, se puede definir un intervalo [0.9709, 0.9895], que da la incertidumbre de pertenencia, la cual indica que la señal de entrada corresponde al fonema "a" de la base de datos. En seguida, se tiene que usar la defuzzificación centriode (centroid deffuzification) para obtener un valor de pertenencia único. El mismo procedimiento se repite para los cinco fonemas involucrados en la aplicación. Los resultados para los 5 fonemas se muestran en la tabla.

/a/		/e/		/i/	
0,9895	0,9709	0,0992	0,0247	0,0011	-0,0002
-0,0007	-0,0102	0,9991	0,9758	-0,0044	-0,0034
0,0000	0,0128	0,0043	0,0145	0,9385	0,9372
0,0059	0,0552	-0,0709	-0,0236	-0,0007	-0,0037
0,0054	-0,0287	-0,0316	0,0087	0,0657	0,0700

/o/		/u/	
0,0381	0,0338	0,0015	-0,0020
0,0279	0,0061	0,0077	0,0186
-0,0011	0,0004	-0,0025	-0,0227
0,7110	0,7068	0,1818	0,1888
0,2241	0,2529	0,8116	0,8172

Tabla 4. Resumen de los resultados para los módulos para un conjunto de fonemas.

La arquitectura modular completa del sistema se muestra en la figura 44.

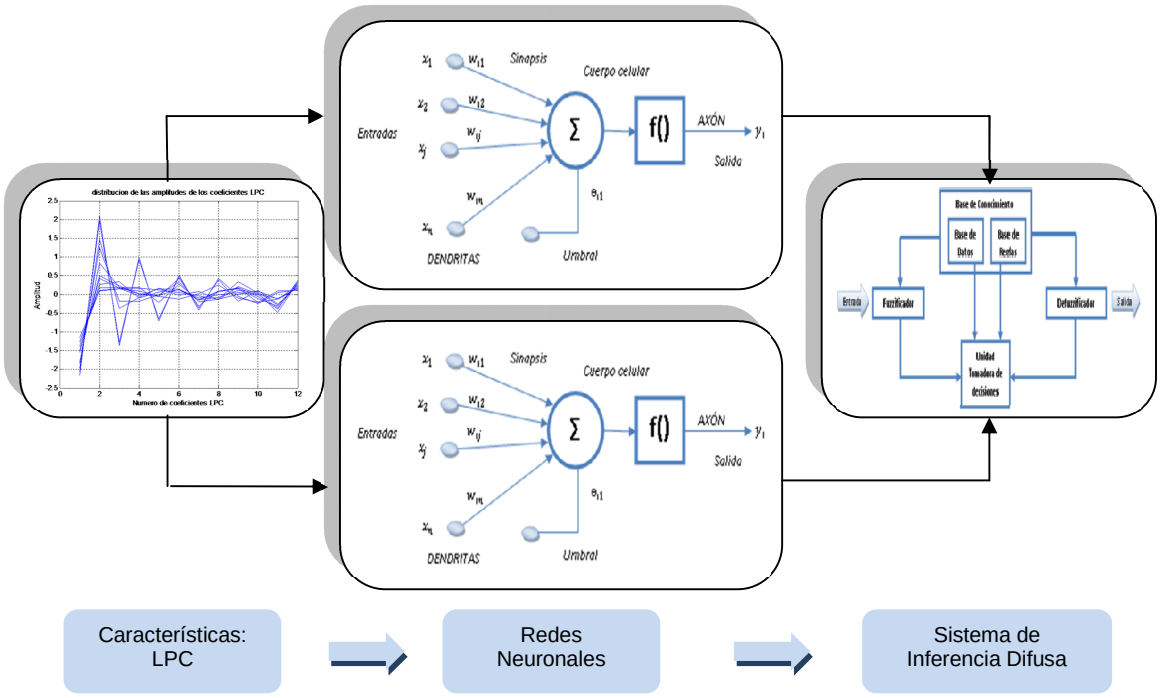


Figura 44. Arquitectura modular del sistema de reconocimiento de voz.

El mismo enfoque modular de la red neuronal se extendió a los 13 fonemas involucrados en la primera aplicación, sin embargo, debido al gran número de reglas que se utilizan para trabajar con los 5 fonemas (30 reglas), la implementación del sistema descrito no resulta adecuado para los 13 fonemas, pues la cantidad de reglas que definen el sistema se incrementarían considerablemente. No obstante, este enfoque presentaría muy buenos resultados si es aplicado al reconocimiento del hablante, en donde la lógica difusa permitiría mediante reglas definir las características propias de un hablante. Por otro lado, si bien existen algunas investigaciones con RNA en el reconocimiento de voz, las aplicaciones que involucran la lógica difusa son casi nulas, por lo que ofrecen un campo bastante amplio para ser explotado.

7.2.3 Algoritmo de reconocimiento mediante modelos ocultos de Markov (HMM). Siguiendo con el desarrollo de algoritmos de reconocimiento, en la presente sección se desarrolla un sistema de reconocimiento de voz aplicando la técnica de los modelos Ocultos de Markov, HMM (véase sección 4.7), pues esta técnica es usada en la mayoría de reconocedores comerciales como el incorporado en Windows o el desarrollado por Nuance debido a su gran porcentaje de reconocimiento.

Formalmente un HMM se compone del número de estados N, el número de símbolos de observaciones posibles M, una matriz de probabilidades transición

$\mathbf{A}$, una matriz de probabilidades de observación $\mathbf{B}$ y una matriz de distribución inicial de probabilidades $\boldsymbol{\pi}$. Todas estas componentes definen el comportamiento del algoritmo.

Para desarrollar la aplicación tenemos un vocabulario de V palabras ($V = 10$), que van a ser reconocidas, y se quiere diseñar un HMM distinto con N estados para cada palabra ($N = 5$). Supongamos además, que para cada palabra en el vocabulario se tiene un conjunto de entrenamiento de K ocurrencias o repeticiones habladas por una persona ($K = 50$), en donde cada ocurrencia de la palabra constituye una secuencia de observación, y donde las observaciones son alguna representación adecuada (espectral y/o temporal) de la señal de voz. En general, la señal correspondiente a cada palabra se representa con una secuencia temporal de vectores espectrales codificados. Se asume que la codificación se realiza con una plantilla con M palabras. Cada observación resulta entonces igual al índice del vector en la plantilla que está más cerca del vector espectral correspondiente a la señal de voz.

El algoritmo de reconocimiento puede resumirse como sigue:

1. Para cada palabra v en el vocabulario V se debe construir un HMM $\boldsymbol{\lambda_v}$, es decir se deben estimar los parámetros del modelo $(\mathbf{A}_v, \mathbf{B}_v, \boldsymbol{\pi}_v)$ que optimizan la probabilidad del conjunto de vectores de entrenamiento asociados a la v -ésima palabra (problema de entrenamiento).

2. Para cada palabra desconocida que se desea reconocer, se debe obtener la secuencia de vectores de observación $\mathbf{O} = [\mathbf{O}_1, \mathbf{O}_2, \dots, \mathbf{O}_T]$ y luego calcular las probabilidades de que esa secuencia haya sido generada por cada uno de los modelos posibles $P(\mathbf{O}\,|\lambda_v)$, con $1 \leq v \leq V$, para finalmente seleccionar la palabra cuyo modelo tenga la más alta probabilidad (problema de evaluación), es decir:

   $$v^* = \underset{1 \leq v \leq V}{\mathbf{argmax}}[P(\mathbf{O}\,|\lambda_v)]$$

   El cálculo de probabilidad se realiza generalmente mediante el algoritmo de Viterbi, es decir, se utiliza el camino de máxima verosimilitud. En la figura 45 se muestra el diagrama de bloques del sistema de reconocimiento basado en los Modelos Ocultos de Markov.

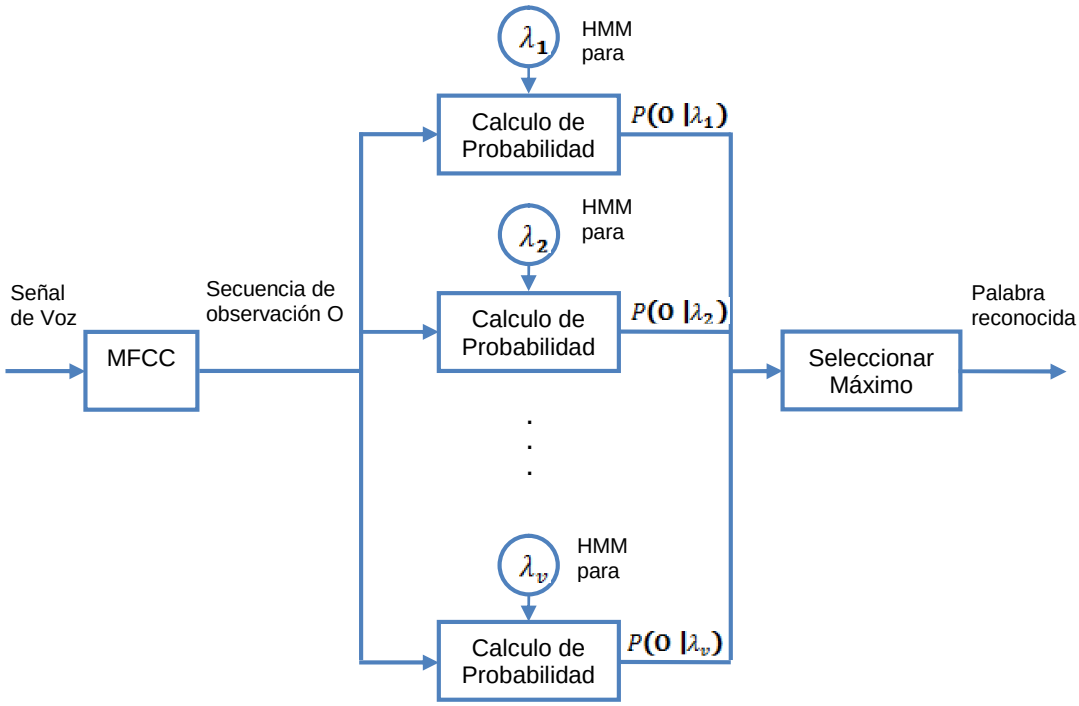


Figura 45. Diagrama de bloques de un reconocedor de palabras aisladas.

Sin embargo, antes de aplicar las fórmulas para el entrenamiento y la evaluación, es importante seleccionar el tipo de modelo y los parámetros iníciales, de manera que se alcance al máximo global o lo más cercano posible a él. Este proceso se conoce como inicialización.

Para el reconocimiento de palabras aisladas con un Modelo Oculto de Markov diferente para cada palabra en el vocabulario como el desarrollado, un modelo de izquierda-derecha resulta más adecuado que un modelo ergódico, ya que se puede asociar el tiempo y los estados del modelo como distintos sonidos de la palabra que está siendo modelada. En cuanto a los parámetros iníciales $\boldsymbol{\pi}$ y $\mathbf{A}$ una opción adecuada sería una distribución uniforme, pero como se trabaja con un modelo de izquierda-derecha, $\boldsymbol{\pi}$ tendrá una probabilidad de uno par el primer estado y cero para los otros estados. Concretamente $\boldsymbol{\pi}$ y $\mathbf{A}$ tienen inicialmente los siguientes valores:

$$\boldsymbol{\pi} = [1 \quad 0 \quad 0 \quad 0 \quad 0]_{1x5}$$

$$\mathbf{A} = \begin{bmatrix} 0.95 & 0.95 & 0 & 0 & 0 \\ 0 & 0.95 & 0.95 & 0 & 0 \\ 0 & 0 & 0.95 & 0.95 & 0 \\ 0 & 0 & 0 & 0.95 & 0.95 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}_{5x5}$$

En cuanto al número de estados N a utilizar en cada modelo de la palabra, se puede trabajar desde dos prospectivas diferentes. La primera, se basa en que el número de estados corresponden aproximadamente con el número de sonidos (fonemas) dentro de la palabra, por lo tanto, en esta perspectiva sería apropiado utilizar modelos que van desde 2 a 10 estados. Por otra parte, la segunda perspectiva consiste en dejar que el número de estados correspondan aproximadamente con el número promedio de observaciones en una versión hablada de la palabra. De esta manera, cada estado corresponde a un intervalo de observación. Este enfoque es el que se utiliza en el desarrollo, pues restringe cada modelo de palabra a tener el mismo número de estados, lo que implica que los modelos funcionen mejor ya que las palabras se representan con el mismo número de sonidos.

Por otra parte, se sabe que los M símbolos de emisión son valores discretos, estos

son los datos observables, pero lo que se tiene como salida de la extracción de características son vectores (vectores característicos) y no valores discretos, por lo tanto no es posible entrenar directamente los HMMs teniendo como entrada los MFCC o LPC. Entonces para lograr el entrenamiento de los HMMs a partir de los MFCC o LPC, se incorpora un algoritmo de agrupamiento intermedio que como su nombre lo indica agrupa los vectores en valores discretos de acuerdo a sus características. Para el presente trabajo se utiliza el algoritmo denominado k-means.

Finalmente, y antes de iniciar con el entrenamiento de los HMM, se obtiene la matriz de probabilidades de observación $\mathbf{B}$, la matriz con los valores medios asociados a las distribuciones Gaussianas Mu y matriz con los desvíos estándar asociados a las distribuciones Gaussianas $Sigma$. La matriz $\mathbf{B}$ se obtiene de manera aleatoria mediante la función `mk_stochastic`; Mu y $Sigma$ se obtienen mediante la función `mixgauss_init`[16].

Una vez obtenidos todos los parámetros iníciales se procede con el entrenamiento y la evaluación que se mencionaron anteriormente. Parte del código desarrollado en MATLAB se muestra en la figura 46.

[16] Todas las funciones se encuentran en el Toolbox de HMM, desarrollado por Kevin Murphy, disponible en: http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html

```
clear all; clc
%-------------------------------------------------------------------%
...
    <Cargar Audio>
...
nrepeticiones=50;                   %Numero de repeticiones
digit=['0123456789'];
%-------------------------------------------------------------------%
% INICIALIZACIÓN
%-------------------------------------------------------------------%
% Matriz de probabilidades de transmisión A
transmat0 = [0.95 0.05 0 0 0;0 0.95 0.05 0 0;0 0 0.95 0.05 0;...
             0 0 0 0.95 0.05;0 0 0 0 0.95];
%-------------------------------------------------------------------%
% Matriz de distribución inicial de probabilidades pi
prior0 = [1 0 0 0 0];
%-------------------------------------------------------------------%
cov_type = 'diag';
O = 13;          %Numero de coeficientes en un vector
M = 1;           %Numero de mixturas por estado
Q = 5;           %Numero de estados
T = 96;          %Numero de vectores en una secuencia
%-------------------------------------------------------------------%
for d=1:length(digit) %contador pra digitos del 0-1
    disp('Entrenamiento para el digito:');disp(digit(d));
...
    <Obtención de Coeficientes>
...
            [mu0, Sigma0] = mixgauss_init(Q*M, data, cov_type);
            mu0 = reshape(mu0, [O Q M]);
            Sigma0 = reshape(Sigma0, [O O Q M]);
            mixmat0 = mk_stochastic(rand(Q,M));

%         Obtencion del modelo por palabra
        [LL, prior1, transmat1, mu1, Sigma1, mixmat1] = ...
            mhmm_em(data, prior0, transmat0, mu0, Sigma0, ...
            mixmat0, 'max_iter', 10);
...
    <Guardar HHM obtenido por palabra>
...
```

Figura 46. Código en MATLAB para entrenamiento de los Modelos Ocultos de Markov.

Hasta el momento se diseñaron sistema que realizan un reconocimiento dependiente del locutor. Estos sistemas pueden ser generalizados para lograr un reconocimiento multiusuario dentro de un vocabulario limitado. Pero, para implementar el sistema, era necesario crear plantillas para cada una de las palabras dentro de la base de datos y para cada locutor, lo cual resultaba bastante extenuante e ineficaz. Por tal motivo, se vio la necesidad de explorar otras técnicas que permitieran ampliar el vocabulario y enfrentar la independencia del locutor, esto llevo a desarrollar una aplicación que utiliza el .Net Framework.

7.2.4 Algoritmo de reconocimiento usando .NET Framework desde MATLAB.

Dado que ninguno de los algoritmos desarrollados permiten alcanzar completamente el objetivo general del proyecto, se decide utilizar el Microsoft .NET Framework, que como se mencionó en la sección 6.1, es un componente integral de Windows que provee un extenso conjunto de soluciones pre-codificadas para requerimientos comunes, por lo que presenta una serie de beneficios que pueden ser utilizados desde la interfaz de MATLAB .NET como crear instancias de clases .NET o interactuar con aplicaciones .NET. En consecuencia, y aprovechando las capacidades de Microsoft .NET Framework, se puede realizar un reconocimiento de voz a través de la interfaz de MATLAB .NET, que para el desarrollo del proyecto significa:

- Optimización en el uso del tiempo máquina, haciendo más eficiente la aplicación.
- Eliminar la necesidad de crear una base de datos para cada hablante.
- Versatilidad del software, pues se puede instalar en cualquier computador con sistema operativo Windows Vista o superior.
- Aumento del vocabulario a 511palabras.
- Independencia del locutor.

Este aprovechamiento de las capacidades de .NET Framework se consigue haciendo uso de la Interfaz de Programación de Aplicaciones API (del inglés *Application Programming Interface*) y más específicamente la Speech API (SAPI), que son un conjunto de funciones y procedimientos que brinda cierta librería para ser utilizado por otro software.

El desarrollo del sistema de reconocimiento de voz, que utiliza los ensamblados (véase sección 6.2.1) incluidas en el .NET Framework, se realiza mediante programación en MATLAB y está dividido en varias fases que se describen a continuación.

Inicialmente y cuando se trabaja con una aplicación .NET es necesario hacer visibles los ensamblados para MATLAB. La forma de hacerlo depende del ensamblado, pues hay dos tipos de ensamblados: ensamblados privados y ensamblados compartidos. Los ensamblados privados se almacenan en el mismo directorio que la aplicación que los usa y sólo puede usarlos ésta. Los ensamblados compartidos se almacenan en un caché de ensamblado global (GAC) y pueden usarlos cualquiera que haya sido compilada referenciándolos.

El ensamblado compartido en MATLAB, se carga usando el nombre corto del ensamblado, que es el nombre del archivo sin la extensión. El ensamblado privado, por el contrario, necesita la ruta completa (carpeta y nombre de archivo con la extensión) del ensamblado. Para hacer uso de cualquier tipo de ensamblado, primero es necesario cargarlo utilizando el comando

`NET.addAssembly`, y después se puede trabajar con las clases definidas por el ensamblado[17]. No puede descargar un ensamblado desde MATLAB.

Dentro del contexto del .NET, también aparece el término *espacio de nombres (*en inglés *namespace)* que agrupa clases con funcionalidades similares y que está relacionado con el ensamblado. En MATLAB, un espacio de nombres es un paquete. Los *espacios de nombres* en .NET usados en la aplicación se muestran en la tabla 5.

Namespace	Descripción
System	Contiene clases fundamentales y clases base que definen los valores y tipos de datos de referencia, eventos y controladores de eventos, interfaces, atributos y excepciones de procesamiento comúnmente utilizados. Otras clases proporcionan servicios para convertir tipos de datos, manipular parámetros de métodos, realizar cálculos matemáticos, invocar programas remotos y locales, administrar el entorno de aplicaciones y supervisar aplicaciones administradas y no administradas.
System.IO	Contiene tipos que permiten la lectura y escritura sincrónicas y asincrónicas en archivos y secuencias de datos.
System.Xml	Proporciona compatibilidad basada en estándares para procesar XML.
System.Speech	El espacio de nombres `System.Speech` contiene tipos que apoyan el reconocimiento de voz.
System.Windows.Forms	Contiene clases para crear aplicaciones basadas en Windows y que aprovechan plenamente las características avanzadas de interfaz de usuario disponibles en este sistema operativo.
System.Runtime	Contiene tipos avanzados que admiten diversos espacios de nombres tales como *System*, los espacios de nombres Runtime y los espacios de nombres Security.

[17] MATLAB carga los ensamblados `mscorlib.dll` y `system.dll` desde .NET Framework cuando el programa es iniciado.

System.Runtime.Serialization	Contiene clases que se pueden utilizar para serializar y deserializar objetos. La serialización es el proceso de convertir un objeto o un gráfico de objetos en una secuencia lineal de bytes para su almacenamiento o transmisión a otra ubicación. La deserialización es el proceso de volver a generar los objetos a partir de la información almacenada.

Tabla 5. Espacios de Nombres utilizados dentro de la aplicación y su descripción general.

En MATLAB, se puede referir a cualquier clase por su nombre totalmente cualificado (namespace.ClassName)[18], que incluye el nombre del espacio de nombres. Un nombre totalmente calificado puede ser bastante largo, por lo que resulta difícil editar y leer comandos y funciones. Sin embargo, se puede hacer referencia a las clases por el nombre de la clase sola (sin el espacio de nombres), si primero importa el nombre totalmente cualificado en MATLAB. Esto se realiza mediante la función `import` que agrega todas las clases que se importan a una lista llamada lista de importación. Por ejemplo, para eliminar la necesidad de escribir `System.` antes de cada comando, se escribe `import System.*`.

Una vez puntualizada la manera cómo se carga los ensamblados de .NET Framework en MATLAB, a continuación se describe el uso de las clases contenidas en los espacios de nombres, mencionados en la tabla 1, para desarrollar el sistema de reconocimiento de voz.

En la primera fase, se crea una lista de opciones utilizando la clase `Choices`. Esta lista es creada con el fin de evitar resultados inútiles para la aplicación, pues elimina el problema de escuchar palabras que no se encuentren en la lista y limita el motor de reconocimiento para que pueda tomar mejores decisiones entre sonidos que puedan ser confusos.

La lista de opciones constituye la base de datos del sistema, base que es creada en forma matricial haciendo uso del comando `NET.createArray`. Sin embargo, crear la matriz cada vez que se carga el sistema implica un gasto computacional, en consecuencia y para mejorar el rendimiento del sistema, esta base de datos se serializa en un archivo denominado *diccionario.dat* (ver figura 47), que permite tener la base de datos disponible para las diferentes usos dentro del sistema.

[18] Un nombre totalmente cualificado viene del inglés *fully qualified name*

```
%Matriz de palabras
strArr = NET.createArray('System.String', 511);
%-----------------------------------------------------------------------%
strArr.Set(0,'Adulto')
strArr.Set(1,'El')
strArr.Set(2,'Ellas')
strArr.Set(3,'Hombre')
...
%-----------------------------------------------------------------------%
%Serializacion del archivo (Esto es como si guardara la informacion)
stream=File.Open('diccionario.dat', FileMode.Create);
formatter=BinaryFormatter();
formatter.Serialize(stream, strArr);
stream.Close();
```

Figura 47. Código en MATLAB para crear el archivo *diccionario.dat.*

En la figura 47 se hace uso del espacio de nombres `System.Runtime.Serialization`, para serializar, y posteriormente para deserializar[19], la matriz *strArr* que corresponde a un objeto .NET.

Luego de crear la base de datos y de añadirla al objeto .NET, obtenido a partir del uso de la clase `Choices`, en la segunda etapa se crea la gramática del sistema. Este proceso se consigue mediante el uso de las clases `GrammarBuilder` y `Grammar`. La clase `GrammarBuilder` proporciona un mecanismo para construir una gramática compleja a partir de entradas simples; es decir, define la forma en que las palabras pueden ser combinadas. La clase `Grammar` proporciona soporte para la obtención y manejo de la información gramatical.

Una vez establecida la gramática del sistema, la tercera etapa, corresponde a la creación del reconocedor empleando la clase `SpeechRecognitionEngine`. En esta fase el reconocedor empezará a escuchar el habla que se ajuste a los patrones definidos por la gramática. Cuando el motor de reconocimiento de voz reconoce algo que se ajusta a los patrones definidos por gramática, se produce el evento `SpeechRecognized`, el cual ejecuta el controlador de eventos que recibe dicho evento y que determina las acciones que deben ejecutarse, como por ejemplo: “mostrar la palabra reconocida”. Cuando el motor de reconocimiento de voz no identifica el habla de entrada, se produce el evento `SpeechRecognitionReject`, el cual ejecuta el controlador de eventos que recibe dicho evento y que realizara otra acción, como mostrar por ejemplo: “Palabra no Identificada”.

Cabe resaltar que cada objeto creado, contiene sus propiedades, métodos y eventos, que son utilizados de acuerdo a las necesidades del sistema de

[19] Los procesos de serialización y deserialización permiten almacenar y transferir fácilmente datos

reconocimiento de voz realizado. La figura 48 muestra parte de la programación para obtener el reconocimiento y la figura 49 muestra el esquema del algoritmo utilizado.

```
...
    <Carga de Ensamblados>
...
    <Importación de espacio de nombres>

%Opciones a reconocer
opciones = Choices();
%Carga de la base de datos previamente creada
...
    <Deserializacion del diccionario>
...
%Adición de la base de datos al objeto
opciones.Add(obj1)

%Gramatica del sistema
constructor=GrammarBuilder(choices);
...
    <establecimiento de la gramatica del sistema>
...
%Creación y activación del objeto reconocedor de habla
recocedor=SpeechRecognitionEngine();
...
    <Caracteristicas del reconocedor>
...

%Eventos del reconocedor
%-------------------------------------------------------------------%
h=event.listener (recognizer, 'SpeechRecognized', ...
                  @(hObject, eventdata)Interfaz2_reconocedor1...
                  (hObject, eventdata, handles));
%-------------------------------------------------------------------%
i=event.listener (recognizer, 'SpeechRecognitionRejected', ...
                  @(hObject, eventdata)Interfaz2_reconocedor2...
                  (hObject, eventdata, handles));
```

Figura 48. Código en MATLAB para realizar reconocimiento a partir de MATLAB y el .NET Framework.

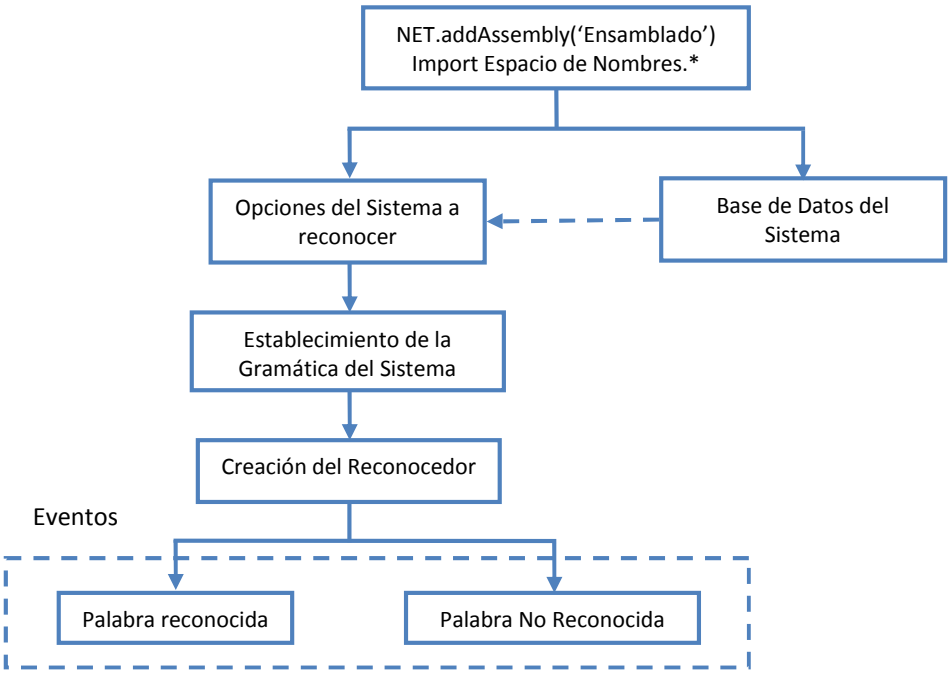


Figura 49. Esquema básico del reconocimiento de voz con programación MATLAB y .NET.

7.3 DESARROLLO DE LA INTERFAZ GRÁFICA PARA LA VISUALIZACIÓN DE LAS PALABRAS RECONOCIDAS EN LENGUA DE SEÑAS

En esta etapa se desarrolla la interfaz grafica de usuario o GUI (del inglés *graphical user interface)*, con el cual es posible llevar a cabo el proceso de reconocimiento y su traducción a lengua de señas, sin la necesidad de ser un desarrollador o basarse en un lenguaje de programación básico. Esta interfaz, que es el medio con que el usuario puede comunicarse con un computador, utiliza un conjunto de objetos gráficos para representar la información y las acciones que se encuentran disponibles.

Su desarrollo se realiza apoyado en MATLAB como software de programación, debido a su gran robustez, características simples en el manejo matemático y gran capacidad en el procesamiento. Además, proporciona un entorno visual sencillo que permite la comunicación con el computador.

La figura 50 orienta el proceso del sistema de reconocimiento de voz, en donde la señal de voz capturada es procesada, analizada y reconocida en diferentes etapas. Luego, el resultado es mostrado en lengua de señas, a través de la interfaz grafica de usuario.

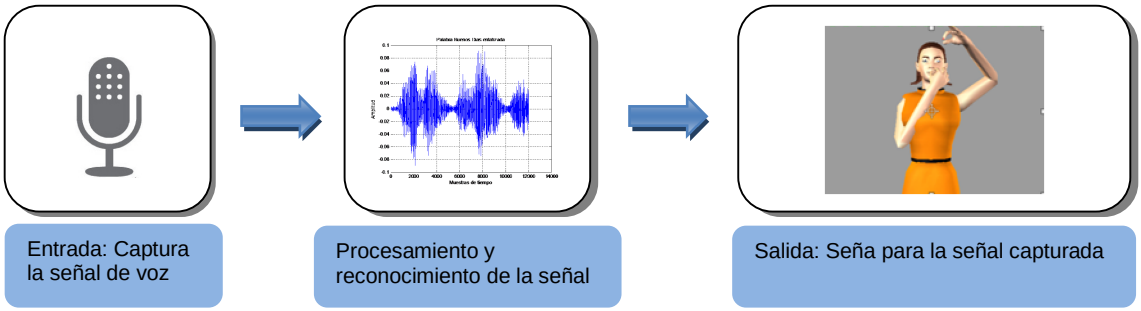


Figura 50. Diagrama de bloques del Sistema de reconocimiento y traducción a LSC.

Tras una serie de análisis en cuanto al procesamiento, mejora de código, recopilación de algoritmos y distribución ordenada para realizar los procesos necesarios, se procedió a realizar en este proyecto el software denominado *LSC* con base a las siglas de Lengua de Señas Colombiana, provisto de características que permiten realizar el reconocimiento de voz y traducir la palabra reconocida a lengua de señas. El software de sistema está basado en el algoritmo de reconocimiento que hace uso del .Net Framework desde MATLAB, pues con este algoritmo se obtuvo un mayor porcentaje de reconocimiento por palabra. Además, brinda la posibilidad de aumentar el vocabulario del sistema.

Las diferentes características que componen el sistema de reconocimiento, están basadas en diversas funciones y se encuentran en varias ventanas que se describen en la sección 7.3.1 y 7.3.2.

7.3.1 Ventana Principal. De forma similar a muchos de los programas basados en presentación de ventanas a los cuales se está acostumbrado, la ventana principal, o ventana de reconocimiento de palabras, es un área visual que ha sido diseñada con unos menús desplegables para acceder a las diferentes funcionalidades que ofrece el sistema.

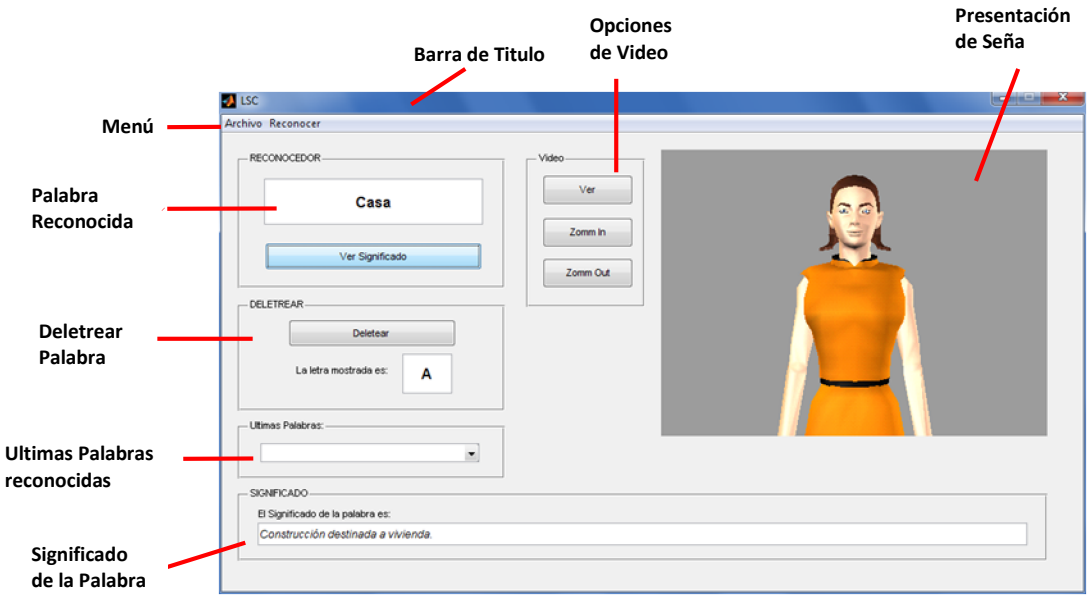


Figura 51. Ventana Principal.

La ventana principal, primordialmente reconoce las palabras y muestra cada palabra reconocida en lengua de señas siempre y cuando se encuentre en la base de datos. Los menús incluidos en esta ventana, que se denominan *Archivo* y *Reconocer,* se presentan en la figura 52 y permiten realizar otros tipos de procesamiento o acceder a otras ventanas.

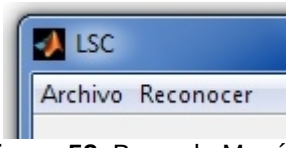


Figura 52. Barra de Menús.

Archivo: Menú que se despliega para dar lugar a *Buscar Palabra*, que permite realizar la búsqueda de una palabra de manera escrita dentro del sistema. De igual forma permite activar o desactivar el reconocedor y cerrar el programa de forma general al presionar la pestaña *Salir* (figura 53).

Figura 53. Menú *Archivo* desplegado.

Reconocer: Muestra la pestaña *Reconocer Oración* (figura 54), que presenta la ventana auxiliar LCS-O, en la que se realiza un reconocimiento a nivel de oración y que se detalla en la sección 7.3.2.

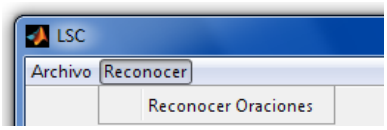


Figura 54. Menú *Reconocer* desplegado.

Adicional a las características referidas en la barra de menús, la ventana principal posee otras secciones denominadas *RECONOCEDOR*, *Video*, *DELETREAR*, Ultimas palabras y *SIGNIFICADO*, que se detallan a continuación.

- *RECONOCEDOR:* Esta sección contiene dos elementos: una caja de texto y un botón. En la caja de texto se muestra la palabra que ha sido reconocida por el sistema y que ha sido traducida a lengua de señas. El Botón denominado *Ver Significado* permite, como su nombre lo indica, ver el significado de la palabra reconocida según el Diccionario Básico de la Lengua de Señas Colombiana (DBLSC)[20] en la sección *SIGNIFICADO*.

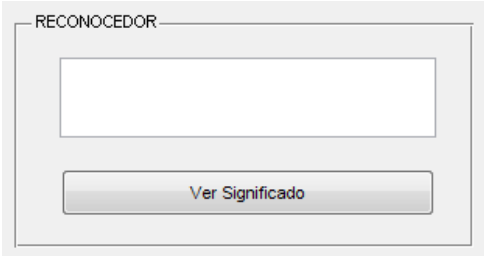
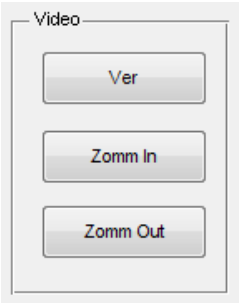


Figura 55. Sección *RECONOCEDOR.*

- *Video:* Aunque la señа se muestra automáticamente una vez reconocida la señal de voz, esta sección contiene varios botones denominados *Repetir*, *Zoom In* y *Zoom Out*, que permiten realizar acciones sobre la seña mostrada. Las acciones realizadas son repetir, alejar o acercar la seña respectivamente.



[20] El DBLSC es un documento desarrollado por el Instituto Nacional Para Sordos INSOR.

Figura 56. Sección *Video.*

- *DELETREAR:* Esta sección posee un botón, denominado *Deletrear*, con el cual el usuario puede visualizar cada una de las letras que conforman la palabra reconocida a través del alfabeto manual, es decir, deletrea la palabra reconocida en lengua de señas. Cada letra es mostrada en la caja de texto de esta sección y su correspondiente seña en el campo *Presentación de Seña*.

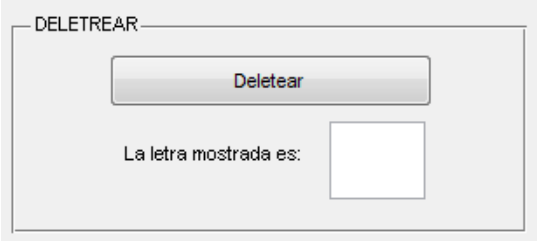


Figura 57. Sección DELETREAR.

- *SIGNIFICADO:* En esta sección se muestra el significado de la palabra reconocida haciendo uso de la base de datos del sistema. La base de datos contiene los significados de cada una de las palabras que el sistema es capaz de reconocer y como se menciono anteriormente está basada en el Diccionario Básico de la Lengua de Señas Colombiana (DBLSC), que es un documento desarrollado por INSOR.

Figura 58. Sección *SIGNIFICADO.*

- *Últimas Palabras*: Contiene un menú desplegable que exhibe las últimas cinco palabras reconocidas por el sistema y que permite acceder a ellas sin la necesidad de volver a pronunciarlas.

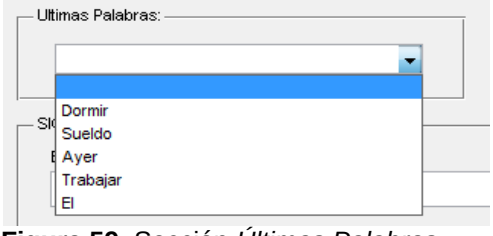


Figura 59. Sección *Últimas Palabras.*

- *Presentación de Seña*: Se presenta las señas correspondientes a las palabras reconocidas. El desarrollo de las señas a nivel gráfico, es decir las animaciones, se realizaron haciendo uso del ejecutable Vsign Builder, basado en Macromedia Director Shockwave Studio 8.5, debido a su facilidad de uso.

Las señas desarrolladas, que corresponde a las palabras de la base de datos del sistema, se guardan en archivo *.swf y se muestran de forma automática una vez reconocida la palabra.

Figura 60. Sección *Presentación de Seña.*

7.3.2 Ventana Auxiliar. En la ventana auxiliar o ventana de reconocimiento de oraciones (figura 61), se ha realizado un compilado con el que se puede llevar a cabo un reconocimiento de oraciones.

Esta ventana, que es accedida a través del menú *Reconocer* de la ventana principal, básicamente traduce las oraciones reconocidas a lengua señas. Sin embargo, la traducción se realiza de manera elemental, pues la estructura gramatical de la lengua de señas resulta bastante compleja de abordar dentro de los alcances del proyecto.

Para orientar el funcionamiento de la gramática del sistema, imagine que se reconoce la oración "*El niño está comiendo hamburguesa*", lo cual es interpretado por el sistema como "El *NIÑO COMER HAMBURGUESA*". Una vez interpretado, el sistema pasa a mostrar las señas correspondientes a "NIÑO" y "COMER", pero como la palabra "HAMBURGUESAS" no está definida dentro diccionario del sistema, entonces el sistema simplemente la deletrearía utilizando el alfabeto manual.

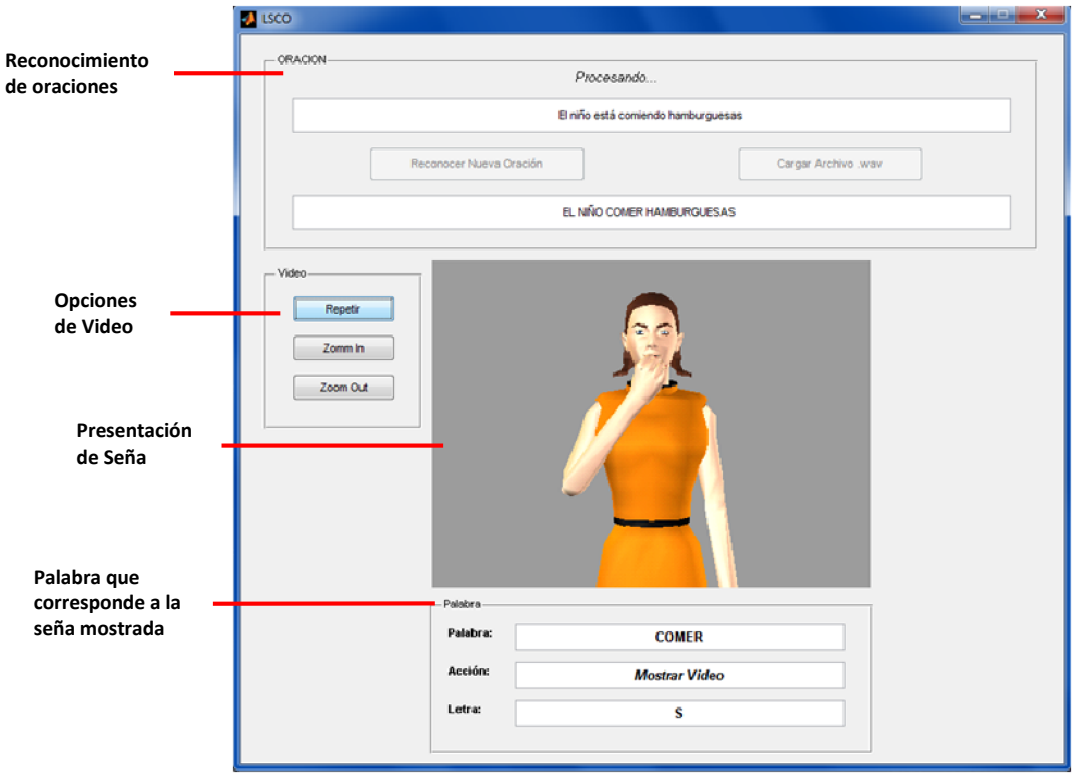


Figura 61. Ventana Auxiliar.

Continuando con la descripción de la ventana auxiliar, se observa que esta ventana posee, al igual que la ventana principal, varias secciones denominadas *ORACIÓN*, *Video*, y *Palabra*, que se precisan a continuación.

- *ORACIÓN*: Esta sección contiene dos cajas de texto y dos botones. La caja de texto superior muestra la oración pronunciada por el usuario y la inferior la misma oración después de pasar por un proceso gramatical elemental que dé lugar a la interpretación en lengua de señas. El botón *Reconocer Nueva Oración*, activa el reconocedor para realizar un nuevo reconocimiento y procesamiento de oraciones, pues una vez la oración es reconocida, el reconocedor del sistema se desactiva.

  El botón *Cargar Archivo .wav*, como su nombre lo indica, carga un archivo de audio .wav almacenado en el computador y realiza el procesamiento y reconocimiento de la señal de voz contenida en este, para luego traducirlo a lengua de señas.

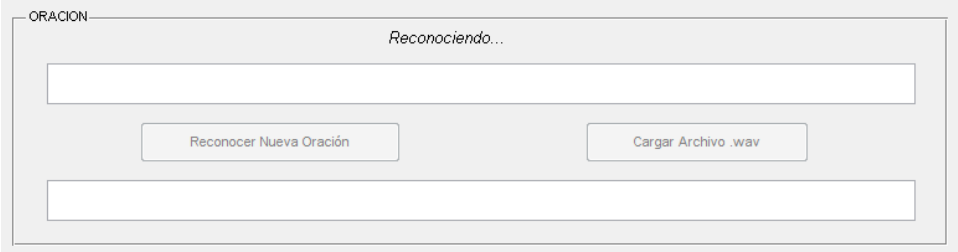


Figura 62. Sección *ORACIÓN.*

- *Video*: Esta sección permite realizar acciones para la visualización de la oración en lengua de señas. Posee opciones como *Repetir* para volver a ver la oración en lengua de señas y opciones de zoom, para acercar o alejar la seña.

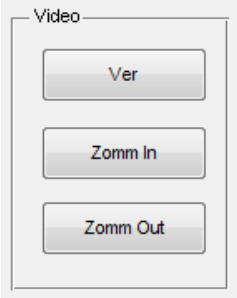


Figura 63. Sección *Video.*

- *Palabra*: Aquí se muestra cada una de las palabras que componen la oración reconocida y que son mostradas por el sistema. Como hay palabras dentro de la oración que no tienen un equivalente en señas, el sistema automáticamente deletrea la palabra usando el alfabeto manual. Recapitulando se dice que si la palabra posee un equivalente en señas, el sistema muestra la señ mediante la acción *"Mostrar video"*; por el contrario, si la palabra no poseen un equivalente en señas, simplemente la deletrea mediante la acción *"Deletrear".* Este proceso se realiza para facilitar su uso y no confundir al usuario.

Figura 64. Sección Palabra.

7.3.3 Requerimientos. Finalmente se expone que el software en el que se basó el desarrollo fue en Matlab R2009a versión 7.8.0.347.

El equipo de cómputo sobre el cual fue desarrollado posee las siguientes características básicas y de operación satisfactoria:

- Sistema operativo: Windows Vista o Windows 7
- Equipo Intel® Pentium® Dual Core CPU 3.00GHz
- 3GB de RAM

7.4 PRUEBAS DE VALIDACIÓN DEL SISTEMA Y MODIFICACIONES

Una vez culminado el diseño de los algoritmos para la adquisición, entrenamiento y reconocimiento que dieron lugar a pruebas preliminares y al desarrollo de la interfaz gráfica para la visualización de los resultados, en la presente sección se realiza una serie de pruebas con el fin de validar el correcto funcionamiento y detectar posibles fallas del sistema desarrollado. Así mismo, se muestran una comparación de los diferentes resultados obtenidos con los algoritmos mencionados y se realiza una aplicación no contemplada dentro de los objetivos del proyecto.

7.4.1 Resultados de los algoritmos. Los algoritmos de reconocimiento presentados en la sección 7.2 pueden ser usados en sistemas de reconocimiento de voz, dependiendo de la complejidad de la aplicación, sin embargo, el que tiene mayor rendimiento es el de .NET Framework basados en los HMM.

7.4.1.1 Resultados DTW. Inicialmente se trabajo con la aplicación DTW para el reconocimiento de palabras realizándose dos pruebas. En la primera prueba se realiza un reconocimiento de dígitos del cero al nueve y en la segunda se incrementa el número de palabras a 20 (las palabras se muestran en la tabla 1). Los resultados obtenidos fueron los siguientes:

	10 Palabras	**20 Palabras**
palabras clasificadas correctamente	83,5%	79,75%
palabras clasificadas incorrectamente	16,5%	20,25%
Total	100%	100%

Tabla 6. Porcentaje de reconocimiento del algoritmo DTW con 10 y 20 palabras.

En la primera prueba, se observa que dependiendo del grado de parentesco entre los fonemas de la secuencia de entrada con los de las plantillas de la base de datos, el porcentaje de reconocimiento es afectado. Por ejemplo, hubo un bajo porcentaje de acierto con las palabras tres y siete, las cuales eran confundidas con las palabras dos y cinco respectivamente. En estos casos el porcentaje de acierto alcanzo un valor de 50%.
En la segunda prueba se observa una reducción en el porcentaje de aciertos con el algoritmo DTW, esto ocurre debido a que la comparación de la entrada se

realiza con un número mayor de plantillas, incurriendo en un mayor grado de desacierto por el parentesco entre fonemas de varias palabras.

7.4.1.2 Resultado híbrido RNA/DTW. En la aplicación mediante el híbrido RNA/DTW se trabajo inicialmente con 10 palabras, que involucran los 13 fonemas mostrados en la tabla 2 y posteriormente se incrementó el número de palabras a 20, palabras que se muestran en la tabla 1 y que involucran los mismos 13 fonemas. Los resultados obtenidos se muestran a continuación.

	10 Palabras	20 Palabras
Palabras clasificadas correctamente	98,5%	96,3%
Palabras clasificadas incorrectamente	1,5%	3,7%
Total	100%	100%

Tabla 7. Porcentaje de reconocimiento mediante el híbrido RNA/DTW con 10 y 20 palabras.

En esta aplicación, el porcentaje de reconocimiento aumenta si se compara con los resultados obtenidos mediante la aplicación DTW (sección 7.4.1.1), y aunque al aumentar el número de palabras el porcentaje de acierto disminuye, en general el sistema implementado mediante este híbrido funciona satisfactoriamente.

7.4.1.3 Resultado RNA/FIS. Esta prueba difiere un poco de las demás, pues aunque tuvo un porcentaje de acierto de 94.7% con los cinco fonemas correspondientes a las vocales, cuando su implementación se trato realizar con las 10 palabras (o 13 fonemas), si bien el porcentaje de reconocimiento fue mayor que el obtenido mediante el algoritmo de DTW, no fue mayor que el obtenido mediante el híbrido RNA/DTW. Esto sumado al incremento en el tiempo de procesamiento y a la gran cantidad de reglas a definir, dio como resultado la inviabilidad de desarrollar un sistema de reconocimiento basado en un sistema neuro-difuso secuencial. Por tal motivo, los valores de la tabla 8 y la tabla 9 no se tienen en cuenta en la comparación de resultados.

	Porcentaje
Vocales clasificadas correctamente	94.7%
Vocales clasificadas incorrectamente	5.3%
Total	100%

Tabla 8. Porcentaje de reconocimiento mediante el híbrido RNA/FIS con 5 vocales.

	Porcentaje
Palabras clasificadas	85.5%

correctamente	
Palabras clasificadas incorrectamente	14.5%
Total	100%

Tabla 9. Porcentaje de reconocimiento mediante el híbrido RNA/FIS con 10 palabras.

7.4.1.4 Resultado HMM. Para la aplicación que involucra los Modelos ocultos de Markov, se realizo una prueba similar a las anteriores en donde se reconocen los dígitos del cero al nueve con porcentajes de acierto de 94%. Los resultados obtenidos se muestran en la tabla 10.

	Porcentaje
palabras clasificadas correctamente	94%
palabras clasificadas incorrectamente	6%
Total	100%

Tabla 10. Porcentaje de reconocimiento del algoritmo HMM con 10 palabras.

Si bien los resultados presentados en la tabla 10 fueron bastante satisfactorios, estos valores se omiten en la comparación de resultados pues la aplicación está enfocada a explicar el funcionamiento básico de los Modelos Ocultos de Markov, lo cuales están incorporados en *.Net Framework* y permiten realizar un reconocimiento más robusto.

7.4.1.5 Resultado Interfaz MATLAB/.NET. De igual forma que en los casos anteriores, se realizaron dos pruebas, la primera con 10 palabras (números del cero al nueve) y la segunda con 20 palabras. En ambos casos se midió el porcentaje de aciertos que ofrece la aplicación, obteniendo los siguientes resultados:

	10 Palabras	**20 Palabras**
Palabras clasificadas correctamente	98%	98,75%
Palabras clasificadas incorrectamente	2%	1,25%
Total	100%	100%

Tabla 11. Porcentaje de reconocimiento mediante la interfaz MATLAB y .NET con 10 palabras.

En las dos pruebas se observa un porcentaje de acierto bastante alto y aunque con 10 palabras el acierto es menor que con el híbrido RNA/DTW, esto queda justificado si se tiene en cuenta que el sistema es independiente del locutor.

7.4.2 Comparación de los algoritmos. En la figura 65 se puede observar los resultados obtenidos con los diferentes algoritmos. EL resultado obtenido mediante la interfaz MATLAB/.NET es el que presenta mejor porcentaje de reconocimiento cuando el número de palabras es mayor. Mientras que para la primera aplicación, con un menor número de palabras, el porcentaje de reconocimiento más alto se presenta con híbrido RNA/DTW. En consecuencia, el algoritmo implementado finalmente, es el realizado mediante la interfaz MATLAB/.NET dado su porcentaje de acierto y su robustez para la aplicación.

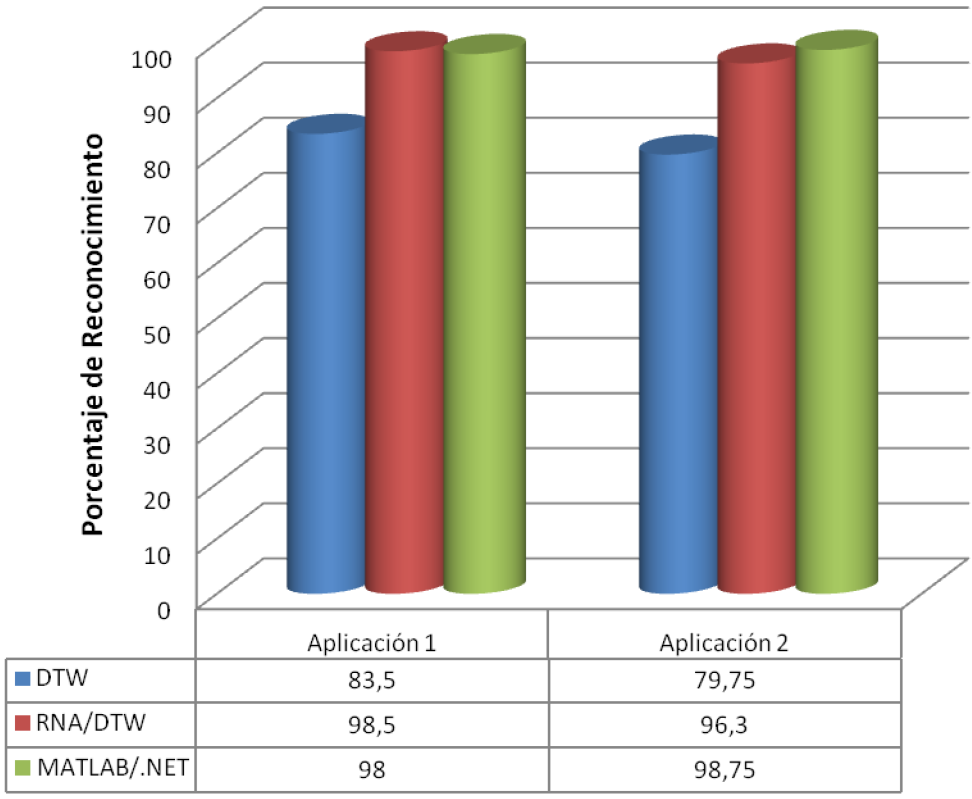


Figura 65. Grafica comparativa de los algoritmos de reconocimiento para las aplicaciones.

En las figuras 66 y 67 se muestra el porcentaje de aciertos y desaciertos con cada una de las técnicas empleadas, para un vocabulario con 10 palabras y para un vocabulario de 20 palabras respectivamente.

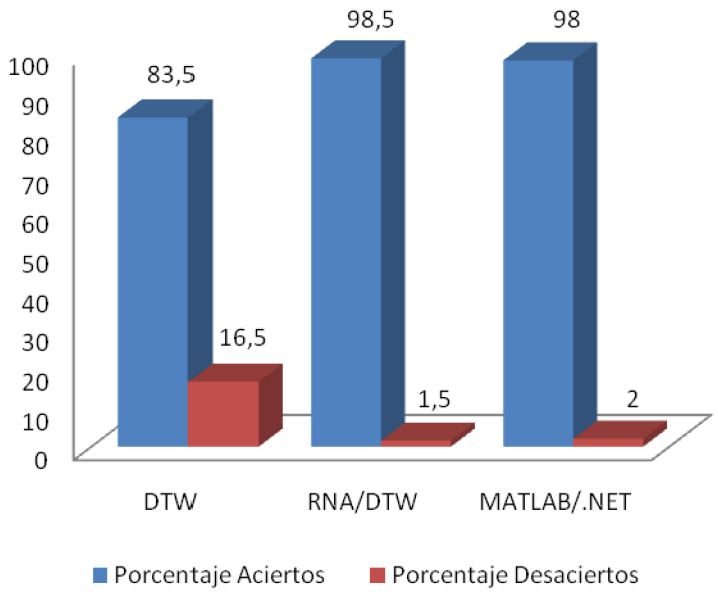


Figura 66. Porcentaje de reconocimiento de la aplicación con 10 palabras para cada uno de los algoritmos utilizados.

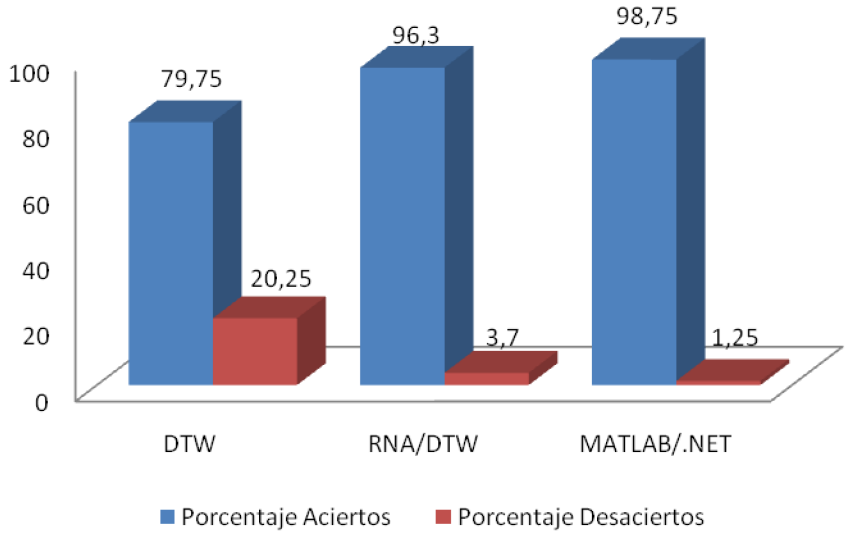


Figura 67. Porcentaje de reconocimiento de la aplicación con 20 palabras para cada uno de los algoritmos utilizados.

7.4.3 Aplicación control motor. Como se mencionó al principio de la sección, se desarrollo una aplicación que no estaba contemplada dentro de los objetivos del presente trabajo. La aplicación, que muestra la versatilidad del sistema desarrollado, utiliza los comandos de voz para controlar los grados de giro de un motor *paso a paso* (PaP). El diagrama general de la aplicación se muestra en la figura 68.

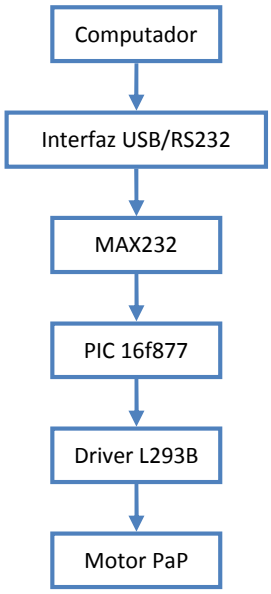


Figura 68. Diagrama de bloques del sistema electrónico.

Para llevar a cabo la aplicación, se dispone de un computador que posee una conexión de tipo USB, a través de la cual se pretende realizar una conexión serial RS232, ya que la mayoría de los equipos actualmente construidos poseen el estándar USB. Sin embargo, el sistema puede ser conectado directamente al puerto serial si el equipo en uso lo permite.

Habiendo resuelto el medio de comunicación, la información es acondicionada a niveles de voltaje comprendidos por el microcontrolador PIC16f877, el cual realiza el proceso de codificación y transmisión de la señal de control hacia los diferentes dispositivos de actuación. Este control se realiza mediante elementos denominados *drivers,* que para la aplicación corresponden a los circuitos integrados L293B.

El motor utilizado, es un motor *paso a paso* debido a características como un control de ángulo sencillo a diferencia de trabajar con motores de corriente continua. El funcionamiento de la aplicación, hace girar al motor un determinado número de grados, en dirección derecha o izquierda dependiendo del comando que pronunciado por el locutor. El diagrama del circuito electrónico se muestra en la figura 69.

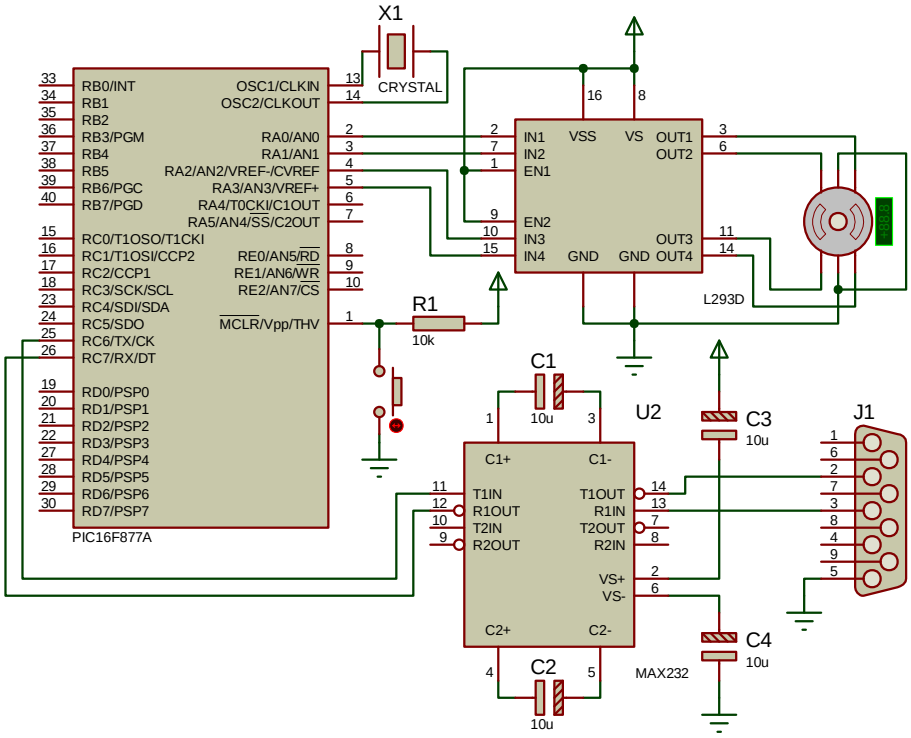


Figura 69. Diagrama del circuito electrónico.

Un ejemplo de un posible comando utilizado por el usuario dentro de la aplicación seria "*Girar 30 grados Izquierda*" en donde el motor giraría 30 grados en sentido contrario a las manecillas de reloj. La tabla 12 muestra una generalización de los comandos implicados para el funcionamiento de la aplicación, y la figura 70 la interfaz que muestra el proceso del control del motor.

Palabras			Función
Acción	**Dirección**	**No de grados**	
Girar	Izquierda	0-360	Girar X grados izquierda
	Derecha		Girar X grados derecha

Tabla 12. Comandos implicados en la aplicación.

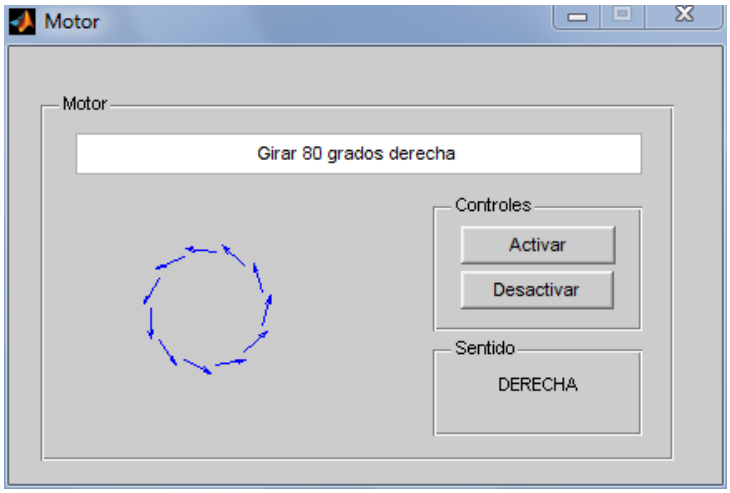


Figura 70. Ventana principal de la aplicación.

8. CONCLUSIONES

Aunque el uso de los Coeficientes Cepstrales en la escala de Mel (MFCC) está bastante extendido dentro del reconocimiento de voz, los Coeficientes de Predicción Lineal (LPC), resultan una solución atractiva para la extracción de características de voz, pues dentro del proyecto se presentaron casos en que el uso de los LPC mejoró significativamente el reconocimiento de voz y redujo el gasto computacional, debido a su simplicidad.

Mediante las pruebas realizadas durante esta investigación se logro determinar la efectividad de las técnicas empleadas para el reconocimiento de voz. Sin embargo, dicha efectividad depende de la aplicación que vaya a hacer uso del reconocimiento de voz, pues a mayor tamaño de vocabulario, el procesamiento y el porcentaje de acierto se ve afectado. En consecuencia, el DTW y el híbrido RNA/DTW presentan un funcionamiento bastante bueno si el vocabulario de la aplicación es reducido y si se trata de sistemas multiusuario. No obstante, para aplicaciones que requieran un vocabulario mayor los HMM presentan un proceder más adecuado, de ahí que su uso este bastante extendido en reconocedores comerciales como el incluido en Windows, desarrollado por Microsoft, o el software Dragon Naturally Speaking, desarrollado por Nuance.

Mediante las pruebas experimentales se determinó que los sistemas neuro-difusos no presentan un buen funcionamiento cuando son aplicadas al reconocimiento de voz. Sin embargo, estos sistemas pueden implementarse para realizar reconocimiento del locutor con dependencia de texto, como también en el tratamiento de imágenes, en donde su utilización podría generan mayores beneficios.

En lo que respecta al proyecto de investigación es el primero encaminado en el área de la lógica difusa y a los sistemas neuro-difusos, dentro de la Universidad de Nariño. Pues es un área relativamente nueva y apenas está siendo utilizada en diversas aplicaciones.

Es uno de los primeros trabajos en los cuales se desarrolla reconocimiento de voz utilizando el .NET Framework desde MATLAB, y en general uno de los primeros en los cuales se utiliza la programación .NET desde el MATLAB, pues la interacción entre estas dos tecnologías aun no está muy desarrollada.

El software desarrollado paso por varios prototipos que hacían uso de diversas técnicas para el reconocimiento voz, técnicas que pueden servir de base para futuros proyectos que involucren cualquier tipo de reconocimiento. Sin embargo, la implementación final del software utiliza la incorporación del .NET Framework para realizar un sistema de reconocimiento robusto e independiente del locutor que es útil para comunicarse con personas sordas.

Como subproducto del trabajo realizado se logró implementar un módulo en el software, diseñado para el reconocimiento a nivel de oraciones, el cual es capaz reconocer oraciones y de traducirlas a lengua de señas, basado en una estructura gramatical simple.

Conforme a los actuales procesos de inclusión de los estudiantes en situación de discapacidad en todos los ámbitos académicos de la Universidad de Nariño, se desarrollo el software LSC, que integra el reconocimiento de voz y la Lengua de Señas Colombiana, como una herramienta de apoyo docente, creando un medio de comunicación con el estudiante con discapacidad auditiva o sordera y que apoya los procesos de integración de la comunidad sorda.

9. RECOMENDACIONES

Ampliar el número de palabras del sistema desarrollado para que sea más robusto e implementar una gramática más compleja para que la traducción a lengua de señas sea de mayor utilidad.

Aplicar las técnicas desarrolladas en esta investigación a sistemas enmarcados en el campo de la domótica y el control de equipo médico especializado, donde podrían ser de gran ayuda, para personas que presenten discapacidad física y para facilitar la utilización del equipo médico, respectivamente.

Posibilidad de trasladar el código a un software de programación libre, como java que permita ampliar el área de distribución del mismo independizándolo de MATLAB y de .NET Framework.

Realizar estudios e investigaciones encuadrados en el área de la lógica difusa y los sistemas Neuro-difusos, que si bien no tuvieron unos buenos resultados en el presente proyecto, aplicados a otras áreas podrían ser de gran utilidad. Pues como se mencionó en el anterior capítulo, esta es la primera investigación realizada sobre esta área dentro de la Universidad de Nariño.

Como la comunicación con personas sordas se realiza en ambos sentidos sería una buena idea desarrollar un proyecto que realice lo contrario a lo plasmado en el proyecto, es decir, que traduzca de lengua de señas a lenguaje verbal.

Teniendo en cuenta las aplicaciones de los sistemas de reconocimiento de voz y las actuales tecnologías que se encuentran en el mercado como los denominados Real-time Target Machine; que son sistemas embebidos que incorporan procesadores Intel Core 2 Duo de 2.13GHz y memorias RAM de 2 GB; se podría Implementar los métodos aplicados en este proyecto haciendo uso de esta tecnología, lo que haría más robusto el sistema y facilitaría su uso en ambientes industriales, educativos, sociales, etc.

Por último continuar con la expansión de este proyecto, implementándolo a gran escala en diferentes centros educativos, lo que podría llegar a facilitar el proceso de inclusión que actualmente está manejando la educación.

BIBLIOGRAFÍA

ABEJÓN, Alejandro. Reconocimiento de idioma en voz espontanea mediante reconocimiento fonético multilingüe en paralelo y modelado estadístico del idioma. Universidad autónoma de Madrid. 2007.

COLOMBIA APRENDE. Lengua de señas colombiana. [En Línea]. Disponible en Internet: http://mail.colombiaaprende.edu.co:8080/recursos/lengua_senas/

ESTEVE, Cristina. Reconocimiento de locutor dependiente de texto mediante adaptación de modelos ocultos de Markov fonéticos. Universidad autónoma de Madrid. 2007.

ETTER, Delores M. Solución de problemas de ingeniería con MATLAB. Segunda edición. PRENTICE HALL . 1997.

FRANCO, Jorge Andrés. Reconocimiento de voz para niños con discapacidad en el habla. Universidad de las Américas Puebla. 2004.

GÓMEZ, J.C. Reconocimiento Automático de Voz basado en HMM. Articulo. 2007. [En línea], http://www.fceia.unr.edu.ar/prodivoz/apuntes.html. [Argentina], 2009. Disponible en: http://www.fceia.unr.edu.ar/prodivoz/RAV_HMM_bw.pdf

GRANADOS, Jorge. Redes neuronales. Introducción a la computación. Universidad de la República. 1996.

GUSTAVO, Romero. Sistema de reconocimiento automático del habla. Argentina: Universidad Nacional de Entre Ríos 2001.

HERNANDO, Francisco Javier. Técnicas de procesado y representación de la señal de voz para el reconocimiento del habla en ambientes ruidosos. Universidad politécnica de Cataluña. Departamento de teoría de la señal y comunicaciones. 1993.

JUANG, B. y RABINER, L. Hidden Markov Models for Speech Recognition. Technometrics, Vol. 33, No. 3. 1991.

MARTIN DEL BRIO, Bonifacio y MOLINA, Alfredo. Redes Neuronales y Sistemas Borrosos. 3ra edición 2007.

MIYARA, Federico. Acustica del tracto vocal. Articulo. 2001 [En línea], http://www.fceia.unr.edu.ar/prodivoz/apuntes.html. [Argentina], 2009. Disponible en: http://www.fceia.unr.edu.ar/prodivoz/fonatorio.pdf

MIYARA, Federico. Alineación temporal para el reconocimiento de palabras. 2002. [En línea], http://www.fceia.unr.edu.ar/prodivoz/apuntes.html. [Argentina], 2009. Disponible en: http://www.fceia.unr.edu.ar/prodivoz/alinea.pdf

NILSSON, Mikael y EJNARSSON, Marcus. Speech Recognition using Hidden Markov Model. Degree of Master of Science in Electrical Engineering. Department of Telecommunications and Signal Processing. Blekinge Institute of Technology. 2002.

OLABE, Xavier Bosogain. Redes neuronales artificiales y sus aplicaciones. Escuela superior de ingeniería de Bilbao. Departamento se sistemas y automática. 2008.

OROPEZA, José Luis. Algoritmos y métodos para el reconocimiento de voz en español mediante silabas. Centro de investigación en computación-IPN. Computación y Sistemas Vol. 9. México D.F. 2006.

PLAZA, José Leonardo. Reconocimiento de voz para personas con discapacidad en el habla. Universidad de las Américas Puebla. 2004.

PUYIN, Liu y HONGXING, Li .Fuzzy Neural Network Theory And Application. Series in Machine Perception and Artificial Intelligence. Vol. 59. 2004.

RABINER, Lawrence. A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE, Vol. 77, No 2. 1989.

SANDMANN, H. Predição não-Linear de séries temporais usando sistemas de arquitectura neuro-fuzzy. Tesis de Maestría. Universidad de São Paulo. 2006.

SERRANO, Antonio J., SORIA, Emilio y MARTIN, José D. Redes Neuronales Artificiales. Programa 3er ciclo doctorado. Universidad de Valencia. Curso 2009 2010.

VILLAMI, Iván Horacio. Aplicaciones en reconocimiento de voz utilizando HTK. Pontificia universidad javeriana. Santa fe de Bogotá DC. 2005.

WIKIPEDIA. Lengua De Señas. [En Línea] Disponible en Internet: http://es.wikipedia.org/wiki/Lengua_de_se%C3%B1as

ANEXOS

ANEXO A. CÓDIGO FUENTE DEL MICROCONTROLADOR

```
#include <16F877A.h> // PIC a usar
#fuses XT,NOWDT,NOPROTECT,NOLVP
#use delay(clock=4000000) // Cristal
#use rs232(baud=2400, xmit=PIN_C6, rcv=PIN_C7,bits=8) // configuracion
serial
#BYTE TRISA = 0x85 // Configura Puerto A
#BYTE PORTA = 0x05 // Habilita Puerto A
INT8 valor; // variable para recepcion serial
INT8 mH; // motores referencia
INT8 n; // Incremento para el for
INT16 pase,pase1,pase2,pase3; // valor de pasos
float angulo=3.75; // angulo por paso
INT16 t=100; // Tiempo
INT8 estado; // estado del puerto

void der() // Rutina de giro hacia la derecha
{
   switch (estado)
   {
      case 3:estado=2;break;
      case 2:estado=6;break;
      case 6:estado=4;break;
      case 4:estado=12;break;
      case 12:estado=8;break;
      case 8:estado=9;break;
      case 9:estado=1;break;
      case 1:estado=3;break;
      return;
   }
}

void izq() // Rutina de giro hacia la izquierda
{
   switch (estado)
   {
      case 3:estado=1;break;
      case 1:estado=9;break;
      case 9:estado=8;break;
      case 8:estado=12;break;
      case 12:estado=4;break;
      case 4:estado=6;break;
      case 6:estado=2;break;
      case 2:estado=3;break;
      return;
   }
}

void main()
{
   TRISA = 0x00; // Puerto A como salida
```

```
   mH=0x09; // motor
   PORTA = mH; // Inicia el puerto A en un estado

   while(true) // mientras se ejecute
   {
      valor = getch(); // Obtiene valor del serial
      switch (valor) // Seleccion segun valor
         {
            case 'd': // giro derecha
            estado=mH;
            pase1 = getch(); // Obtiene valor del serial
            pase2 = getch(); // Obtiene valor del serial
            pase3 = getch(); // Obtiene valor del serial
            pase = ((pase1-48)*100 + (pase2-48)*10 + pase3-48)/angulo; //
numero de pasos
            for (n=1;n<=pase;n++)
            {
               der();
               mH=estado;
               portA=mH;
               delay_ms(t);
            }
            break;

            case 'i': // giro izquierda
            estado=mH;
            pase1 = getch(); // Obtiene valor del serial
            pase2 = getch(); // Obtiene valor del serial
            pase3 = getch(); // Obtiene valor del serial
            pase = ((pase1-48)*100 + (pase2-48)*10 + pase3-48)/angulo; //
numero de pasos
            for (n=1;n<=pase;n++)
            {
               izq();
               mH=estado;
               portA=mH;
               delay_ms(t);
            }
            break;
         }
      }
   }
```

ANEXO B. CÓDIGO FUENTE DEL PROGRAMA DE RECONOCIMIENTO – VENTANA PRINCIPAL

```
function varargout = LSC(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @LSC_OpeningFcn, ...
                   'gui_OutputFcn',  @LSC_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

function LSC_OpeningFcn(hObject, eventdata, handles, varargin)
NET.addAssembly('System')
NET.addAssembly('System.Xml')
NET.addAssembly('System.Speech')
NET.addAssembly('System.Windows.Forms')
NET.addAssembly('System.Runtime.Serialization.Formatters.Soap')

import System.*
import System.IO.*
import System.Xml.*
import System.Speech.*
import System.Windows.Forms.*
import System.Runtime.Serialization.*
import System.Runtime.Serialization.Formatters.Binary.*
set(gcf,'CloseRequestFcn',{@CloseFigure,handles})
global recognizer
choices=System.Speech.Recognition.Choices();
stream = File.Open('diccionario.dat', FileMode.Open);
formatter=BinaryFormatter();
obj1=formatter.Deserialize(stream);
stream.Close();
choices.Add(obj1)
grambuil=System.Speech.Recognition.GrammarBuilder(choices);
grammar=System.Speech.Recognition.Grammar(grambuil);
recognizer=System.Speech.Recognition.SpeechRecognitionEngine();
recognizer.UnloadAllGrammars();
recognizer.LoadGrammar(grammar);
recognizer.SetInputToDefaultAudioDevice();
recognizer.RecognizeAsync(System.Speech.Recognition.RecognizeMode.Multipl
e)
```

```
h=event.listener (recognizer, 'SpeechRecognized', ...
                  @(hObject, eventdata)LSC_reconocedor1...
                  (hObject, eventdata, handles));
H=h;
handles.LSC_reconocedor1=H;
i=event.listener (recognizer, 'SpeechRecognitionRejected', ...
                  @LSC_reconocedor2);
I=i;
handles.LSC_reconocedor2=I;
utpl=recognizer;
handles.recognizer=utpl;
global cadena M
cadena={};
M=1;
% Choose default command line output for LSC
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

function varargout = LSC_OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;


function LSC_reconocedor1(hObject, eventdata, handles)
import System.*
import System.IO.*
import System.Xml.*
import System.Windows.Forms.*
import System.Speech.Recognition.*
import System.Runtime.Serialization.*
import System.Runtime.Serialization.Formatters.Binary.*
global cadena M
xDoc=System.Xml.XmlDocument();
disp(eventdata.Result.Text)
g=eventdata.Result.Text.ToString;
G=char(g); set(handles.edit1,'String',G);
try
    archivo=['D:\Pruebas Matlab y .NET\Otro1\LCS\Video\' G '.swf'];
    file=['D:\Pruebas Matlab y .NET\Otro1\LCS\Video\' G '_config.xml'];
    xDoc.Load(file);
    Tiempo1=xDoc.GetElementsByTagName('video1').ItemOf...
        (1).ChildNodes.Item(0).InnerText;
    Tiempo=str2double(char(Tiempo1));
    set(handles.activex3,'Movie',archivo)
    t=timer('TimerFcn', 'stat=false;','StartDelay',Tiempo);
    start(t); waitfor(t); stop(t); delete(t);
    handles.activex3.Rewind()
catch ME1
    disp(ME1)
    disp('Archivo NO encontrado')
end
M=M+1;
```

```
cadena{M}=G;
set(handles.popupmenu1,'String',cadena);
if M==6
    for i=2:5
        cadena{i}=cadena{i+1};
        M=5;
    end
end


function LSC_reconocedor2(hObject, eventdata, handles)
disp('palabra no reconocida')
set(handles.edit1,'String','Palabra no Reconocida');


function pushbutton1_Callback(hObject, eventdata, handles)
import System.Xml.*
xDoc=System.Xml.XmlDocument();
strArr = NET.createArray('System.String', 1);
pala=get(handles.edit1,'String');
strArr.Set(0,pala)
obj=strArr.Get(0);
obj=obj.ToUpper;
arr=obj.ToCharArray(0, obj.Length);
L=arr.Length;
for X=0:L-1
    letra=arr.Get(X);
    try
        archivo=['D:\Pruebas Matlab y .NET\Otro1\LCS\Video\' letra
'.swf'];
        file=['D:\Pruebas Matlab y .NET\Otro1\LCS\Video\' letra
'_config.xml'];
        xDoc.Load(file);
        Tiempo1=xDoc.GetElementsByTagName('video1').ItemOf...
            (1).ChildNodes.Item(0).InnerText;
        Tiempo=str2double(char(Tiempo1));
        set(handles.activex3,'Movie',archivo)
        handles.activex3.Play()
        letra1=char(letra); set(handles.edit2,'String',letra1);
        t=timer('TimerFcn', 'stat=false;','StartDelay',Tiempo);
        start(t); waitfor(t); stop(t); delete(t);
        handles.activex3.Rewind()
    catch ME2
        disp(ME2)
        disp('Letra NO Encontrada')
    end

end

function pushbutton4_Callback(hObject, eventdata, handles)
handles.activex3.Zoom(96)

function pushbutton5_Callback(hObject, eventdata, handles)
handles.activex3.Zoom(100)
```

```
function pushbutton6_Callback(hObject, eventdata, handles)
import System.*
import System.IO.*
import System.Xml.*
import System.Windows.Forms.*
import System.Speech.Recognition.*
import System.Runtime.Serialization.*
import System.Runtime.Serialization.Formatters.Binary.*
nombre_arch=get(handles.edit1,'String');
file=['D:\Pruebas Matlab y .NET\Otro1\Diccionario\' nombre_arch '.txt'];
d=strcmp(file,'D:\Pruebas Matlab y .NET\Otro1\Diccionario\.txt');
if d==1
    return
end
fileName=file;
stream=FileStream(fileName, FileMode.Open, FileAccess.Read);
reader=StreamReader(stream, System.Text.Encoding.Default);
while(reader.Peek() > -1)
    base=reader.ReadToEnd();
    obj=base;
end
reader.Close();
obj1=obj.ToString;
obj2=char(obj1);
set(handles.edit3,'String',obj2)


function popupmenu1_Callback(hObject, eventdata, handles)
import System.Xml.*

valor=get(handles.popupmenu1,'Value');
cadena=get(handles.popupmenu1,'String');
F=cadena{valor};
if F==0
    return
end
set(handles.edit1,'String',F)
xDoc=System.Xml.XmlDocument();
try
    archivo=['D:\Pruebas Matlab y .NET\Otro1\LCS\Video\' F '.swf'];
    file=['D:\Pruebas Matlab y .NET\Otro1\LCS\Video\' F '_config.xml'];
    xDoc.Load(file);
    Tiempo1=xDoc.GetElementsByTagName('video1').ItemOf...
        (1).ChildNodes.Item(0).InnerText;
    Tiempo=str2double(char(Tiempo1));
    set(handles.activex3,'Movie',archivo)
    handles.activex3.Play()
    t=timer('TimerFcn', 'stat=false;','StartDelay',Tiempo);
    start(t); waitfor(t); stop(t); delete(t);
    handles.activex3.Rewind()
catch ME3
    disp(ME3)
    disp('Archivo NO encontrado')
```

```
end

function popupmenu1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function pushbutton7_Callback(hObject, eventdata, handles)
xDoc1=System.Xml.XmlDocument();
H=get(handles.edit1,'String');
try
    archivo=['D:\Pruebas Matlab y .NET\Otro1\LCS\Video\' H '.swf'];
    file=['D:\Pruebas Matlab y .NET\Otro1\LCS\Video\' H '_config.xml'];
    xDoc1.Load(file);
    Tiempo1=xDoc1.GetElementsByTagName('video1').ItemOf...
        (1).ChildNodes.Item(0).InnerText;
    Tiempo=str2double(char(Tiempo1));
    set(handles.activex3,'Movie',archivo)
    handles.activex3.Play()
    t=timer('TimerFcn', 'stat=false;','StartDelay',Tiempo);
    start(t); waitfor(t); stop(t); delete(t);
    handles.activex3.Rewind()
catch ME4
    disp(ME4)
    disp('Archivo NO encontrado')
end


function Oracion_Callback(hObject, eventdata, handles)
global recognizer
recognizer.RecognizeAsyncStop;
LSCO


function Salir_Callback(hObject, eventdata, handles)
closereq;


function CloseFigure(src,event,handles)
opcion =questdlg('¿Desea Salir del Programa?','Salir','Si','No','No');
if strcmp(opcion,'No')
    return
end
closereq;
```

ANEXO C. CÓDIGO FUENTE DEL PROGRAMA DE RECONOCIMIENTO – VENTANA AUXILIAR

```
function varargout = LSCO(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @LSCO_OpeningFcn, ...
                   'gui_OutputFcn',  @LSCO_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


function LSCO_OpeningFcn(hObject, eventdata, handles, varargin)
NET.addAssembly('System')
NET.addAssembly('System.Xml')
NET.addAssembly('System.Speech')
NET.addAssembly('System.Windows.Forms')
NET.addAssembly('System.Runtime.Serialization.Formatters.Soap')

import System.*
import System.IO.*
import System.Xml.*
import System.Speech.*
import System.Windows.Forms.*
import System.Runtime.Serialization.*
import System.Runtime.Serialization.Formatters.Binary.*
set(gcf,'CloseRequestFcn',{@CloseFigure,handles})
gram=System.Speech.Recognition.DictationGrammar();
recognizerO=System.Speech.Recognition.SpeechRecognitionEngine;
recognizerO.LoadGrammar(gram);
recognizerO.SetInputToDefaultAudioDevice();
recognizerO.RecognizeAsync(System.Speech.Recognition.RecognizeMode.Single
);
utpl=recognizerO;
handles.recognizerO=utpl;
h=event.listener (recognizerO, 'SpeechRecognized', ...
                  @(hObject, eventdata)LSCO_reconocedor1...
                  (hObject, eventdata, handles));
H=h;
handles.LSCO_reconocedor1=H;
```

```
i=event.listener (recognizerO, 'SpeechRecognitionRejected', ...
                  @LSCO_reconocedor2);
I=i;
handles.LSCO_reconocedor2=I;
set(handles.pushbutton6,'Enable','off')
set(handles.pushbutton8,'Enable','off')
set(handles.text1,'String','Reconociendo...')
% Choose default command line output for LSCO
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

function varargout = LSCO_OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;

function LSCO_reconocedor1(hObject, eventdata, handles)
import System.*
import System.IO.*
import System.Xml.*
import System.Windows.Forms.*
import System.Speech.Recognition.*
import System.Runtime.Serialization.*
import System.Runtime.Serialization.Formatters.Binary.*
xDoc=System.Xml.XmlDocument();
disp(eventdata.Result.Text)
g=eventdata.Result.Text.ToString;
G=char(g); set(handles.edit1,'String',G);
s=grammatica(g);
glosa=s.ToUpper;
S=char(glosa);
set(handles.edit2,'String',S)
set(handles.text1,'String','Procesando...')
Limite=char(' ');
a=s.Split(Limite);
b=a.Length;
for k=0:b-1
    palabra=a.Get(k);
    e=palabra.ToUpper;
    E=char(e); set(handles.edit3,'String',E);
    try
        set(handles.edit4,'String','Mostrar Video');
        archivo=['D:\Pruebas Matlab y .NET\Otro1\LCS\Video\' E '.swf'];
        file=['D:\Pruebas Matlab y .NET\Otro1\LCS\Video\' E
'_config.xml'];
        xDoc.Load(file);
        Tiempo1=xDoc.GetElementsByTagName('video1').ItemOf...
            (1).ChildNodes.Item(0).InnerText;
        Tiempo=str2double(char(Tiempo1));
        set(handles.activex3,'Movie',archivo)
        t=timer('TimerFcn', 'stat=false;','StartDelay',Tiempo);
        start(t); waitfor(t); stop(t); delete(t);
        handles.activex3.Rewind()
```

```
    catch ME1
        set(handles.edit4,'String','Deletrear');
        set(handles.edit5,'Enable','inactive')
        arr=e.ToCharArray(0, e.Length);
        L=arr.Length;
        for X=0:L-1
            letra=arr.Get(X);
            set(handles.edit5,'String',letra)
            try
                archivo=['D:\Pruebas Matlab y .NET\Otro1\LCS\Video\'
letra '.swf'];
                file=['D:\Pruebas Matlab y .NET\Otro1\LCS\Video\' letra
'_config.xml'];
                xDoc.Load(file);
                Tiempo1=xDoc.GetElementsByTagName('video1').ItemOf...
                    (1).ChildNodes.Item(0).InnerText;
                Tiempo=str2double(char(Tiempo1));
                set(handles.activex3,'Movie',archivo)
                handles.activex3.Play()
                t=timer('TimerFcn', 'stat=false;','StartDelay',Tiempo);
                start(t); waitfor(t); stop(t); delete(t);
                handles.activex3.Rewind()
            catch ME2
                disp('Letra NO Encontrada')
            end
        end
        set(handles.edit5,'Enable','off')
    end
end
set(handles.pushbutton6,'Enable','on')
set(handles.pushbutton8,'Enable','on')
set(handles.edit4,'String','');
set(handles.text1,'String','Pausa...')

function LSCO_reconocedor2(hObject, eventdata, handles)
import System.*
import System.IO.*
import System.Windows.Forms.*
import System.Speech.Recognition.*
import System.Runtime.Serialization.*
import System.Runtime.Serialization.Formatters.Binary.*
recognizerO=handles.recognizerO;
disp('palabra no reconocida')
recognizerO.RecognizeAsync();
set(handles.text1,'String','Reconociendo...')
set(handles.pushbutton6,'Enable','off')
set(handles.pushbutton8,'Enable','off')

function pushbutton1_Callback(hObject, eventdata, handles)
handles.activex3.Zoom(100)


function pushbutton2_Callback(hObject, eventdata, handles)
xDoc=System.Xml.XmlDocument();
```

```
set(handles.text1,'String','Procesando...')
s=get(handles.edit2,'String');
strArr = NET.createArray('System.String',1);
strArr.Set(0,s);
s=strArr.Get(0);
Limite=char(' ');
a=s.Split(Limite);
b=a.Length;
for k=0:b-1
    palabra=a.Get(k);
    e=palabra.ToUpper;
    E=char(e); set(handles.edit3,'String',E);
    try
        set(handles.edit4,'String','Mostrar Video');
        archivo=['D:\Pruebas Matlab y .NET\Otro1\LCS\Video\' E '.swf'];
        file=['D:\Pruebas Matlab y .NET\Otro1\LCS\Video\' E
'_config.xml'];
        xDoc.Load(file);
        Tiempo1=xDoc.GetElementsByTagName('video1').ItemOf...
            (1).ChildNodes.Item(0).InnerText;
        Tiempo=str2double(char(Tiempo1));
        set(handles.activex3,'Movie',archivo)
        handles.activex3.Play()
        t=timer('TimerFcn', 'stat=false;','StartDelay',Tiempo);
        start(t); waitfor(t); stop(t); delete(t);
        handles.activex3.Rewind()
    catch ME1
        set(handles.edit4,'String','Deletrear');
        arr=e.ToCharArray(0, e.Length);
        L=arr.Length;
        for X=0:L-1
            letra=arr.Get(X);
            try
                archivo=['D:\Pruebas Matlab y .NET\Otro1\LCS\Video\'
letra '.swf'];
                file=['D:\Pruebas Matlab y .NET\Otro1\LCS\Video\' letra
'_config.xml'];
                xDoc.Load(file);
                Tiempo1=xDoc.GetElementsByTagName('video1').ItemOf...
                    (1).ChildNodes.Item(0).InnerText;
                Tiempo=str2double(char(Tiempo1));
                set(handles.activex3,'Movie',archivo)
                handles.activex3.Play()
                t=timer('TimerFcn', 'stat=false;','StartDelay',Tiempo);
                start(t); waitfor(t); stop(t); delete(t);
                handles.activex3.Rewind()
            catch ME2
                disp('Letra NO Encontrada')
            end
        end
    end
end
```

```
function pushbutton4_Callback(hObject, eventdata, handles)
handles.activex3.Zoom(96)


function pushbutton6_Callback(hObject, eventdata, handles)
import System.*
import System.IO.*
import System.Windows.Forms.*
import System.Speech.Recognition.*
import System.Runtime.Serialization.*
import System.Runtime.Serialization.Formatters.Binary.*
recognizerO=handles.recognizerO;
set(handles.edit1,'String','');
set(handles.edit2,'String','');
recognizerO.RecognizeAsyncCancel
recognizerO.SetInputToDefaultAudioDevice();
recognizerO.RecognizeAsync();
set(handles.text1,'String','Reconociendo...')
set(handles.pushbutton6,'Enable','off')
set(handles.pushbutton8,'Enable','off')


function pushbutton8_Callback(hObject, eventdata, handles)
recognizerO=handles.recognizerO;
[Filename Path]=uigetfile({'*.wav'},'Abrir');
wavfile=strcat(Path,Filename);
recognizerO.SetInputToWaveFile(wavfile);
recognizerO.RecognizeAsync();
set(handles.pushbutton8,'Enable','off')

function CloseFigure(src,event,handles)
global recognizer
recognizer.RecognizeAsync(System.Speech.Recognition.RecognizeMode.Multipl
e)
closereq;
```

ANEXO D. CÓDIGO FUENTE DE LAS SUBFUNCIONES

- Gramática

```
function [ OUT ] = grammatica( O )
g=O;
NET.addAssembly('System'); import System.*; import System.IO.*
NET.addAssembly('System.Xml'); import System.Xml.*
NET.addAssembly('System.Speech'); import System.Speech.Recognition.*
NET.addAssembly('System.Windows.Forms'); import System.Windows.Forms.*
NET.addAssembly('System.Runtime.Serialization.Formatters.Soap')
import System.Runtime.Serialization.*
import System.Runtime.Serialization.Formatters.Binary.*
oracion=ordenar(g); oracion=char(oracion);
estado,sinonimos] = diccionario(oracion);
logica=cellfun('isempty',sinonimos);
m=length(sinonimos(:,1)); n=length(sinonimos);
stream = File.Open('diccionario1.dat', FileMode.Open);
formatter=BinaryFormatter(); obj=formatter.Deserialize(stream);
stream.Close(); base=String.Concat(obj);
base=base.ToLower; strArr = NET.createArray('System.String', m);
sb=System.Text.StringBuilder();
for M=1:m
    for N=1:n
        resultado=sinonimos(M,N); resultadoL=logica(M,N);
        if resultadoL==1 || N==n
            resultado=sinonimos(M,1);
            resultado=char(resultado);
            strArr.Set(M-1,resultado);
            resultado=strArr.Get(M-1);
            sb.Append(resultado);
            sb.Append(' ');
            break
        else
            resultado=char(resultado);
            strArr.Set(M-1,resultado)
            resultado=strArr.Get(M-1);
            resultado1=resultado.Insert(resultado.Length,' ');
            if base.Contains(resultado1)
                sb.Append(resultado);
                sb.Append(' ');
                break
            else
                continue
            end
        end

    end
end
str = sb.ToString();
str=ordenar1(str);
OUT=str;
end
```

- Diccionario

```
function [estado,varargout] = diccionario(texto)
k=1;
temp='';
for n=1:length(texto);
    if ~isspace(texto(n))
        temp = [temp texto(n)];
    else
        if ~isspace(texto(n-1))
            words{k} = temp;
        end
        temp='';
        k=k+1;
    end
end
words{k} = temp;

Doc = actxserver('Word.Application');
m=1;
for n=1:length(words);
    if ~isempty(words{n})
        estado(m) = invoke(Doc,'CheckSpelling',words{n},[],1,1);
        if nargout==2
            X = invoke(Doc,'SynonymInfo',words{n});
            sinonimos = get(X,'MeaningList');
            Significados{m,1} = words{n};
            if isempty(sinonimos) && estado(m)==1
                Significados{m,2} = '';
            elseif estado(m)==0
                Significados{m,2} = '';
            else
                for k=2:length(sinonimos)+1
                    Significados{m,k} = sinonimos{k-1};
                end
            end
        end
        m=m+1;
    end
end

if exist('Significados','var')
    varargout = {Significados};
end
estado = all(estado);

invoke(Doc,'Quit');
delete(Doc);
```