UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO
FACULTAD DE INGENIERÍA ELÉCTRICA

MÉTODOS PARA EL RECONOCIMIENTO DE VOZ UTILIZANDO MODELOS OCULTOS DE MARKOV Y DTW

TESIS

Que para obtener el grado de
INGENIERO EN COMPUTACIÓN

presenta

C. José María Valencia Ramírez

-

Dr. José Antonio Camarena Ibarrola

Director de Tesis

Abril 2013

Deseo expresar mis agradecimientos a:

A mi asesor José Antonio Camarena Ibarrola por el apoyo brindado con el cual he podido culminar con esta tesis, gracias a su orientación, aportación y asesoramiento.

A mis hermanos Luciana, Hortencia, Hector, María de Jesús, Baltazar y Moisés por su vital apoyo, gracias por creer en mí, comparto con ellos este logro.

A mi amada esposa María Lucero Martínez Ledezma, por quererme, apoyarme siempre, por su gran paciencia y creer en mí.

Un agradecimiento muy especial a mi madre Silvina Ramírez Gaytan, por su apoyo incondicional, comprensión, cariño y confianza brindada para llegar a lograr esta meta. Agradezco infinitamente su esfuerzo por educarme y formarme. Mamá gracias por darme una carrera para mi futuro, todo esto te lo debo a ti.

A mis amigos, que nos apoyamos mutuamente en nuestra formación profesional y que hasta ahora, seguimos siendo amigos.

Resumen

El análisis de la señal de voz y su posterior reconocimiento deben superar diversos problemas que para los seres humanos son triviales, algunos de estos problemas son, la correcta elección y extracción de las características de la señal de voz, tamaños de los vocabularios a reconocer, ruido y distorsión del entorno.

En el presente trabajo se implementaron los siguientes métodos de extracción de características: Energía por banda (Espectrogramas), MFCC (Mel-frequency Cepstral Coefficients) y Codificación Lineal Predictiva (LPC, Linear Prediction Coefficients). Las técnicas de reconocimiento que se utilizaron son, Dynamic Time Warping (DTW) y Discrete Hidden Markov Models (DHMM) utilizando las distancias coseno, euclidiana e Itakura. Se utilizaron las curvas ROC (Receiver Operating Characteristic) para obtener cual de los métodos de extracción de características es el mejor en el reconocimiento de palabras aisladas, utilizando un diccionario de 100 palabras monolocutor muestreada a 8000.0 Hz cuantizado a 16 bits, monocanal, little-endian. Para la implementación de Espectrogramas se utilizaron las primeras 18 bandas críticas de la escala de Bark, para MFCC se utilizaron 16 filtros de Mel y para LPC se utilizo un predictor de orden 16.

De las pruebas realizadas se obtuvieron los siguientes resultados para las técnicas de reconocimiento empleadas: utilizando DTW, el mejor sistema de reconocimiento de palabras aisladas es LPC como método para extraer características, el área bajo su curva ROC obtenida fue de: 0.988 usando la distancia de Itakura, 0.995 usando la distancia coseno y 0.991 usando la distancia euclidiana (considerando un área de 1.0 para un sistema perfecto). En cuanto a DHMM el mejor sistema de reconocimiento de palabras aisladas fue MFCC utilizando un modelo de 9 estados y 64 símbolos de observaciones usando la distancia coseno en la etapa de cuantización vectorial y en la creación del codebook obteniendo una área bajo su curva de 0.96741. De donde puede apreciarse que los resultados obtenidos por DTW fueron superiores.

Contenido

Lista de Figuras

Lista de Tablas

Lista de Símbolos

$x(n)$	Secuencia de números.
$y(n)$	Secuencia de números.
$x_a(t)$	Forma de onda analógica.
$x_a(nT)$	Secuencia de muestras tomadas periódicamente de una señal analógica.
T	Periodo de una señal.
$\delta(n)$	Impulso unitario.
$u(n)$	Escalón unitario.
ω_0	Frecuencia angular.
r	Modulo de un número complejo.
$h(n)$	Respuesta de un sistema al impulso unitario.
$*$	Símbolo de convolución.
E_n	Energía de tiempo corto.
N	Tamaño de una secuencia de símbolos.
$w(n)$	Función de una ventana.
f	Frecuencia de una señal.
f_s	Frecuencia de muestreo.
Z_n	Régimen de cruces por cero de tiempo corto.
$\{v_i\}$	Conjunto de los posible valores que pueden tomar las muestras de una señal de audio.
I	Información propia.
p_i	Probabilidad de un evento.
$P(v_i)$	Probabilidad del valor v_i.
$I(p_i)$	Información propia del valor v_i.
H	Entropía de una señal de voz.
H_{min}	Entropía mínima de una señal de voz.
H_{max}	Entropía máxima de una señal de voz.
δ^2	Varianza.
$\bar{x}$	Media de $x(n)$.
$\phi(k)$	Autocorrelación de una señal discreta.
$R_n(k)$	Autocorrelación de tiempo corto de una señal discreta.
$\hat{R}_n(k)$	Autocorrelación Modificada de tiempo corto de una señal discreta.
$h_k(n)$	Multiplicación de una ventana consigo misma trasladada k posiciones.
$\mathcal{F}$	Transformada de Fourier.

$X(f)$	Transformada Continua de Fourier de $x_a(t)$.
$X(k)$	Transformada Discreta de Fourier de $x(n)$.
$Y(k)$	Transformada Discreta de Fourier de $y(n)$.
W_T	Exponencial compleja con periodo T.
$x^*(n)$	Conjugado de $x(n)$.
$X^*(k)$	Conjugado de $X(k)$.
f_β	Frecuencia en Barks.
ϖ	Frecuencia en Mels.
$f_c(m)$	Frecuencia central en Hertz del m-ésimo filtro de Mel.
$H(k,m)$	Banco de filtros de Mel.
M	Número de filtros que conforman el banco de filtros de Mel.
$\Delta\varpi$	Separación entre cada filtro triangular de Mel.
f_{max}	Frecuencia máxima de la señal bajo análisis.
f_{min}	Frecuencia mínima de la señal bajo análisis.
ϖ_{max}	Límites del rango de frecuencias de la señal bajo análisis correspondientes a f_{max}.
ϖ_{min}	Límites del rango de frecuencias de la señal bajo análisis correspondientes a f_{min}.
$\varpi_c(m)$	Frecuencia central en Mels del m-ésimo filtro de Mel.
$\|X(k)\|$	Magnitud de $X(k)$.
$X'(m)$	$\|X(k)\|$ escalado logarítmicamente tanto en frecuencia como en magnitud.
$\ll$	Símbolo matemático "mucho menor que".
$c(l)$	l-ésimo coeficiente cepstral de Mel.
$S(z)$	Transformada Z de la respuesta del filtro digital.
$U(z)$	Transformada Z de la señal de entrada al filtro digital.
G	Ganancia del filtro digital.
a_k	Coeficiente k de la señal de voz.
α_k	Coeficiente k del LP.
$s(n)$	Muestras reales de voz.
$\tilde{s}(n)$	Muestras obtenidas por el predictor.
$u_e(n)$	Señal de excitación del sistema.
$e(n)$	Error de predicción.
$s_n(m)$	m-ésima muestra contada desde el inicio del marco que comienza en la n-ésima muestra, es decir, $s(n+m)$.
$e_n^2(m)$	Error de predicción del marco que comienza en la n-ésima muestra al cuadrado, es decir, $e^2(n+m)$.
$E(n)$	Error de predicción de tiempo corto.
p	Orden del LP.
$\phi_n(i,k)$	Sumatoria del producto de $s_n(m-i)$ y $s_n(m-k)$.
τ	Variable auxiliar para realizar un cambio de variable.
Λ	Vector de LPC.
R	Matriz de autocorrelaciones de $(p)(p)$.
R_v	Vector de autocorrelaciones $R_n(1), R_n(2), \ldots, R_n(p)$.
$\alpha_i^{(p)}$	i-ésimo coeficiente de un predictor de orden p.

$E_n^{(p)}$	Error de predicción de tiempo corto de un predictor de orden p.
k_i	i-ésimo coeficiente PARCOR (PARtial CORrelation).
V_n	Error de predicción medio cuadrático normalizado.
$\hat{\Lambda}$	Vector de LPC observados.
$D(\Lambda, \hat{\Lambda})$	Distancia de Itakura entre el vector Λ y $\hat{\Lambda}$.
$\vec{x}$	Vector de tamaño N.
$\vec{y}$	Vector de tamaño N.
θ	Menor ángulo que hay entre los vectores $\vec{x}$ y $\vec{y}$.
$d_e(\vec{x}, \vec{y})$	Distancia euclidiana entre el vector $\vec{x}$ y $\vec{y}$.
$d_c(\vec{x}, \vec{y})$	Distancia coseno entre el vector $\vec{x}$ y $\vec{y}$.
$\in$	Símbolo matemático "pertenencia de conjuntos", se lee "en; está en; es elemento de; es miembro de; pertenece a".
$\mathbb{N}$	Conjunto de los números naturales.
$\wp(n)$	Función de doblado.
$d_{n,m}$	Distancia entre el vector de características correspondiente al marco n de la elocución $x(n)$, y el vector de características correspondiente al marco m de la elocución $y(m)$.
$D_{n,m}$	Distancia acumulada de DTW.
Γ	Conjunto de tiempos.
$\mho$	Espacio muestral.
$X(t, \dot{s})$	Familia de variables aleatorias, donde $t \in \Gamma$ y $\dot{s} \in \mho$.
N_s	Número de estados de un HMM.
S	Conjunto finito de N_s estados.
N_o	Número de posibles simbolos en un HMM.
q_t	Estado en el tiempo t.
Ω	Conjunto finito de simbolos de observaciones $\{o_1, o_2, \ldots, o_{N_o}\}$.
$a_{i,j}$	Probabilidad de que el sistema estando en el estado s_i cambie al estado s_j.
A	Conjunto de probabilidades de transición entre estados de un HMM; $\{a_{i,j}\}$.
$P(q_t = s_j \vert \text{Œ})$	Probabilidad que en el tiempo t, se este en el estado s_j; dado el conjunto de eventos Œ.
Π	Distribución de probabilidad de estado iniciales de un HMM; $\{\pi_i\}$.
$b_j(o_t)$	Probabilidad de la observación o en el instante t, estando en el estado s_j.
B	Probabilidades de generación de observaciones de un HMM; $\{b_j(k)\}$.
O	Secuencia de observaciones; $\{o_t\}$; $t = 1, 2, \ldots, \Gamma$.
o_t	valor de la observación en el instante t, correspondiente a la secuencia de observaciones O.
$\vert\Gamma\vert$	Cardinalidad de Γ, es decir, el número de elementos de Γ.
λ	Modelo Oculto de Markov determinado por (Π, A, B).
$\bar{V}$	Conjunto discreto y finito de símbolos llamado alfabeto; $\{\bar{v}_k\}$.
N_a	Tamaño del alfabeto $\bar{V}$.

Q	Secuencia de estados; $\{q_t\}$.
$\alpha_t(i)$	Variable Forward.
$\beta_t(i)$	Variable Backward.
Q^*	Secuencia de estados más probable; $\{q_t^*\}$.
$P[q, O]$	Probabilidad conjunta.
$P[q\|O]$	Probabilidad condicional.
$\delta_t(i)$	Variable Viterbi.
$\psi_t(i)$	Matriz para recuperar la secuencia de estados de probabilidad máxima.
P^*	Probabilidad de la secuencia óptima.
q_t^*	Secuencia de estados óptima.
λ^*	Modelo HMM que máximiza de manera local la probailidad $P[O\|\lambda]$.
$\xi_t(i, j)$	Probabilidad de que el modelo se encuentre en el estado s_i en el instante t, y se produzca una transición de forma que en el instante $t + 1$ el estado sea el s_j; dada la secuencia de observaciones, y el modelo λ.
$\gamma_t(i)$	Probabilidad de estar en el estado s_i en el instante t.
$\bar{\pi}_i$	Número de veces en el estado s_i en el tiempo $(t = 1) = \gamma_1(i)$.
$\bar{a}_{i,j}$	Número esperado de transiciones desde el estado s_i al estado s_j, entre el número esperado de transiciones desde el estado s_i.
$\bar{b}_j(k)$	Número esperado de veces en el estado s_j, y observando el símbolo $\bar{v}_k$, entre el número esperado de veces en el estado s_j.
$\bar{\Pi}$	Distribución de probabilidad de estado iniciales de un HMM; $\{\bar{\pi}_i\}$.
$\bar{A}$	Conjunto de probabilidades de transición entre estados de un HMM; $\{\bar{a}_{i,j}\}$.
$\bar{B}$	Probabilidades de generación de observaciones de un HMM; $\{\bar{b}_j(k)\}$.
$\bar{\lambda}$	Modelo estimado utilizando el algoritmo Baum-Welch; $(\bar{\Pi}, \bar{A}, \bar{B})$.
$\bar{\alpha}_t(i)$	Variable Forward auxiliar para el escalamiento de $\alpha_t(i)$.
c_t	Vector de escalamiento de $\alpha_t(i)$.
C_t	Coeficiente de escalado de la variable Forward.
$\hat{\alpha}_t(i)$	Variable Forward escalada.
$\bar{\beta}_t(i)$	Variable Backward auxiliar para el escalamiento de $\beta_t(i)$.
D_t	Coeficiente de escalado de la variable Backward.
$\hat{\beta}_t(i)$	Variable Backward escalada.
K	Número de secuencias de observación O.
$\|\Gamma\|_k$	Tamaño de la secuencias de observación k.
η	Umbral del método floor smoothing.
λ_κ	Modelo más probable perteneciente al conjunto de modelos correspondientes a las diferentes palabras a reconocer.
Υ	Umbral de la Entropía H para identificar voz de ruido.
$x'(n)$	Señal $x(n)$ pré-enfatizada.
σ	Nivel de pré-enfasis.
$d\vec{f}t_{real}$	Parte real de $X(k)$.
$d\vec{f}t_{imag}$	Parte imaginaria de $X(k)$.
$f(n)$	Función para obtener la frecuencia en Hertz.

Δ	Avance del marco de la señal de voz.
ϑ	Palabra con menor distancia acumulada.
N_{pal}	Número de palabras almacenadas en el diccionario.
$\mathcal{P}_{di}$	$i-$ésima palabra almacenada en el diccionario.
$\mathcal{P}_{ri}$	$i-$ésima palabra a reconocer.
N_{rec}	Número de palabras a reconocer.
Δ_d	Ancho de la diagonal de la matriz A.
O_{pi}	Secuencia de observaciones de la $i-$ésima palabra a reconocer.
λ_i	$i-$ésimo HMM almacenado en el diccionario.
$N_{hmm's}$	Número de HMMs almacenados en el diccionario.

Capítulo 1

Introducción

En este capítulo se describe brevemente los principales problemas que los humanos se han enfrentado en el reconocimiento de voz, así como una breve reseña histórica de cómo ha ido avanzando y surgiendo nuevas maneras de extraer características a la señal de voz y sus principales aplicaciones.

1.1. Planteamiento del Problema

El reconocimiento de voz constituye uno de los campos en que la computación moderna mantiene un desarrollo constante con el objetivo de encontrar soluciones cada vez más útiles y mejores al problema de la comunicación hombre-máquina. Es por esta razón que existen varios métodos para la extracción de características de la señal de voz, entre las que destacan [Cowling and Sitte, 2002]:

- Energía por Banda o Espectrogramas
- Coeficientes de Predicción Lineal (LPC por sus siglas en inglés)
- Coeficientes Cepstrales en las Frecuencias de Mel (MFCC por sus siglas en inglés)
- La Transformada Wavelet Discreta

Actualmente existe una gran controversia en cual es el mejor método para extraer características de la señal de voz [Camarena, 2011], y si a esto se le agregan las diferentes

técnicas para el reconocimiento en las que destacan [Cowling and Sitte, 2002]:

- Doblado Dinámico en Tiempo (DTW por sus siglas en inglés)
- Modelos Ocultos de Markov (HMM por sus siglas en inglés)
- Vector Quantization (VQ) / Learning Vector Quantization (LVQ)
- Self-Organizing Maps (SOM)
- Artificial Neural Networks (ANN)

Y las diferentes distancias o métricas que se pueden aplicar, las siguientes distancias sirven para comparar vectores [Camarena, 2011]:

- Distancia Euclidiana
- Distancia Manhattan
- Distancia Coseno
- Distancia Itakura

Todo esto hace que surjan preguntas importantes como ¿Cuál es el mejor método de extracción de características de la señal de voz?, ¿Cuál es la mejor técnica para el reconocimiento de voz?, ¿Cuál es la mejor métrica o distancia que se puede utilizar?, las respuestas de estas preguntas son muy complejas; ya que dependen del contexto en el que se esté desarrollando. Por ejemplo: un sistema que converse de manera continua con un humano, una conversación básica texto-voz o voz-texto. También depende si es un sistema monolocutor o multilocutor. Sin embargo, como dice la frase célebre de Julio César "Divide y Vencerás", es decir, es posible experimentar en los diferentes contextos y obtener respuestas de manera específica a cada uno de ellos, puesto que obtener una respuesta que contemple todos los posibles entornos elevaría su complejidad exponencialmente.

1.2. Antecedentes

1.2.1. Historia del reconocimiento de voz

La historia esencial de los sistemas de reconocimiento de voz se resume a continuación mencionando los eventos más importantes:

Años 50's, los inicios

En 1952 en los Laboratorios Bell, K. Davis, R. Biddulph y S. Balashek crearon un sistema electrónico que permitía el reconocimiento de dígitos aislados monolocutor, el fundamento de esta máquina se basaba en medidas de las resonancias espectrales del tracto vocal para cada dígito, estas se obtenían mediante el uso de filtros analógicos. Los Laboratorios RCA implementaron sistemas de reconocimiento de 10 sílabas monolocutor. En 1959 en la University College in England, P. Denes trataba de desarrollar un sistema para reconocer 4 vocales y 9 consonantes. Laboratorio MIT Lincoln implemento un reconocedor de vocales independiente del hablante. Hasta este momento, todos los sistemas son dispositivos electrónicos [Álvarez, 2012, Oropeza, 2006, Rabiner and Juang, 1993].

Años 60's, los fundamentos

Aparecen los primeros desarrollos realizados en Japón aplicando todavía, piezas de hardware específico aplicadas al reconocimiento de: vocales (J. Suzuki y K. Nakata Radio Research Lab. de Tokio, 1961), fonemas (T. Sakai y S. Doshita, Universidad de Kioto, 1962) y dígitos (K. Nagata, Y. Kato y S. Chiba laboratorios NEC, 1963). En la Unión Soviética T. K. Vintsyuk propone la utilización de métodos de programación dinámica para conseguir el alineamiento temporal de pares de señales de voz, esta es la esencia de los conceptos relativos al algoritmo de DTW. En CMU (Carnegie Mellon University) realiza investigaciones en el campo del reconocimiento del Habla Continua. Durante este periodo se generaliza el uso de computadores en este campo [Álvarez, 2012, Oropeza, 2006].

Años 70's, reconocimiento de palabras aisladas y primeros intentos del habla continua

Surge el reconocimiento de palabras aisladas. IBM (International Business Machines) realizó desarrollos de proyectos de reconocimiento de grandes vocabularios. Investigadores Japoneses establecieron de manera formal los algoritmos; fundamentados en la programación dinámica para aplicarse a la resolución de este tipo de problemas. Los trabajos de F. Itakura mostraban cómo los principios de las técnicas LPC podían extenderse al reconocimiento, empleados anteriormente en la codificación y compresión de la voz. El sistema Hearsay I, construido por la CMU (Carnegie Mellon University) en 1973 era capaz de emplear información de tipo semántico para reducir el número de posibles alternativas que el reconocedor debía evaluar. Finalmente, en los AT&T Bell Labs (ahora Bell Labs, Lucent Technologies y AT&T Labs-Research), los investigadores comenzaron una serie de experimentos orientados a conseguir reconocedores realmente independientes del locutor para su uso en aplicaciones telefónicas [Álvarez, 2012, Oropeza, 2006].

Años 80's, reconocimiento del habla continua

Si en la década de los 70 los sistemas de reconocimiento de vocablos aislados alcanzan una cierta madurez, los años 80 se caracterizaron por generalizar la construcción de sistemas de reconocimiento de voz. Surge un giro metodológico que se produce como consecuencia de pasar de métodos basados en comparación y programación dinámica, a los métodos basados en modelado estadístico como son los HMM; métodos surgidos en la década de los 70's pero su aceptación generalizada fue hasta los 80's. Se reintroducen las redes neuronales, los primeros modelos neuronales como por ejemplo: el perceptrón; inicialmente propuesto en los años 50, volvieron a aparecer a finales de esta década gracias al desarrollo de algoritmos de aprendizaje mucho más eficaces. La tecnología de sistemas expertos había sido postulada como base para diseñar unidades de decodificación fonética, que se sirvieran de la experiencia de fonetistas en tareas de interpretación de espectrogramas. En esta época también se realizaron grabaciones de bases de datos de voz como por ejemplo TIMIT (1986), que contribuyen a los avances en la disciplina y que permiten comparar los resultados entre diferentes grupos de trabajo. Durante este mismo periodo, el programa

DARPA (Defence Advance Research Agency), impulsó en Estados Unidos el desarrollo de mejores sistemas de reconocimiento para habla continua y vocabularios de tamaño medio y grande con independencia del locutor [Álvarez, 2012, Oropeza, 2006].

Años 90's, mejora y diversificación de aplicaciones

La década de los 90's supone en cierta manera la continuidad en los objetivos ya propuestos, ampliando el tamaño de los vocabularios a la vez que se diversifican los campos de aplicación. Los servicios de telefonía son unos de los campos que más atención acaparan en la actualidad. En los últimos años a aumentado el interés por el estudio de reconocimiento de voz en condiciones de ruido y ambientes adversos. Surgieron los sistemas de dictado e integración entre reconocimiento de voz y procesamiento del lenguaje natural [Álvarez, 2012, Oropeza, 2006].

Siglo XXI, integración de aplicaciones

Se integra el software de reconocimiento de voz en el Sistema Operativo, integración de aplicaciones por teléfono y sitios de Internet dedicados a la gestión de reconocimiento de voz (Voice Web Browsers) y aparece el estándar VoiceXML [Oropeza, 2006].

El rendimiento de los sistemas de reconocimiento actuales están muy por debajo de las capacidades del ser humano. Para obtener un reconocimiento y entendimiento automático del habla cercano al del humano no será de manera inmediata por el trabajo de un investigador ingenioso, sino el trabajo continuo que incorpore conocimientos multidisciplinares, incluyendo trabajos de investigación básica en los campos de producción y percepción del habla [Álvarez, 2012].

1.2.2. Aplicaciones del reconocimiento de voz

Actualmente existe una gran cantidad de aplicaciones enfocadas al reconocimiento y síntesis de voz. El rendimiento de estas aplicaciones es aceptable. Sin embargo deben estar en constante mejora e investigando nuevas técnicas que optimicen el rendimiento. Algunos de los desarrollos y aplicaciones del renoconocimiento de voz son:

- Software de Reconocimiento de Voz integrado en Windows, usando solamente la voz se puede: iniciar programas, dictar texto dentro de un documento, escribir y enviar e-mails. Casi todo lo que se puede hacer con el teclado y el mouse puedes hacerlo usando solo la voz [Serrano, 2012].

- EL proyecto LISTEN (Literacy Innovation that Speech Technology ENables). Es un proyecto de investigación interdisciplinario de la Universidad de Carnegie Mellon CMU (Carnegie Mellon University) para desarrollar una nueva herramienta para mejorar la alfabetización. Un tutor de lectura automatizada muestra historias en la pantalla de un ordenador, y escucha a los niños leer en voz alta. Para proporcionar una experiencia agradable y auténtica en la lectura asistida, el tutor de lectura permite al niño elegir en un menú las historias que más le interesen, incluyendo cuentos creados por el usuario [CMU, 2008].

- En el desarrollo de Sistemas Empotrados (Embedded Systems), resulta interesante la integración del reconocedor de voz en los mismos, con todas las ventajas que aporta en el desarrollo de aplicaciones. El reconocimiento de voz se puede utilizar prácticamente en cualquier aparato, por ejemplo: un televisor, una lavadora, un carro, una puerta, una lámpara, etcétera [Puertas, 2000].

- Aplicaciones Telefónicas. Estas aplicaciones presentan unas propiedades muy particulares puesto que utilizan la red telefónica como canal de comunicación. Suelen ser aplicaciones de tipo conversacional, donde hay un diálogo en el que se guía al usuario a través de un conjunto de preguntas hasta conseguir el objetivo deseado. Suelen emplearse reconocedores de palabras aisladas o palabras conectadas con vocabularios no muy grandes [Puertas, 2000].

- Una de las ramas de la tecnología donde se aplica el reconocimiento de voz es la Robótica. Por ejemplo: controlar un robot mediante el dictado de órdenes o simplemente para hacerlos más inteligentes e interactivos con los humanos.

En las últimas décadas la telefonía móvil se ha convertido en una herramienta indispensable en la Sociedad de la Informática, y es un factor clave para asegurar la competitividad ya que se ha incorporado en la vida cotidiana. La telefonía móvil se revela como el medio más eficaz y personalizado de comunicación del futuro. Por esta razón se han desarrollado una gran cantidad de aplicaciones de reconocimiento de voz para dispositivos móviles, muchas de ellas con características muy similares. A continuación se describen algunas:

- Voice Translator. Es una aplicación para PC con Windows en la que puedes aprender y practicar otros idiomas de una manera muy natural. Solamente se tiene que hablar al audífono integrado en el equipo y obtén como respuesta la traducción en el idioma de la elección, o simplemente selecciona un texto con el ratón y escucha cómo se pronuncia en otro idioma. Ni la pronunciación texto a voz, ni el reconocimiento de voz, ni el proceso de traducción funcionan siempre con un 100 % de precisión [VTRA]. Actualmente está disponible para Android [Software, 2012].

- Voice Search (búsqueda por voz). Es una aplicación para buscar en la web, en el celular y además para buscar locaciones cercanas. Solo se necesita hablar en voz alta lo que deseas buscar en Google. Actualmente tiene más de 100 millones de usuarios [Play, 2011].

- VoiceLink. Aplicación que reconoce voz y permite enviar mensajes, llamar a contactos, actualizar el estado de Twitter, enviar un correo, etc., de una manera muy simple [Play, 2009].

- Shazam. Es una aplicación para telefonía móvil; permite a los usuarios identificar el artista, nombre, álbum, enlaces de interés a servicios como iTunes, Youtube, Spotify o Zune, así como también sugerencias de otras canciones relacionadas con tan solo escuchar un fragmento de la canción a identificar [Barton et al., 1999]. Si bien no es un "reconocedor de voz", aplica la misma tecnología que se utiliza para la voz.

El reconocimiento de voz es una tecnología, que promete cambiar la forma de cómo se interactua con las computadoras. Sin embargo, todavía falta tiempo para que esta sustituya la manera en la que actualmente se desarrollan tareas.

1.3. Objetivos de la Tesis

1.3.1. Objetivo general

Obtener el rendimiento mediante curvas ROC de los sistemas de reconocimiento de voz de palabras aisladas monolucutor implementados utilizando LPC, Energía por Banda (Espectrogramas) y MFCC; aplicando DTW y DHMM como técnicas de reconocimiento.

1.3.2. Objetivos particulares

- Tomar como entrada un archivo WAV (WAVEform audio file format) muestreado a 8000.0 Hz cuantizado a 16 bits, mono, little-endian grabados en un entorno de baja interferencia acústica.
- Realizar la segmentación de palabras aisladas y aplicar el filtro de pre-énfasis a la palabra segmentada.
- Implementar los algoritmos de extracción de características LPC, Energía por Banda y MFCC.
- Implementar el algoritmo de DTW.
- Resolver los problemas básicos que se tienen al usar DHMM, implementar los algoritmos de Forward-Backward para resolver el problema de evaluación, algoritmo de Viterbi para resolver el problema de decodificación y el algoritmo de Baum-Welch para resolver el problema de entrenamiento.
- Implementar la distancia euclidiana, coseno e Itakura.
- Construir sistemas de reconocimiento de palabras aisladas monolocutor.
- Evaluar el rendimiento de los diferentes sistemas de reconocimiento de voz de palabras aisladas implementados, se hace uso de las curvas ROC para obtener el mejor método de extracción de características, la mejor técnica de reconocimiento y para cada método de extracción de características obtener la mejor distancia que se debe utilizar.

- Apoyar con este documento a personas que incursionan el tema de reconocimiento de voz y servir como base para futuros trabajos de investigación.

1.4. Descripción de Capítulos

El capítulo 2 explica la teoría bajo la cual se fundamenta el procesamiento digital de señales, conceptos como energía de tiempo corto, régimen de cruces por cero de tiempo corto, Entropía de la señal de voz, Autocorrelación de tiempo corto. También se introduce como realizar la segmentación de palabras aisladas.

El capítulo 3 explica los métodos de extracción de características de la señal de voz utilizando Energía por Banda o Espectrogramas, MFCC y LPC. Se introducen los temas de transformada de Fourier de tiempo corto, como determinar el espectrograma de una señal usando transformada de Fourier. Explica cómo obtener los coeficientes MFCC de una señal de voz, se introducen algunos conceptos como Cepstrum y el Cepstrum complejo, se introduce la escala y filtros triangulares de Mel. También se explica cómo obtener los coeficientes LPC de una señal de voz utilizando el método recursivo de Durbi.

El capítulo 4 describe a grandes rasgos las técnicas de reconocimiento como son el algoritmo de Doblado Dinámico en Tiempo (DTW, Dynamic Time Warping) y los Modelos Ocultos de Markov (HMM, Hidden Markov Models), se explica el algoritmo Forward-Backward, algoritmo de Viterbi, el algoritmo de Baum-Welch y se estudiará la aplicación de los HMM al reconocimiento de voz, incluyendo como solucionar algunos problemas prácticos de implementación.

El capítulo 5 expone con detalle la implementación de los sistemas de reconocimiento de palabras aisladas implementados.

El capítulo 6 muestra los resultados obtenidos al comparar el rendimiento de los diferentes sistemas de reconocimiento de palabras aisladas implementados, haciendo uso de las curvas ROC.

Finalmente en el capítulo 7 se exponen las conclusiones, se mencionan algunas mejoras que se pueden realizar al sistema implementado y desarrollos futuros.

Capítulo 2

Procesamiento digital de la señal de voz

El procesamiento digital de una señal se refiere a la obtención de la representación discreta de la señal, con la teoría, diseño y la implementación de procedimientos numéricos. Las técnicas de procesamiento digital de la señal inicialmente fueron aplicadas en problemas de procesamiento del habla, así como en la simulación de sistemas analógicos complejos.

A continuación se abordan algunos conceptos preliminares así como los esquemas más populares acorde a [Camarena, 2011], que se emplean para determinar el inicio y el fin de una palabra.

2.1. Introducción

Se denota como $x_a(t)$, a una forma de onda continuamente variante en el tiempo (o analógica) mientras que se representa como $x(n)$, a una secuencia de números. Una secuencia de muestras tomadas periódicamente de una señal analógica se puede representar como $x_a(nT)$, donde T es el periodo de muestreo.

La Figura 2.1 muestra un ejemplo de representación de una señal de voz tanto como una señal analógica, como una secuencia de muestras a una tasa de muestreo de 8

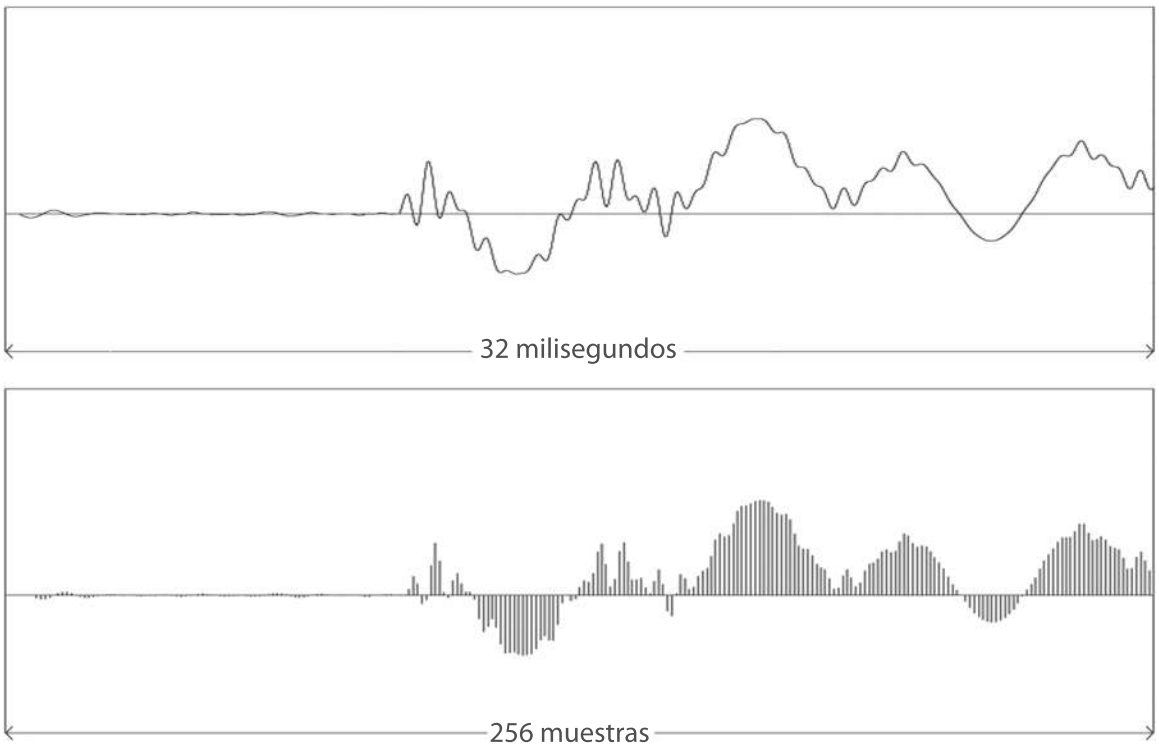

Figura 2.1: Representaciones de una señal de voz

kHz. En las figuras subsiguientes, por conveniencia en las gráficas se utiliza la representación analógica, aún cuando se esté usando la representación discreta. En estos casos la curva continúa que se muestre, significa la envolvente de la secuencia de muestras.

Existen algunas secuencias especiales de tiempo que son de gran utilidad en el estudio de sistemas de procesamiento digital de la señal de voz [Camarena, 2011]. Varias de estas secuencias son representadas en la Figura 2.2.

El impulso unitario:

$$\delta(n) = \begin{cases} 1 & n = 0 \\ 0 & \text{De lo contrario} \end{cases} \tag{2.1}$$

El escalón unitario:

$$u(n) = \begin{cases} 1 & n \geq 0 \\ 0 & \text{De lo contrario} \end{cases} \tag{2.2}$$

La secuencia exponencial:

$$x(n) = a^n \tag{2.3}$$

si a es complejo, es decir, $a = re^{j\omega_0}$, entonces

$$x(n) = r^n e^{j\omega_0 n} = r^n[cos(\omega_0 n) + jsen(\omega_0 n)] \tag{2.4}$$

Donde r es el modulo del número complejo, y ω_0 es la frecuencia angular. Si $r = 1$ y $\omega_0 \neq 0$, entonces $x(n)$ es una senoide, si $\omega_0 = 0$, $x(n)$ es real; y si $r < 1$ y $\omega_0 \neq 0$, entonces $x(n)$ es una secuencia oscilatoria que decae exponencialmente.

2.1.1. Sistemas lineales e invariantes en el tiempo

El procesamiento de señales se puede definir como la transformación de una señal a una forma más deseable en cierto sentido, tales transformaciones se pueden denotar como

$$y(n) = T\left[x(n)\right] \tag{2.5}$$

La clase especial de sistemas lineales e invariantes en el tiempo LTI (Linear Time-Invariant) (o en el espacio) son útiles para implementar filtros para señales de voz y quizás son más útiles para implementar modelos para la producción de voz, estos sistemas quedan completamente caracterizados por su respuesta al impulso unitario. Para estos sistemas la salida $y(n)$ se puede determinar como

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) = x(n) * h(n) \tag{2.6}$$

Donde $h(n)$, es la respuesta del sistema al impulso unitario, y $*$ es el símbolo de convolución [Calderón, 2006] (no confundir con la multiplicación).

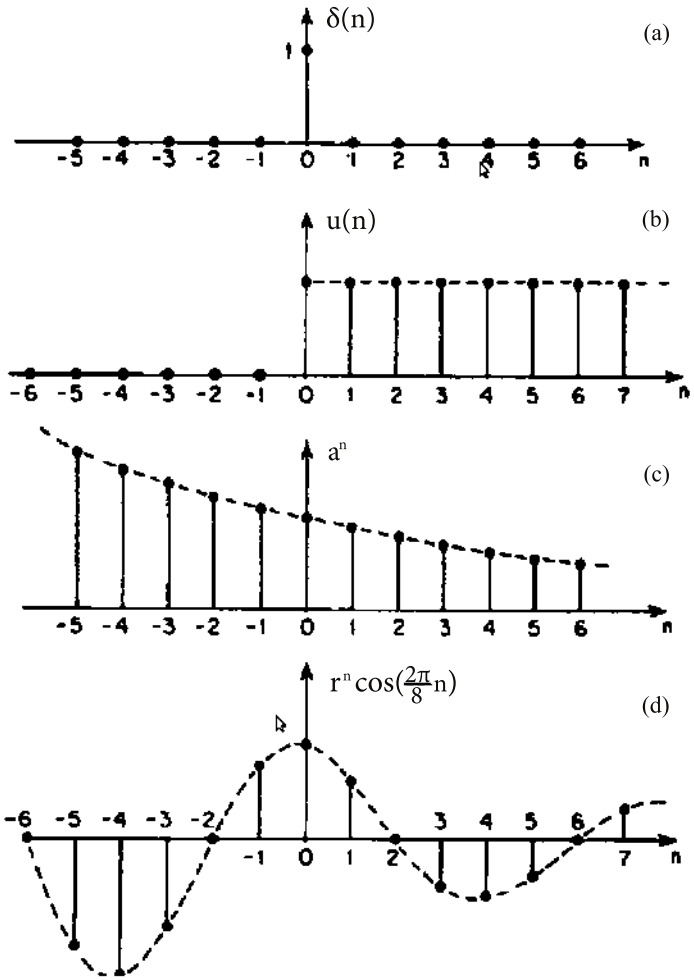

Figura 2.2: (a) Impulso unitario; (b) Escalón unitario; (c) Exponencial real; y (d) Coseno amortiguado [Rabiner and Schafer, 1978]

La ecuación 2.6 es equivalente a:

$$y(n) = \sum_{k=-\infty}^{\infty} h(k)x(n-k) = h(n) * x(n) \tag{2.7}$$

2.2. Energía de tiempo corto

La amplitud de una señal de voz varia con el tiempo, en particular la amplitud de segmentos no vocalizados es generalmente mucho menor que la amplitud de segmentos vocalizados. La energía de tiempo corto de una señal de voz provee una representación que refleja las variaciones de amplitud. En general, se puede definir la energía de tiempo corto como

$$E_n = \sum_{m=-\infty}^{\infty} \Big[x(m)w(n-m)\Big]^2 \tag{2.8}$$

Donde E_n es la energía de tiempo corto de la ventana ubicada en la posición n, $x(m)$ es la señal discreta de audio y $w(n-m)$ es una ventana de tamaño N que se desliza sobre la señal y solo se permite ver un segmento corto de la señal de audio a la vez. La ventana se puede definir de varias maneras, en la sección 2.2.1 se muestran algunas ventanas que se pueden utilizar.

La ecuación 2.8 puede escribirse como

$$E_n = \sum_{m=-\infty}^{\infty} x^2(m) \cdot h(n-m) \tag{2.9}$$

donde

$$h(n) = w^2(n) \tag{2.10}$$

La ecuación 2.9 puede ser interpretada gráficamente como se representa en la Figura 2.4. La señal $x^2(n)$ es filtrada por un filtro lineal con respuesta al impulso $h(n)$ como se muestra en la ecuación 2.10.

En la Figura 2.3, la señal $x(m)$ se eleva al cuadrado (observe en la parte inferior como ya no hay valores negativos) y luego se ve a través de una ventana (en realidad marco)

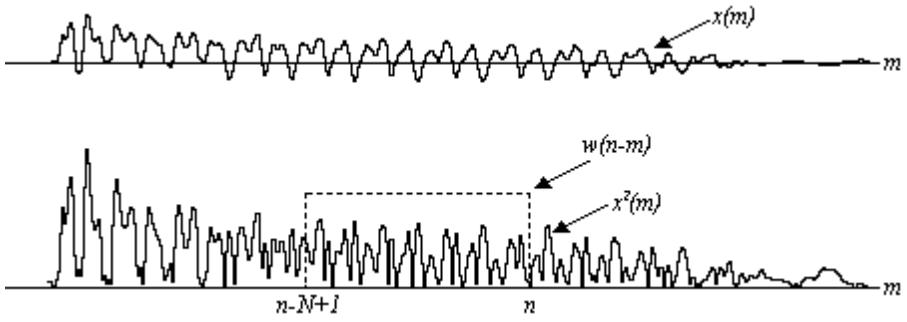

Figura 2.3: Calculando la Energía de tiempo corto

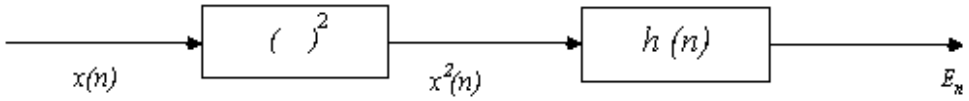

Figura 2.4: Diagrama de flujo representando la energía de tiempo corto

de tamaño N, de manera que las únicas muestras que se van a sumar para determinar la energía de tiempo corto son las que se ubican dentro de la ventana, esta energía se denota como E_n.

Las Figuras 2.5 y 2.6 se muestra el efecto de variar el tamaño de la ventana (para las ventanas rectangular y Hamming, respectivamente) sobre E_n, para la palabra "atrás"pronunciada por un hombre a 8000Hz. Se puede observar claramente que al aumentar el tamaño de N, E_n se vuelve más suave para ambas ventanas.

Para comprender mejor el significado de E_n se debe saber que provee una base para distinguir segmentos de voz vocalizados de segmentos de voz no vocalizados. Como se puede observar en la Figuras 2.5 y 2.6, los valores de E_n para segmentos no vocalizados son significativamente más pequeños que para segmentos vocalizados. La función de energía puede ser utilizada para localizar el tiempo aproximado en el cual el sonido cambia de vocalizado a no vocalizado y viceversa, y la energía se puede utilizar para distinguir voz de

silencio.

2.2.1. Aplicación de ventanas

Las ventanas son funciones matemáticas usadas con frecuencia en el análisis y el procesamiento de señales para evitar las discontinuidades al principio y al final de los bloques analizados.

La *ventana rectangular* es equivalente a no aplicar ninguna ventana, esta definida como

$$w(n) = \begin{cases} 1 & 0 \leq n \leq N-1 \\ 0 & \text{De lo contrario} \end{cases} \tag{2.11}$$

La *ventana de Welch* definida mediante

$$w(n) = 1 - \left(\frac{n-N/2}{N/2}\right)^2 \tag{2.12}$$

La *ventana de Barlett* definida mediante

$$w(n) = \begin{cases} 1 - \frac{2n}{N} & 0 \leq n \leq \frac{N}{2} \\ 1 + \frac{2n}{N} & -\frac{N}{2} \leq n \leq 0 \\ 0 & \text{otro lugar} \end{cases} \tag{2.13}$$

La *ventana de Hamming* está definida mediante

$$w(n) = 0.54 + 0.46cos\left(\frac{2\pi n}{N}\right) \tag{2.14}$$

La *ventana de Hann* está definida mediante

$$w(n) = 0.5 + 0.5cos\left(\frac{2\pi n}{N}\right) \tag{2.15}$$

La ventana de Hamming y la ventana de Hann ambas forman parte de una familia de ventanas genéricas definidas mediante

$$w(n) = \alpha + (1-\alpha)cos\left(\frac{2\pi n}{N}\right) \tag{2.16}$$

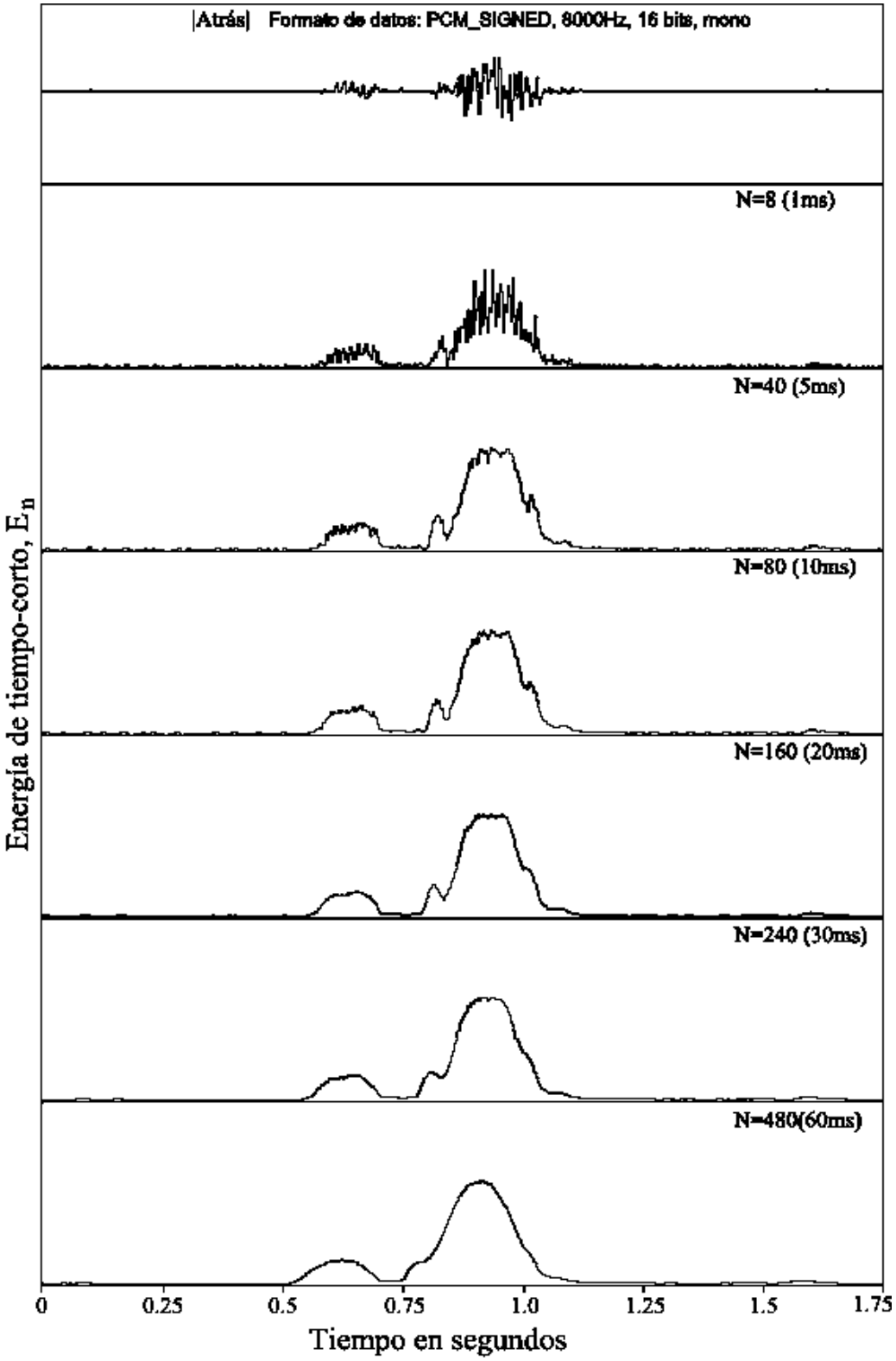

Figura 2.5: Energía de tiempo corto para ventanas rectangulares de varios tamaños

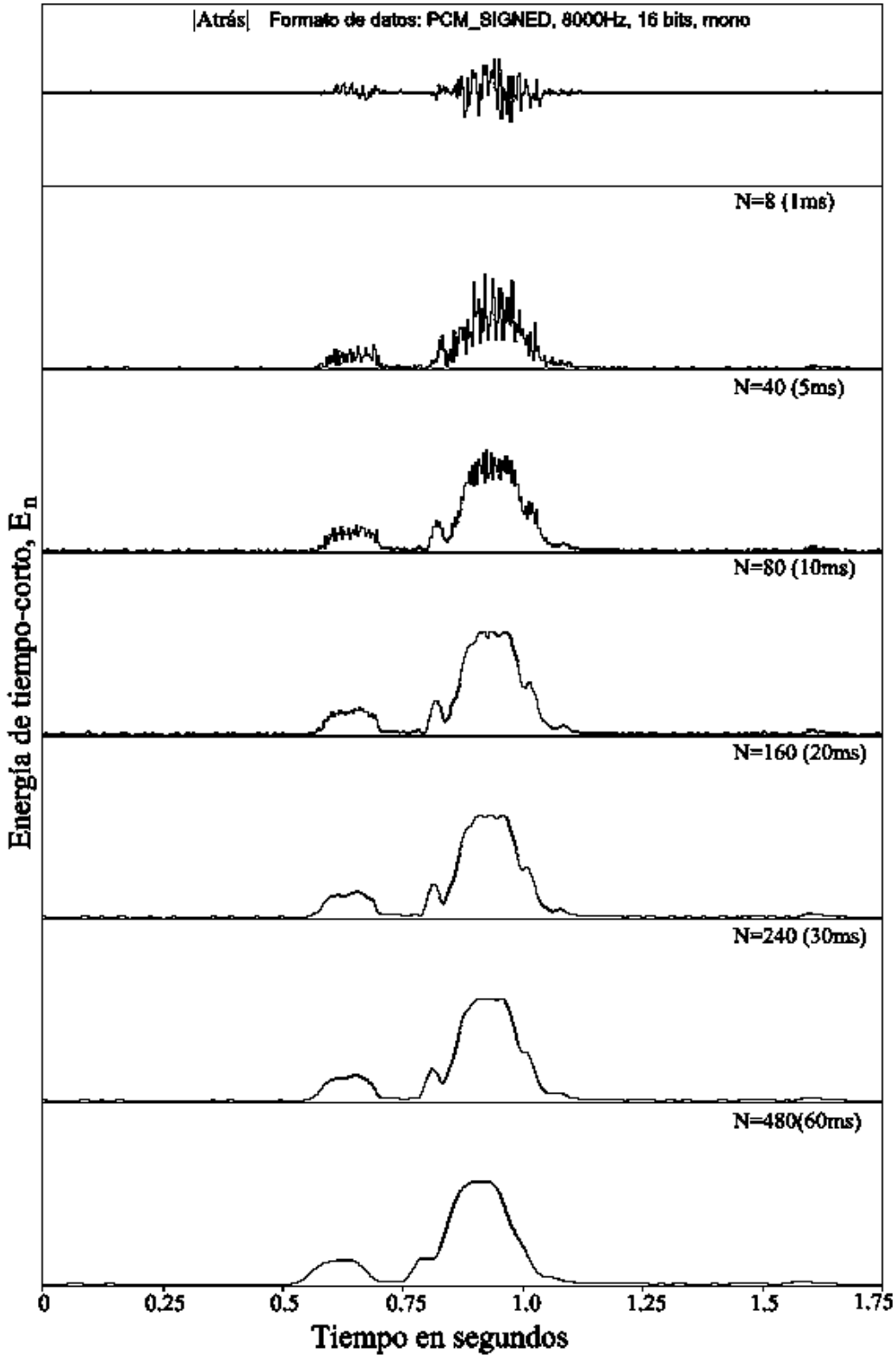

Figura 2.6: Energía de tiempo corto para ventanas Hamming de varios tamaños

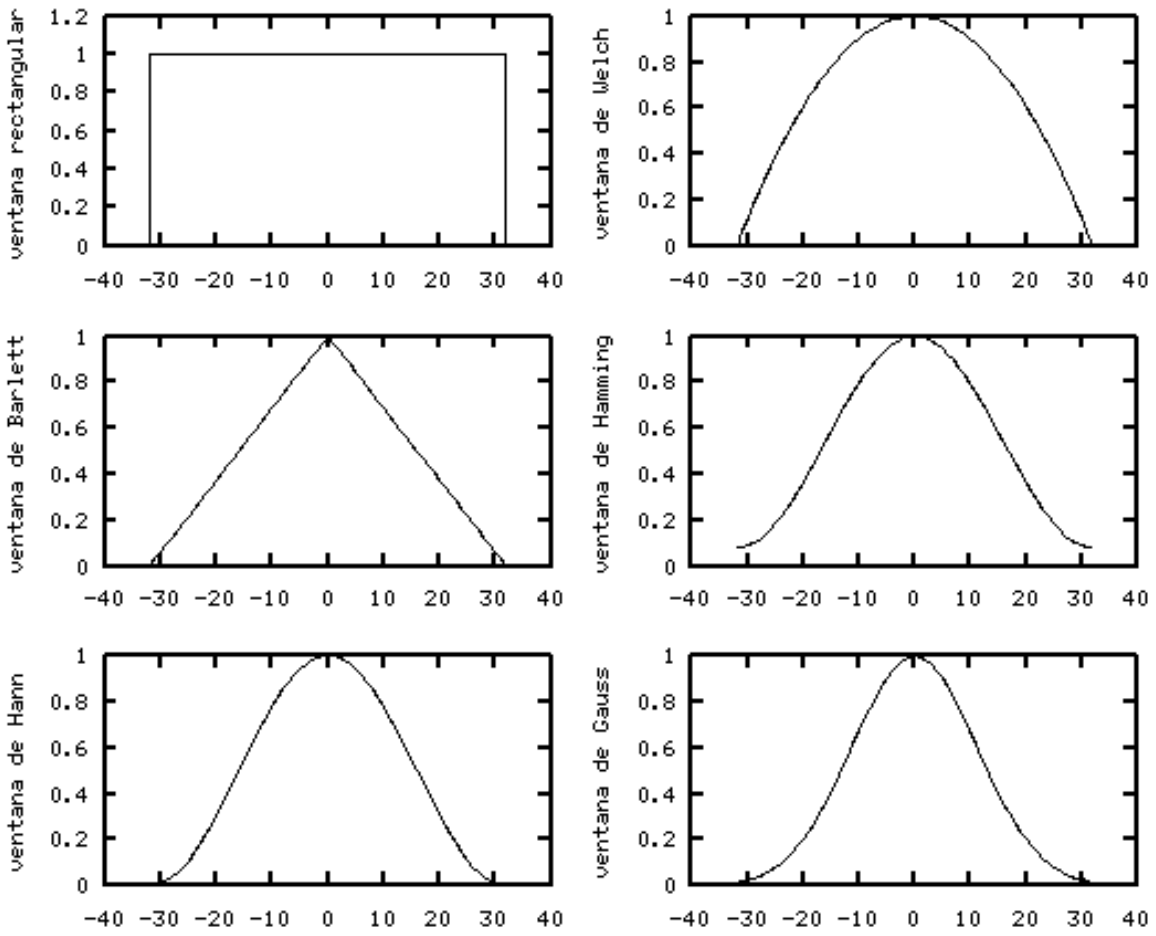

Figura 2.7: Ventanas rectangular, Welch, Barlett, Hamming, Hann y Gauss

La *ventana de Gauss* está definida mediante

$$w(n) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(n-\mu)^2}{2\sigma^2}} \tag{2.17}$$

Donde μ es la media y σ^2 es la varianza.

Las ventanas rectangular, Welch, Barlett, Hamming, Hann y Gauss se muestran gráficamente en la Figura 2.7, en los ejemplos se utiliza un tamaño de ventana N de 64 y para la ventana de Gauss se utiliza $\sigma = 11$ y $\mu = 0$.

El hecho de utilizar ventanas hace que se convolucione la transformada de la señal con la transformada de la ventana. Por lo tanto, se debe elegir aquella ventana que produzca menor distorsión. En el presente trabajo se utiliza la ventana de Hamming, por que de las ventanas expuestas esta ventana es la más utilizada [Bernal et al., 2012, pág. 13-15].

2.3. Régimen de cruces por cero de tiempo corto

Un cruce por cero ocurre cundo dos muestras consecutivas de la señal tienen signos distintos. El régimen al cual ocurren los cruces por cero es una medida del contenido de frecuencia en la señal, por ejemplo, una senoide de frecuencia f muestreada a una frecuencia f_s será digitalizada de manera que habrá f_s/f muestras por periodo, y como hay dos cruces por cero en cada periodo, se tendra un régimen de cruces por cero $Z_n = 2f/f_s$. Una definición adecuada del régimen de cruces por cero de tiempo corto [Camarena, 2011] es:

$$Z_n = \sum_{m=-\infty}^{\infty} \Big[|signo[x(m)] - signo[x(m-1)]|\, w(n-m) \Big] \tag{2.18}$$

donde:

$$signo[x(n)] = \begin{cases} 1 & x(n) \geq 0 \\ -1 & x(n) < 0 \end{cases} \tag{2.19}$$

además:

$$w(n) = \begin{cases} \frac{1}{2N} & 0 \leq n \leq N-1 \\ 0 & \text{De lo contrario} \end{cases} \tag{2.20}$$

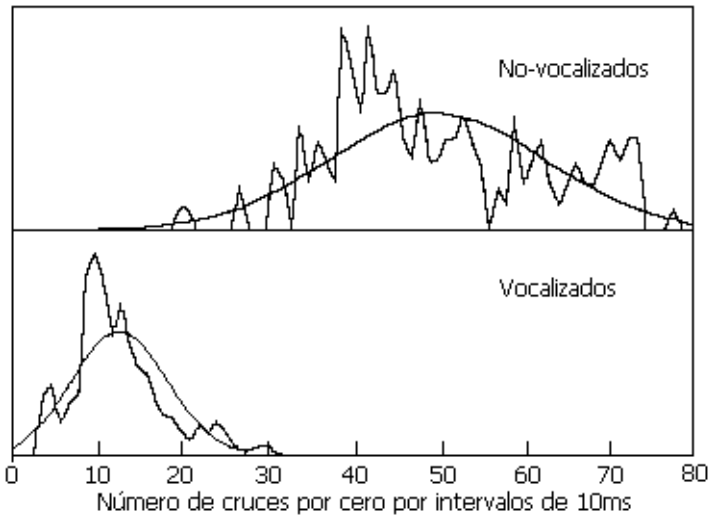

Figura 2.8: Distribución del régimen de cruces por cero en sonidos vocalizados y en sonidos no-vocalizados

Un uso que se le da al régimen de cruces por cero de tiempo corto es el de discriminar sonidos vocalizados de sonidos no-vocalizados; como se puede ver en la Figura 2.8 es que en general, los sonidos vocalizados tiene un régimen de cruces por cero bajo, comparado con el régimen de cruces por cero de los sonidos no vocalizados.

En la Figura 2.9 se muestran algunos ejemplos de mediciones del régimen de cruces por cero. En estos ejemplos, la duración de la ventana es de 15 milisegundos (frecuencia de muestreo de 8kHz-120 muestras). Los valores de cruces por cero varían consideradamente en las regiones de sonidos vocalizados y no-vocalizados, son bastante prominentes como se muestra en la Figura 2.9.

2.4. Entropía de la señal de voz

El contenido de información de un mensaje es desde el punto de vista de Claude Shannon proporcional al nivel de sorpresa obtenida en el lector, si es fácil de adivinar el contenido no habrá mucha información ahí. Sea $\{v_i\}$, el conjunto de los posible valores que pueden tomar las muestras de una señal de audio. v_i tiene la probabilidad p_i de ocurrir y la secuencia $p_1, p_2, \ldots, p_n$, es la Función de Distribución de Probabilidades (PDF por sus

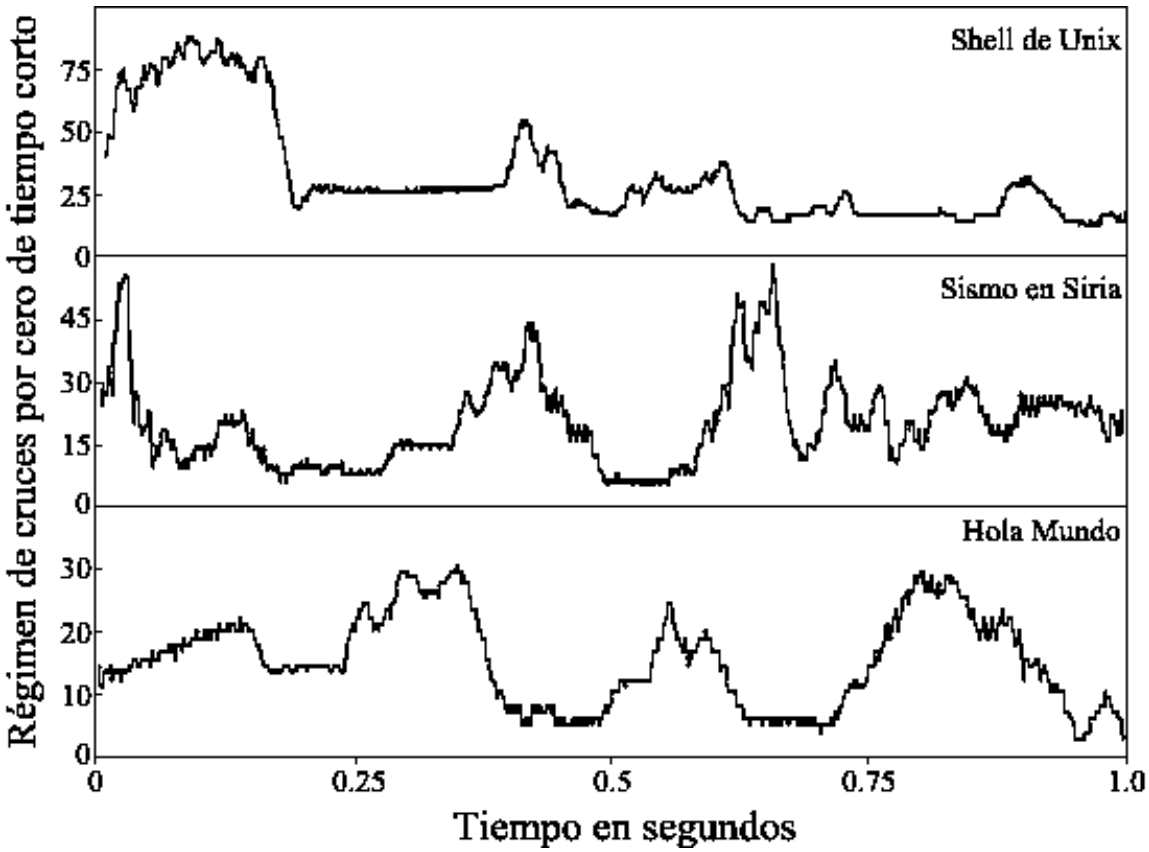

Figura 2.9: Régimen de cruces por cero para tres diferentes locuciones

siglas en inglés) sujeta a:

$$\sum_{i=1}^{n} p_i = 1 \tag{2.21}$$

El contenido de información I en un valor v_i también llamada "información propia", depende exclusivamente de su probabilidad de ocurrir [Camarena, 2011, pág. 54-57], es decir, $p_i = P(v_i)$.

Mientras menos probable sea que un valor se presente, mayor será la información que traerá consigo, entonces, la información propia I, debe ser una función monotónicamente decreciente de la probabilidad; además $I(p_i)$, debe ser calculada de manera que si v_i depende de dos o más eventos independientes con probabilidades: $p_{i1}, p_{i2}, \ldots, p_{in}$, entonces la contribución al contenido de información de cada evento debe ser tal que la suma total coincida con $I(pi)$, para poderse manejar como información, así si $p_i = p_{i1}, p_{i2}, \ldots$; entonces, $I(p_i) = I(p_{i1}) + I(p_{i2}), \ldots$. Estas restricciones entre otras llevaron a Shannon a la conclusión de que la única función que podía cumplirlas era la función logarítmica, y que la información propia debe de calcularse mediante la Ec. 2.22 (la base del logaritmo no es importante).

$$I(p_i) = ln\left(\frac{1}{p_i}\right) = -ln(p_i) \tag{2.22}$$

La entropía H es igual a la esperanza matemática del contenido de información en una secuencia [Camarena, 2011, pág. 54-57], es decir, es el promedio de los contenidos de información de todos los valores posibles ponderado por sus probabilidades de ocurrir, tal y como lo indica la ecuación 2.23. Como la entropía de una señal es una medida de lo impredecible que esta es, resulta mínima para una señal constante de valor k, su PDF es un impulso unitario ubicado en k, es decir, $p_i = \delta(k)$, y su entropía es cero como se muestra en la ecuación 2.24, en el otro extremo la entropía es máxima para la distribución uniforme cuya PDF es $p_i = 1/n$ para n valores posibles, la entropía máxima es $ln(n)$ (ver ecuación 2.25).

$$H = E[I(p)] = \sum_{i=1}^{n} p_i I(p) = \sum_{i=1}^{n} p_i ln(p_i) \tag{2.23}$$

$$H_{min} = -\sum_{i} \delta(k) ln[\delta(k)] = -ln(1) = 0 \tag{2.24}$$

$$H_{max} = -\sum_{i} \frac{1}{n} ln[\frac{1}{n}] = ln(n) \tag{2.25}$$

Para calcular la entropía de una señal usando métodos paramétricos se debe asumir el tipo de distribución al que se apega la señal, una vez elegido el tipo de distribución se deben estimar sus parámetros [Camarena, 2011, pág. 54-57]. Si por ejemplo, se asume que la señal de audio sigue una distribución Gaussiana, se puede utilizar la ecuación 2.26 que es la entropía de una variable aleatoria con distribución $N(0, R)$.

$$H = \frac{n}{2} ln(2\pi) + \frac{1}{2} ln(|R|) \tag{2.26}$$

Donde R es la matriz de covarianzas de grado n, la ecuación 2.26 es equivalente a:

$$H = \frac{n}{2} ln(2\pi) + \frac{1}{2} ln(\delta^2) \tag{2.27}$$

Donde δ^2 es la varianza y está definida como:

$$\delta^2 = \frac{1}{N-1} \sum_{i=0}^{N-1} (x_i - \overline{x})^2 \tag{2.28}$$

donde $\overline{x}$ es la media, H se mide en nats.

2.5. Segmentación de palabras aisladas

Los algoritmos que se describirán en esta sección uno está basado en dos simples mediciones en el dominio del tiempo, lo que es energía de tiempo corto y régimen de cruces por cero de tiempo corto, el otro es usando la Entropía de la señal de voz.

El uso más importante del régimen de cruces por cero está en la segmentación de palabras aisladas y mas específicamente en la discriminación entre silencio y la voz. Está por si sola funciona bien solamente si la palabra inicia con un fricativo (algunas consonantes

fricativas son la f, s, x, j). En vista de que no todas las palabras comienzan con un fricativo, conviene combinar el régimen de cruces por cero con la energía de tiempo corto; para implementar un discriminador silencio/voz, de esta manera si se mantiene por encima de un cierto valor umbral el Z_n o por encima de un cierto valor umbral la E_n, se puede considerar que la señal de audio tiene contenido de voz, de lo contrario se puede considerar que la señal de audio tiene solo silencio o ruido ambiental, es decir, es una señal de poca energía y de baja frecuencia.

En la Figura 2.10 a) y c) se observa como se encuentra el inicio y el final de las palabras *atrás y seis*, monitoreando únicamente el régimen de cruces por cero y la energía de tiempo corto.

La Entropía se puede utilizar para la discriminación entre silencio/voz, de manera que si la Entropía se mantiene por encima de un cierto umbral se puede considerar que la señal de audio tiene contenido de voz, es decir, contiene un gran contenido de información, de lo contrario solo contiene silencio o ruido ambiental. Es común que ciertas palabras tengan silencio en medio de la palabra (ver Figura 2.10 a) y b)), por esta razón para encontrar el inicio de la palabra se monitorea la señal de izquierda a derecha, para encontrar el final se puede monitorear de derecha a izquierda o continuar monitoreando de izquierda a derecha, pero una vez que se encuentre el final de la palabra seguir monitoreando un cierto tiempo para observar si la palabra continua.

En la Figura 2.10 b) y d) se observa como se encuentra el inicio y el final de las palabras *atrás y seis*, monitoreando la Entropía de la señal.

2.6. Autocorrelación de tiempo corto

La autocorrelación de una señal discreta x es una función definida como:

$$\phi(k) = \sum_{m=-\infty}^{\infty} x(m)x(m+k) \tag{2.29}$$

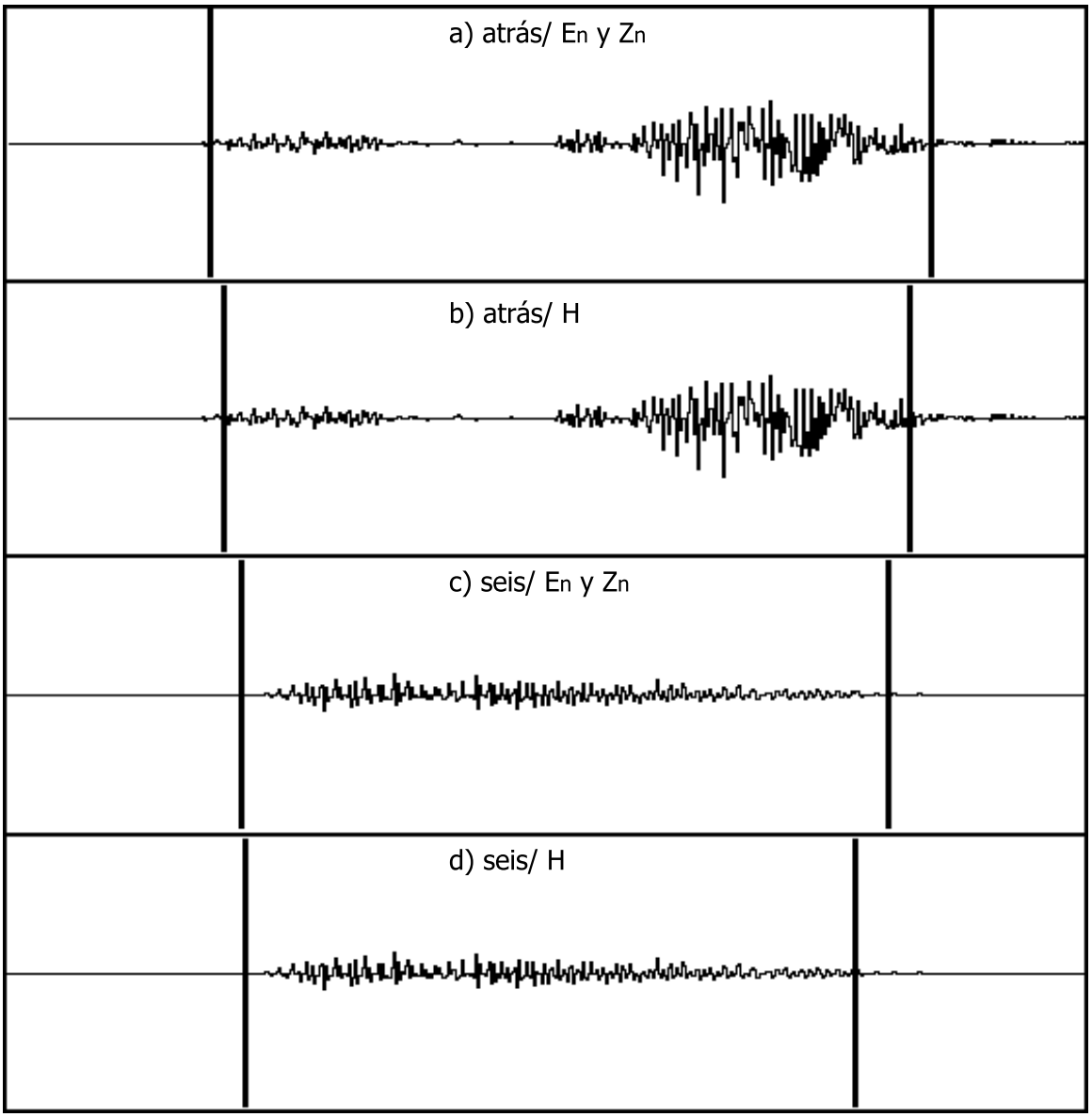

Figura 2.10: Inicio y fin de las palabras a) *"atrás"* utilizando Energía de tiempo corto y Régimen de cruces por cero, b) *"atrás"* utilizando Entropía, c) *"seis"* utilizando Energía de tiempo corto y Régimen de cruces por cero, d) *"seis"* utilizando Entropía

Si la señal es aleatoria o periódica la definición apropiada es:

$$\phi(k) = \lim_{N\to\infty} \frac{1}{(2N+1)} \sum_{m=-N}^{N} x(m)x(m+k) \tag{2.30}$$

La autocorrelación de una señal es una forma de obtener ciertas propiedades de la señal, por ejemplo, si la señal es periódica con periodo P muestras, entonces es fácil demostrar que

$$\phi(k) = \phi(k+P) \tag{2.31}$$

Propiedades de la autocorrelación:

1. La función de autocorrelación de una señal periódica también es periódica y con el mismo periodo.

2. Es una función par; es decir, $\phi(k) = \phi(-k)$.

3. Tiene su máximo en $k = 0$; es decir, $|\phi(k)| \leq \phi(0)\ \forall k$.

4. La energía de la señal es igual a $\phi(0)$.

5. La ubicación del máximo global diferente de $\phi(0)$ es el periodo, si la señal es periódica.

Si se toma en cuenta las propiedades (2) y (3), se puede concluir que la autocorrelación tiene máximos locales en $0, \pm P, \pm 2P, \ldots$, por lo tanto el periodo de la señal se puede determinar encontrando la ubicación del primer máximo local de la función de autocorrelación (5). Esta propiedad hace a la función de autocorrelación muy atractiva como estimador de la periodicidad de señales cortas.

La autocorrelación retiene tanta información de la señal de voz que se pueden usar solo los primeros 10-15 valores de la autocorrelación de tiempo corto, a partir de ellos, se pueden determinar coeficientes LPC utilizados para comprimir y descomprimir la señal.

La autocorrelación de tiempo corto de la señal x se puede definir de manera similar a como se definió la energía de tiempo corto o el régimen de cruces por cero de tiempo corto:

$$R_n(k) = \sum_{m=-\infty}^{\infty} x(m)w(n-m)x(m+k)w(n-k-m) \tag{2.32}$$

Esta ecuación se puede entender como tomar un segmento de la señal (mediante la multiplicación por la ventana) y luego determinar la autocorrelación de dicho segmento de señal. Usando la propiedad (2) $R_n(k) = R_n(-k)$ la ecuación 2.32 es equivalente a

$$R_n(k) = \sum_{m=-\infty}^{\infty} x(m)x(m-k)[w(n-m)w(n+k-m)] \tag{2.33}$$

Si se define

$$h_k(n) = \sum_{n=-\infty}^{\infty} w(n)w(n+k) \tag{2.34}$$

Entonces la ecuación 2.33 se puede reescribir como

$$R_n(k) = \sum_{m=-\infty}^{\infty} x(m)x(m-k)h_k(n-m) \tag{2.35}$$

Al programar la función de autocorrelación de tiempo corto es usualmente realizado usando la ecuación 2.32 después de reescribirla en la forma:

$$R_n(k) = \sum_{m=0}^{N-1-k} [x(n+m)]w'(m)][x(n+m+k)w'(k+m)] \tag{2.36}$$

donde $w'(n) = w'(-n)$.

El cálculo de la autocorrelación tiene complejidad cuadrática pero hay métodos como el de Blankenship que aprovechan sus propiedades para ahorrar cálculos en base a una formulación recursiva de la autocorrelación.

La Figura 2.11 muestra tres ejemplos de funciones de autocorrelación determinadas para una señal de voz muestreada a 10kHz usando N = 401. Como se observa la autocorrelación fue evaluada para valores de $0 \leq k \leq 250$. En los primeros dos casos (a) y (b) se trata de segmentos vocalizados y en el tercero (c) de un segmento no-vocalizado, en el primer segmento los picos ocurren aproximadamente en múltiplos de 72 indicando un periodo de 7.2ms o una frecuencia fundamental de 140Hz, en el segundo caso (b) los

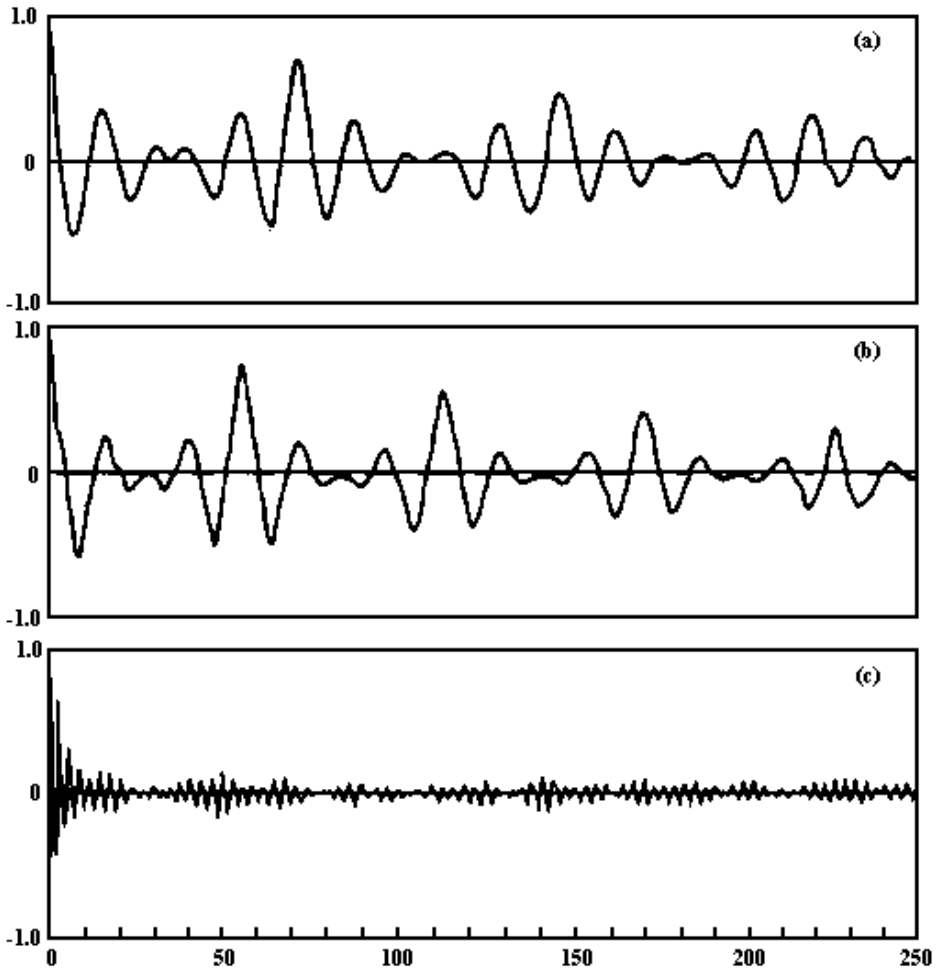

Figura 2.11: Función de autocorrelación para dos segmentos vocalizados (a) y (b); y un segmento no-vocalizado (c), usando una ventana rectangular de tamaño $N = 401$

picos ocurren en múltiplos de 58 muestras indicando un periodo de 5.8ms o una frecuencia fundamental aprox. de 173Hz. Finalmente el último caso (c) se trata de un segmento no-vocalizado no hay picos periódicos indicando ausencia de periodicidad en la señal, es decir, la función de autocorrelación para segmentos no-vocalizados es una forma de onda de alta frecuencia similar al ruido.

En [Rabiner and Schafer, 1978] dice que es mejor utilizar la ventana rectangular en lugar de la ventana de Hamming puesto que la ventana rectangular se puede observar más claramente los picos de la periodicidad que si se usara la ventana de Hamming.

El tamaño de la ventana N debe ser lo suficientemente pequeño como para reflejar los cambios rápidos de la señal pero lo suficientemente grande como para lograr estimar la periodicidad de la señal. Como regla general acorde a [Camarena, 2011], la ventana debe contener al menos dos periodos de la forma de onda. Sin embargo, es cumplicado saber cual es la forma de la onda, por lo que se recomienda utilizar tamaños de ventana de 30ms [Camarena, 2011], por ejemplo si la $f_s = 10,000Hz$, la ventana debe ser de tamaño $N = 300$, en el presente trabajo se utilizo un tamaño de ventana de 32ms (256muestras, potencia de 2 para FFT) y una $f_s = 8000Hz$.

2.6.1. Autocorrelación Modificada de tiempo corto

Debido a la longitud finita de la ventana utilizada en el cálculo de la autocorrelación de tiempo corto, hay cada vez menos datos involucrados en el cálculo de $R_n(k)$; a medida que k aumenta, esto conduce a a una reducción en la amplitud de los picos a medida que k aumenta, este efecto se puede apreciar en la Figura 2.12.

La autocorrelación de una onda estrictamente periódica tendría picos equiespaciados exáctamente de la misma amplitud, sin embargo eso no ocurre en la autocorrelación de tiempo corto, debido al efecto que se acaba de explicar (ver Figura 2.12); donde para la misma onda, se calcula $R_n(k)$ para diferentes tamaños de ventana, observe en la Figura 2.12 (c) el pico ubicado en $k = 72$ no es el del mismo tamaño comparado con los demás picos, esto debido a que para un tamaño de ventana de 125 ya son pocas muestras las involucradas en la sumatoria cuando se determina $R_n(75)$.

Para evitar este problema se puede utilizar la función de autocorrelación modificada de tiempo corto, la cual se define como:

$$\hat{R}_n(k) = \sum_{m=-\infty}^{\infty} x(m)w_1(n-m)x(m+k)w_2(n-m-k) \tag{2.37}$$

La ventana w_2 es una ventana mas grande que w_1 para permitir que se tome en cuenta muestras que quedan fuera del intervalo definido por w_1, para la ventana rectangular

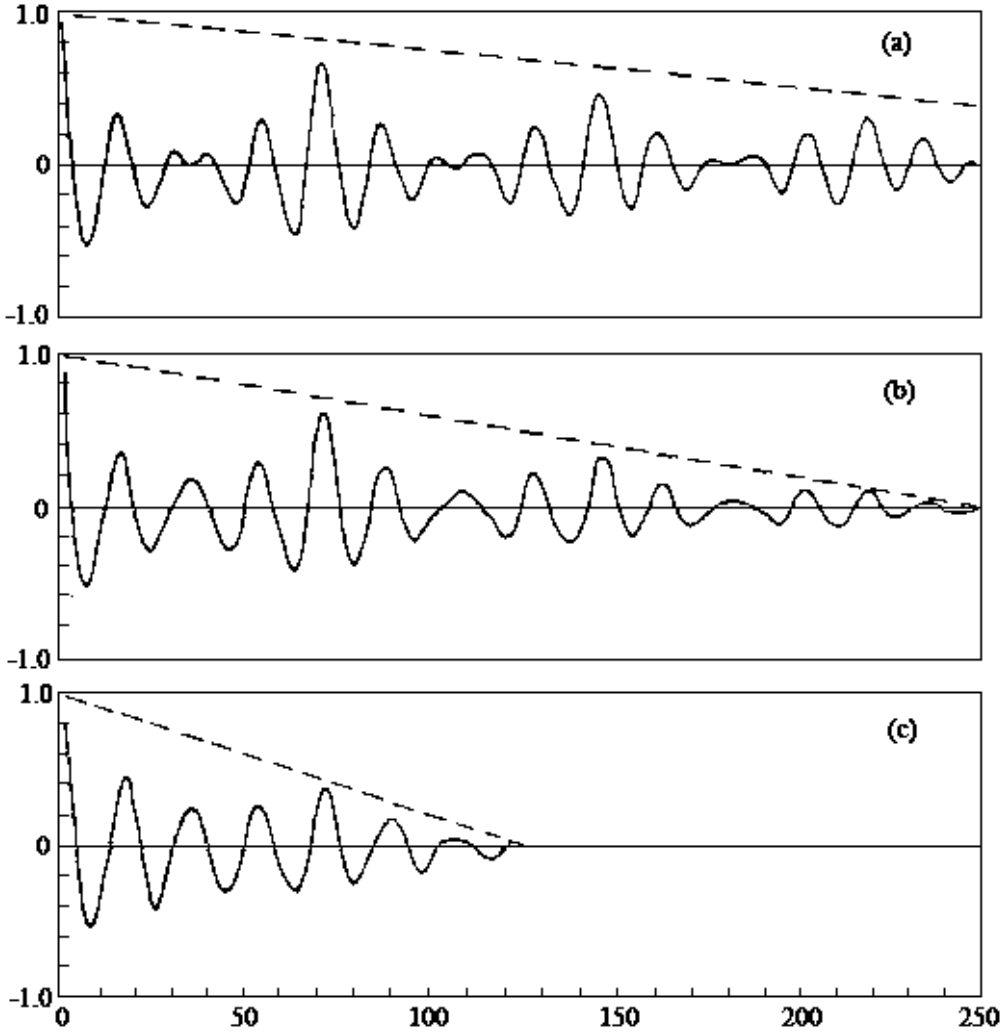

Figura 2.12: Efecto en la Autocorrelación de tiempo corto debido a que al aumentar k se involucran menos datos en el cálculo de $R_n(k)$

w_1 y w_2 se definen como:

$$w_1(m) = \begin{cases} 1 & 0 \leq m \leq N-1 \\ 0 & \text{De lo contrario} \end{cases} \tag{2.38}$$

$$w_2(m) = \begin{cases} 1 & 0 \leq m \leq N-1+K \\ 0 & \text{De lo contrario} \end{cases} \tag{2.39}$$

En lugar de desplazar la ventana hasta la posición m, se puede de manera equivalente, es decir, dejar la ventana fija y desplazar la señal para que la posición m coincida con el origen (donde se ubica la ventana), de manera que la siguiente definición es igualmente válida.

$$\hat{R}_n(k) = \sum_{m=-\infty}^{\infty} x(n+m)w_1(m)x(n+m+k)w_2(m+k) \tag{2.40}$$

Para la ventana rectangular, la definición de la ecuación 2.40 es equivalente a

$$\hat{R}_n(k) = \sum_{m=0}^{N-1} x(n+m)x(n+m+k) \ \forall \ 0 \leq k \leq K \tag{2.41}$$

donde K es el número de valores de $\hat{R}_n(k)$ que se ocuparán para el análisis de la señal.

2.7. Conclusiones

Para procesar una señal de audio analógica (por ejemplo el habla) esta debe ser digitalizada, es decir, muestreada a una cierta frecuencia f_s, y es necesario tener el modelo matemático que la describa. Para obtener el modelo matemático de algún sistema se hace uso de los sistemas LTI.

Por lo general cuando se graban palabras no se empieza a grabar en el momento que se activa el micrófono, esto hace que el micrófono grabe el ruido del ambiente o silencio, lo mismo ocurre cuando se termina de pronunciar la palabra es por esta razón que se necesita segmentar la señal de voz para encontrar donde inicia y donde termina la palabra,

y así poder separar la información a la cual se le quiere extraer características.

La autocorrelación es capaz de retener mucha información de señales de audio, es por esta razón que se utiliza para comprimir-descomprimir señales y para estimar la periodicidad de señales cortas.

El algoritmo de Blankenship permite calcular de manera eficiente la autocorrelación, y se utiliza cuando interesa obtener únicamente algunos valores, por ejemplo al momento de calcular los coeficientes LPC se necesita únicamente p valores, donde p es el orden del predictor, en estos casos se recomienda utilizar este algoritmo.

En este capítulo se explico como extraer características de una señal de audio en el dominio del tiempo, se abordo como segmentar una señal de voz para encontrar el inicio y final de una palabra. Sin embargo, una vez segmentada la señal se necesita algún tipo de representación paramétrica sumamente compactada para su posterior análisis, existe un gran número de posibilidades para representar a la señal de voz, como son los Espectogramas, los MFCC y los coeficientes LPC. Estas técnicas serán abordadas en el Capítulo 3.

Capítulo 3

Técnicas de extracción de características de la voz

Se ha visto que procesando la señal en el dominio del tiempo se puede determinar si hay voz en la señal de audio, en caso de que haya voz, el tipo de sonido, es decir vocalizado o no-vocalizado. Sin embargo, el reconocimiento de voz, ya sea de palabras aisladas o de habla continua dificilmente se puede hacer en el dominio del tiempo, de hecho, el sistema auditivo humano realiza un ánalisis espectral de la señal de audio dado que dentro de la cóclea se tiene una colección de neuronas en forma de vellosidades que oscilan a frecuencias que dependen de la longitud de estas [Camarena, 2011].

El proceso de convertir la señal de voz en algún tipo de representación paramétrica sumamente compactada para su posterior análisis se conoce como extracción de características. A continuación se describen las técnicas de extracción de características: Energía por Banda o Espectrogramas, MFCC y LPC.

3.1. Energía por Banda o Espectrogramas

3.1.1. Transformada de Fourier

Una de las herramientas más utilizada para el procesamiento digital de señales es la transformada de Fourier discreta de la función x. La Transformada de Fourier descompone las señales estacionarias en combinaciones lineales de ondas senos y cosenos. La transformada de Fourier continua se encuentra definida como

$$X(f) = \int_{-\infty}^{+\infty} x(t) \cdot e^{-j2\pi ft} dt \tag{3.1}$$

Para utilizar esta herramienta en computadoras se puede obtener una función de "tiempo discreto"muestreando una función de tiempo continuo $x_a(t)$, la cual produce una secuencia $x_a(nT)$ para valores enteros de n [Calderón, 2006, Camarena, 2011].

3.1.2. La Transformada Discreta de Fourier

En el caso de que una secuencia $x(n)$ sea periódica (con periodo T), es decir $x(n) = x(n+T)$ entonces se puede usar una serie de Fourier para representarla como una suma de senoides. La Transformada Discreta de Fourier (DFT, Discrete Fourier Transform) se define como:

$$X(k) = \sum_{k=0}^{T-1} x(n) e^{\frac{-j2\pi kn}{T}} \tag{3.2}$$

La exponencial compleja $e^{j(2\pi n/T)}$ es periódica con periodo T y existe una familia de exponenciales complejas también con periodo T dadas por

$$\phi_k(n) = e^{\frac{-j2\pi kn}{T}} \tag{3.3}$$

todas estas señales tienen frecuencias que son múltiplos de la misma frecuencia fundamental $2\pi/T$ [Calderón, 2006]. Así cuando $k = 0$ se le llama componente de CD de la señal, cuando $k = 1$ como armónico de frecuencia fundamental, $k = 2$ segundo armónico y así sucesivamente. La Transformada Discreta Inversa de Fourier (IDFT, Inverse Discrete Fourier Transform) se define por la ecuación 3.4

$$x(n) = \frac{1}{T}\sum_{k=0}^{T-1} X(k)e^{\frac{j2\pi kn}{T}} \tag{3.4}$$

La DFT $[X(k) = \sum_{n=0}^{T-1} x(n)e^{j2\pi kn/T}]$ tiene un kernel de tamaño proporcional al de la señal

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ \vdots \\ X(T) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & W_T^1 & W_T^2 & \dots & W_T^{T-1} \\ 1 & W_T^2 & W_T^4 & \dots & W_T^{2(T-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_T^{T-1} & W_T^{2(T-1)} & \dots & W_T^{(T-1)^2} \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ \vdots \\ x(T) \end{bmatrix}$$

donde $W_T = e^{-j2\pi/T}$.

La complejidad para determinar directamente la DFT es $O(T^2)$. Sin embargo, existe un algoritmo conocido como la Transformada Rápida de Fourier (FFT, Fast Fourier Transform), este puede determinar la DFT con solo $O(Tlog_2T)$ operaciones. El algoritmo FFT se conoció hasta mediados de los años sesentas a partir del trabajo de J.W. Cooley y J.W. Tukey. Se sabe de varios métodos de cálculo eficiente de la DFT, el de Danielson and Lanczos en 1942, proveé una de las mas claras derivaciones del algoritmo [Calderón, 2006, Camarena, 2011]. La implementación del algoritmo FFT en Java se puede consultar en el apéndice B.1.

3.1.3. Propiedades de la Transformada Discreta de Fourier

La Transformada de Fourier Discreta, presenta algunas propiedades un cuanto diferentes que la transformada continua. En la Tabla 3.1 y 3.2, se muestra un resumen de las propiedades de la DFT [Calderón, 2006].

Propiedad	**Señal en el tiempo** $x(n)$ y $y(n)$, son secuencias de números en el tiempo.	**Señal en la frecuencia** $X(k)$ y $Y(k)$, son las DFT de $x(n)$ y $y(n)$ respectivamente.
Periodicidad: Si $x(n)$ es una señal periódica con periódo T, el espectro $X(k)$ es también periódico con el mismo periódo T.	$x(n)$ tiene periodo T.	$X(k)$ tiene periodo T.
Linealidad: La suma ponderada de dos secuencias, es igual a la suma ponderada de sus DFT.	$ax(n) + by(n)$ a y b son números reales o complejos.	$aX(k) + bY(k)$ a y b son números reales o complejos
Desplazamiento en tiempo: Un desplazamiento en el tiempo es equivalente a multiplicar el espectro por una fase lineal.	$x(n - n_0)$ n_0 es un número entero. $x(n - n_0)$ es $x(n)$ desplazada n_0 posiciones.	$e^{-j\left(\frac{2\pi}{T}\right)n_0 k}X(k)$ n_0 es un número entero.
Desplazamiento en frecuencia: Un desplazamiento en el espectro de frecuencias es equivalente a multiplicar la secuencia en el tiempo por una fase lineal.	$e^{-j\left(\frac{2\pi}{T}\right)nk_0}x(n)$ k_0 es un número entero.	$X(k - k_0)$ k_0 es un número entero. $X(k - k_0)$ es $X(k)$ desplazada k_0 posiciones.
Inversión en tiempo: Si una secuencia se invierte en el tiempo, el espectro de frecuencias también se invierte.	$x(-n)$ $x(-n)$ es la secuencia invertida de $x(n)$.	$X(-k)$ El rango de la DFT, también se invierte.
Expansión en tiempo: Un escalamiento en el tiempo, provoca un escalamiento en la DFT.	$x(cn)$ c es un entero. $x(cn)$ es una señal multiplicada por un número.	$X(k/c)$ El rango de la DFT, también se escala.
Conjugación: La DFT de una secuencia $x(n)$ conjugada es el conjugado de la DFT de la secuencia $x(n)$ con el rango invertido.	$x^*(n)$ $x^*(n)$ es el conjugado de $x(n)$.	$X^*(-k)$ $X^*(-k)$ es el conjugado de la DFT con el rango invertido.
Convolución: Una convolución en tiempo es una multiplicación en la frecuencia.	$x(n) * y(n)$ $*$ es el símbolo de convolución.	$TX(k)Y(k)$
Multiplicación: Una multiplicación en el dominio del tiempo es equivalente a una convolución en la frecuencia.	$x(n)y(n)$	$X(k) * Y(k)$ $*$ es el símbolo de convolución.

Tabla 3.1: Resumen de propiedades de la Transformada Discreta de Fourier

Propiedad	**Señal en el tiempo**	**Señal en la frecuencia**
Diferenciación en tiempo: La derivada de una secuencia $x(n)$, es el espectro de frecuencia multiplicado por una fase lineal.	$x(n) - x(n-1)$	$\left(1 - e^{-j\left(\frac{2\pi}{T}\right)k}\right) X(k)$
Diferenciación en frecuencia: La derivada de la DFT se puede obtener a partir de multiplicar $x(n)$ por una fase lineal.	$\left(1 - e^{-j\left(\frac{2\pi}{T}\right)k}\right) x(n)$	$X(k) - X(k-1)$
Simetría conjugada: Si $x(n)$ es real, estonces la DFT es igual a su conjugada.	$x(n)$ es real.	$X(k) = X^*(k)$ $X^*(k)$ es el conjugado de $X(k)$.
Simetría: Si $x(n)$ es real y par, entonces la DFT es real y par.	$x(n)$ es real y par.	$X(k)$ es real y par.
Simetría: Si $x(n)$ es real e impar, entonces la DFT es imaginaria e impar	$x(n)$ es real e impar.	$X(k)$ es imaginaria e impar.
Relación de Parseval: Dice la relación que hay entre la energía de la secuencia $x(n)$ con la energía de su DFT.	$\sum_{n=0}^{T-1} \lvert x(n)\rvert^2$	$T \sum_{k=0}^{T-1} \lvert X(k)\rvert^2$

Tabla 3.2: Resumen de propiedades de la Transformada Discreta de Fourier (continuación).

3.1.4. Transformada discreta de Fourier de tiempo corto

La transformada de toda la señal de voz capturada no permite saber en que instante se presentaron los componentes de frecuencia que reporta la DFT, por ejemplo si se implementara un reconocedor de palabras aisladas no se podrían diferenciar carro de roca, saco de cosa, etc. puesto que ambas palabras tienen los mismos componentes de frecuencias en diferentes instantes de tiempo, sin embargo al aplicar DFT a la señal de cada una de dichas palabras, se pierde el dominio del tiempo (ver Figura 3.1). Lo que se necesita es hacer un análisis espectral, pero de segmentos lo suficientemente cortos de audio para que se pueda considerar que la señal en ese segmento corto es una señal estacionaria, es decir, que los componentes de frecuencia no cambian significatívamente. Los segmentos (o marcos) no deben ser demasiado cortos tampoco, por regla general, deben ser lo suficientemente grandes para que contenga dos periodos completos del componente de frecuencia mas baja que se esté considerando.

Es muy común utilizar marcos de 30ms, si se quiere utilizar el algoritmo FFT el tamaño de los marcos debe ser potencia de 2, en el presente trabajo se utilizan señales muestreadas a 8kHz, por lo tanto se utilizan marcos de 32ms (256 muestras). Se aplica DFT a la señal contenida en el marco (Ej. de 32ms), se avanza el marco y se vuelve a extraer la DFT, al desplazar el marco se sabrá cuales son las componentes de frecuencia de la señal contenida en el tiempo, es decir, se tiene información de la señal en el dominio de la frecuencia pero sin perder el dominio del tiempo (ver Figura 3.3).

En [Rabiner and Schafer, 1978, Camarena, 2011] indican que la frecuencia mínima que un tracto vocal puede emitir es 80 Hertz. Por lo tanto, los marcos se deben avanzar 12.5ms, sin embargo lo común es avanzar los marcos en 10ms [Camarena, 2011].

En la Figura 3.1 muestra a la izquierda tres señales diferentes entre sí. La primer señal y_1 es estacionaria, definida como:

$$y_1 = sen[2\pi(500)t] + sen[2\pi(1000)t] + sen[2\pi(2000)t] + sen[2\pi(3500)t] \qquad (3.5)$$

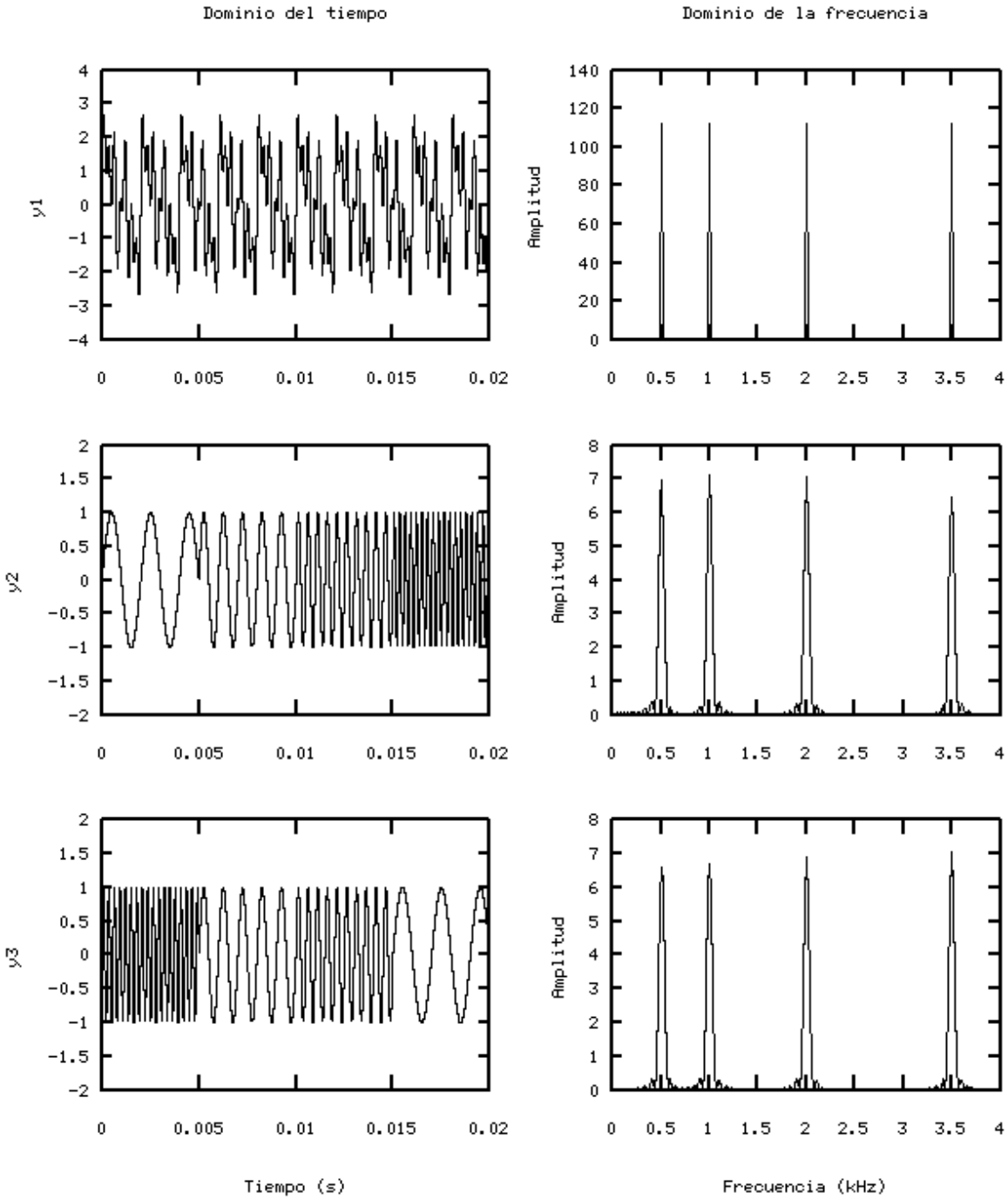

Figura 3.1: Señal y_1 (ecuación 3.5), señal estacionaria (izquierda) con su espectro de frecuencia (derecha). Señal y_2 (ecuación 3.6), señal no estacionaria (izquierda) con su espectro de frecuencia (derecha). Señal y_3 (ecuación 3.7), señal no estacionaria (izquierda) con su espectro de frecuencia (derecha).

Las siguientes dos señales son no estacionarias y_2 y y_3, definidas como:

$$y_2 = \begin{cases} sen[2\pi(500)t] & 0 \leq t < 0.005 \\ sen[2\pi(1000)t] & 0.005 \leq t < 0.01 \\ sen[2\pi(2000)t] & 0.01 \leq t < 0.015 \\ sen[2\pi(3500)t] & 0.015 \leq t < 0.02 \end{cases} \tag{3.6}$$

$$y_3 = \begin{cases} sen[2\pi(3500)t] & 0 \leq t < 0.005 \\ sen[2\pi(1000)t] & 0.005 \leq t < 0.01 \\ sen[2\pi(2000)t] & 0.01 \leq t < 0.015 \\ sen[2\pi(500)t] & 0.015 \leq t < 0.02 \end{cases} \tag{3.7}$$

A la derecha de la Figura 3.1 se observan los correspondientes espectros de las magnitudes de las DFTs de cada una de las señales de la izquierda. Si se desprecian las diferencias en amplitud y los rizos de magnitudes relativamente pequeñas que aparecen en la transformada de las señales y_2 y y_3, el aspecto de las tres señales en el dominio de la frecuencia es el mismo, por lo tanto no se puede distinguir una señal estacionaria de una no estacionaria si ambas señales tienen los mismos componentes de frecuencias, este es el problema que se resuelve al aplicar la DFT de tiempo corto.

Al tomar segmentos de tiempo corto de la señal de audio es altamente probable que dichos segmentos comiencen y/o terminen con un valor diferente de cero y además de valores distintos (entre el inicio y el final). La DFT entenderá esta diferencia entre el primer valor y el último como una discontinuidad, el cambio brusco que ve la DFT implicará un gran esfuerzo para reconstruirlo como una suma infinita de senoides lo cual se traduce en coeficientes de Fourier de alta frecuencia con alto contenido de energía [Camarena, 2011]. La ener-gía total de la señal en el dominio de la frecuencia es igual a la energía de la señal en el dominio del tiempo, eso es lo que dice el Teorema de Parseval, por lo tanto, la energía de los coeficientes de Fourier espurios de alta frecuencia será tomada de la energía de los coeficientes de Fourier verdaderos, a este efecto se le conoce como escurrimiento o efecto

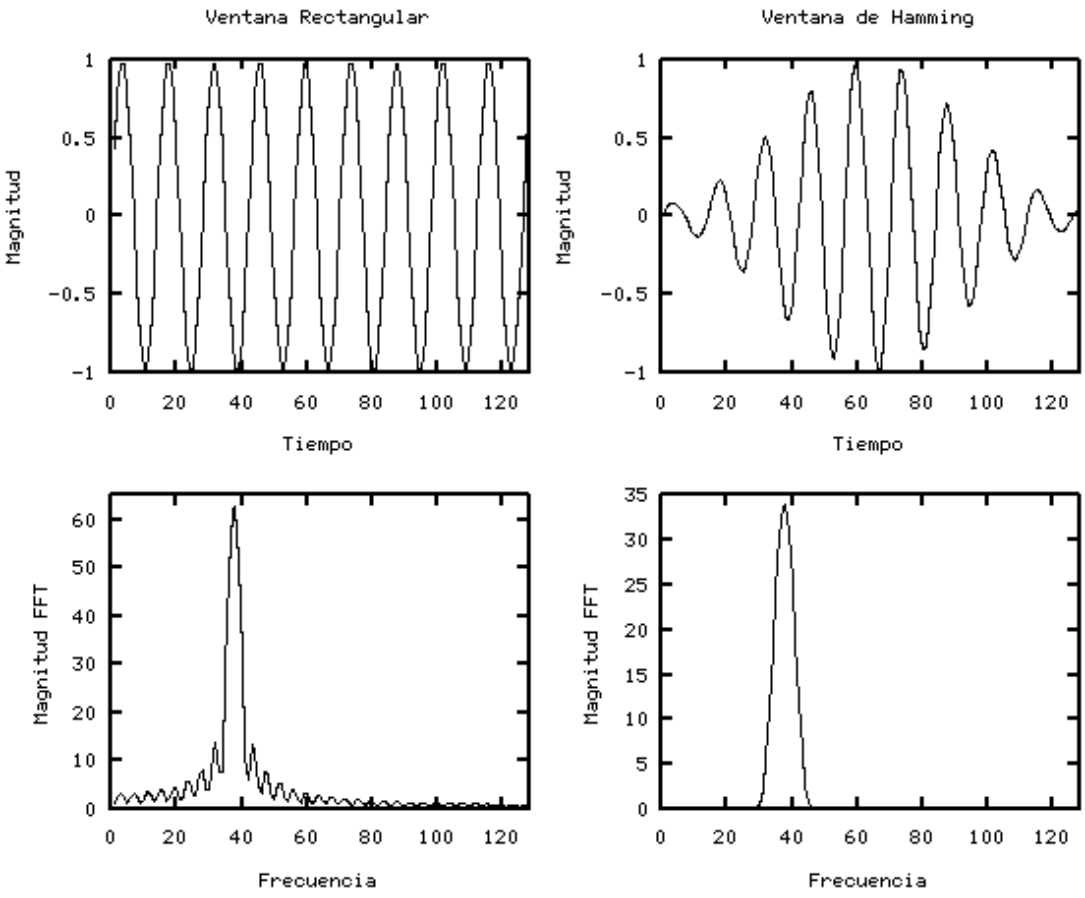

Figura 3.2: Efecto "leakage"(escurrimiento)

"leakage".

En la Figura 3.1 se aprecia el escurrimiento en las DFTs de las señales no estacionarias y_1 y y_2, y en la Figura 3.2. Para reducir el escurrimiento se debe aplicar una ventana como las mostradas en la Figura 2.7.

En la Figura 3.2 se muestra el espectro de frecuencias obtenido después de aplicar la DFT a una senoide pura, la cual no termina en el mismo valor con el que empieza, observe como parece que se escurre lo que debería ser un pico, en la misma figura se muestra la DFT de la misma senoide después de aplicarle la ventana de Hamming.

3.1.5. Escala de Bark

Es conveniente reproducir la manera en la que las personas identifican los sonidos, no todas las frecuencias se perciben con la misma sensibilidad, el oído humano percibe mejor las frecuencias bajas que las altas. La escala Bark fue diseñada para modelar el oído humano [Camarena, 2011], es una escala psicoacústica que fué propuesta por Eberhard Zwicker en 1961 en honor a Heinrich Barkhausen quien realizó las primeras mediciones subjetivas de la percepción de la intensidad sonora, también conocida como sensación sonora o sonoridad (En Inglés loudness).

La escala de Bark define 25 bandas críticas [Camarena, 2011], cada banda crítica corres-ponde con un segmento de la cóclea, en la Tabla 3.3 se describen dichas bandas, cada banda tiene un ancho de un Bark, la ecuación 3.8 se utiliza para pasar de Hertz a Barks. En el presente trabajo solo se utilizan 18 bandas por que la máxima frecuencia que se puede recuperar de la señal es 4000Hz, debido a que la frecuencia de muestreo es de 8000Hz.

$$f_\beta = 13tan^{-1}\left(\frac{0.76f}{1000}\right) + 3.5tan^{-1}\left(\frac{f}{750}\right)^2 \tag{3.8}$$

Donde f_β es frecuencia en Barks y f es la frecuencia en Hertz.

3.1.6. Determinación del espectrograma de la señal de voz mediante Bancos de Filtros

Para poder implementar un reconocedor de voz se necesita tanto del dominio del tiempo como del dominio de la frecuencia, para ello, se hace uso de la DFT de tiempo corto, de esta forma para cada marco de tiempo se obtiene su DFT, previa aplicación de una ventana como las mostradas en la Figura 2.7, en el presente trabajo se utiliza la ventana de Hamming, luego se recorre la ventana y se repite el proceso, el resultado es una secuencia de espectros de frecuencia.

Banda crítica	**Frecuencia inicial (Hz)**	**Frecuencia final (Hz)**
0	0	100
1	100	200
2	200	300
3	300	400
4	400	510
5	510	630
6	630	770
7	770	920
8	920	1080
9	1080	1270
10	1270	1480
11	1480	1720
12	1720	2000
13	2000	2320
14	2320	2700
15	2700	3150
16	3150	3700
17	3700	4400
18	4400	5300
19	5300	6400
20	6400	7700
21	7700	9500
22	9500	12000
23	12000	15500
24	15500	20000

Tabla 3.3: La escala de Bark (las 25 bandas críticas)

El espectrograma es un diagrama donde se despliega la cantidad de energía en función del tiempo, y de la frecuencia simultáneamente, este diagrama tiene una resolución en el tiempo que depende del ancho de los marcos; así como del traslape entre los marcos, mientras que la resolución en la frecuencia depende del ancho de los marcos y de la frecuencia de muestreo [Rabiner and Schafer, 1978].

Al implementar un espectrograma de la manera anterior se obtiene un arreglo por cada marco con demasiados valores, esto resulta demasiado costoso para almacenar, y casi imposibilita la comparación entre espectrogramas de dos elocuciones.

Un espectrograma con resolución en frecuencia considerablemente menor se puede obtener mediante un banco de filtros [Camarena, 2011]; a la salida de cada filtro se determina la energía de tiempo corto en el dominio del tiempo, de esa manera la resolución en frecuencia depende del número de filtros que conforman el banco, si por ejemplo se utiliza un filtro pasabanda por cada banda crítica de Bark (con ancho de banda de un Bark), se usarían 18 filtros y la resolución en frecuencia sería de 18 números por cada marco para construir el espectrograma de baja resolución.

Un espectograma de baja resolución presenta muchas ventajas [Rabiner and Schafer, 1978, Camarena, 2011], es fácil de implementar en hardware, es paralelizable y se puede usar como característica fundamental para reconocimiento de voz.

La Figura 3.3 muestra la señal definida por la ecuación 3.9 (imagen superior), y su espectrograma (imagen inferior); se utiliza la escala de Bark y para el suavizado del marco se utiliza la ventana de Hamming con un tamaño de 32ms (256 muestras). Como se puede observar, el espectrograma muestra el dominio de la frecuencia pero sin perder el dominio del tiempo, por que dice para un cierto tiempo cuales son los componentes de frecuencia. El espectrograma esta en escala de grises, es decir, negro significa cero en la magnitud del espectro de frecuencias, entre más blanco más grande es su magnitud del espectro de frecuencias.

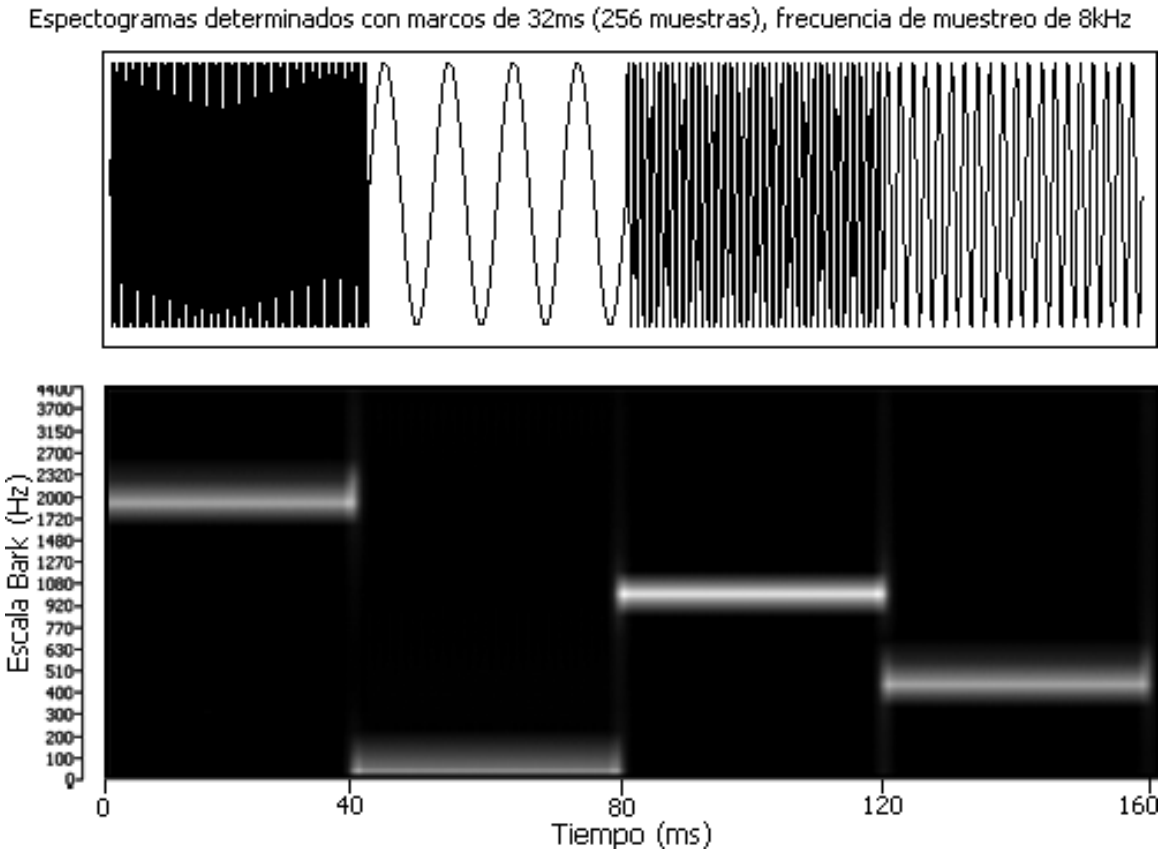

Figura 3.3: Señal determinada por la ecuación 3.9, en la parte de abajo se muestra su Espectograma usando marcos de 32ms (256muestras) con un desplazamiento de 10ms.

$$f(t) = \begin{cases} sin[2\pi(2000)t] & 0 \leq t < 40 \\ sin[2\pi(100)t] & 40 \leq t < 80 \\ sin[2\pi(1000)t] & 80 \leq t < 120 \\ sin[2\pi(500)t] & 120 \leq t < 160 \end{cases} \tag{3.9}$$

En resumen para obtener el espectrograma de una señal de voz (palabra o frase) se debe hacer lo siguiente:

1. Encontrar el inicio y final de la palabra o frase.

2. Obtener la transformada de Fourier de todos los marcos de la señal previamente suavizados aplicando una ventana (Hamming, Gaussiana, Hann, etc).

3. Calcular la potencia de la señal, $Im^2 + Re^2$.

4. Graficar tiempo con frecuencia, la frecuencia tiene que graficarse en escala logarítmica (escala de Bark).

3.1.7. Implementación de un sistema de reconocimiento de palabras aisladas mediante espectrogramas

Los ingenieros que trabajaban en los Laboratorios Bell de la At&T imprimían los espectrogramas de un conjunto reducido de palabras con las que hacían pruebas, entonces se dieron cuenta de que eran capaces de identificar una palabra con solo ver su espectrograma, esto significaba trasladar un problema de identificación de una señal de audio a un problema de reconocimiento de imágenes, si bien esto podría significar una herramienta para los sordos (que no sean ciegos), el hecho es que los espectrogramas se pueden utilizar como una caracterización de la señal de voz para realizar reconocimiento de palabras aisladas.

En la Figura 3.4 se muestran los espectrograma de la palabra /atrás/ y /bueno/, ambas palabras pronunciadas dos veces cada una, se puede observar que los espectrogramas de la palabra /atrás/ (a), ambos son muy parecidos, igualmente los espectrogramas de la palabra /bueno/ (b), sin embargo el espectrograma de la palabra /atrás/ (a) comparado con el de la palabra /bueno/ (b) ambos son muy diferentes.

El espectrograma queda almacenado en una matriz, el tamaño de esta será de número de bandas críticas por número de marcos de la señal. Por ejemplo, si la señal es muestreada a 8kHz con las 18 primeras bandas críticas de Bark es suficiente, la palabra tiene una duración de 0.5s, los marcos son de 32ms (256 muestras) y se avanza cada 10ms, entonces se tendra una matriz de (18)(50), es decir, 900 valores. La matriz obtenida es almacenada en el diccionario con la etiqueta del nombre de la palabra y el número de marcos de la misma, el formato se describe a continuación.

Con este sistema de extracción de características se puede implementar un progra-

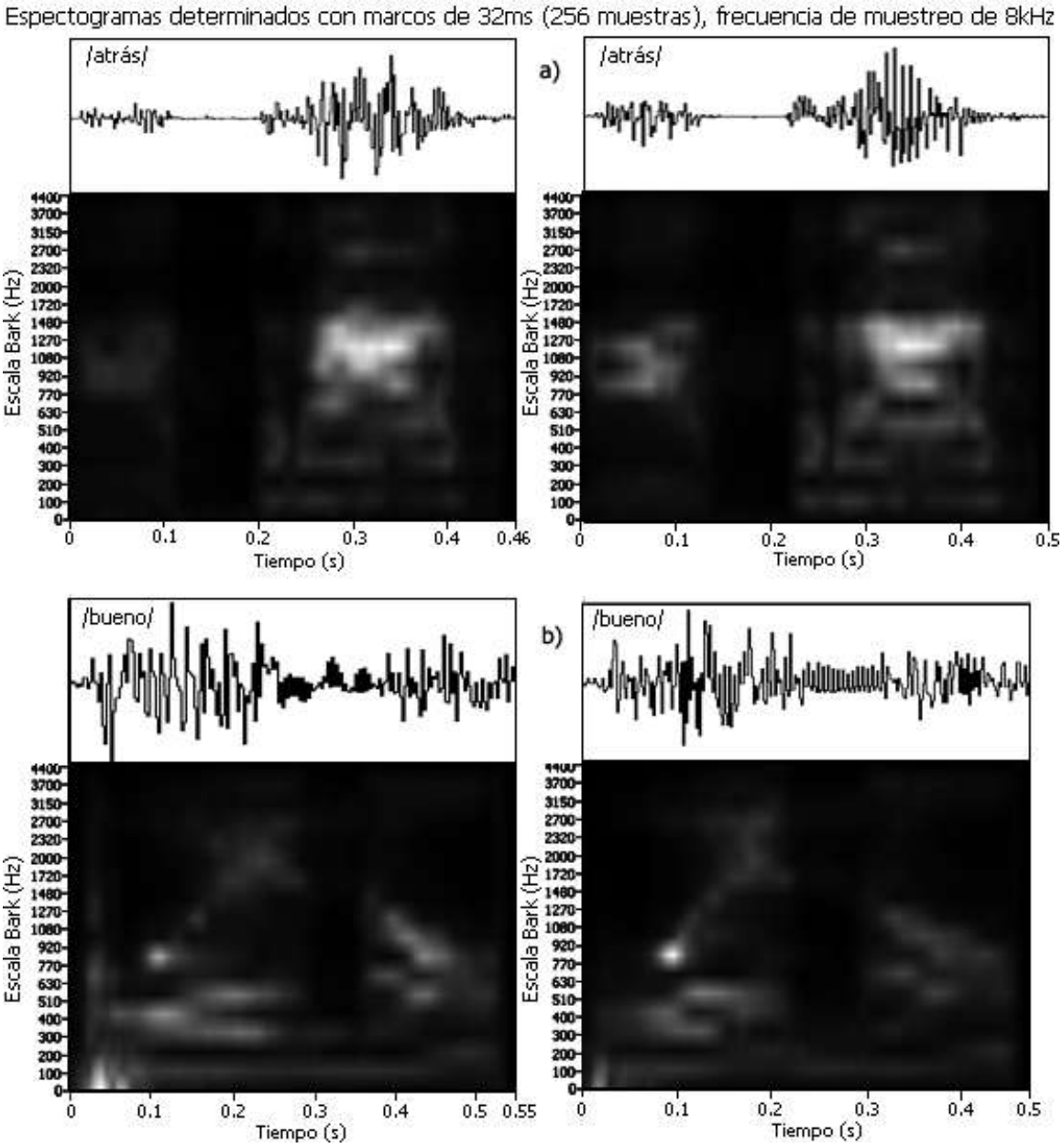

Figura 3.4: Espectrogramas de las palabras, a) /atrás/, b) /bueno/ ambas pronunciadas dos veces cada una por el mismo locutor.

ma que por cada palabra pronunciada ante el micrófono, agregue la entrada correspondiente al diccionario del sistema de reconocimiento de voz.

El diccionario es un archivo con las etiquetas y características de las palabras conocidas por el sistema, en el presente trabajo se utiliza siguiente formato:

<Etiqueta> <Núm. de marcos>
1, <Energía banda 0>, <Energía banda 1>, ..., <Energía banda 17>
2, <Energía banda 0>, <Energía banda 1>, ..., <Energía banda 17>
⋮
Núm. de marcos, <Energía banda 0>, <Energía banda 1>, ..., <Energía banda 17>

<Etiqueta> <Núm. de marcos>
1, <Energía banda 0>, <Energía banda 1>, ..., <Energía banda 17>
2, <Energía banda 0>, <Energía banda 1>, ..., <Energía banda 17>
⋮
Núm. de marcos, <Energía banda 0>, <Energía banda 1>, ..., <Energía banda 17>

Donde *<Etiqueta>* es el nombre de la palabra almacenada, y *<Núm. de marcos>* depende de la duración de la palabra (número de columnas de la matriz).

El sistema de reconocimiento procede de la misma manera que el programa con el que se creó el diccionario en lo concerniente a la extracción de características, excepto que en lugar de agregar otra entrada, se recorre comparando (la comparación se puede hacer usando DTW, HMM, etc.) cada matriz de energía de la palabra recién capturada con cada matriz de energía de las palabras almacenadas, detectando aquella con la que la distancia es menor (en el caso de DTW) o con la que la probabilidad es mayor (en el caso de HMM), si esta distancia es menor o si la probabilidad es mayor a un cierto umbral, entonces reportar ocurrencia de la palabra, de lo contrario declarar que la palabra es desconocida para el sistema. Las técnicas de reconocimiento DTW y HMM se abordan en el capítulo 4.

3.2. MFCC (Mel Frequency Cepstral Coefficients)

Los MFCC se basan en la variación de las bandas críticas del oído con la frecuencia [Camarena, 2011]. Para obtener las características fonéticas más importantes de una señal de audio se usan filtros separados de forma lineal para las bajas frecuencias, y de manera logarítmica para las altas frecuencias. Esta separación se representa mediante la escala de Mel.

3.2.1. Escala de Mel

La Escala Mel [Camarena, 2011], propuesta por Stevens, Volkman y Newmann en 1937, es una escala musical perceptual de tonos juzgados como intervalos equiespaciados por parte de observadores. Los estudios psicoacústicos muestran que la percepción humana de las frecuencias no sigue una escala lineal. Por esto se debe hacer un escalado de las frecuencias en Hz a una escala subjetiva conocida como la escala Mel. Lo que se hace es dividir el espectro en un banco de filtros, mucho más estrechos y espaciados en forma lineal por debajo de 1 kHz y muy amplios y espaciados de manera logarítmica por encima de esta cantidad. De este modo, se da mayor importancia a la información contenida en las bajas frecuencias en consonancia con el comportamiento del oído humano [Rabiner and Schafer, 1978].

Para convertir f de Herts a Mels se emplea:

$$\varpi = 2595 log_{10}\left(\frac{f}{700} + 1\right) \tag{3.10}$$

En la Figura 3.5 se muestra la curva de Mels vs Hertz.

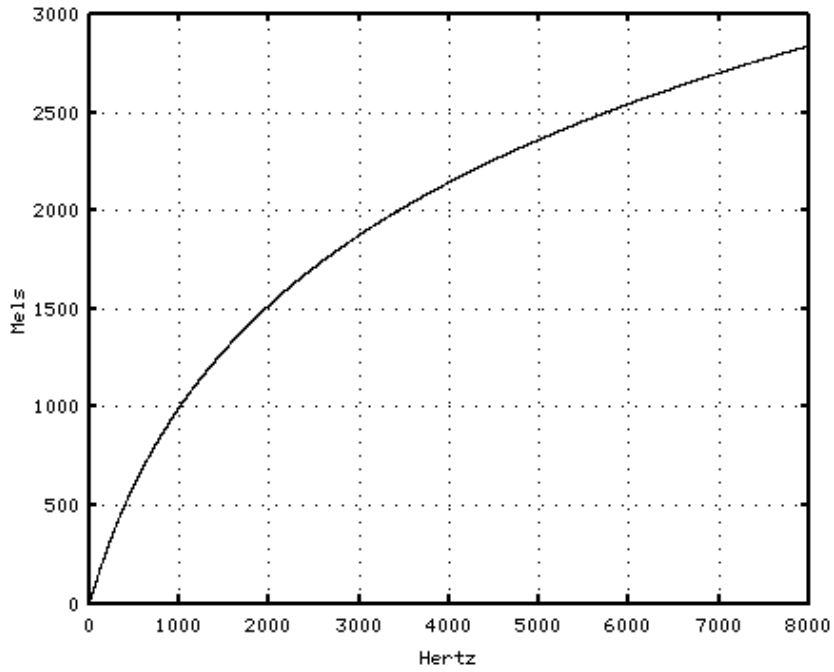

Figura 3.5: Escala de Mel

3.2.2. Filtros triangulares de Mel

El banco de filtros de Mel [Sigurdsson et al., 2006], es una colección de filtros definidos por sus frecuencias centrales $f_c(m)$.

$$H(k,m) = \begin{cases} 0 & f(k) < f_c(m-1) \\ \frac{f(k)-f_c(m-1)}{f_c(m)-f_c(m-1)} & f_c(m-1) \leq f(k) < f_c(m) \\ \frac{f(k)-f_c(m+1)}{f_c(m)-f_c(m+1)} & f_c(m) \leq f(k) < f_c(m+1) \\ 0 & f(k) \geq f_c(m+1) \end{cases} \quad k = 1, 2, \ldots, N \; ; \; m = 1, 2, \ldots, M \tag{3.11}$$

Donde k es el $k-$ésimo valor del m-ésimo filtro triangular. N es el tamaño de la señal y M es el número de filtros que conforman el banco de filtros.

Este banco de filtros tiene un diseño triangular 50 % solapado, con ancho de banda y separación determinado por un intervalo Mel constante. En la Figura 3.6 se muestra un banco de filtros triangulares de Mel con un ancho de banda de 177.5 Mels. Aunque en

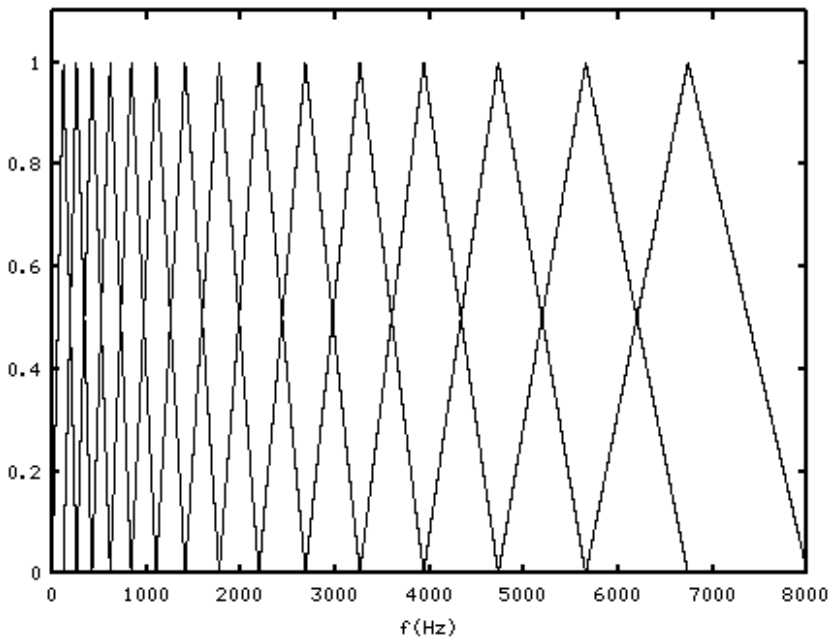

Figura 3.6: Filtros Triangulares de Mel, ancho de banda de 177.5 Mels

Hertz, las frecuencias centrales están repartidas logarítmicamente, en la escala de Mel las frecuencias centrales están linealmente distribuidas [Sigurdsson et al., 2006], la separación entre cada filtro triangular de Mel y el que le sigue esta dada por la ecuación 3.12:

$$\Delta\varpi = \frac{\varpi_{max} - \varpi_{min}}{M+1} \tag{3.12}$$

donde ϖ_{max} y ϖ_{min} son los límites del rango de frecuencias de la señal bajo análisis correspondientes a f_{max} y f_{min} respectivamente, y M es el número de filtros que conforman el banco de filtros. Las frecuencias centrales de Mel están dadas por:

$$\varpi_c(m) = m(\Delta\varpi)\ m = 1, 2, \ldots, M \tag{3.13}$$

Para obtener las frecuencias centrales en Hertz, se utiliza la inversa de la ecuación 3.10 y se obtiene lo siguiente:

$$f_c(m) = 700\left(10^{\varpi_c(m)/2595} - 1\right) \tag{3.14}$$

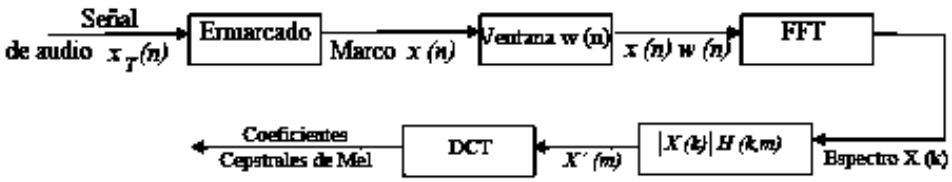

Figura 3.7: Proceso de obtención de los MFCC

3.2.3. Determinación de los coeficientes MFCC

Los coeficientes cepstrales de Mel [Camarena, 2011], se determinan de la siguiente manera:

1. La señal de audio es dividida en marcos de tiempo (ejemplo de 30ms), se aplica una ventana (por ejemplo de Hamming, Hann, Gaussiana, etc.) a cada marco de tiempo.

2. Se aplica la DFT, en otras palabras se determina $X(k)$ mediante:

$$X(k) = \sum_{n=0}^{N-1} [x(n)w(n)]e^{-j2\pi kn/N} \;\; k = 0, 1, \ldots, N-1 \tag{3.15}$$

   donde N es el tamaño del marco (para utilizar la FFT se necesita que sea potencia de 2), k corresponde a la frecuencia $f(k) = kf_s/N$, (f_s es la frecuencia de muestreo) y $w(n)$ es la ventana por ejemplo la de Hamming o alguna de las que se muestran en la Figura 2.7.

3. La magnitud del espectro $|X(k)|$, se escala logarítmicamente tanto en frecuencia como en magnitud, para escalar logarítmicamente en frecuencia se utiliza el banco de filtros de Mel $H(k, m)$ (ecuación 3.11), y luego se escala logarítmicamente la magnitud para obtener:

$$X'(m) = ln\left(\sum_{k=0}^{N-1} [|X(k)|\, H(k, m)]\right) \tag{3.16}$$

   para $m = 1, 2, \ldots, M$, donde M es el número de filtros del banco, por supuesto $M \ll N$ (M debe ser mucho menor que N).

4. El último paso para obtener los Coeficientes Cepstrales de Mel es aplicar la Transformada Coseno Discreta (DCT, Discrete Cosine Transform) mediante:

$$c(l) = \sum_{m=1}^{M} X'(m) cos \left[\frac{l\pi\left(m - \frac{1}{2}\right)}{M} \right] \quad l = 1, 2, \ldots, M \tag{3.17}$$

donde $c(l)$ es el l-ésimo coeficiente cepstral de Mel.

En la Figura 3.7 se muestra el proceso para obtener los coeficientes MFCC de una señal de audio.

3.2.4. Sistema de reconocimiento de voz usando los MFCC

El reconocedor de palabras aisladas en base a coeficientes cepstrales de Mel es prácticamente igual al sistema de reconocimiento de palabras aisladas mediante espectrogramas descrito en la sección 3.1.7. Ambos sistemas caracterizan una palabra mediante una matriz de flotantes, en ambos casos la matriz tiene un número fijo de renglones, por que este depende del número de filtros de mel (en el caso de MFCC) o del número de bandas críticas (en el caso de Espectrogramas), y un número variable de columnas, pues este depende de la duración de la locución. La única diferencia entre ambos sistemas es el denominado *front end*, es decir, el módulo de extracción de características de la señal de voz y por supuesto la interpretación que debe de hacerse de dicha matriz, en el caso del reconocedor mediante espectrogramas descrito en 3.1.7 cada columna corresponde con una banda crítica de Bark, mientras que en el reconocedor hecho mediante coeficientes MFCC, cada columna corresponde con los coeficientes de la DCT del banco de filtros triangulares de Mel.

Para implementar un sistema de reconocimiento de voz usando los MFCC [Camarena, 2011], se propone utilizar 15 filtros triangulares de Mel, es decir $M = 15$, una frecuencia inferior f_{min} de 80 Hertz puesto que es la frecuencia mínima que un tracto vocal puede emitir, y la frecuencia superior f_{max} de 4kHz si la frecuencia de muestreo f_s es de 8Khz.

3.3. Codificación Lineal Predictiva

Una de las técnicas mas potentes y predominante en el análisis de la señal de voz es el método de análisis lineal predictivo [Camarena, 2011]. La importancia de esta técnica se encuentra tanto en su capacidad para proporcionar estimaciones muy precisas de los parámetros de voz, como en su relativo bajo costo computacional.

La idea básica del análisis lineal predictivo [Camarena, 2011], es que una muestra de voz puede ser aproximada como una combinación lineal de muestras de voz anteriores. Para poder hacer esto, es necesario minimizar la suma de las diferencias al cuadrado (en un intervalo finito) entre la actual muestra de voz y una linealmente prevista.

3.3.1. Principios básicos del análisis lineal predictivo

Aunque la codificación lineal predictiva se utilice ampliamente para extraer características de la señal de voz, esta se contempló originalmente como un problema independiente del reconocimiento y de la síntesis de la voz [Rabiner and Schafer, 1978], se deseaba codificar la voz para: poder transmitirla por un canal de comunicación sin demandar mucho ancho de banda; así como poder almacenarla sin que requiera mucho espacio de disco.

Los parámetros del modelo de la Figura 3.8 son: la frecuencia f del tren de impulsos, la ganancia G, la posición del interruptor vocalizado/no-vocalizado y los coeficientes a_k del filtro digital, todos estos parámetros varían con el tiempo aunque no presentan cambios bruscos, excepto el interruptor. El tracto vocal y la excitación glotal [Rabiner and Schafer, 1978], son representados mediante un filtro digital variante en el tiempo cuya función del sistema en estado estacionario es de la forma:

$$\frac{S(z)}{U(z)} = \frac{G}{1 - \sum_{k=1}^{p} a_k z^{-k}} \tag{3.18}$$

Donde $S(z)$ es la Transformada Z de la respuesta del filtro digital, $U(z)$ es la Transformada Z de la señal de entrada al filtro digital y p es el orden del predictor.

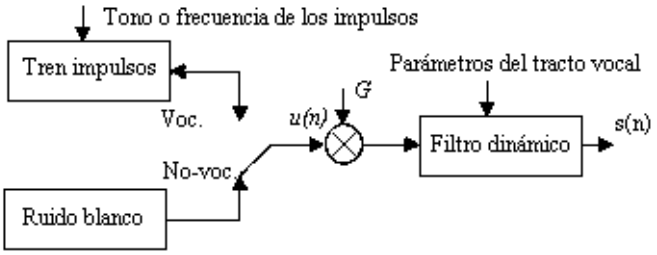

Figura 3.8: Diagrama de bloques del modelo simplificado para la producción de voz

Aplicando Transformada Z inversa a la ecuación 3.18:

$$s(n) = \sum_{k=1}^{p} a_k s(n-k) + G u_e(n) \tag{3.19}$$

Un predictor lineal predice una muestra de voz en base a las p muestras anteriores, así:

$$\tilde{s}(n) = \sum_{k=1}^{p} \alpha_k s(n-k) \tag{3.20}$$

El error de predicción es la diferencia entre las muestras reales de voz y las muestras obtenidas por el predictor:

$$e(n) = s(n) - \tilde{s}(n) = s(n) - \sum_{k=1}^{p} \alpha_k s(n-k) \tag{3.21}$$

El problema básico del análisis lineal predictivo [Rabiner and Schafer, 1978], es determinar el conjunto de coeficientes predictores α_k directamente de la señal de voz de tal manera que se obtenga una buena estimación de las propiedades espectrales de la señal de voz.

Comparando las ecuaciónes 3.19 y 3.21 se concluye que para poder afirmar que los coeficientes predictores y los parámetros del tracto vocal coinciden, es decir $\alpha_k = a_k$ entonces deberá cumplirse $e(n) = G u_e(n)$ lo cual significa que $e(n)$ consiste de un tren de impulsos, o sea que $e(n)$ vale cero la mayor parte del tiempo [Rabiner and Schafer, 1978, Camarena, 2011]. El error de predicción de tiempo corto se define como:

$$E(n) = \sum_m e_n^2(m) = \sum_m [s_n(m) - \tilde{s}_n(m)]^2 = \sum_m \left[s_n(m) - \sum_{k=1}^{p} \alpha_k s_n(m-k) \right]^2 \tag{3.22}$$

donde $s_n(m)$ es la m-ésima muestra contada desde el inicio del marco que comienza en la n-ésima muestra, es decir $s(n+m)$, y $e_n^2(m)$ es el error de predicción del marco que comienza en la n-ésima muestra al cuadrado, es decir $e^2(n+m)$.

Para encontrar los valores de α_k que minimizan E_n se hace $\partial E_n / \partial \alpha_i = 0$, para todo $i = 1, 2, \ldots, p$.

$$2 \sum_m \left[s_n(m) - \sum_{k=1}^{p} \alpha_k s_n(m-k) \right] \frac{\partial \left[- \sum_{k=1}^{p} \alpha_k s_n(m-k) \right]}{\partial \alpha_i} = 0 \; i = 1, 2, \ldots, p \tag{3.23}$$

donde:

$$\frac{\partial \left[- \sum_{k=1}^{p} \alpha_k s_n(m-k) \right]}{\partial \alpha_i} = -s_n(m-i) \; i = 1, 2, \ldots, p \tag{3.24}$$

por la tanto:

$$2 \sum_m \left[s_n(m) - \sum_{k=1}^{p} \alpha_k s_n(m-k) \right] [-s_n(m-i)] = 0 \; i = 1, 2, \ldots, p \tag{3.25}$$

de donde:

$$\sum_m s_n(m-i) s_n(m) = \sum_{k=1}^{p} \alpha_k \sum_m s_n(m-i) s_n(m-k) \; i = 1, 2, \ldots, p \tag{3.26}$$

Definiendo:

$$\phi_n(i,k) = \sum_m s_n(m-i) s_n(m-k) \tag{3.27}$$

Se puede escribir la ecuación 3.26 en forma compacta como:

$$\sum_{k=1}^{p} \alpha_k \phi_n(i,k) = \phi_n(i,0) \; i = 1, 2, \ldots, p \tag{3.28}$$

Para encontrar la solución para los coeficientes del predictor óptimo, se debe calcular los valores de $\phi_n(i-k)$ para $1 \leq i \leq p$ y $0 \leq k \leq p$, una vez que esto se hace únicamente se tiene que resolver la ecuación 3.28 para obtener los $\alpha_k's$.

3.3.2. Método de la Autocorrelación

Para resolver el sistema de ecuaciones dado por la ecuación 3.28, existen varios métodos, entre ellos el método de la autocorrelación [Camarena, 2011]. En las ecuaciones anteriores, los límites de las sumatorias que utilizan a m como índice se han dejado sin especificar, sin embargo, al referirnos al error de predicción de tiempo corto, el intervalo coincide con el ancho N de los marcos, así $s(n+m)w(m)$ vale cero fuera del intervalo $[0, N-1]$ y el error de predicción de tiempo corto solo tiene valores distintos de cero dentro del intervalo $[0, N+p-1]$, por esta razón (3.27) se convierte en:

$$\phi_n(i,k) = \sum_{m=0}^{N+p-1} s_n(m-i)s_n(m-k) \; i = 1, 2, \ldots, p \quad k = 0, 1, 2, \ldots, p \tag{3.29}$$

Haciendo el cambio de variable $\tau = m - i$, se tiene que $m - k = \tau + i - k$

$$\phi_n(i,k) = \sum_{\tau=0}^{N-1-(i-k)} s_n(\tau)s_n(\tau+i-k) \; i = 1, 2, \ldots, p \quad k = 0, 1, 2, \ldots, p \tag{3.30}$$

Función de autocorrelación:

$$R_n(k) = \sum_{m=0}^{N-1-k} s_n(m)s_n(m+k) \tag{3.31}$$

En la ecuación 3.30, se puede apreciar que $\phi_n(i,k)$ es idéntica a la función de autocorrelación (3.31) evaluada en $(i-k)$. En vista de que la función de autocorrelación es una función par, se cumple que:

$$\phi_n(i,k) = R_n(i-k) = R_n(|i-k|) \tag{3.32}$$

Entonces la ecuación 3.28 se puede expresar como:

$$\sum_{k=1}^{p} \alpha_k R_n(|i-k|) = R_n(i) \; i = 1, 2, \ldots, p \tag{3.33}$$

El conjunto de ecuaciones dadas por la ecuación 3.33 puede ser expresada en forma

matricial como:

$$\begin{bmatrix} R_n(0) & R_n(1) & R_n(2) & \cdots & R_n(p-1) \\ R_n(1) & R_n(0) & R_n(1) & \cdots & R_n(p-2) \\ R_n(2) & R_n(1) & R_n(0) & \cdots & R_n(p-3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ R_n(p-1) & R_n(p-2) & R_n(p-3) & \cdots & R_n(0) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \vdots \\ \alpha_p \end{bmatrix} = \begin{bmatrix} R_n(1) \\ R_n(2) \\ R_n(3) \\ \vdots \\ R_n(p) \end{bmatrix} \tag{3.34}$$

3.3.3. Método recursivo de Durbin para solucionar las ecuaciones de autocorrelación

Para encontrar los coeficientes LP α_k, se debe resolver el sistema de ecuaciones 3.34, este sistema de ecuaciones se puede representar como:

$$R\Lambda = R_v \tag{3.35}$$

Donde: Λ es el vector de $\alpha_1, \alpha_2, \ldots, \alpha_p$, R es la matriz de autocorrelaciones de $(p)(p)$, y R_v es el vector de autocorrelaciones $R_n(1), R_n(2), \ldots, R_n(p)$.

La matriz de autocorrelación R es una matriz de Toeplitz dado que es simétrica y los valores a lo largo de cualquier diagonal son un mismo valor. Los métodos mas conocidos acorde a [Camarena, 2011] para solucionar un sistemas de ecuaciones con estas características son los de Levinson y Robinson, pero el método más conocido se le conoce como *procedimiento recursivo de Durbin.*

El procedimiento recursivo de Durbin comienza con un predictor de primer orden, es decir, de un solo coeficiente e incrementa el orden recursívamente utilizando las soluciones de orden inferior, para obtener soluciones de órdenes superiores. La notación $\alpha_i^{(p)}$ denota el i-ésimo coeficiente de un predictor de orden p. El algoritmo de Durbin resuelve primero un sistema predictor de primer orden ($p = 1$). El sistema de ecuaciones se reduce a:

$$\alpha_1^{(1)} R_n(0) = R_n(1) \tag{3.36}$$

Despejando $\alpha_1^{(1)}$ de la ecuación anterior:

$$\alpha_1^{(1)} = \frac{R_n(1)}{R_n(0)} \tag{3.37}$$

El error de predicción para el predictor de primer orden se calcula mediante:

$$E_n = R_n(0) - \sum_{k=1}^{p} \alpha_k R_n(k) \tag{3.38}$$

El error de predicción para el predictor de primer orden se calcula mediante:

$$E_n^{(1)} = R_n(0) - \alpha_1^{(1)} R_n(1) \tag{3.39}$$

Posteriormente se soluciona un sistema predictor de segundo orden ($p = 2$), el sistema de ecuaciones a solucionar es:

$$\begin{bmatrix} R_n(0) & R_n(1) \\ R_n(1) & R_n(0) \end{bmatrix} \begin{bmatrix} \alpha_1^{(2)} \\ \alpha_2^{(2)} \end{bmatrix} = \begin{bmatrix} R_n(1) \\ R_n(2) \end{bmatrix} \tag{3.40}$$

Solucionando el sistema de ecuaciones para $\alpha_2^{(2)}$, se obtiene:

$$\alpha_2^{(2)} = \frac{R_n(0)R_n(2) - [R_n(1)]^2}{[R_n(0)]^2 - [R_n(1)]^2} \tag{3.41}$$

Dividiendo numerador y denominador por $R_n(0)$

$$\alpha_2^{(2)} = \frac{R_n(2) - [R_n(1)]^2/R_n(0)}{R_n(0) - [R_n(1)]^2/R_n(0)} \tag{3.42}$$

Sustituyendo ahora el resultado al que se llegó con el predictor de primer orden, se obtiene:

$$\alpha_2^{(2)} = \frac{R_n(2) - R_n(1)\alpha_1^{(1)}}{R_n(0) - R_n(1)\alpha_1^{(1)}} \tag{3.43}$$

Pero se sabe que $E_n^{(1)} = R_n(0) - \alpha_1^{(1)} R_n(1)$, por tanto:

$$\alpha_2^{(2)} = \frac{R_n(2) - R_n(1)\alpha_1^{(1)}}{E_n^{(1)}} \tag{3.44}$$

Se observa como un coeficiente de predicción de segundo orden $\alpha_2^{(2)}$ se puede calcular con pocas operaciones aritméticas si se conocen ya los coeficientes del predictor del orden inmediato inferior (en este caso $\alpha_1^{(1)}$). Algo similar se hace para $\alpha_1^{(2)}$. Una vez que se ha solucionado el sistema de segundo orden se procede a solucionar el sistema de tercer orden y así sucesivamente hasta llegar al orden deseado [Camarena, 2011].

Algoritmo de Levinson-Durbin [Iosu, 1999, Rabiner and Schafer, 1978]:

1. $E_n^{(0)} = R_n(0)$

2. Desde $i = 1$ hasta p

   $$k_i = \left[R_n(i) - \sum_{j=1}^{i-1} \alpha_j^{(i-1)} R_n(i-j) \right] / E_n^{(i-1)}$$

   $$\alpha_i^{(i)} = k_i$$

   Desde $j = 1$ hasta $i - 1$

   $$\alpha_j^{(i)} = \alpha_j^{(i-1)} - k_i \alpha_{i-j}^{(i-1)}$$

   Si $i < p$ entonces

   $$E_n^{(i)} = (1 - k_i^2) E_n^{(i-1)}$$

En el proceso de calcular los coeficientes predictores para un predictor de orden p se deben calcular los coeficientes de todos los predictores de órdenes inferiores a p. El error de predicción se puede estar monitoreando así como los coeficientes PARCOR (PARtial CORrelation) (k_i) [Camarena, 2011].

El espectro de voz a ser analizado se puede representar adecuadamente con 2 polos (Ej. una pareja de polos complejos conjugados) por kHz [Camarena, 2011, Rabiner and Schafer, 1978], entonces se requiere un número de polos igual a la frecuencia de muestreo en kHz debido a la contribución del tracto vocal al espectro de la voz. Adicionalmente, se requieren 3 o 4 polos mas para representar adecuadamente la fuente de excitación (pulmones, valvula glotal y cuerdas vocales), y la radiación del sonido hacia el medio (labios). En vista de que el número de polos coincide con el grado del polinomio en el denominador de la función sistema, el número total de polos debe ser igual a p. Por ejemplo, para una frecuencia de muestreo de 8 KHz se recomienda p = 12.

En la Figura 3.9, se observa como el rms del error de predicción disminuye al aumentar p, también el hecho de que el error de predicción siempre es mayor para sonidos

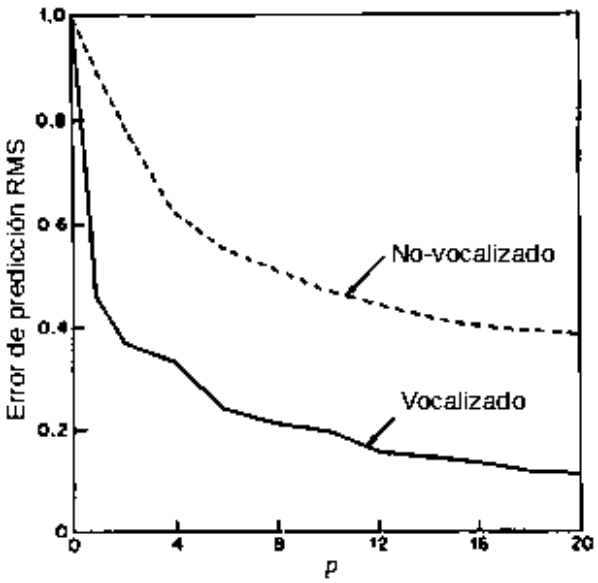

Figura 3.9: Error de predicción en RMS vs p

no-vocalizados que para sonidos vocalizados.

3.3.4. Ventajas de utilizar el Algoritmo de Durbin

1. Se puede monitorear la señal de error $e(n)$ definida como:

$$e(n) = s(n) - \sum_{k=1}^{p} \alpha_k s(n-k) = G u_e(n) \tag{3.45}$$

Esto quiere decir que la señal de error es una buena aproximación de la fuente de excitación de un sistema de producción de voz el cual es modelado por un predictor de orden p [Camarena, 2011, Rabiner and Schafer, 1978]. Por lo tanto, es de esperarse que el error de predicción aumente drásticamente cada vez que inicia otro periodo del tono para sonidos vocalizados, dicho periodo (el tono) se puede determinar detectando las posiciones de las muestras de la señal $e(n)$ que son de mayor amplitud, y midiendo entonces la diferencia de tiempo que hay entre muestras de $e(n)$ que superen un valor umbral.

2. El error de predicción medio cuadrático normalizado [Camarena, 2011], se puede calcular de manera muy simple mediante la ecuación 3.46.

$$V_n = \prod_{i=1}^{p}(1 - k_i^2) \tag{3.46}$$

3. El algoritmo de Durbin-Levinson determinan de manera colateral k_i, se les conoce como coeficientes PARCOR [Rabiner and Schafer, 1978]; debido a que son una medida de la correlación parcial entre el error de predicción hacia adelante y el error de predicción hacia atrás, un concepto que utilizan los métodos basados en retículas para determinar los coeficientes predictores.

En el modelo de producción de voz, cuando el sonido es vocalizado, la excitación $u_e(n)$ es un tren de impulsos. Del análisis expuesto aquí se concluye que la señal de error de predicción es proporcional a la excitación ($e(n) = Gu_e(n)$) real, se sabe que la excitación en un tracto vocal humano consiste de un tren de pulsos glotales y que la forma del pulso glotal cambia de un individuo a otro, y por esa razón es muy utilizado para la identificación de individuos por su voz. En conclusión la forma de la señal de error para un sonido vocalizado específico se podría utilizar para la identificación del emisor de la señal de voz bajo análisis.

Con los parámetros obtenidos en el análisis de predicción lineal [Camarena, 2011, Rabiner and Schafer, 1978], se puede hacer: síntesis de voz, determinar el tono de la señal, obtener los formantes de la señal e implementar un reconocedor de palabras aisladas monolocutor. En el presente trabajo se describe únicamente como implementar un reconocedor de palabras aisladas.

3.3.5. Distancia de Itakura

Tanto el reconocimiento de voz como la identificación de individuos por su voz es necesario comparar cuantitativa y efeciéntemente dos marcos de voz para los cuales el análisis LPC ha extraído diferentes vectores de coeficientes LPC. Entonces, se requiere una medida de distancia entre marcos de voz $D(\Lambda, \hat{\Lambda})$, donde $\Lambda = (\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_p)$ es

el vector aumentado de los coeficientes LP de referencia y $\hat{\Lambda} = (\hat{\alpha}_0, \hat{\alpha}_1, \hat{\alpha}_2, \ldots, \hat{\alpha}_p)$ es el vector aumentado de los coeficientes LP observados [Iosu, 1999], algunos autores definen $\alpha_0 = 1$ [Rabiner and Schafer, 1978, pág. 498-500], y otros definen $\alpha_0 = -1$ [Iosu, 1999]. Sin embargo se obtienen mejores resultados usando $\alpha_0 = -1$ (ver Apéndice B.3.1). Como $D(\Lambda, \hat{\Lambda})$ es una medida de distancia se requiere que:

$$D(\Lambda, \hat{\Lambda}) \geq 0 \tag{3.47}$$

y que:

$$D(\Lambda, \Lambda) = 0 \tag{3.48}$$

Itakura propuso una distancia basado en el siguiente razonamiento: Debido al ruido y a la imprecisión del modelo de voz basado en predicción lineal, no es posible determinar los verdaderos valores LPC del segmento de voz en cuestión, solo se hace una estimación de los mismos. Suponga que se tiene un segmento de voz y los coeficientes estimados son los coeficientes $\hat{\Lambda}$, entonces el problema es determinar la probabilidad de que $\hat{\Lambda}$ haya sido extraído de un segmento de voz cuyos coeficientes LPC verdaderos son los de Λ [Camarena, 2011, Iosu, 1999]. Por supuesto una probabilidad pequeña implica una distancia grande y viceversa, eso y algunas consideraciones respecto al ahorro de cómputos condujo a Itakura [Camarena, 2011], a proponer la siguiente medida de distancia:

$$D(\Lambda, \hat{\Lambda}) = log\left[\frac{\Lambda R \Lambda^t}{\hat{\Lambda} R \hat{\Lambda}^t}\right] \tag{3.49}$$

La clave para entender esta distancia es que dos vectores de coeficientes LPC deben tener una distancia corta entre ellos si fueron extraídos de segmentos de voz que suenan muy parecido.

3.3.6. Sistema de reconocimiento de voz basado en coeficientes LPC

El reconocedor de palabras aisladas basado en coeficientes LP es prácticamente igual al sistema de reconocimiento de palabras aisladas mediante espectrogramas descrito en la sección 3.1.7. Ambos sistemas caracterizan una palabra mediante una matriz de flotantes, en ambos casos la matriz tiene número fijo de renglones, pues este depende del orden

del predictor (en el caso de LPC) o del número de bandas críticas (en el caso de Espectrogramas), y un número variable de columnas, por que este depende de la duración de la locución. La única diferencia entre ambos sistemas es el denominado *front end*, es decir, el módulo de extracción de características de la señal de voz y por supuesto la interpretación que debe de hacerse de dicha matriz, en el caso del reconocedor mediante espectrogramas descrito en 3.1.7 cada columna corresponde con una banda crítica de Bark, mientras que en el reconocedor hecho en base a coeficientes LPC, cada columna corresponde con uno de los parámetros del filtro dinámico de puros polos que modela al tracto vocal.

Para comparar las matrices se puede usar DTW, HMM, etc.; utilizando la distancia euclidiana, coseno, etc. Sin embargo, conviene aprovechar que se trata de LPC, y utilizar la distancia de Itakura [Camarena, 2011], la cual fué diseñada específicamente para comparar vectores cuyos componentes son precisamente LPC.

Para implementar un reconocedor de palabras aisladas [Camarena, 2011], el orden del predictor debe de ser igual a la frecuencia de muestreo en kHz y tres o cuatro polos adicionales, por lo tanto si la frecuencia de muestreo que se usa es de 8kHz, se recomienda usar $p = 12$.

3.4. Conclusiones

La extracción de características de la señal de voz se necesita para convertir la señal de voz en algún tipo de representación paramétrica sumamente compactada con la finalidad de comprimir audio, reconocimiento de voz, hacer síntesis de voz, etc.

Los espectrogramas y los MFCC extraen características de la señal de voz basándose en la forma en que las personas identifican los sonidos, es decir tratando de modelar el oído humano. Estas dos técnicas de extracción de características usan la DFT, la cual se puede calcular eficientemente usando el algoritmo de FFT, para utilizar el algoritmo FFT se necesita que el tamaño del marco sea potencia de dos y el marco tiene que ser suavizado

aplicandole una ventana para evitar el efecto de escurrimiento.

Los espectrogramas son fácil de implementar y entender su funcionamiento, sólo permite el reconocimiento de voz monolocutor.

Los MFCC es la técnica de extracción de características más utilizada actualmente en reconocedores comerciales porque permite reconocimiento de voz multilocutor [Camarena, 2011]. Su desventaja es que tiene un alto costo computacional a causa de que utiliza operaciones coseno y logarítmicas para poder obtener los coeficientes cepstrales de Mel.

Los LPC extraen características de la señal tratando de modelar el tracto vocal y la excitación glotal, es una de las técnica más potentes y predominantes ya que proporciona estimaciones muy precisas de los parámetros de la voz y es muy eficiente. Los LPC no hace uso de la DFT, por lo cual no es necesario que el tamaño del marco sea potencia de dos, tampoco se necesita aplicar ventana para suavizar el marco.

La eficiencia de los LPC reside en el hecho de que únicamente se necesitan calcular los $p+1$ primeros valores de autocorrelación usando el algoritmo de Blankenship, y resolver un sistema de ecuaciones de tamaño $(p)(p)$, usando el eficiente algoritmo de Levinson-Durbin.

Otra ventaja de los LPC respecto a los espectrogramas y los MFCC es que permite hacer síntesis de voz usando los coeficientes predictores o los coeficientes de reflección.

Los LPC permiten hacer identificación de individuos por su voz, porque sólo acepta reconocimiento de voz monolocutor.

El objetivo es implementar un sistema de reconocimiento de voz, hasta el momento se contempla como encontrar el inicio y final de una palabra contenida en una señal de audio, y como extraer las características de la señal empleando espectrogramas, MFCC y LPC; estas técnicas caracterizan una señal de audio mediante una matriz de flotantes teniendo un número fijo de renglones y un número variable de columnas. La etapa que falta es la de reconocimiento, es decir, la que se encarga de comparar las matrices de dos locuciones y decidir si son iguales o diferentes, existen varios métodos para realizar esta tarea entre los

que destacan DTW y HMM, estos métodos se describen a detalle en el siguiente capítulo.

Capítulo 4

Métodos de reconocimiento

Los siguiente métodos de reconocimiento [Cowling and Sitte, 2002], se utilizan comúnmente para el reconocimiento de voz:

- Dynamic Time Warping (DTW).
- Hidden Markov Models (HMM).
- Vector Quantization (VQ)/ Learning Vector Quantization (LVQ).
- Artificial Neural Networks (ANN).

Cada uno de estos métodos tiene sus particularidades, así como sus ventajas y desventajas. En este capítulo se describen en detalle los métodos DTW y HMM, que fueron los empleados en este trabajo por ser las técnicas más relevantes acorde a [Puertas, 2000, pág. 7].

4.1. Distancias

DTW calcula la distancia total entre dos matrices, para esto se calcula la distancia entre una columna de una matriz (palabra de referencia) con la columna de la otra matriz (palabra a reconocer).

El codebook se obtiene de aplicar algún algoritmo de agrupamiento, en el presente trabajo se aplica el algoritmo K-medias, este algoritmo realiza la agrupación de una serie de vectores de acuerdo con un criterio de cercanía. Esta cercanía se define en términos de una determinada función de distancia.

Para realizar la VQ se necesita la distancia entre el vector a cuantizar con todos los vectores almacenado en el codebook.

4.1.1. Distancia Euclidiana

Esta distancia considera que los puntos que están en la periferia de un círculo están a la misma distancia del centro, es quizás la distancia mas fácil de concebir puesto que se sabe el concepto de que la distancia mas corta entre dos puntos es la línea recta y se puede deducir esta distancia utilizando el Teorema de Pitágoras [Camarena, 2011]. Está distancia esta definida como:

$$d_e(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{N} (\vec{x}_i - \vec{y}_i)^2} \tag{4.1}$$

donde $\vec{x}$ y $\vec{y}$ son vectores de tamaño N.

4.1.2. Distancia Coseno

La distancia coseno entre dos vectores [Camarena, 2011], depende de la orientación de estos y no de su magnitud, básicamente se trata de determinar en ángulo menor que hay entre dos vectores, el producto punto entre dos vectores se determina mediante:

$$\vec{x} \cdot \vec{y} = |\vec{x}|\,|\vec{y}|\,cos(\theta) \tag{4.2}$$

donde θ es el menor ángulo que hay entre los vectores $\vec{x}$ y $\vec{y}$.

$$cos(\theta) = \frac{\vec{x} \cdot \vec{y}}{|\vec{x}|\,|\vec{y}|} = \frac{\sum_{i=1}^{N} \vec{x}_i \vec{y}_i}{\sqrt{\sum_{i=1}^{N} (\vec{x}_i)^2} \sqrt{\sum_{i=1}^{N} (\vec{y}_i)^2}} \tag{4.3}$$

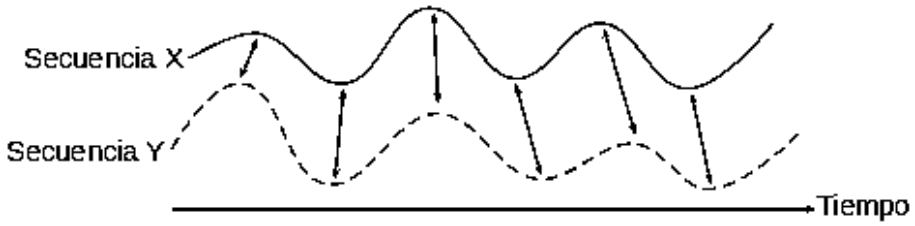

Figura 4.1: Alineación de dos secuencias en función del tiempo. Los puntos alineados se indica por las flechas

La ecuación 4.3 es en realidad una medida de similitud no una distancia [Camarena, 2011], es decir, mientras menos se parecen los vectores la medida de similitud se aproxima mas a cero, el mayor valor que el coseno devuelve es 1.0, se puede convertir la medida de similitud en distancia de la siguiente manera:

$$d_c(\vec{x},\vec{y}) = 1 - |cos(\theta)| = 1 - \left|\frac{\vec{x}\cdot\vec{y}}{|\vec{x}|\,|\vec{y}|}\right| = 1 - \left|\frac{\sum_{i=1}^{N}\vec{x}_i\vec{y}_i}{\sqrt{\sum_{i=1}^{N}(\vec{x}_i)^2}\sqrt{\sum_{i=1}^{N}(\vec{y}_i)^2}}\right| \tag{4.4}$$

4.2. Doblado Dinámico en Tiempo

Doblado Dinámico en Tiempo (DTW) [Muller, 2007], es una técnica bien conocida para encontrar una alineación óptima entre dos secuencias dadas (dependientes del tiempo), con determinadas limitaciones Figura 4.1. Intuitivamente, las secuencias se deforman de manera no lineal para hacerlas coincidir entre sí. Originalmente, DTW ha sido utilizada para comparar diferentes patrones en el reconocimiento de voz automático, y en campos tales como la minería de datos y recuperación de información [Muller, 2007].

DTW se ha aplicado con éxito para hacer frente automáticamente con deformaciones de tiempo y velocidades diferentes asociados con datos dependientes del tiempo [Muller, 2007].

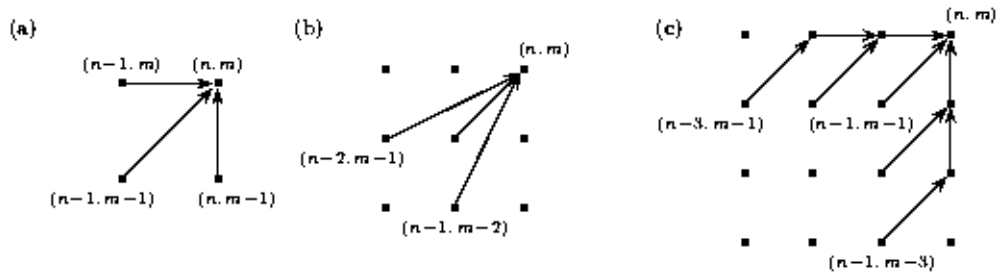

Figura 4.2: Ilustración de tres condiciones con diferentes tamaño de paso, que expresan diferentes restricciones locales sobre las posibles deformaciones admisibles [Muller, 2007]

4.2.1. Restricciones locales

Alinear dos series de tiempo $x(n)$, $0 \leq n \leq N$, de tamaño $N \in \mathbb{N}$, y $y(m)$, $0 \leq m \leq N_2$, de tamaño $N_2 \in \mathbb{N}$, es equivalente a encontrar una función de doblado $m = \wp(n)$; que mapea cada índice n de la serie $x(n)$ en un índice de la serie $y(m)$, de manera que se realice un registro entre las dos series de tiempo. La función $\wp(n)$ está sujeta a las condiciones de frontera $\wp(0) = 0$ y $\wp(N) = N_2$, y a restricciones locales.

Posíblemente la restricción local más utilizada [Muller, 2007], es la que indica que si la trayectoria óptima pasa por el punto (n, m), entonces debió pasar por $(n-1, m-1)$, por $(n, m-1)$ o por $(n-1, m)$; como se muestra en la Figura 4.2 (a). Una penalización de 2 es impuesta cuando se elige $(n-1, m-1)$ y de 1 si se eligen $(n, m-1)$ o $(n-1, m)$, de esta manera, las tres posibles trayectorias de $(n-1, m-1)$ a (n, m) (Ej. ir primero a $(n, m-1)$ y luego a (n, m)) tendran todas el mismo costo de 2.

Sea $d_{n,m}$, la distancia entre el vector de características correspondiente al marco n de la elocución $x(n)$, y el vector de características correspondiente al marco m de la elocución $y(m)$. Entonces la función óptima entre $x(n)$ y $y(m)$ es aquella que minimiza la distancia acumulada $D_{n,m}$, definida mediante la ecuación 4.5.

$$D_{n,m} = \sum_{i=1}^{n} d_{x(i),y(\wp(i))} \tag{4.5}$$

Una vez elegida una restricción local, D_{N,N_2} puede calcularse utilizando cualquiera

de las recurrencias definidas por las ecuaciones (4.6), (4.7) o (4.8) [Muller, 2007], las cuales corresponde a la restricciónes locales mostradas en la Figura 4.2 (a), (b) y (c) respectivamente. Basandose en estas recurrencias, D_{N,N_2} se puede determinar utilizando programación dinámica.

$$D_{0,0} = d_{0,0}$$

$$D_{n,0} = d_{n,0} + D_{n-1,0}$$

$$D_{0,m} = d_{0,m} + D_{0,m-1}$$

$$D_{n,m} = min \begin{cases} D_{n-1,m-1} + 2d_{n,m} \\ D_{n-1,m} + d_{n,m} \\ D_{n,m-1} + d_{n,m} \end{cases} \tag{4.6}$$

$$D_{n,m} = min \begin{cases} D_{n-1,m-1} + d_{n,m} \\ D_{n-2,m-1} + 2d_{n,m} \\ D_{n-1,m-2} + 2d_{n,m} \end{cases} \tag{4.7}$$

$$D_{n,m} = min \begin{cases} D_{n-1,m-1} + d_{n,m} \\ D_{n-2,m-1} + 2d_{n,m} \\ D_{n-1,m-2} + 2d_{n,m} \\ D_{n-3,m-1} + 3d_{n,m} \\ D_{n-1,m-3} + 3d_{n,m} \end{cases} \tag{4.8}$$

En la Figura 4.3 se muestra la matriz de distancias que se debe llenar para implementar el doblado dinámico en tiempo mediante programación dinámica. En la misma figura se muestra la trayectoria óptima para llegar de la localidad $(1, 1)$ a la localidad (N, N_2), y que corresponde con la forma de la función de doblado óptima para alinear las series de tiempo de longitud N y N_2 respectivamente. La distancia entre las dos series de tiempo será la última en determinarse al llenar la matriz, es decir D_{N,N_2}, conviene normalizar esta

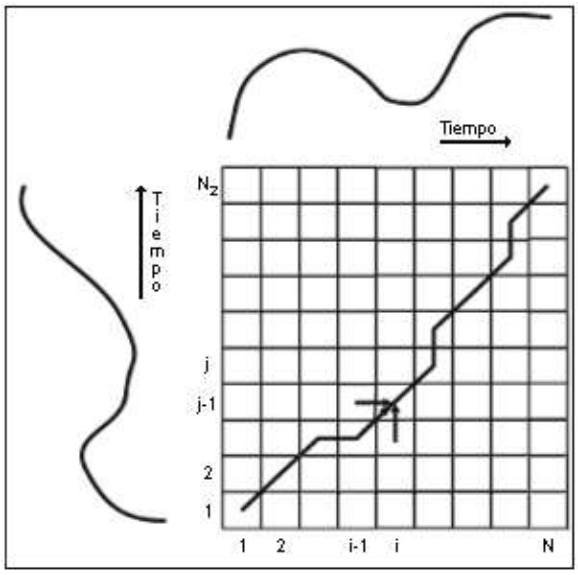

Figura 4.3: Matriz de distancias para implementar doblado dinámico en tiempo y la trayectoria óptima de acuerdo a la restricción local simétrica de primer orden, ecuación (4.6)

distancia [Camarena, 2011], es decir usar:

$$\frac{D_{N,N_2}}{(N+N_2)} \tag{4.9}$$

De esta manera, la distancia entre dos elocuciones de la misma palabra no será grande solo porque la palabra es larga, dicho de otra manera, la distancia entre dos elocuciones no deberá ser pequeña solo porque la palabra es corta.

4.2.2. Restricciones globales

Una variante común DTW [Muller, 2007], es imponer restricciones globales en los caminos de deformaciones admisibles. Estas limitaciones no sólo aceleran los cálculos DTW sino también previenen las alineaciones patológicas controlando globalmente la ruta de un camino de deformación.

Dos regiones de restricción global muy conocidas son la banda de Sakoe-Chiba y el paralelogramo de Itakura [Muller, 2007], estas regiones se muestran en la Figura 4.4.

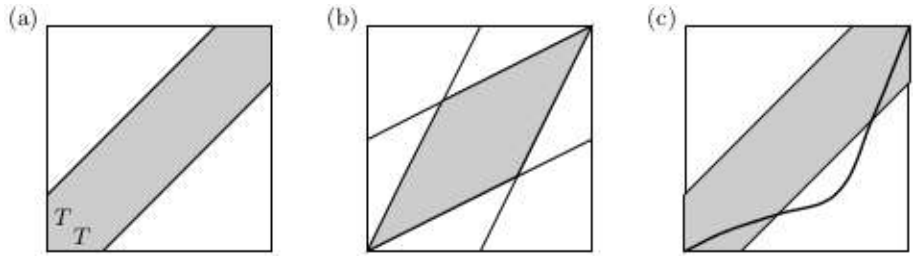

Figura 4.4: (a) Banda de Sakoe-Chilba. (b) Paralelogramo de Itakura. (c) Ruta óptima (linea negra) se observa que no se ejecuta dentro de la región de restricción [Muller, 2007]

Sin embargo, el uso de regiones de restricción global también es problemático [Muller, 2007], ya que el camino de deformación óptimo puede pasar por fuera de la región de limitación. En otras palabras, el resultado de la ruta óptima de deformación (restringido) generalmente no coincide con la trayectoria de deformación óptima (sin restricciones), este efecto se observa en la Figura 4.4 (c). Este hecho puede dar lugar a resultados no deseados o incluso de alineación completamente inútil.

4.3. Cuantización Vectorial

La aplicación de los DHMM al reconocimiento del habla requiere la obtención de una secuencia de símbolos correspondiente a un alfabeto discreto y finito a partir de una secuencia de vectores que representan las características espectrales de la señal de voz [Hernando, 1993]. Por esta razón es necesario realizar una discretización del espacio de características de la señal de voz mediante cuantización vectorial (VQ, Vector Quantization); este procedimiento consiste en establecer una partición del espacio vectorial en un conjunto finito de clases (libro de codigos o codebook en ingles).

Las clases se definen utilizando método iterativos, el más simple, conocido y usado en las pruebas experimentales realizadas en este trabajo, es el algoritmo de K-medias (Mac Queen 1967) [Pascual et al., 2007], el cual se describe a continuación.

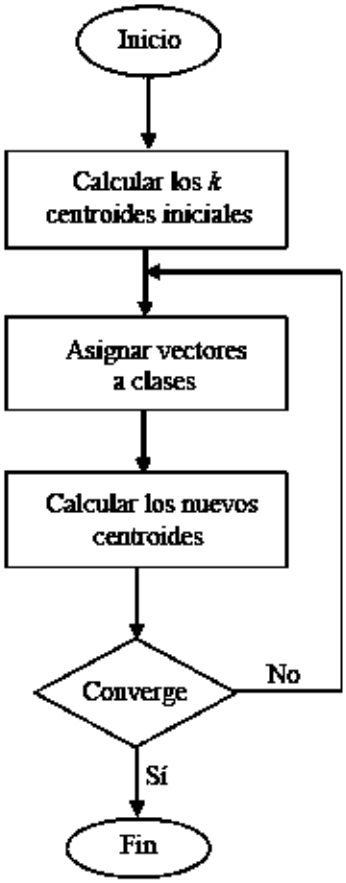

Figura 4.5: Diagrama de flujo del algoritmo de K-medias

4.3.1. Algoritmo de agrupamiento K-medias

La idea principal del algoritmo K-medias es definir N_a centroides (uno para cada grupo o clase, N_a es el número de símbolos del HMM), y luego tomar cada punto del conjunto de vectores y situarlo en la clase más cercana. El próximo paso es recalcular los centroides y volver a distribuir todos los objetos según el grupo más cercano. El proceso se repite hasta alcanzar la condición de convergencia (ver Figura 4.5).

El problema del empleo del algoritmo K-medias [Pascual et al., 2007], es que falla cuando los puntos de un grupo están muy cerca del centroide de otro grupo, también cuando los grupos tienen diferentes tamaños y formas.

Para la condición de convergencia se pueden usar varias condiciones; cuando alcanza un número de iteraciones dado, cuando no existe un intercambio de objetos entre los grupos o converger cuando la diferencia entre los centroides de dos iteraciones consecutivas es más pequeño que un umbral dado.

En este trabajo se utiliza la condición de converger cuando se alcance un número de $1,000$ iteraciones, porque es la más simple de implementar y evita ciclos infinitos cuando el algoritmo falla.

4.4. Modelos Ocultos de Markov

Un modelo oculto de Markov asume que el proceso subyacente es una cadena de Markov cuyos estados internos están ocultos para el observador. Por lo general, se supone que el número de estados del sistema y las probabilidades de transición de los estados son conocidos [Ibe, 2008]. Por lo tanto, hay dos parámetros asociados con cada estado de la cadena de Markov:

1. Probabilidades de emisión de los símbolos que describen las probabilidades de los diferentes resultados posibles del estado.

2. Las probabilidades de transición que describen la probabilidad de entrar a un nuevo estado desde el estado actual.

Los HMM [Ibe, 2008] se utilizan en una gran variedad de aplicaciones, pero las dos áreas de aplicación más importantes son el reconocimiento de voz y análisis de secuencias biológicas tales como la modelización de secuencia de ADN; fueron aplicados por primera vez para el reconocimiento de voz a principios de 1970 [Ibe, 2008], sin embargo, fueron reportados por Lawrence R. Rabiner en 1989 [Rabiner, 1989]; y Juang y Rabiner en 1991 [Ibe, 2008].

En los últimos años los HMM, se han convertido en la aproximación predominante en reconocimiento del habla [Hernando, 1993], superando la técnica de comparación de patrones, debido a la simplicidad de su estructura algorítmica y a sus buenas prestaciones.

A continuación se presentará la estructura de los HMM. Posteriormente, se presentarán los algoritmos para la evaluación, decodificación y entrenamiento de los HMM, se tratarán aspectos prácticos de implementación, como la inicialización, el escalado y el suavizado de los parámetros, y se estudiará su aplicación al reconocimiento de voz.

4.4.1. Elementos de un HMM

Un modelo oculto de Markov [Ibe, 2008], es la representación de un proceso estocástico[1] que consta de dos mecanismos interrelacionados: una cadena de Markov de primer orden subyacente, con un número finito de estados, y un conjunto de funciones aleatorias, cada una de las cuales se asociada a un estado.

Un HMM supone que el proceso está en un estado determinado en un instante discreto de tiempo, y que genera una observación mediante la función aleatoria asociada. Al instante siguiente, la cadena subyacente de Markov cambia de estado siguiendo su matriz de probabilidades de transición, produciendo una nueva observación mediante la función aleatoria correspondiente [Hernando, 1993].

Un HMM [Ibe, 2008], queda caracterizado por los siguientes elementos:

1. $Q = \{s_1, s_2, \ldots, s_{N_s}\}$, es un conjunto finito de N_s estados y al estado en el tiempo t como q_t.

2. $\Omega = \{o_1, o_2, \ldots, o_{N_o}\}$, es un conjunto finito de N_o posibles simbolos.

3. $A = \{a_{i,j}\}$, es un conjunto de probabilidades de transición entre estados, donde $a_{i,j}$ es la probabilidad de que el sistema estando en el estado s_i cambie al estado s_j. Para el

[1] Un proceso estocástico es una colección o familia de variables aleatorias $\{X(t, \dot{s}) | t \in \Gamma, \dot{s} \in \mho\}$, definidas sobre un espacio de probabilidad dado e indexado por el parámetro de tiempo t [Ibe, 2008].

caso especial de una cadena de Markov de primer orden, esta descripción probabilística se trunca en el estado actual y el último predecesor, es decir,

$$P(q_t = s_j | q_{t-1} = s_i, q_{t-2} = s_k, \ldots) = a_{i,j} = P(q_t = s_j | q_{t-1} = s_i) \tag{4.10}$$

Además, se considera que esta última probabilidad es independiente del tiempo (propiedad de homogeneidad temporal), lo cual da lugar a un conjunto de probabilidades de transición entre estados que se denotará con la matriz $A = \{a_{i,j}\}$ $i, j = 1, 2, \ldots, N_s$, donde

$$a_{i,j} = P(q_t = s_j | q_{t-1} = s_i) \; i, j = 1, 2, \ldots, N_s \tag{4.11}$$

La matriz A determinará la topología del modelo. A es una matriz de probabilidades, para que sea considerado probabilidad debe cumplir las siguientes restricciones:

$$\sum_{i=1}^{N} p_i = 1 \; ; \; j = 1, 2, \ldots, N \tag{4.12}$$

$$p_i \geq 0 \; ; \; i = 1, 2, \ldots, N \tag{4.13}$$

4. $\Pi = \{\pi_i\}$ es la distribución de probabilidad de estados iniciales, definida de la forma:

$$\pi_i = P(q_1 = s_i) \; ; \; i = 1, 2, \ldots, N_s \tag{4.14}$$

5. Las probabilidades de generación de observaciones, que caracterizan el proceso asociado a cada uno de los estados del modelo y que se denotarán como $B = \{b_j(o_t)\}$; $j = 1, 2, \ldots, N_s$ con:

$$b_j(o_t) = P(o_t | q_t = s_j) \; ; \; j = 1, 2, \ldots, N_s \tag{4.15}$$

donde o_t, representa el valor de la observación en el instante t, correspondiente a la secuencia de observaciones $O = \{o_t\}$; $t = 1, 2, \ldots, |\Gamma|$. Se supone que el proceso de generación de observaciones es independiente del tiempo y que únicamente depende del estado actual del modelo.

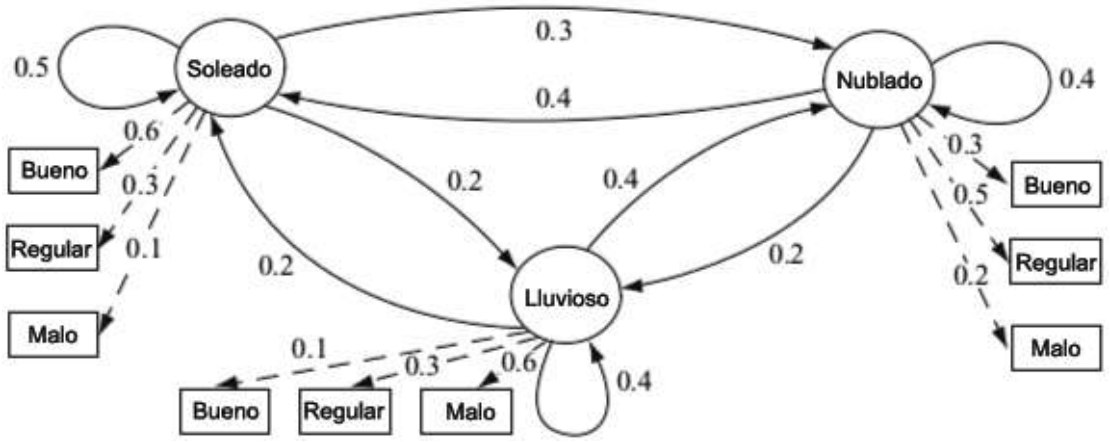

Figura 4.6: Representación de un Modelo Oculto de Markov

Debido a que los estados y la secuencia de salida se conocen, es habitual denotar los parámetros de un HMM [Ibe, 2008], por:

$$\lambda = (\Pi, A, B) \tag{4.16}$$

Una representación esquemática de un modelo oculto de Markov de tres estados $S = \{\text{Soleado, Nublado, Lluvioso}\}$, y un conjunto de observaciones $\Omega = \{\text{Bueno, Regular, Malo}\}$; puede verse en la Figura 4.6.

En los llamados modelos discretos (DHMM), estas probabilidades están representadas a través de distribuciones de probabilidad discretas, ya que las observaciones o_t toman valores dentro de un conjunto discreto y finito de símbolos llamado alfabeto $\bar{V} = \{\bar{v}_k\}$; $k = 1, 2, \ldots, N_a$, siendo N_a el tamaño de alfabeto. Las probabilidades de observación forman un conjunto que se denota como $B = \{b_j(k)\}$; $j = 1, 2, \ldots, N_s$ donde:

$$b_j(k) = P(v_k \text{ en } t | q_t = s_j) \text{ ; } j = 1, 2, \ldots, N_s \text{ ; } k = 1, 2, \ldots, N_a \tag{4.17}$$

En el caso de la aplicación al reconocimiento del habla [Hernando, 1993], los vectores característicos de las tramas de voz son cuantificados vectorialmente, y estos símbolos $\bar{v}_k$ se corresponden con las etiquetas de las palabras-código producto de dicha cuantificación vectorial.

En los modelos continuos (CHMM) [Hernando, 1993], las probabilidades de observación están representadas a través de funciones de densidad de probabilidad multivariadas, ya que las observaciones toman valores dentro de un espacio continuo multidimensional. En el caso del reconocimiento del habla, las observaciones consisten simplemente en los vectores espectrales de las tramas de voz sin cuantificación.

4.4.2. Tres problemas fundamentales

Hay tres problemas fundamentales en HMM [Ibe, 2008]:

1. *Problema de evaluación*: Dada una secuencia de observaciones $O = \{o_t\}$; $t = 1, 2, \ldots, |\Gamma|$, y un modelo $\lambda = (\Pi, A, B)$, cómo evaluar eficientemente $P(O|\lambda)$, la probabilidad de la secuencia de observación dado el modelo. Esta probabilidad se puede utilizar para clasificar las secuencias de observación como sucede en las aplicaciones de reconocimiento de voz.

2. *Problema de decodificación*: Dada una secuencia de observaciones $O = \{o_t\}$; $t = 1, 2, \ldots, |\Gamma|$, y un modelo $\lambda = (\Pi, A, B)$, cómo elegir la correspondiente secuencia de estados $Q = \{q_t\}$; $t = 1, 2, \ldots, |\Gamma|$, que es óptima en algún sentido, que mejor "explica"las observaciones. Su solución permite obtener información sobre el proceso oculto, por ejemplo, el significado de los estados del modelo. También puede utilizarse, como se verá, para obtener una aproximación eficiente al problema de evaluación.

3. *Problema de entrenamiento*: Dada una secuencia de observaciones $O = \{o_t\}$; $t = 1, 2, \ldots, |\Gamma|$, cómo ajustar los parámetros del modelo $\lambda = (\Pi, A, B)$; de forma que se maximice $P(O|\lambda)$, la probabilidad de generación de dicha secuencia por el modelo. Su solución permite desarrollar un método para obtener los parámetros de un modelo en base a secuencias de observaciones que se pretenden modelar.

4.4.3. Solución a los tres problemas fundamentales

A continuación se describe la solución de los tres problemas anteriores para el caso de DHMM, que fue la técnica aplicada en este trabajo.

1. ***Solución al problema de evaluación.***

Se desea calcular la probabilidad de que una secuencia de observaciones $O = \{o_t\}$; $t = 1, 2, \ldots, |\Gamma|$, dado un modelo λ, es decir, $P(O|\lambda)$. Existe un algoritmo recursivo que permite obtener esta probabilidad de forma eficiente, el algoritmo Forward-Backward [Ibe, 2008], que se describirá a continuación. Aunque la parte forward del algoritmo es suficiente para resolver el problema de la evaluación, se presentará también la parte backward, ya que se utilizará en la solución del problema de entrenamiento.

Algoritmo Forward-Backward

El procedimiento Forward-Backward [Ibe, 2008], considera la variable forward $\alpha_t(i)$; está definida como:

$$\alpha_t(i) = P(o_1 o_2 \cdots o_t, q_t = s_i|\lambda) \tag{4.18}$$

es decir, la probabilidad de generar la secuencia parcial de observaciones, $o_1 o_2 \cdots o_t$, (hasta el tiempo t) de manera que el modelo queda en el estado s_i en el instante t, dado el modelo. Se puede resolver $\alpha_t(i)$ inductivamente, de la siguiente manera:

1. Inicialización. Se calcula $\alpha_1(i)$ como la probabilidad conjunta de generar la primera observación o_1 y terminar en el estado s_i, para cada uno de los N_s estados.

$$\alpha_1(i) = \pi_i b_i(o_1) \;\; ; 1 \leq i \leq N_s \tag{4.19}$$

2. Inducción. Se calcula $\alpha_{t+1}(j)$, multiplicando la probabilidad de generar la observación o_{t+1} en el estado s_j, $b_j(o_{t+1})$, por la suma de las probabilidades de generar la secuencia parcial de las t observaciones previas finalizando en cada estado s_j, $\alpha_t(i)$, multiplicadas por las probabilidades de transición entre este estado y s_j, $a_{i,j}$. Este proceso puede verse esquematizado en la Figura 4.7.

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N_s} \alpha_t(i) a_{i,j}\right] b_j(o_{t+1}) \;\; ; 1 \leq t \leq |\Gamma| - 1 \; ; \;\; 1 \leq j \leq N_s \tag{4.20}$$

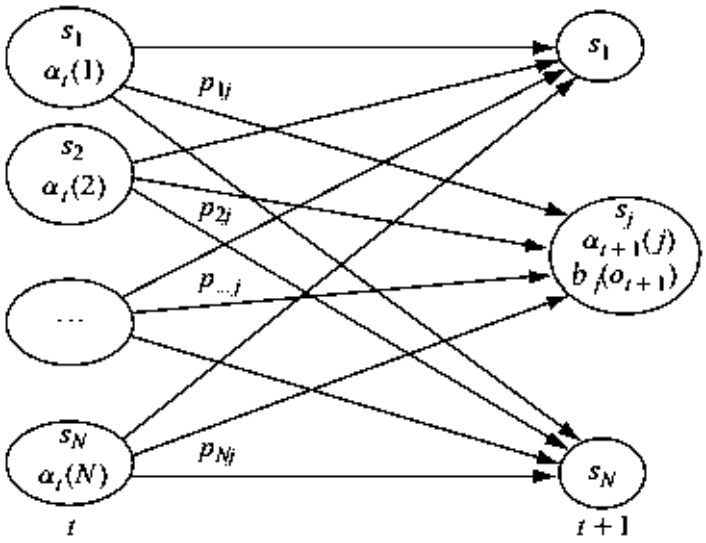

Figura 4.7: Paso de inducción del algoritmo Forward [Ibe, 2008]

3. Terminación.

$$P(O|\lambda) = \sum_{i=1}^{N_s} \alpha_{|\Gamma|}(i) \tag{4.21}$$

Los cálculos de este algoritmo pueden realizarse eficientemente si se considera un entramado de observaciones y estados por la que se avanza, y se van considerando las probabilidades de observación y transición, como es muestra en la Figura 4.8. La complejidad de este algoritmo es del orden de $N_s^2|\Gamma|$, cálculos [Ibe, 2008, Hernando, 1993].

Algoritmo Backward

Se define la variable backward [Ibe, 2008], como:

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, o_{|\Gamma|}|q_t = s_i, \lambda) \tag{4.22}$$

es decir, la probabilidad de la secuencia parcial de observaciones desde $t+1$ hasta el final, dados el estado s_i en el instante t y el modelo λ. También puede realizarse el cálculo de $\beta_t(i)$ recursivamente como sigue:

1. Inicialización.

$$\beta_{|\Gamma|}(i) = 1 \quad ; 1 \leq i \leq N_s \tag{4.23}$$

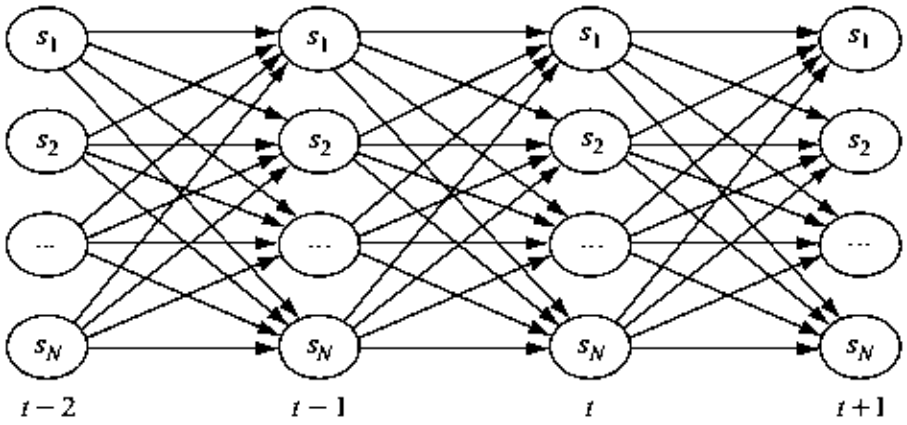

Figura 4.8: Esquema de cálculo del algoritmo Forward [Ibe, 2008]

2. Inducción. Se calcula $\beta_t(i)$, para $t = |\Gamma| - 1, |\Gamma| - 2, \ldots 1$, e $i = 1, 2, \ldots, N_s$, de forma análoga al cálculo recursivo de la variable Forward, teniendo en cuenta que del estado s_i puede pasarse a cualquiera de los N_s estados. Este proceso está ilustrado en la Figura 4.9.

$$\beta_t(i) = \sum_{j=1}^{N_s} a_{i,j} b_j(o_{t+1}) \beta_{t+1}(j) \ ; \ t = |\Gamma| - 1, |\Gamma| - 2, \ldots 1 \ ; \ i = 1, 2, \ldots, N_s \qquad (4.24)$$

3. Terminación [Ibe, 2008]. Se considera que la primera observación puede producirse en cualquier estado:

$$P(O|\lambda) = \sum_{i=1}^{N_s} \pi_i b_i(o_1) \beta_1(i) \qquad (4.25)$$

El coste computacional es del mismo orden que el de la evaluación forward ($N_s^2|\Gamma|$), y también pueden realizarse los cálculos en una estructura análoga a la de la Figura 4.8.

2. ***Solución al problema de decodificación.***

El problema de la obtención de la secuencia óptima de estados dada una secuencia de observaciones y el modelo λ [Hernando, 1993], no tiene una solución única. Depende del criterio con que se defina esta secuencia óptima. Se suele utilizar el criterio de seleccionar la

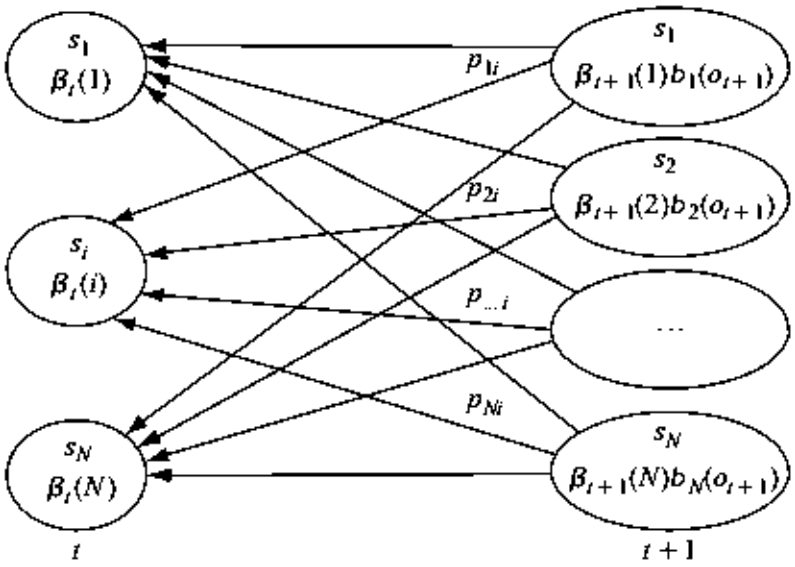

Figura 4.9: Paso de inducción del algoritmo Backward [Ibe, 2008]

secuencia de estados para la que la probabilidad de generación condicionada es máxima. Para solucionar este problema se utiliza un algoritmo recursivo análogo al Forward-Backward, basado en técnicas de programación dinámica, el algoritmo de Viterbi.

Algoritmo de Viterbi

El algoritmo de Viterbi fue diseñado originalmente para descodificar códigos convolucionales y ahora se aplica en muchas otras áreas [Ibe, 2008]. En los HMM se utiliza para encontrar la secuencia de estados más probable $Q^* = \{q_1^*, q_2^*, \ldots, q_T^*\}$, para una determinada secuencia de observación $O = \{o_1, o_2, \ldots, O_|\Gamma|\}$. El algoritmo de Viterbi simultáneamente máximiza tanto la probabilidad conjunta $P[q, O]$ y la probabilidad condicional $P[q|O]$ [Ibe, 2008], debido al hecho de que:

$$\underset{argmax}{Q}\{P[Q|O,\lambda]\} = \underset{argmax}{Q}\left\{\frac{P[Q,O|\lambda]}{P[O|\lambda]}\right\} = \underset{argmax}{Q}\{P[Q,O|\lambda]\} \tag{4.26}$$

El algoritmo define $\delta_t(i)$ de la siguiente manera:

$$\delta_t(i) = \underset{q_1,q_2,\ldots,q_{t-1}}{max} P[q_1, q_2, \ldots, q_{t-1}, q_t = s_i, o_1, o_2, \ldots, o_{t-1}, o_t|\lambda] \tag{4.27}$$

es decir, la probabilidad máxima de generación de las primeras t observaciones sobre cualquier secuencia de estados cuyo estado final sea el s_i, dado el modelo λ. Por

inducción se obtiene:

$$\delta_{t+1}(j) = [\underset{i}{max}\ \delta_t(i)a_{i,j}] \cdot b_j(O_{t+1}) \tag{4.28}$$

Para recuperar la secuencia de estados de probabilidad máxima es necesario almacenar los valores del argumento que maximizan (4.28), para cada t y j. Para ello se utiliza la matriz $\psi_t(j)$.

El algoritmo de Viterbi consta de los siguientes pasos:

1. Inicialización:

$$\delta_t(i) = \pi_i b_i(o_1)\ ;\ i = 1, 2, \ldots, N_s \tag{4.29}$$

$$\psi_t(i) = 0 \tag{4.30}$$

2. Recursión:

$$\delta_t(j) = \underset{0\leq i\leq N_s}{max}[\delta_{t-1}(i)a_{i,j}]b_j(o_t)\ ;\ \ 2 \leq t \leq |\Gamma|\ ;\ 1 \leq j \leq N_s \tag{4.31}$$

$$\psi_t(j) = \underset{0\leq i\leq N_s}{argmax}[\delta_{t-1}(i)a_{i,j}]\ \ ;\ 2 \leq t \leq |\Gamma|\ ;\ \ 1 \leq j \leq N_s \tag{4.32}$$

3. Terminación:

$$P^* = \underset{0\leq i\leq N_s}{max}[\delta_{|\Gamma|}(i)] \tag{4.33}$$

$$q^*_{|\Gamma|} = \underset{0\leq i\leq N_s}{argmax}[\delta_{|\Gamma|}(i)] \tag{4.34}$$

Note que este paso es similar al paso de inducción del algoritmo Forward. La diferencia principal entre los dos, es que el algoritmo Forward utiliza sumas sobre los estados anteriores, mientras que el algoritmo de Viterbi utiliza minimización.

4. Recursión para obtener la secuencia de estados óptima:

$$q^*_t = \psi_{t+1}(q^*_{t+1})\ ;\ t = |\Gamma| - 1, |\Gamma| - 2, \ldots, 1 \tag{4.35}$$

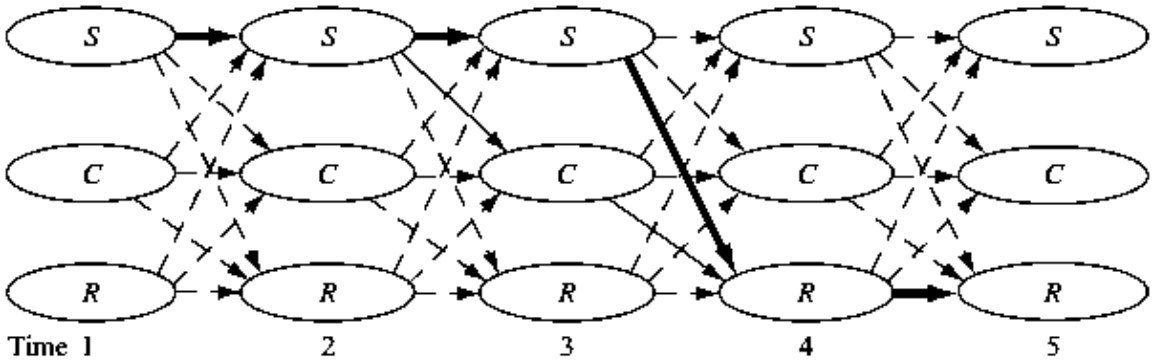

Figura 4.10: Esquema de cálculo del algoritmo Viterbi [Ibe, 2008]

También en este caso la estructura implementa eficientemente los cálculos (ver Figura 4.10). Hay que destacar que en este caso no se consideran todas las transiciones hasta cada estado, sino solamente aquellas que dan lugar a una probabilidad máxima.

3. ***Solución al problema de entrenamiento.***

El tercer problema relacionado con el modelado HMM es el de ajustar los parámetros del modelo $\lambda = (\Pi, A, B)$, para maximizar la probabilidad de generación de una secuencia de observaciones $O = \{o_t\}\ t = 1, 2, \ldots, |\Gamma|$; dado el modelo. Dada una secuencia finita de observaciones no es posible estimar de forma óptima los parámetros del modelo. Sin embargo, se pueden elegir los parámetros del modelo de forma que se maximice localmente la probabilidad $P(O|\lambda)$, mediante un procedimiento iterativo llamado algoritmo de Baum-Welch [Ibe, 2008], a veces llamado el algoritmo forward-backward.

$$\lambda^* = \underset{\lambda}{argmax}\{P[O|\lambda]\} \tag{4.36}$$

Algoritmo Baum-Welch

El algoritmo Baum-Welch (como forward-backward) usa las mismas variable forward $\alpha_t(i)$ y la variable backward $\beta_t(i)$ usadas en el problema de evaluación, estas variables se definen como:

$$\alpha_t(i) = P[o_1, o_2, \ldots, o_t, q_t = s_i|\lambda]\ ;\ 1 \leq t \leq |\Gamma|\ ;\ 1 \leq i \leq N_s \tag{4.37}$$

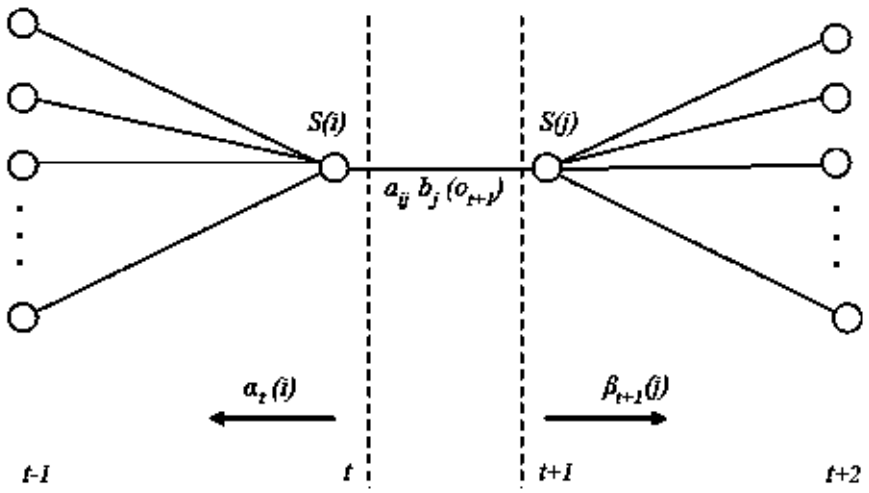

Figura 4.11: Cálculo de $\xi_t(i,j)$

$$\beta_t(i) = P[o_{t+1}, o_{t+2}, \ldots, o_{|\Gamma|}|q_t = s_i, \lambda] \ ; \ 1 \leq t \leq |\Gamma| \ ; \ 1 \leq i \leq N_s \tag{4.38}$$

Se define $\xi_t(i,j)$ como la probabilidad de que el modelo se encuentre en el estado s_i en el instante t, y se produzca una transición de forma que en el instante $t+1$ el estado sea el s_j dada la secuencia de observaciones y el modelo λ, es decir,

$$\xi_t(i,j) = P(q_t = s_i, q_{t+1} = s_j|O, \lambda) = \frac{P(q_t = s_i, P(q_{t+1} = s_j, O|\lambda)}{P(O|\lambda)} \tag{4.39}$$

Este valor puede expresarse en función del las probabilidades forward y backward, en la forma (ver Figura 4.11).

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{i,j}b_j(o_{t+1})\beta_{t+1}(j)}{P(O|\lambda)} = \frac{\alpha_t(i)a_{i,j}b_j(o_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N_s}\sum_{j=1}^{N_s}\alpha_t(i)a_{i,j}b_j(o_{t+1})\beta_{t+1}(j)} \tag{4.40}$$

También es conveniente definir la variable $\gamma_t(i)$ como la probabilidad de estar en el estado s_i en el instante t o, equivalentemente, el número esperado de transiciones que se hacen desde el estado s_i dada la secuencia de observaciones y el modelo λ, es decir,

$$\gamma_t(i) = P(q_t = s_i|O, \lambda) = \frac{P(q_t = s_i, O|\lambda)}{P(O|\lambda)} = \sum_{j=1}^{N_s}\xi_t(i,j) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N_s}\alpha_t(i)\beta_t(i)} \tag{4.41}$$

Sumando $\gamma_t(i)$ para $t = 1, 2, \ldots, |\Gamma|$ [Hernando, 1993], se obtiene el número esperado (en el tiempo) de veces que el modelo se encuentra en el estado s_i, y sumando para $t = 1, 2, \ldots, |\Gamma| - 1$, se obtiene el número esperado de veces que el modelo realiza un transición desde el estado s_i. Además, sumando $\gamma_t(i)$ para $t = 1, 2, \ldots, |\Gamma|$, con la restricción de que el símbolo observado sea $\bar{v}_k$, se obtiene el número esperado de veces que el modelo genera el símbolo $\bar{v}_k$ en el estado s_i. Por último, sumando $\xi_t(i, j)$ para $t = 1, 2, \ldots, |\Gamma| - 1$, se obtiene el número esperado de veces que se produce una transición entre los estados s_i y s_j. Con estas definiciones y estos resultados, se pueden establecer las siguientes fórmulas de reestimación de los parámetros, que dan lugar a nuevos parámetros que verifican automáticamente las restricciones estocásticas:

$\bar{\pi}_i$ = número de veces en el estado s_i en el tiempo $(t = 1) = \gamma_1(i)$.

$$\gamma_1(i) = \frac{\alpha_1(i)\beta_1(i)}{\sum_{i=1}^{N_s} \alpha_1(i)\beta_1(i)} \; ; \; i = 1, 2, \ldots, N_s \tag{4.42}$$

$$\bar{a}_{i,j} = \frac{\text{número esperado de transiciones desde el estado } s_i \text{ al estado } s_j}{\text{número esperado de transiciones desde el estado } s_i} =$$

$$= \frac{\sum_{t=1}^{|\Gamma|-1} \xi_t(i, j)}{\sum_{t=1}^{|\Gamma|-1} \gamma_t(i)} = \frac{\sum_{t=1}^{|\Gamma|-1} \alpha_t(i) a_{i,j} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{t=1}^{|\Gamma|-1} \alpha_t(i)\beta_t(i)} \; ; \; i, j = 1, 2, \ldots, N_s \tag{4.43}$$

$$\bar{b}_j(k) = \frac{\text{número esperado de veces en el estado } s_j \text{ y observando el símbolo } \bar{v}_k}{\text{número esperado de veces en el estado } s_j} =$$

$$= \frac{\sum_{t=1, o_t = \bar{v}_k}^{|\Gamma|} \gamma_t(j)}{\sum_{t=1}^{|\Gamma|} \gamma_t(j)} = \frac{\sum_{t=1, o_t = \bar{v}_k}^{|\Gamma|} \alpha_t(j)\beta_t(j)}{\sum_{t=1}^{|\Gamma|} \alpha_t(j)\beta_t(j)} \; ; \; j = 1, 2, \ldots, N_s \; ; \; k = 1, 2, \ldots, N_a \tag{4.44}$$

Se demuestra que si a partir de un modelo $\lambda = (\Pi, A, B)$, utilizando las fórmulas de reestimación (4.42), (4.43) y (4.44), se obtiene un nuevo modelo $\bar{\lambda} = (\bar{\Pi}, \bar{A}, \bar{B})$, la probabilidad de generación de la secuencia de observaciones dado el modelo $\bar{\lambda}$ es siempre mayor que la obtenida para el modelo inicial λ, excepto cuando se alcanza un valor crítico

de la función probabilidad, en cuyo caso las dos probabilidades coinciden [Hernando, 1993]. Esta prueba garantiza una convergencia uniforme de este método de reestimación hacia este punto límite, llamado Estimación de Máxima Probabilidad.

El algoritmo Baum-Welch sólo conduce a máximos locales, y en muchos problemas de interés la superficie de optimización es muy compleja y tiene muchos máximos locales [Hernando, 1993]. De ahí que en muchos casos sea importante el problema de la inicialización de los parámetros.

4.4.4. Problemas de implementación de los HMM

A continuación se discutirán cuestiones relacionadas con la implementación de dicho modelado como: los problemas derivados del rango de valores de las probabilidades de observación, la elección de los valores iniciales de los parámetros, la estimación de los parámetros del modelo con múltiples secuencias de observaciones y los efectos de la existencia de un número finito de observaciones.

- **Escalado dinámico y escalado logarítmico**

Los parámetros π_i, $a_{i,j}$ y $b_j(k)$ tienen valores muy inferiores a la unidad, las definiciones de las variables $\alpha_t(i)$ y $\beta_t(i)$; vistas en el apartado 4.4.3, dan lugar a valores que decaen exponencialmente a cero con el tiempo [Rabiner, 1989]. Para un valor de $|\Gamma|$, moderadamente alto el rango dinámico de estas variables excede la precisión de cualquier máquina.

Para solucionar el problema anterior, se debe escalar dinámicamente las variables $a_{i,j}$ y $b_j(k)$; multiplicandolas por unos coeficientes de escalado que son independientes de i (es decir, sólo dependen de t), de manera que el valor escalado de estas variables se mantiene dentro del rango dinámico de la máquina para $t = 1, \ldots, |\Gamma|$, y al finalizar los cálculos los coeficientes de escalado se cancelan exactamente [Rabiner, 1989].

Variable Forward escalada

La variable forward escalada [Rabiner, 1989], está definida mediante:

$$\hat{\alpha}_t(i) = \frac{\alpha_t(i)}{\sum_{j=1}^{N_s} \alpha_t(j)} = C_t \alpha_t(i) \tag{4.45}$$

La recursión para calcular la ecuación 4.45 es la siguiente:

$$\bar{\alpha}_1(i) = \alpha_1(i) \tag{4.46}$$

$$\bar{\alpha}_{t+1}(j) = \sum_{i=1}^{N_s} \hat{\alpha}_t(i) a_{i,j} b_j(o_{t+1}) \tag{4.47}$$

$$c_{t+1} = \frac{1}{\sum_i \bar{\alpha}_{t+1}(i)} \tag{4.48}$$

$$\hat{\alpha}_{t+1}(i) = c_{t+1} \bar{\alpha}_{t+1}(i) \tag{4.49}$$

Por lo tanto, C_t se calcula mediante la siguiente ecuación:

$$C_t = \frac{1}{c_{t+1} \sum_{j=1}^{N_s} \alpha_{t+1}(j)} = \frac{C_{t+1}}{c_{t+1}} \tag{4.50}$$

$$C_t = C_{t-1} c_t = \prod_{\tau=1}^{t} c_\tau \tag{4.51}$$

Variable Backward escalada

La variable backward escalada [Rabiner, 1989], está definida mediante:

$$\hat{\beta}_t(i) = D_t \beta_t(i) \tag{4.52}$$

La recursión para calcular la ecuación 4.52 es la siguiente:

$$\bar{\beta}_{|\Gamma|}(i) = \beta_{|\Gamma|}(i) \tag{4.53}$$

$$\bar{\beta}_t(j) = \sum_{i=1}^{N_s} a_{i,j} b_j(o_{t+1}) \hat{\beta}_{t+1}(i) \tag{4.54}$$

$$\hat{\beta}_t(i) = c_t \bar{\beta}_t(i) \tag{4.55}$$

D_t se calcula mediante la siguiente ecuación:

$$D_t = \prod_{\tau=t}^{|\Gamma|} c_\tau \tag{4.56}$$

Calculando $\bar{a}_{i,j}$ y $\bar{b}_i(l)$ usando $\hat{\alpha}$ y $\hat{\beta}$.

$$\bar{a}_{i,j} = \frac{\sum_{t=1}^{|\Gamma|-1} \hat{\alpha}_t(i) a_{i,j} b_j(o_{t+1}) \hat{\beta}_{t+1}(j)}{\sum_{t=1}^{|\Gamma|-1} \hat{\alpha}_t(i) \hat{\beta}_t(i) \frac{1}{c_t}} \tag{4.57}$$

$$\bar{b}_i(l) = \frac{\sum_{t=1,\ \text{sii}\ o_t=\bar{v}_l}^{|\Gamma|-1} \hat{\alpha}_t(i) \hat{\beta}_t(i) \frac{1}{c_t}}{\sum_{t=1}^{|\Gamma|-1} \hat{\alpha}_t(i) \hat{\beta}_t(i) \frac{1}{c_t}} \tag{4.58}$$

En la realización del algoritmo de Viterbi se ha utilizado escalado logarítmico de las probabilidades, que en este caso proporciona una solución exacta al problema sin necesidad de escalado dinámico. El algoritmo de Viterbi se define de la siguiente manera:

$$\delta_{t+1}(j) = \max_{q_1, q_2, \ldots, q_t} \{\log\ P[q_1 q_2 \cdots q_t, o_1 o_2 \cdots o_t | \lambda]\} \tag{4.59}$$

Inicialización:

$$\delta_t(i) = log(\pi_i) + log[b_i(o_1)] \tag{4.60}$$

Recursión:

$$\delta_t(j) = \max_{0 \le i \le N_s} [\delta_{t-1}(i) + log(a_{i,j})] + log[b_j(o_t)] \tag{4.61}$$

Terminación:

$$log\ P^* = \max_{1 \le i \le N_s} [\delta_T(i)] \tag{4.62}$$

- **Inicialización de los parámetros**

El algoritmo de Baum-Welch para la reestimación iterativa de los parámetros del modelo sólo garantiza la obtención de un máximo local de la función probabilidad de generación del modelo. Por lo tanto es importante elegir estimaciones iniciales de los parámetros de manera que el máximo local obtenido corresponda con el máximo global de la función probabilidad. No existe una solución exacta a esta cuestión. La experiencia demuestra que la elección aleatoria (sujeta a las restricciones estocásticas y sin permitir valores nulos para aquellos parámetros que no desean fijarse a cero), o uniforme para los valores de las probabilidades iniciales de los N_s estados y para las probabilidades de transición A; resulta adecuada para obtener parámetros útiles en casi todos los casos [Hernando, 1993]. Sin embargo, en el caso de las probabilidades de observación B [Hernando, 1993], se ha observado que estimaciones iniciales buenas resultan de gran ayuda en el caso de los modelos discretos y son esenciales en los modelos continuos, semicontinuos y de múltiple etiquetado.

En los DHMM utilizados en este trabajo las matrices A y B fueron inicializadas aleatoriamente, porque para inicializar de manera óptima a A y B no existe una solución exacta, por lo general se tiene que hacer un particionado manual de la secuencia de observaciones [Hernando, 1993], Π se ha forzado por razones físicas que la secuencia comience en el estado 1; por tanto $\Pi = \{1, 0, 0, \ldots, 0\}$.

- **Múltiples secuencias de observaciones**

En muchas situaciones no se dispone de una secuencia de observaciones de la suficiente duración como para poder estimar adecuadamente los parámetros del modelo, o existe una gran variabilidad entre las secuencias de observaciones que se desean modelar, la pronunciación de una palabra o un fonema es un ejemplo claro de ambas situaciones. En cualquiera de estos casos, la naturaleza del proceso a modelar hace necesario utilizar varias secuencias de observaciones, en la estimación de dichos parámetros.

En [Rabiner, 1989] Rabiner explica cómo utilizar secuencias de múltiples observaciones para el entrenamiento del DHMM usando el algoritmo Baum-Welch, las matrices de

transición se calcula de acuerdo a las siguientes ecuaciones:

$$\bar{a}_{i,j} = \frac{\sum_{k=1}^{K} \sum_{t=1}^{|\Gamma|_k - 1} \hat{\alpha}_t^{(k)}(i) a_{i,j} b_j(o_{t+1}^{(k)}) \hat{\beta}_{t+1}^{(k)}(j)}{\sum_{k=1}^{K} \sum_{t=1}^{|\Gamma|_k - 1} \hat{\alpha}_t^{(k)}(i) \hat{\beta}_t^{(k)}(i) \frac{1}{c_t^{(k)}}} \tag{4.63}$$

$$\bar{b}_i(l) = \frac{\sum_{k=1}^{K} \sum_{t=1,\ \text{sii}\ o_t = \bar{v}_l}^{|\Gamma|_k - 1} \hat{\alpha}_t^{(k)}(i) \hat{\beta}_t^{(k)}(i) \frac{1}{c_t^{(k)}}}{\sum_{k=1}^{K} \sum_{t=1}^{|\Gamma|_k - 1} \hat{\alpha}_t^{(k)}(i) \hat{\beta}_t^{(k)}(i) \frac{1}{c_t^{(k)}}} \tag{4.64}$$

Donde K es el número de secuencias de observaciones y $|\Gamma|_k$ el tamaño de la secuencia de observacion k.

- **Insuficiencia de datos de entrenamiento (suavizado)**

La secuencia de observaciones usadas en el entrenamiento es necesariamente finita, lo cual da lugar a un número insuficiente de ocurrencias de los diferentes eventos del modelo, para proporcionar buenas estimaciones de los parámetros, especialmente las probabilidades de observación. En el caso de los DHMM, aunque los símbolos más frecuentes serán bien entrenados, si un símbolo no aparece nunca en un estado durante el entrenamiento se dará valor de 0, a su probabilidad en la distribución correspondiente a dicho estado, y si luego aparece en el reconocimiento esta probabilidad cero, puede ser atribuida a todo el modelo.

La solución a este problema son las técnicas de suavizado (smoothing en ingles) [Hernando, 1993], que consisten en un procesado de las distribuciones de probabilidad de observación posterior al entrenamiento.

En [Hernando, 1993] se detallan varios métodos (método floor smoothing, método de Parzen, método de distandas mutuas, método de correlaciones, método de alineación de secuencias, etc.). En el presente trabajo se uso el método llamado floor smoothing por su simpleza, que consiste en asegurarse que ninguna probabilidad de observación esté por

debajo de un determinado umbral η, es decir:

$$b_j(k) \geq \eta \ ; \ j = 1, 2, \ldots, N_s \ ; \ k = 1, 2, \ldots, N_a \tag{4.65}$$

Si esta restricción es violada, se realiza la corrección manualmente modificando la probabilidad en cuestión y reescalando el resto de las probabilidades para que se cumplan las restricciones estocásticas. Esta técnica resuelve el problema mencionado de la probabilidad cero o nula y es suficiente para obtener modelos razonablemente entrenados, por lo cual su uso es general.

4.4.5. Implementación de un reconocedor de voz utilizando DHMM

Para implementar un sistema de reconocimiento de palabras aisladas mediante DHMM se necesitan las siguientes fases:

1. A la señal de voz se le aplica un análisis espectral ya sea Espectrogramas, MFCC o LPC para obtener una secuencia de vectores característicos.

2. Para poder aplicar DHMM se necesita la discretización del espacio vectorial de características espectrales, para pasar de una secuencia de vectores a una secuencia de símbolos correspondiente a un alfabeto finito. Esto se consigue mediante la cuantificación vectorial (VQ) de los vectores de característicos, y la asociación de las palabras-código del diccionario del cuantificador, con los símbolos del alfabeto del modelo.

3. Una vez obtenida la secuencia de símbolos a partir de la señal de voz se procede a la fase de entrenamiento del DHMM, se realiza en base a un conjunto de observaciones (símbolos) obtenidas para cada palabra, y de una determinada inicialización de los parámetros, en el presente trabajo la inicialización se realizó de manera aleatoria y se utilizó el algoritmo de Baum-Welch descrito anteriormente.

4. A continuación se procede al suavizado de los modelos, en este trabajo se aplico el método de floor smoothing.

5. Una vez obtenido los parámetros del DHMM (un HMM por palabra), este se guarda en el diccionario de palabras.

6. La última fase es la de reconocimiento, se selecciona el modelo más probable λ_κ, perteneciente al conjunto de modelos correspondientes a las diferentes palabras a reconocer; dada la secuencia de observación $O = \{o_t\}$; $t = 1, 2, \ldots, |\Gamma|$, correspondientes a la palabra de "prueba". Por tanto, para identificar la palabra de "prueba"basta con calcular las probabilidades a posteriori con un algoritmo eficiente (p. e. Forward-Backward), y seleccionar el modelo de mayor probabilidad.

4.5. Conclusiones

DTW es una técnica de comparación de patrones, y su principal ventaja es que es fácil de implementar comparado con HMM.

La principal ventaja de usar HMM como técnica de reconocimiento, es que se basa en un proceso estocástico que es más versátil que DTW; ya que son muy utilizados en el reconocimiento continuo de voz, mientras que DTW se utiliza básicamente para implementar sistemas de reconocimiento de palabras aisladas.

Se debe tener cuidado a la hora de implementar DHMM, principalmente en el escalado dinámico de $\beta_t(i)$ y $\alpha_t(i)$, y en la etapa de entrenamiento, por que si no se entrena de manera adecuada el HMM, el sistema de reconocimiento no funcionará.

La etapa de entrenamiento de un HMM, es la que requiere más tiempo de cómputo, sin embargo esta etapa se hace offline, por lo tanto no afecta al tiempo de respuesta de la etapa del reconocimiento. Por esta razón es que los HMM tiene una respuesta más rápida en la etapa de reconocimiento que DTW.

La última etapa que faltaba por explorar era la de reconocimiento, así que con este capítulo se concluye la teoría necesaria para empezar a desarrollar sistemas de reconocimiento de voz. En el siguiente capítulo se describe como se implementaron los sistemas de reconocimiento de palabras aisladas.

Capítulo 5

Sistema implementado

En este capítulo se explican los pasos que se utilizan para implementar reconocedores de palabras aisladas monolocutor muestreado a 8000Hz, cuantizado con 16-bits, mono, little-endian. Se utilizan tres métodos de extracción de características: Espectrogramas, MFCC y LPC; así como dos técnicas de reconocimiento DTW y HMM, probando con diferentes métricas o distancias.

5.1. Características del sistema implementado

Se implementaron seis reconocedores de palabras aisladas basados en: Espectrogramas-DTW, MFCC-DTW, LPC-DTW, Espectrogramas-DHMM, MFCC-DHMM y LPC-DHMM; reconocedores ampliamente usados en el estado del arte [Rabiner, 1989, Cowling and Sitte, 2002 Oropeza, 2006, Puertas, 2000].

En los sistemas implementados utilizando DTW el procedimiento es similar, la etapa que cambia es análisis espectral (Espectrogramas, MFCC y LPC). En la Figura 5.1 se muestra el diagrama de bloques utilizado para implementar estos sistemas.

Los sistemas implementados utilizando DHMM la estructura es similar a la utilizada en DTW, la etapa que se modifica es la de crear el diccionario y obviamente la etapa de reconocimiento. En la Figura 5.2 se muestra el diagrama de bloques utilizado para im-

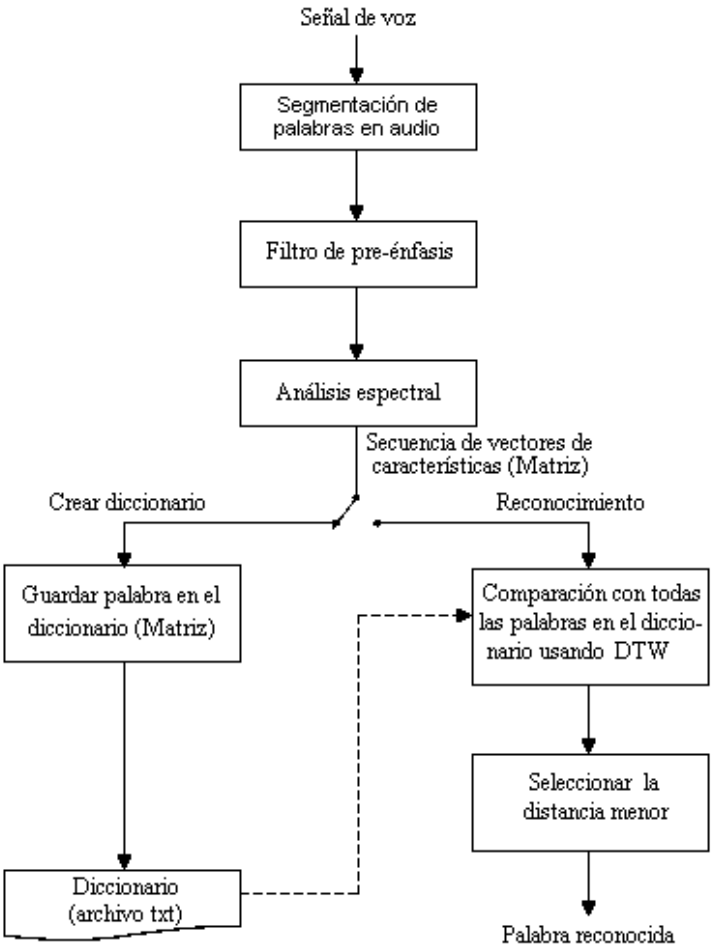

Figura 5.1: Diagrama de bloques para la implementación de un sistema de reconocimiento de palabras aisladas utilizando DTW

plementar reconocimiento de voz utilizando DHMM.

A continuación se detalla cada una de las etapas del sistema de la Figura 5.1 y el sistema de la Figura 5.2.

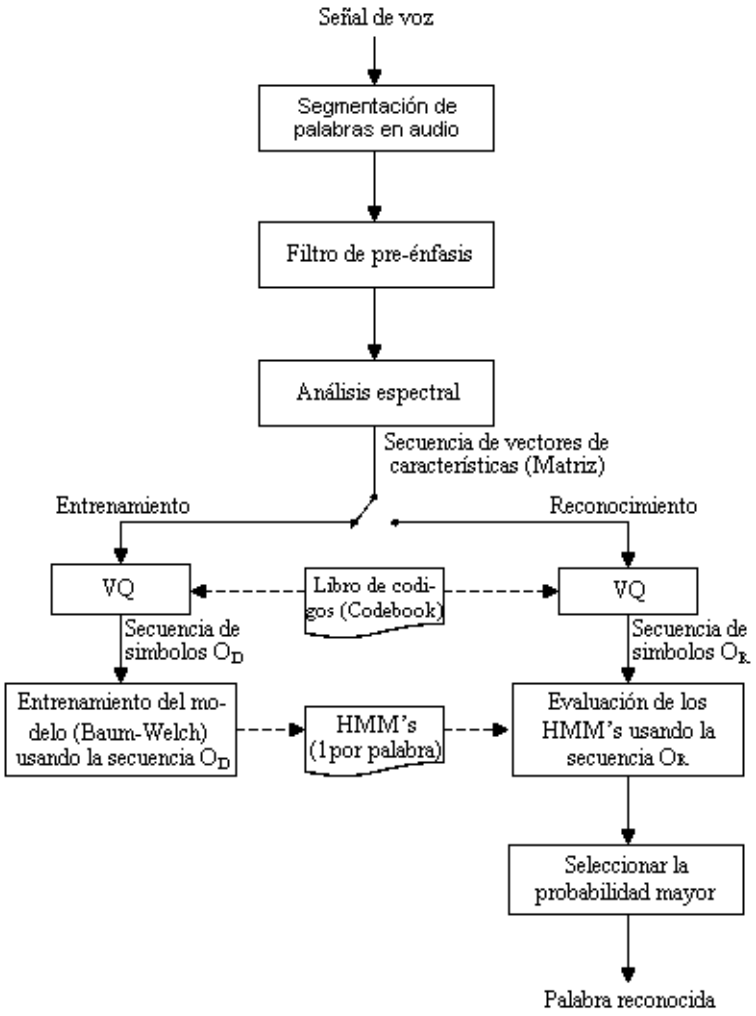

Figura 5.2: Diagrama de bloques para la implementación de un sistema de reconocimiento de palabras aisladas utilizando DHMM

5.2. Segmentación de palabras en audio

La entrada es una señal de voz muestreada a 8000Hz, cuantizado con 16-bits, mono, little-endian grabada con el micrófono integrado de la portatil y a la salida se obtiene la palabra segmentada.

Para segmentar la palabra en audio se utilizó la entropía de tiempo corto de la señal, debido a que la mayoría de las palabras contienen silencio en medio de la palabra (ver Figura 2.10 *a*) y *b*)), el inicio de la palabra se encuentra analizando la señal de voz de izquierda a derecha, y cuando la entropía sea mayor a un umbral Υ, se obtiene el inicio de la palabra. El final se encuentra de la misma forma que el inicio pero analizando la señal de voz de derecha a izquierda.

Realizando pruebas para diferentes valores de Υ, se obtuvieron resultados satisfactorios usando un $\Upsilon = 0.81 \cdot H_{max}$, donde H_{max} es la entropía máxima de la señal. En la Figura 2.10 *b*) y *d*) se muestran ejemplos del segmentador de palabras implementado usando entropía.

Se utilizo un tamaño de la ventana de 30ms (240 muestras), para obtener una mejor precisión de donde inicia o termina la palabra, se utilizo un avance consecutivo.

5.3. Filtro de pre-énfasis

Este corresponde al acondicionamiento necesario para compensar la caída que sufre la señal de voz en las altas frecuencias luego de salir de los labios.

El preénfasis es un filtro con características de pasa altas, donde las frecuencias bajas se reducen y las superiores a 1Khz se amplifican, es decir, se somete la señal a un filtro pasa altas con el objeto de aumentar los niveles de las frecuencias agudas para que no sean despreciadas al realizar el análisis espectral.

Un filtro de pre-énfasis esta dado por:

$$x'(n) = x(n) - \sigma x(n-1) \tag{5.1}$$

Típicamente se utiliza $0.96 \leq \sigma \leq 0.99$ [Iosu, 1999], en el presente trabajo se utiliza $\sigma = 0.98$.

5.4. Análisis espectral

Una vez que se ha aplicado el filtro de pre-énfasis se procede a hacer el análisis espectral, este proceso consiste en convertir la señal de voz en algún tipo de representación paramétrica sumamente compactada para su posterior análisis.

Se implementaron tres métodos para la extracción de características: Espectrogramas, MFCC y LPC.

5.4.1. Espectrogramas

Para obtener el espectrograma de una palabra se necesita:

1. El tamaño del marco sea potencia de 2 para poder aplicar el algoritmo FFT, en el presente trabajo se utiliza un tamaño de marco de 32ms (256 muestras) con un avance de 10ms (80 muestras).

2. A cada marco se le aplica la ventana de Hamming dada por la ecuación 5.2.

3. Se obtiene la DFT usando el algoritmo FFT, como resultado se obtienen dos vectores $\vec{dft}_{real}$ y $\vec{dft}_{imag}$.

4. Se obtiene la magnitud de la DFT ($X(k)$), aplicando la ecuación 5.3.

5. Se le aplica la escala de Bark utilizando 18 bandas (0-17), ya que la frecuencia máxima que se puede tener es de 4000Hz ($f_s = 8000Hz$). Para obtener la frecuencia se utiliza la ecuación 5.4.

En la Figura 5.3 se muestra el diagrama de flujo para obtener el espectrograma de una palabra.

$$w(n) = 0.54 - 0.46cos\left(\frac{2\pi n}{N}\right) = 0.54 - 0.46cos\left(\frac{6.2832n}{256}\right) \tag{5.2}$$

Donde N es el tamaño de la ventana, en este caso $N = 256$ muestras, $n = 0, 1, 2, \ldots, N-1$.

$$|X(k)| = \vec{dft}_{real}(k) \cdot \vec{dft}_{real}(k) + \vec{dft}_{imag}(k) \cdot \vec{dft}_{imag}(k) \tag{5.3}$$

Donde $\vec{dft}_{real}$ es la parte real de la $X(k)$ y $\vec{dft}_{imag}$ es la parte imaginaria.

$$f(n) = n\left(\frac{f_s}{N}\right) = n\left(\frac{8000}{256}\right) \tag{5.4}$$

Donde f_s es la frecuencia de muestreo, en el presente trabajo se utiliza una $f_s = 8000Hz$.

5.4.2. MFCC

Los coeficientes cepstrales de Mel se determinan de la siguiente manera:

1. La señal de audio es dividida en marcos de 32ms (256 muestras), avanzando el marco 10ms (80 muestras).

2. Se aplica la DFT (utilizando FFT, tamaño del marco es potencia de 2), es decir, se obtiene $X(k)$.

3. La magnitud de $X(k)$ se escala logarítmicamente tanto en frecuencia como en magnitud, para escalar logarítmicamente en frecuencia se utiliza el banco de filtros de Mel $H(k, m)$, y posteriormente se escala logarítmicamente la Magnitud. La Magnitud de $X(k)$ se obtiene mediante la ecuación 5.3.

4. Por último se obtienen los coeficientes cepstrales de Mel aplicando la Transformada Coseno Discreta (DCT).

En la Figura 5.4 se muestra de manera detallada el diagrama de flujo que se utiliza para obtener los coeficientes cepstrales de Mel.

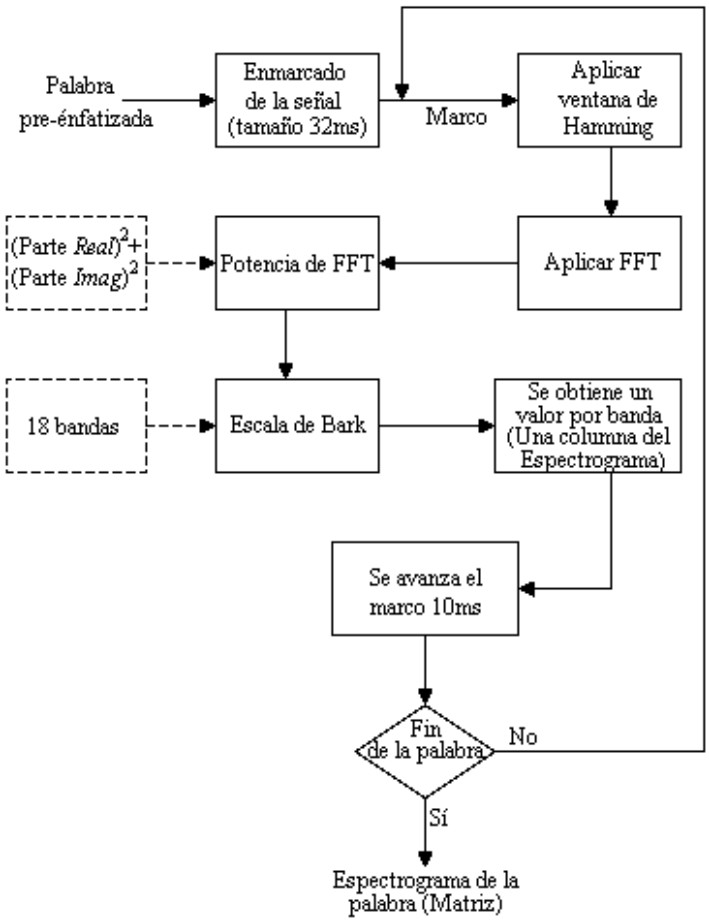

Figura 5.3: Diagrama de flujo para la obtención del espectrograma de una palabra

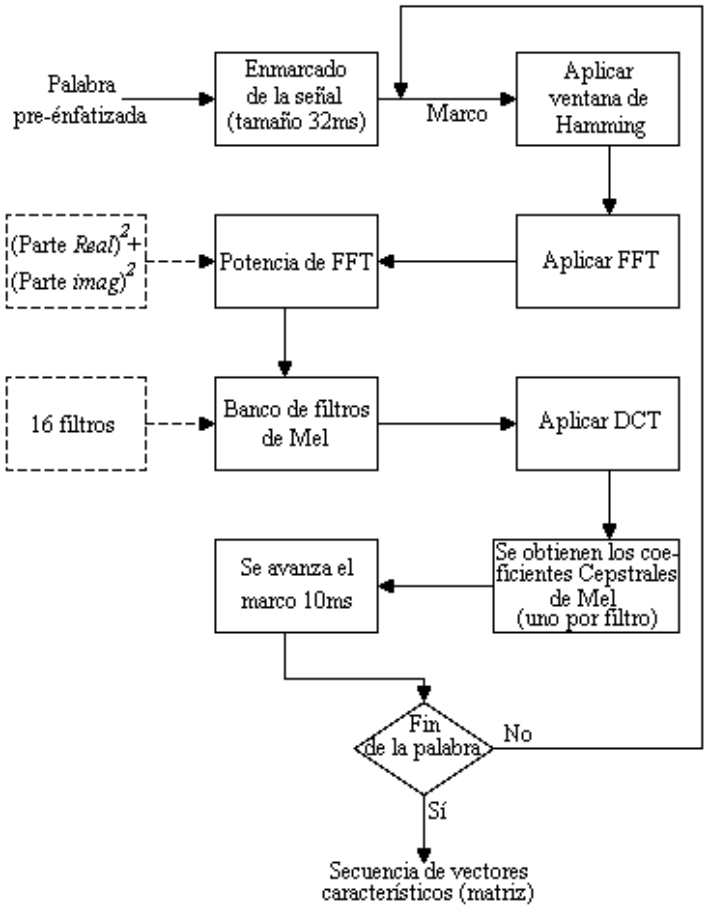

Figura 5.4: Diagrama de flujo para la obtención de los coeficientes MFCC

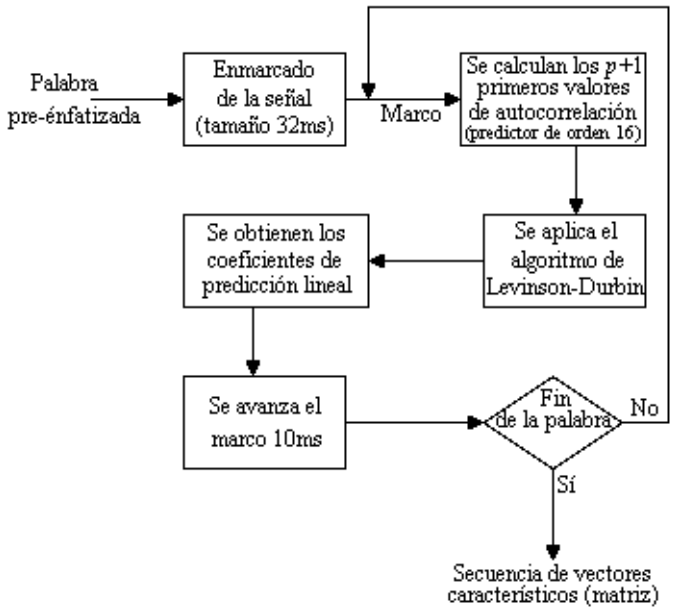

Figura 5.5: Diagrama de flujo para la obtención de los coeficientes LPC

5.4.3. LPC

Para obtener los coeficientes de predicción lineal se utiliza un tamaño de ventana de 32ms (256 muestras) y se avanza el marco 10ms (80 muestras), para cada marco de la señal es necesario calcular lo siguiente:

1. Calcular los $p+1$ primeros valores de autocorrelación $\hat{R}(0), \hat{R}(1), \ldots, \hat{R}(p)$, en este caso se utiliza un predictor de orden 16, es decir, $p = 16$. Los valores de autocorrelación se calculan usando el algoritmo de Blankenship.

2. Se aplica el algoritmo de Levinson-Durbin para resolver el sistema de ecuaciones (ecuación 3.34), y de esta forma obtener los coeficientes de predicción lineal α_k.

En la Figura 5.5 se muestra de manera detallada el diagrama de flujo que se utiliza para obtener los coeficientes de predicción lineal.

5.5. Doblado Dinámico en Tiempo

En este apartado se explica de manera detallada como se crea el diccionario y se hace el reconocimiento mediante DTW (ver Figura 5.1).

5.5.1. Diccionario DTW

El diccionario se utiliza para guardar las palabras que se reconocerán, para crear el diccionario DTW simplemente se almacena la secuencia de vectores característicos obtenidos en la etapa de análisis espectral con la etiqueta del nombre de la palabra y el número de marcos de la misma. Al diccionario es un archivo txt con el siguiente formato:

<Etiqueta>,<Núm. de marcos>
1,<valor 0>,<valor 1>,<valor 2>, ... ,<valor N-1>
2,<valor 0>,<valor 1>,<valor 2>, ... ,<valor N-1>
⋮

Donde *<Etiqueta>* es el nombre de la palabra, *<valor i>* en el caso de espectrogramas es la energía de la banda i y el número de filtros es 18, en el caso de MFCC y LPC es el $i-$ésimo coeficiente, $M = 16$ y $p = 16$ respectivamente. El número de marcos se calcula mediante la siguiente ecuación:

$$\text{Num. de marcos} = \left\lceil \frac{N}{\Delta.} \right\rceil \tag{5.5}$$

Donde N es el tamaño de la señal y Δ es el avance del marco, el avance del marco es de 10ms (80 muestras), por lo tanto:

$$\text{Num. de marcos} = \left\lceil \frac{\text{N}}{80} \right\rceil \tag{5.6}$$

5.5.2. Reconocimiento DTW

Para el reconocimiento de la palabra la secuencia de vectores característicos se compara con todas las palabras almacenadas en el diccionario usando el algoritmo de DTW descrito en la sección 4.2, usando la restricción local de la ecuación 5.7.

$$D_{0,0} = d_{0,0}$$

$$D_{n,0} = d_{n,0} + D_{n-1,0}$$

$$D_{0,m} = d_{0,m} + D_{0,m-1}$$

$$D_{n,m} = min \begin{cases} D_{n-1,m-1} + 2d_{n,m} \\ D_{n-1,m} + d_{n,m} \\ D_{n,m-1} + d_{n,m} \end{cases} \tag{5.7}$$

donde $d_{n,m}$ es la distancia entre el vector n de la palabra a reconocer y el vector m de la palabra en el diccionario, esta distancia se calcula utilizando la distancia euclidiana y posteriormente la distancia coseno descritas en la sección 4.1.

La secuencia de vectores característicos son representados mediante una matriz de tamaño (18)(Núm. de marcos) en el caso de espectrogramas y de tamaño (16)(Núm. de marcos) en el caso de LPC y MFCC.

La palabra con menor distancia acumulada ϑ es la palabra reconocida.

$$\vartheta = \underset{1 \leq i \leq N_{pal.}}{min} \frac{D^{(i)}_{N,N_2}}{N + N_2} \tag{5.8}$$

Donde N_{pal} es el número de palabras almacenadas en el diccionario, $D^{(i)}_{N,N_2}$ es la distancia acumulada por DTW entre la $i-$ésima palabra del diccionario y la palabra a reconocer, N es el tamaño de la palabra en el diccionario y N_2 es el tamaño de la palabra a reconocer.

Para obtener las curvas ROC se necesitan todas las distancias acumuladas para ello las distancias acumuladas $\frac{D^{(i)}_{N,N_2}}{N+N_2}$, para $1 \leq i \leq N_{pal.}$, se almacenan en un archivo txt con el siguiente formato:

$<DTW(\mathcal{P}_{d1},\mathcal{P}_{r1})>$

$<DTW(\mathcal{P}_{d2},\mathcal{P}_{r1})>$

$<DTW(\mathcal{P}_{d3},\mathcal{P}_{r1})>$

$\vdots$

$<DTW(\mathcal{P}_{dN_{pal.}},\mathcal{P}_{r1})>$

$<DTW(\mathcal{P}_{d1},\mathcal{P}_{r2})>$

$<DTW(\mathcal{P}_{d2},\mathcal{P}_{r2})>$

$\vdots$

$<DTW(\mathcal{P}_{dN_{pal.}},\mathcal{P}_{r2})>$

$\vdots$

$<DTW(\mathcal{P}_{dN_{pal.}},\mathcal{P}_{rN_{rec}})>$

Donde $\mathcal{P}_{di}$ es la $i-$ésima palabra almacenada en el diccionario y $\mathcal{P}_{ri}$ es la $i-$ésima palabra a reconocer, en este caso $N_{pal} = 50$ y N_{rec} es el total de palabras a reconocer, en este caso $N_{rec} = 100$.

5.6. Modelos Discretos Ocultos de Markov

En este apartado se explica de manera detallada como se crea el diccionario y se hace el reconocimiento mediante DHMM (ver Figura 5.2).

5.6.1. Creación del libro de códigos (codebook)

La aplicación de los DHMM al reconocimiento de voz requiere la obtención de una secuencia de símbolos O, correspondiente a un alfabeto discreto y finito a partir de una secuencia de vectores que representan las características espectrales de la señal de voz. Por esta razón es necesario hacer una discretización del espacio vectorial, esto se realiza usando el algoritmo K-medias. El diagrama de flujo del algoritmo K-medias se puede ver en la Figura 4.5.

Para realizar el codebook se necesitan los siguientes pasos:

1. Se realiza una grabación de 1.2 minutos.

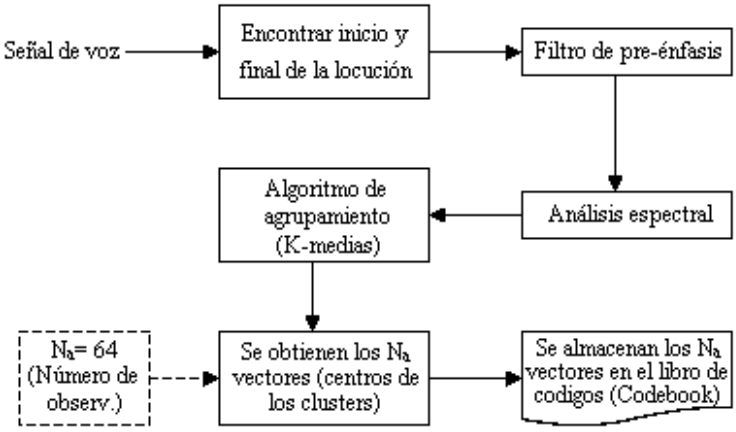

Figura 5.6: Diagrama de flujo para realizar el libro de códigos (codebook)

2. Se segmenta la señal de audio para eliminar el ruido del ambiente o el silencio al inicio y final de la grabación.

3. Una vez segmentada se le aplica el filtro pre-énfasis.

4. Posteriormente se realiza el análisis espectral. El análisis espectral genera una secuencia de vectores característicos.

5. Se aplica el algoritmo K-medias a los vectores característicos para obtener N_a clases. El centroide de cada clase es un vector con el cual se realizará la cuantización.

6. Los N_a (N_a es el número de observaciones, en el presente trabajo $N_a = 64$) centroides se almacenan en un archivo de texto, este archivo se llama codebook.

En la Figura 5.6 se puede ver el diagrama de flujo para realizar el codebook.

El codebook se almacena con el siguiente formato:

$<N_a>$

0, <vector(0)>, <vector(1)>, ..., <vector(N-1)>

1, <vector(0)>, <vector(1)>, ..., <vector(N-1)>

2, <vector(0)>, <vector(1)>, ..., <vector(N-1)>

$\vdots$

$<N_a-1>$, *<vector(0)>, <vector(1)>, ..., <vector(N-1)>*

Donde *<vector(i)>*, es el i-ésimo valor del centroide.

Se crean tres codebooks uno para cada método de extracción de características (Espectrogramas, MFCC y LPC respectivamente), también se crearon tres diccionarios uno para cada método de extracción de características. A continuación se explica como se crearon los diccionarios.

5.6.2. Diccionario DHMM

Para crear el diccionario DHMM se realiza lo siguiente:

1. Una vez obtenida la secuencia de vectores característicos se hace la cuantización vectorial (VQ) utilizando el codebook, esta se realiza tomando vector por vector característico y comparando con todos los vectores almacenados en el codebook; utilizando la distancia euclidiana o coseno, se cambia el vector por el número del vector con el cual la distancia sea menor, es decir, se cambia el vector característico por un número), al final se obtiene una secuencia de simbolos O.

2. El paso 1 se realiza para tres locuciones diferentes de la misma palabra, esto con el fin de obtener múltiples secuencias de observaciones.

3. Una vez obtenidas las tres secuencias de observaciones, se entrena un HMM usando el algoritmo de Baum-Welch, para implementar este algoritmo se necesitan los algoritmos Forward y Backward escalados, para la inicialización de los parámetros la matriz

A y B se inicializan de manera aleatoria.

La matriz A se determina de manera tal que el modelo entrenado sea un modelo de izquierda-derecha como el que se muestra en la Figura 5.7, porque la secuencia de estados subyacente asociado con el modelo tiene la propiedad de que a medida que el tiempo aumenta el índice del estado aumenta (o sigue siendo el mismo), es decir, los estados proceden de izquierda a derecha [Hernando, 1993].

Es evidente que el HMM izquierda-derecha tiene la propiedad deseable que fácilmente pueden modelar señales cuyas propiedades cambian con el tiempo por ejemplo, el habla. Para implementar modelos izquierda-derecha las restricciones adicionales se colocan sobre los coeficientes de transición de estados para asegurarse de que no ahiga cambios grandes en los índices de los estados, por lo que una limitación de la forma:

$$a_{i,j} = 0 \text{ para } (j < i) \text{ ó } (j > i + \Delta_d) \tag{5.9}$$

se utiliza a menudo [Rabiner, 1989].

En particular, para el ejemplo de la Figura 5.7, el valor de Δ_d es 2, es decir, no se permiten saltos de más de dos estados. La forma de la matriz de transición de estados para el ejemplo de la Figura 5.7 es

$$A = \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & 0 \\ 0 & a_{1,1} & a_{1,2} & a_{1,3} \\ 0 & 0 & a_{2,2} & a_{2,3} \\ 0 & 0 & 0 & a_{3,3} \end{bmatrix} \tag{5.10}$$

En el presente trabajo se utiliza un HMM de izquierda-derecha de 5 y 9 estados con un $\Delta_d = 2$.

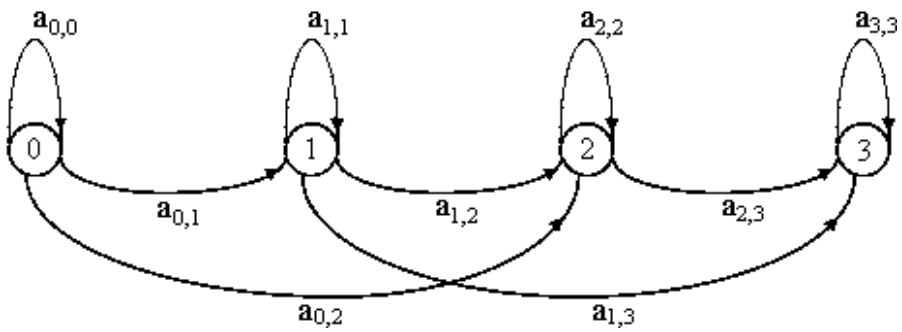

Figura 5.7: HMM izquierda-derecha

El diccionario es un archivo de texto con el siguiente formato:

$<N_s>$, $<N_a>$

<Etiqueta>

A

$< a_{0,0} >,< a_{0,1} >,\cdots,< a_{0,N_s} >$

$< a_{1,0} >,< a_{1,1} >,\cdots,< a_{1,N_s} >$

$\vdots$

$< a_{N_s,0} >,< a_{N_s,1} >,\cdots,< a_{N_s,N_s} >$

B

$< b_0(0) >,< b_0(1) >,< b_0(2) >,\cdots,< b_0(N_a) >$

$< b_1(0) >,< b_1(1) >,< b_1(2) >,\cdots,< b_1(N_a) >$

$\vdots$

$< b_{N_s}(0) >,< b_{N_s}(1) >,< b_{N_s}(2) >,\cdots,< b_{N_s}(N_a) >$

<Etiqueta>

A

$\vdots$

B

$\vdots$

donde N_s es el número de estados y N_a es el número de observaciones, en el diccionario no es necesario almacenar Π, ya que se considera que el modelo inicia en el primer estado, es decir $\Pi = \{1, 0, 0, \ldots, 0\}$. *<Etiqueta>* es el nombre de la palabra almacenada, A y B son

etiquetas para saber donde inicia, y termina cada matriz.

5.6.3. Reconocimiento DHMM

Para el reconocimiento de palabras se realizan los siguientes pasos:

1. Una vez obtenida la secuencia de vectores característicos se realiza la VQ como resultado se obtiene una secuencia de observaciones.

2. Con la secuencia de observaciones se evaluan todos los HMM almacenados en el diccionario y se selecciona el modelo λ_κ tal que la probablidad sea mayor, la etiqueta del modelo λ_κ será la palabra reconocida.

La evaluación de los HMM se realiza utilizando el algoritmo Forward escalado. Antes de realizar la evaluación se realiza el suavizado de los modelos, en este trabajo se aplico el método de floor smoothing (descrito en la sección 4.4.4) con un $\eta = 1e^{-10}$.

Para obtener las curvas ROC se necesitan todas las probabilidades obtenidas de evaluar los modelos para ello las probabilidades obtenidas se guardan en un archivo de texto con el siguiente formato:

$P(O_{p1}|\lambda_1)$
$P(O_{p1}|\lambda_2)$
$P(O_{p1}|\lambda_3)$
$\vdots$
$P(O_{p1}|\lambda_{N_{hmm's}})$
$P(O_{p2}|\lambda_1)$
$P(O_{p2}|\lambda_2)$
$P(O_{p2}|\lambda_3)$
$\vdots$
$P(O_{pN_{rec}}|\lambda_{N_{hmm's}})$

Donde O_{pi} es la secuencia de observaciones de la $i-$ésima palabra a reconocer, λ_i es el $i-$ésimo HMM almacenado en el diccionario, $N_{hmm's}$ es el número de HMMs almacenados

en el diccionario (en este caso $N_{hmm's} = 50$) y N_{rec} es el número de palabras a reconocer (en este caso $N_{rec} = 100$).

5.7. Conclusiones

Con base en el interés por conocer el funcionamiento y el de poder desarrolar sistemas de reconocimiento de voz, es que se han implementado estos sistemas sencillos pero completos para el reconocimiento de palabras aisladas monolocutor haciendo uso de los principios expuestos a lo largo del documento.

Los sistemas se implementaron mediante módulos para facilitar la integración de nuevos métodos de extracción de características y/o técnicas de reconocimiento.

De las técnicas de extracción de características la más fácil de implementar es espectrogramas, y la que más compleja de implementar es MFCC.

Para obtener los MFCC, la etapa más dificil de implementar es aplicar el banco de filtros de Mel a la magnitud de $X(k)$.

Para obtener los LPC se debe de tener especial cuidado al momento de calcular la autocorrelación de manera adecuada, de los contrario el sistema no reconocerá las palabras.

Los resultados obtenidos y el rendimiento de los sistemas se exponen en el siguiente capítulo.

Capítulo 6

Resultados

6.1. Gráficas ROC

Las gráficas Característica de Operación de Receptor (ROC por sus siglas en inglés) [Fawcett, 2004], son una técnica útil para la organización de los clasificadores y visualizar su desempeño. Las gráficas ROC han sido utilizados en la teoría de detección de señales para representar el equilibrio entre la tasa de aciertos y la tasa de falsas alarmas de clasificadores. El análisis ROC se ha ampliado su uso en la visualización y el análisis del comportamiento de los sistemas de diagnóstico.

6.1.1. Rendimiento de un clasificador

Dado un clasificador y una instancia [Fawcett, 2004], existen cuatro posibles resultados:

1. Si la instancia es positiva y es clasificada como positiva se le cuenta como un positivo verdadero (TP por sus siglas en inglés).

2. Si la instancia es positiva y es clasificada como negativa se le cuenta como un negativo falso (FN por sus siglas en inglés).

3. Si la instancia es negativa y es clasificada como negativa, se le cuenta como un negativo

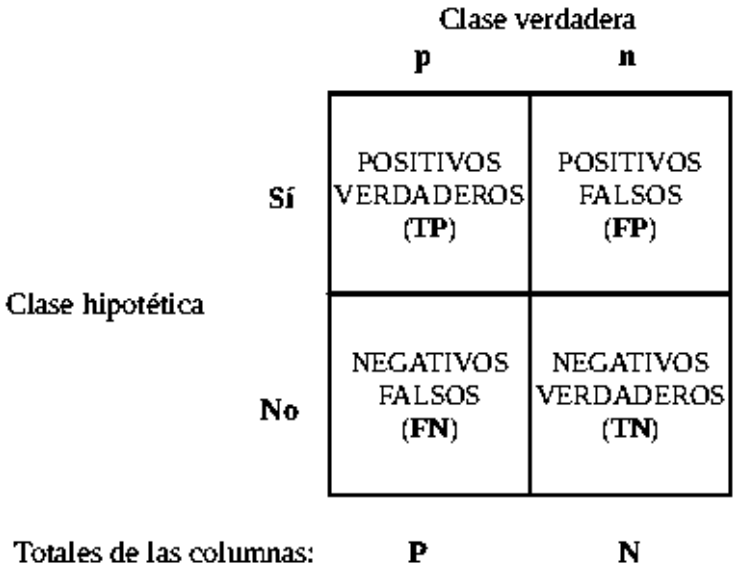

Figura 6.1: Matriz de confusión

verdadero (TN por sus siglas en inglés).

4. Si la instancia es negativa y es clasificada como positiva, se le cuenta como un positivo falso (FP por sus siglas en inglés).

Dado un clasificador y un conjunto de instancias (conjunto de pruebas), se puede construir una matriz de confusión de dos-por-dos (también llamada tabla de contingencia) la cual representa las disposiciones del conjunto de instancias. En la Figura 6.1 se muestra una matriz de confusión a partir de la cual se derivan las siguientes ecuaciones [Fawcett, 2004] (los símbolos utilizados son los que maneja el autor [Fawcett, 2004], por lo tanto no estan incluidos en la tabla de símbolos):

$$tasa\ de\ positivos\ falsos \approx \frac{Negativos\ incorrectamente\ clasificados}{Total\ de\ negativos} = \frac{FP}{N} \tag{6.1}$$

$$tasa\ de\ positivos\ verdaderos \approx \frac{Positivos\ correctamente\ clasificados}{Total\ de\ positivos} = \frac{TP}{P} \tag{6.2}$$

$$P = TP + FN \tag{6.3}$$

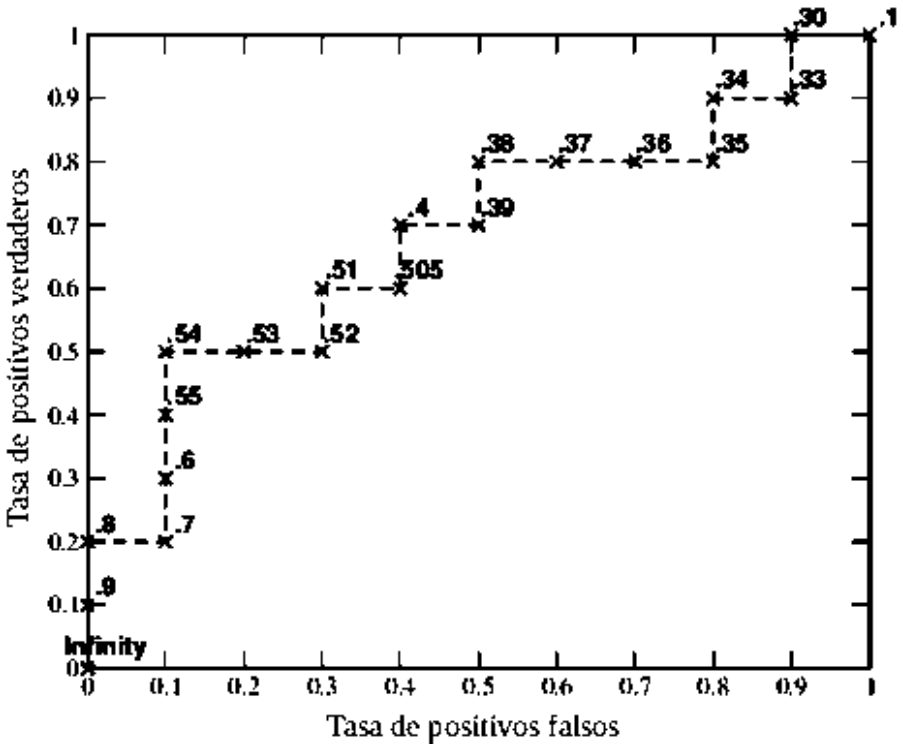

Figura 6.2: Gráfica ROC (Distintos clasificadores en el espacio ROC)

$$N = FP + TN \tag{6.4}$$

Las gráficas ROC son gráficos bidimensionales en los cuales la tasa de positivos verdaderos se dibuja en el eje Y y la tasa de positivos falsos se dibuja en el eje X. Un gráfico ROC revela el equilibrio entre los beneficios (positivos verdaderos) y los costos (los positivos falsos). La Figura 6.2 muestra una gráfica ROC. En el espacio ROC se destacan varios puntos importantes.

- La esquina inferior izquierda $(0, 0)$ representa la estrategia de no emitir nunca una clasificación positiva, un clasificador como tal no comete errores por positivos falsos pero tampoco gana en positivos verdaderos.

- La estrategia opuesta al anterior es emitir clasificaciones positivas incondicionalmente, se representa por el punto en la esquina superior derecha (1,1).

- El punto $(0, 1)$ representa la clasificación perfecta.
- La línea diagonal $y = x$ representa la estrategia de suponer aleatoriamente una clase. Por ejemplo, si un clasificador supone aleatoriamente la clase positiva la mitad del tiempo, se espera obtener la mitad de los positivos y la mitad de los negativos correctos, este es el peor clasificador.

Informalmente, un punto dentro del espacio ROC es mejor que otro si se ubica al noroeste (la tasa de positivos verdaderos es mayor, la tasa de positivos falsos es menor, o ambas cosas) del primero.

6.1.2. Graficación de los resultados obtenidos

Para generar un conjunto de puntos que permitan el trazado de la curva ROC, se utiliza un umbral que determine el resultado de la clasificación: si la salida del clasificador está por encima del umbral (en caso de HMM), el clasificador produce un *Sí* de otro modo produce un *No*, en el caso de DTW el clasificador debe estar por debajo del umbral. Cada valor de umbral produce un punto en el espacio ROC. Se utilizan 5000 umbrales, es decir se generan 5000 puntos (x, y).

El umbral para gráficar los resultados DTW y HMM toma los valores de los resultados obtenidos (distancia acumulada o probabilidad respectivamente) ordenados de menor a mayor. La gráfica que se obtiene esta escalonada así que se suaviza promediando 100 puntos.

6.2. Experimentos realizados

Los diccionarios creados son de 50 palabras, se realizo la elocución de las palabras mostradas en la tabla 6.1 una instancia de cada una, las palabras se eligieron en base a posibles instrucciones que se le puedes dar a un robot o computadora. La caracterización de la señal se lleva a cabo mediante Espectrogramas, MFCC y LPC; usando DHMM y DTW como técnica de reconocimiento.

Palabras				
cero	uno	dos	tres	cuatro
cinco	seis	siete	ocho	nueve
abajo	abrir	accesar	adelante	analizar
apagar	arriba	atras	brincar	caminar
cerrar	completar	configurar	contestar	continuar
corregir	construir	derecha	elegir	encender
escribir	escuchar	frenar	hablar	interpretar
inventar	izquierda	maximizar	minimizar	mover
ordenar	permitir	programar	rechazar	resuelve
suspender	tomar	verificar	viajar	vuelta

Tabla 6.1: Palabras utilizadas para crear los diccionarios

Se hicieron pruebas utilizando la distancias euclidiana, y coseno tanto en la creación del codebook como en la etapa de reconocimiento, para observar el rendimiento de estas en los diferentes sistemas de reconocimiento. En LPC-DTW también se implemento la distancia de Itakura descrita en la sección 3.3.5.

Para la etapa de reconocimiento de los diferentes sistemas descritos en el capítulo anterior, se examinaron las distancias existentes entre las elocuciones de las palabras a reconocer con las 50 elocuciones del diccionario.

En total se utilizaron 100 elocuciones de palabras a reconocer, dos por cada palabra almacenada en el diccionario, es decir, se obtienen 5000 diferentes distancias. Al repetir este procedimiento para los distintos sistemas usando distancia euclidiana y coseno se puede encontrar el rendimiento que demuestran las distintas opciones de caracterización de los sistemas implementados.

6.3. Resultados DHMM

Cuando se implementa DHMM como técnica de reconocimiento hay varias variables que se pueden modificar como por ejemplo, el número de estados, el número de símbolos, el número de secuencia de observaciones de entrenamiento, umbral del método floor smoothing η, etc.

Se observo el rendimiento usando 64 símbolos de observación, un umbral $\eta = 1e^{-10}$ y utilizando un modelo de 9 estados, posteriormente se utilizó un modelo de 5 estados. Para el codebook se utilizó el algoritmo K-medias usando la distancia euclidiana y coseno, para la VQ se utilizó la misma distancia que se utiliza en el codebook.

Los resultados obtenidos se exponen en la tabla 6.2, estos porcentajes de reconocimiento son seleccionando la palabra con la mayor probabilidad obtenida, pero lo que interesa es saber el comportamiento del sistema utilizando umbrales para así poder saber cual es el sistema con mayor tasa de positivos verdaderos pero con menor costo (positivos falsos), es decir que tan bien clasifica las palabras. Para obtener el comportamiento del sistema para diferentes umbrales se utilizan las curvas ROC.

En la Figura 6.3 se observa el comportamiento de los sistemas que usan Espectrogramas donde se puede ver a primera vista que la curva más cercana con la esquina superior izquierda del gráfico es la que utiliza un modelo de 9 estados y la distancia coseno, con una area bajo la curva de 0.90415. En la figura también se observa que la tasa de positivos falsos es muy grande, porque si se quiere reconocer el 90 % de las palabras (tasa de positivos verdaderos), reconocerá 20 % de palabras que no son correctas (tasa de positivos falsos), es decir, cuando se mencione una palabra que se encuentre en el diccionario hay un 90 % de probabilidad que la adivine correctamente y cuando se mencione una palabra que no este en el diccionario hay un 20 % de probabilidad que la reconozca como una palabra válida. En general estos sistemas tienen una baja tasa de reconocimiento y funcionan mucho mejor usando la distancia coseno.

Resultados DHMM			
Análisis espectral	***Distancia***	***Núm. de edos***	***% de palabras reconocidas***
Espectrogramas	Euclidiana	9	60 %
Espectrogramas	Coseno	9	62 %
Espectrogramas	Euclidiana	5	52 %
Espectrogramas	Coseno	5	63 %
MFCC	Euclidiana	9	74 %
MFCC	Coseno	9	85 %
MFCC	Euclidiana	5	78 %
MFCC	Coseno	5	86 %
LPC	Euclidiana	9	80 %
LPC	Coseno	9	76 %
LPC	Euclidiana	5	81 %
LPC	Coseno	5	74 %

Tabla 6.2: Resultados obtenidos de aplicar DHMM como técnica de reconocimiento

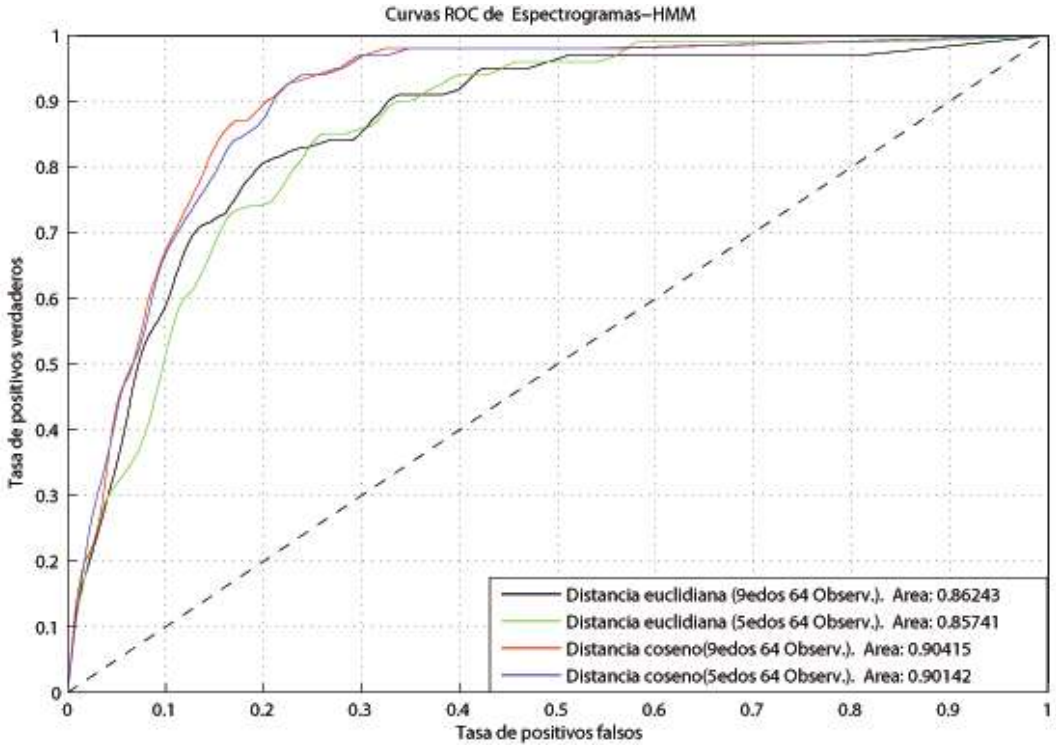

Figura 6.3: Curvas ROC utilizando Espectrogramas como método de extracción de características y DHMM como técnica de reconocimiento

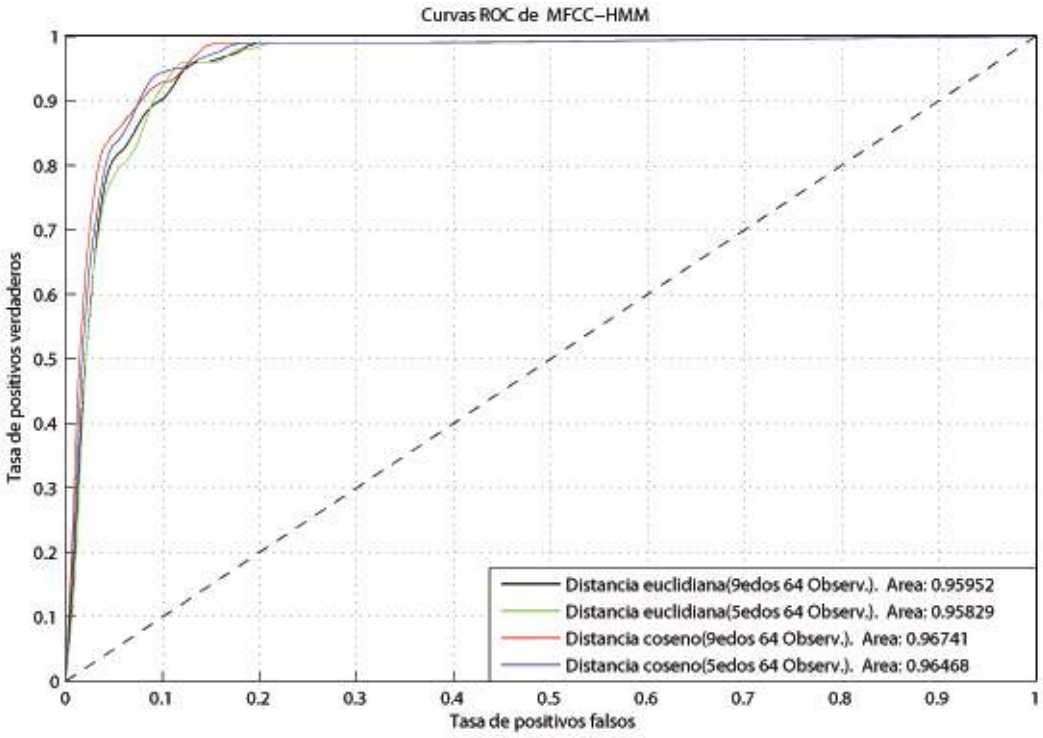

Figura 6.4: Curvas ROC utilizando MFCC como método de extracción de características y DHMM como técnica de reconocimiento

En la Figura 6.4 se observa el comportamiento de los sistemas que usan MFCC donde se observa que la curva más cercana con la esquina superior izquierda del gráfico es la que utiliza un modelo de 9 estados y la distancia coseno con una area bajo la curva de 0.96741. Se observa que la tasa de positivos falsos es grande ya que si se reconoce el 90 % de las palabras bien reconocerá aproximadamente el 7 % de palabras que no son correctas. En la figura también se observa la cercanía de las líneas con la esquina superior izquierda del gráfico, lo cual denota altas tasas de reconocimiento.

En los sistemas que usan MFCC la distancia coseno, es levemente mejor que la distancia euclidiana.

En la Figura 6.5 se observa el rendimiento de los sistemas que usan LPC donde se observa que la curva más cercana con la esquina superior izquierda del gráfico es la que

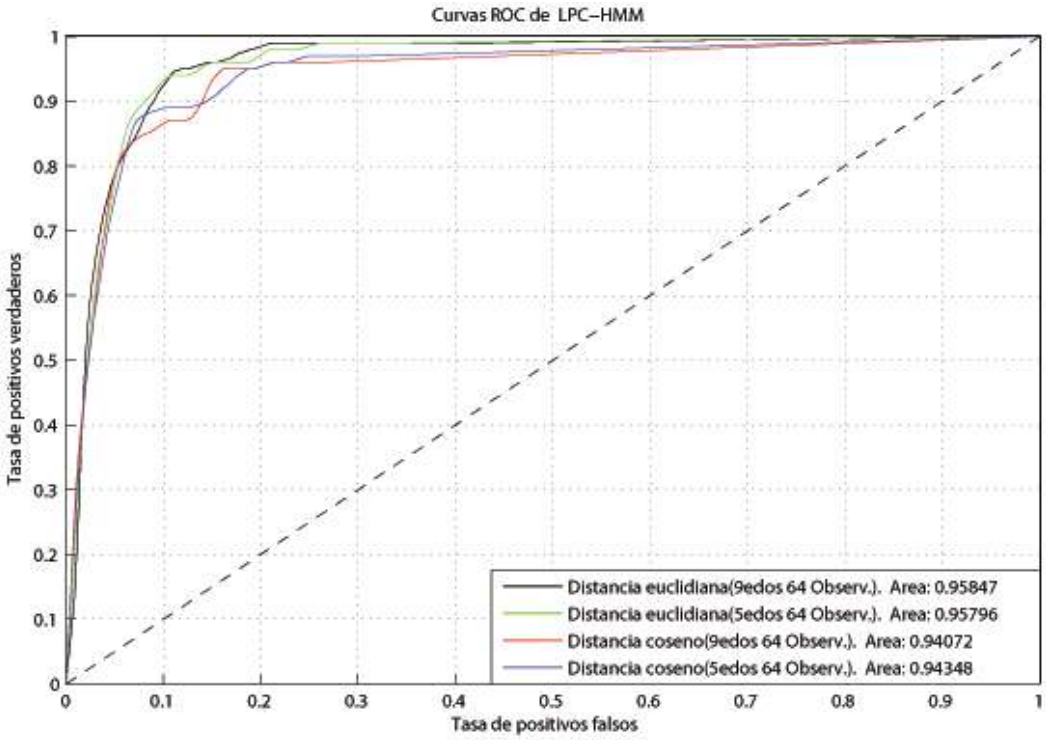

Figura 6.5: Curvas ROC utilizando LPC como método de extracción de características y DHMM como técnica de reconocimiento

utiliza un modelo de 9 estados y la distancia euclidiana con una area bajo la curva de 0.95847. La tasa de positivos falsos es grande ya que si se reconoce el 90 % de las palabras bien reconocerá aproximadamente el 8.5 % de palabras que no son correctas. En la figura también se observa que las líneas más cercanas con la esquina superior izquierda del gráfico son las que utilizan la distancia euclidiana comparadas con la distancia coseno.

En general los sistemas que usan DHMM como técnica de reconocimiento funcionan mejor usando modelos de 9 estados.

Resultados DTW		
Análisis espectral	***Distancia***	***Porcentaje de palabras reconocidas***
Espectrogramas	Euclidiana	24 %
Espectrogramas	Coseno	89 %
MFCC	Euclidiana	93 %
MFCC	Coseno	89 %
LPC	Euclidiana	96 %
LPC	Coseno	93 %
LPC	Itakura	96 %

Tabla 6.3: Resultados obtenidos de aplicar DTW como técnica de reconocimiento

6.4. Resultados DTW

Los resulados obtenidos se exponen en la tabla 6.3, estos porcentajes de reconocimiento son seleccionando la palabra con la menor distancia acumulada, pero al igual que los DHMM lo que interesa saber es el comportamiento del sistema utilizando umbrales, para ello se hace uso de las curvas ROC.

En la Figura 6.6 se observa el rendimiento de los sistemas que usan Espectrogramas donde se observa que la curva más cercana con la esquina superior izquierda del gráfico es la que utiliza distancia coseno con una area bajo la curva de 0.98031. En la figura también se observa que la línea mucho más cercana con la esquina superior izquierda del gráfico es la que utiliza la distancia coseno comparada con la línea de distancia euclidiana, es decir, definitivamente en Espectrogramas funciona mucho mejor la distancia coseno.

La tasa de reconocimiento usando distancia coseno es muy alta, en la Figura 6.6 se observa que para una tasa de positivos verdaderos del 60 % la tasa de positivos falsos es aproximadamente cero.

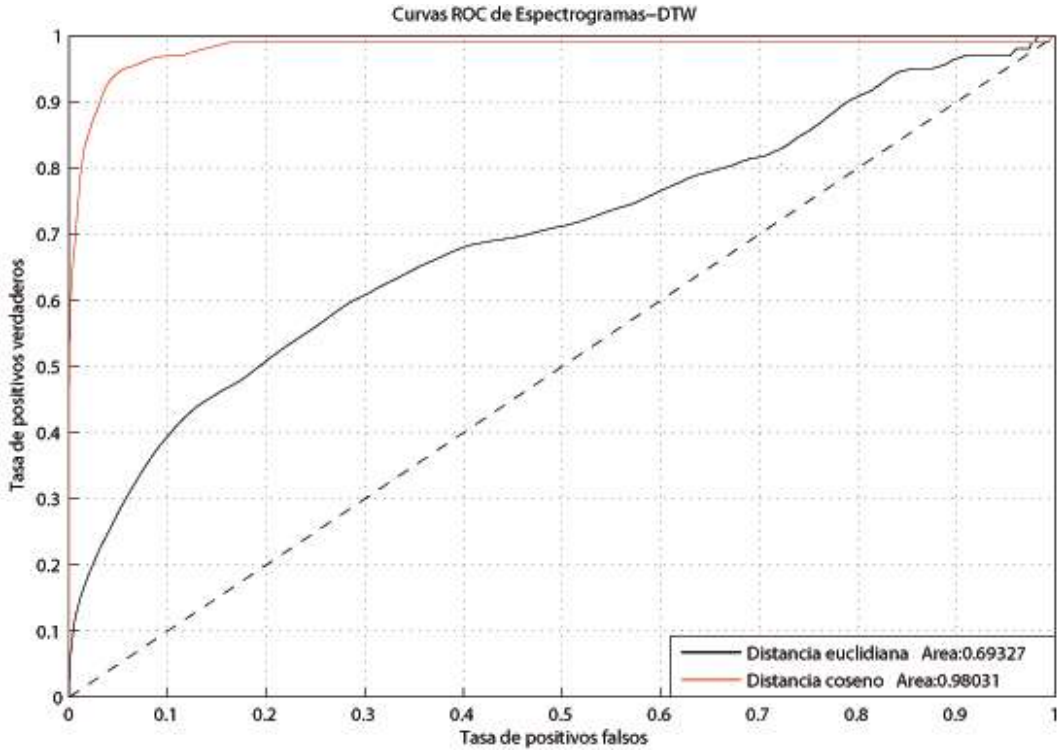

Figura 6.6: Curvas ROC utilizando Espectrogramas como método de extracción de características y DTW como técnica de reconocimiento

En la Figura 6.7 se observa el rendimiento de los sistemas que usan MFCC donde se observa que la curva más cercana con la esquina superior izquierda del gráfico es la que utiliza distancia euclidiana con una area bajo la curva de 0.96911. En la figura también se observa que la línea mucho más cercana con la esquina superior izquierda del gráfico es la que utiliza la distancia euclidiana comparada con la línea de distancia coseno.

MFCC funciona mucho mejor la distancia euclidiana. La tasa de reconocimiento usando MFCC es muy alta, en la Figura 6.7 se observa que para una tasa de positivos verdaderos del 50 % la tasa de positivos falsos es aproximadamente cero en el caso de distancia coseno mientras que para la distancia euclidiana una tasa de positivos verdaderos del 90 % la tasa de positivos falsos es aproximadamente cero.

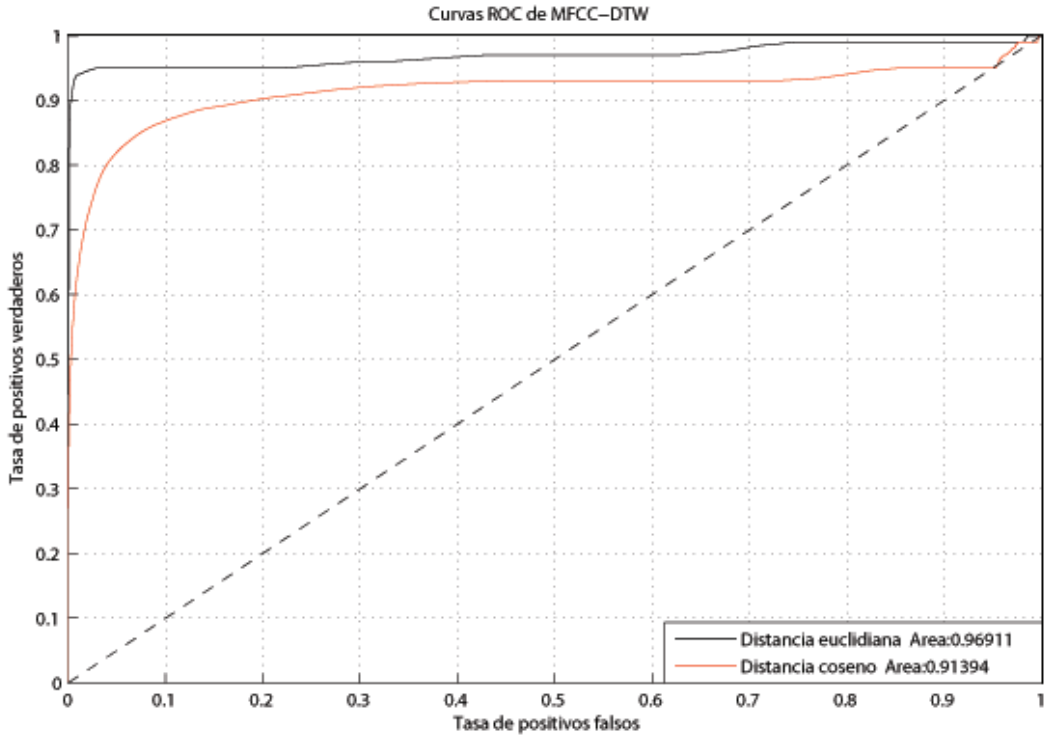

Figura 6.7: Curvas ROC utilizando MFCC como método de extracción de características y DTW como técnica de reconocimiento

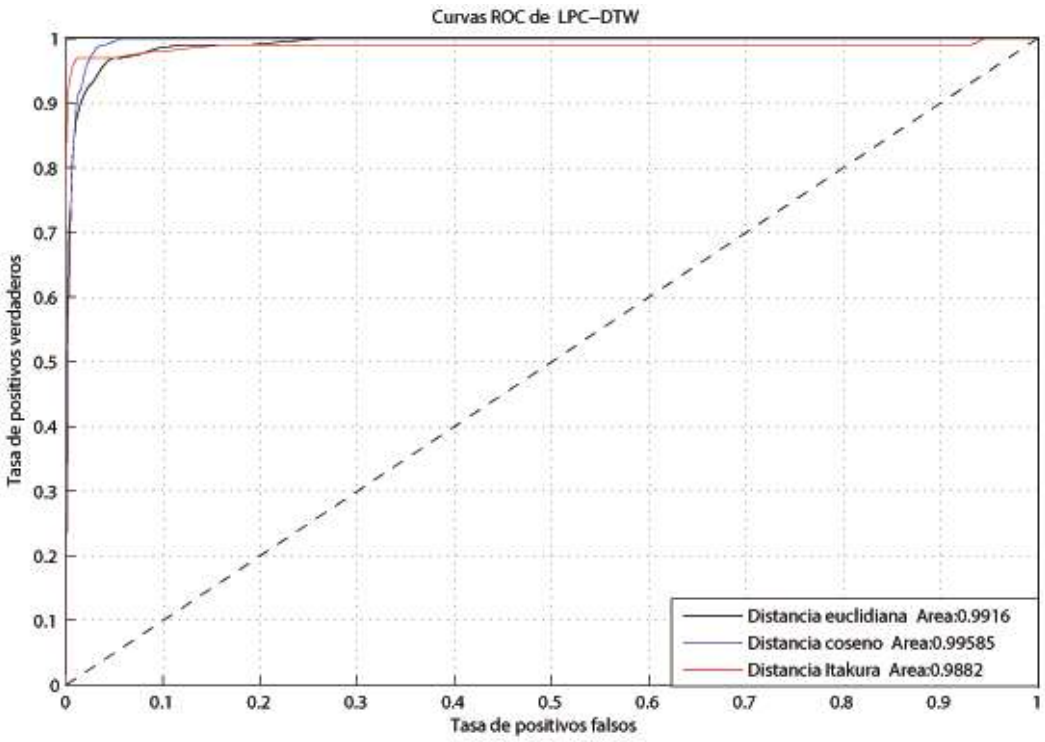

Figura 6.8: Curvas ROC utilizando LPC como método de extracción de características y DTW como técnica de reconocimiento

En el caso de LPC-DTW se hicieron pruebas usando la distancia coseno, euclidiana y la de Itakura, en la Figura 6.8 se observa el rendimiento de los sistemas que usan LPC; donde se observa que la curva más cercana con la esquina superior izquierda del gráfico es la que utiliza la distancia de Itakura con una area bajo la curva de 0.9882.

LPC funciona mucho mejor la distancia de Itakura. La tasa de reconocimiento usando LPC es muy alta, en la Figura 6.8 se observa que para una tasa de positivos verdaderos del 60 % la tasa de positivos falsos es aproximadamente cero en el caso de las distancias euclidiana y coseno mientras que para la distancia Itakura una tasa de positivos verdaderos del 92 % la tasa de positivos falsos es aproximadamente cero.

En general los sistemas que utilizan DTW como técnica de reconocimiento tienen tasas muy altas de reconocimiento con un costo muy bajo (tasa de positivos falsos).

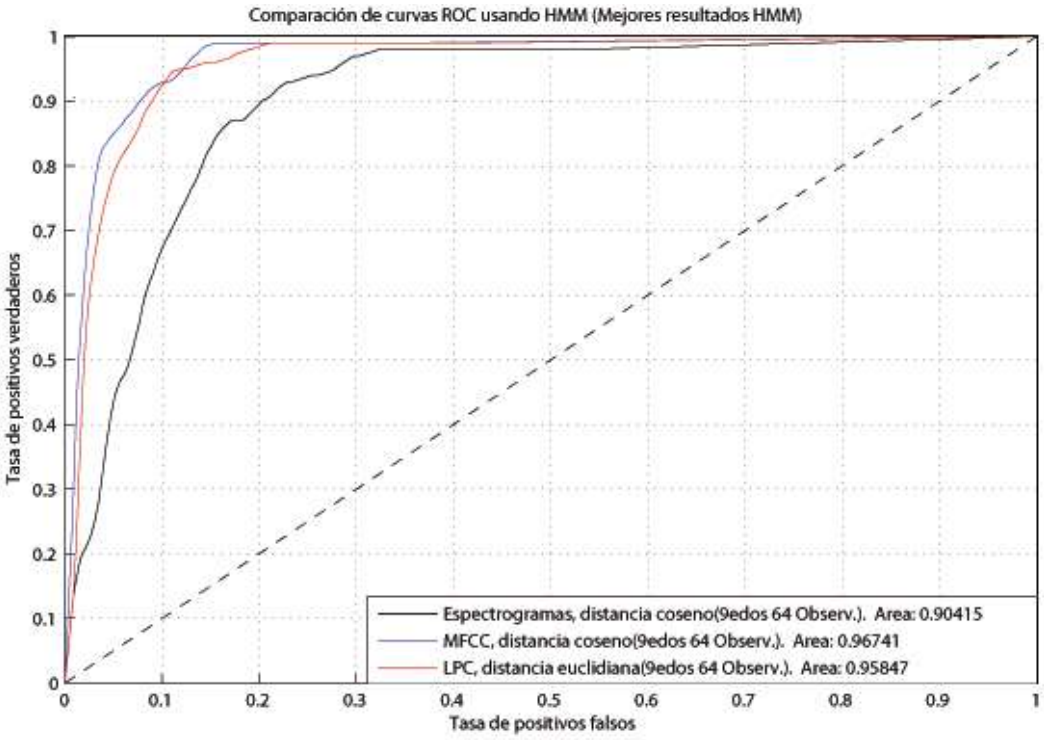

Figura 6.9: Curvas ROC de tres clasificadores DHMM, el mejor clasificador Espectrogramas-DHMM, MFCC-DHMM y LPC-DHMM

6.5. Comparación de resultados

Para obtener la mejor curva ROC para cada método de extracción de características se eligió la curva más cercana al punto $(0, 1)$ (la más cercana a la ezquina superior izquierda), es decir, la que tenga mayor tasa de reconocimiento con un menor costo.

En la Figura 6.9 se observa la mejor curva ROC usando: Espectrogramas-DHMM, MFCC-DHMM y LPC-DHMM. El peor método de extracción de características usando DHMM es Espectrogramas mientras que el mejor método es MFCC.

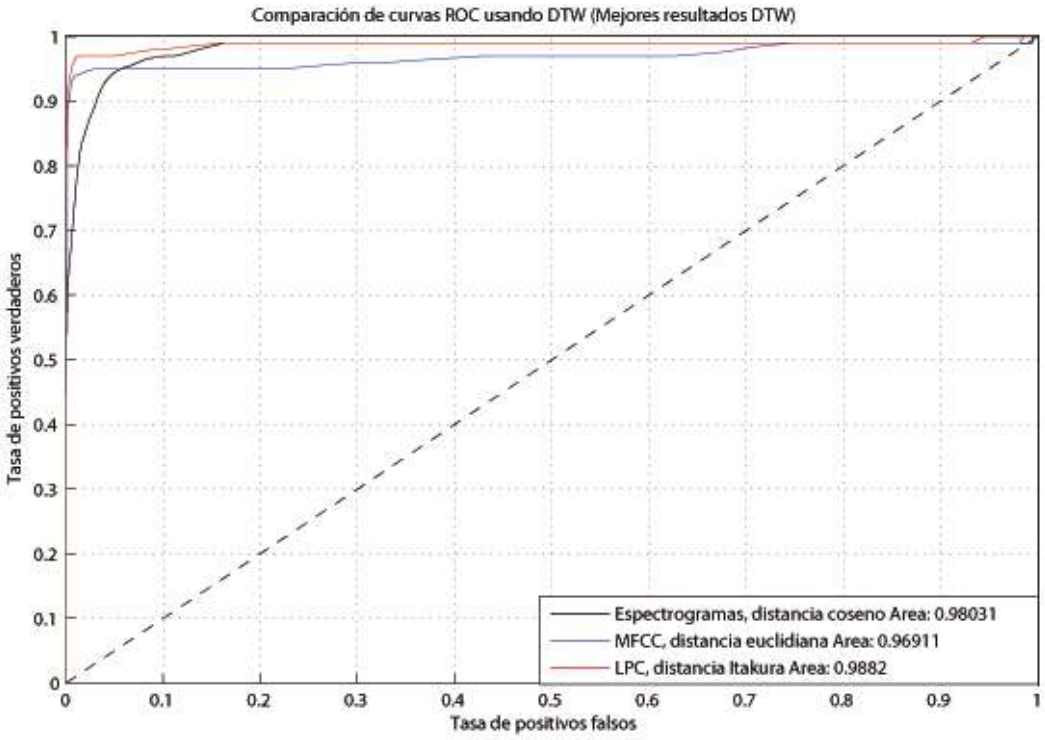

Figura 6.10: Curvas ROC de tres clasificadores DTW, el mejor clasificador Espectrogramas-DTW, MFCC-DTW y LPC-DTW

En la Figura 6.10 se observa la mejor curva ROC usando: Espectrogramas-DTW, MFCC-DTW y LPC-DTW. En la gráfica se observa que el peor método de extracción de características usando DTW es Espectrogramas mientras que el mejor método es LPC.

Uno de los objetivos de este trabajo es obtener cual es la mejor técnica de reconocimiento, por esta razon se comparó la mejor curva ROC DTW vs la mejor curva ROC DHMM. En la Figura 6.11 a simple vista se observa que la mejor curva ROC es la del sistema de reconocimiento que utiliza DTW, comparada con la del sistema que utiliza DHMM. Ambos sistemas tienen altas tasas de reconocimiento, pero DHMM tiene una tasa muy alta de positivos falsos.

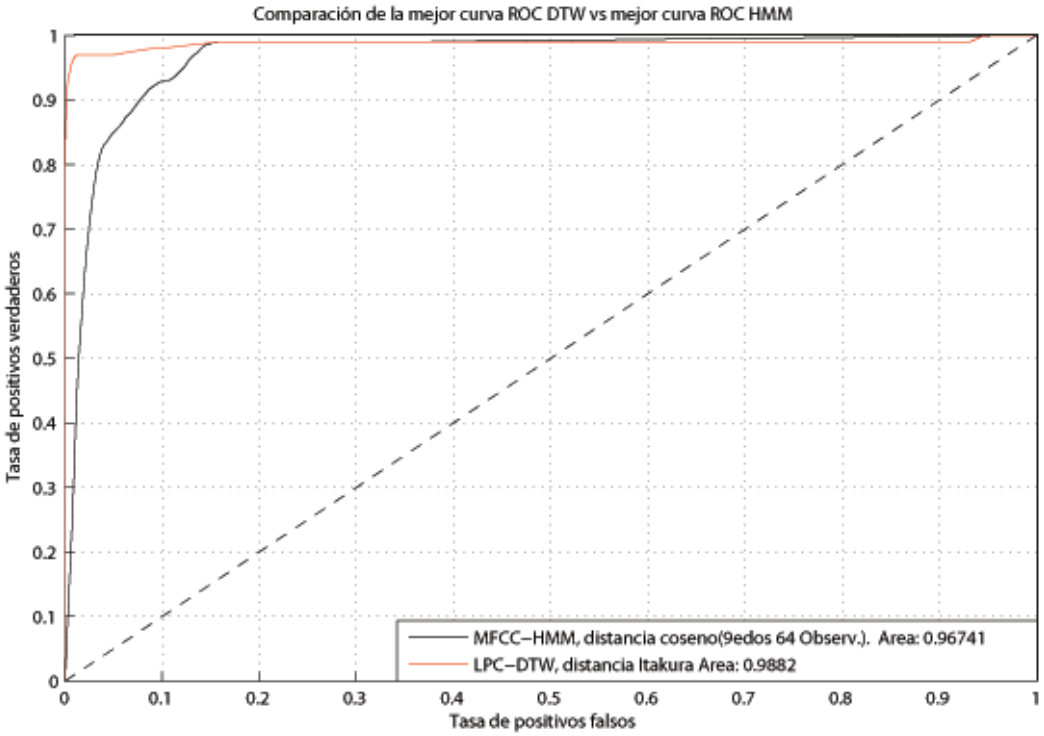

Figura 6.11: Curva ROC del mejor clasificador DTW comparado con la curva ROC del mejor clasificador DHMM

Capítulo 7

Conclusiones

El reconocimiento de voz constituye uno de los campos de la computación moderna y mantiene un desarrollo constante con el objetivo de encontrar nuevas y mejores soluciones al problema de la comunicación hombre-máquina.

7.1. Conclusiones Generales

1. Las dos técnicas más utilizadas en el reconocimiento de voz son las técnicas de comparación de patrones como es el método de DTW y el uso de procesos estocásticos como son los HMM.

2. En los procesos estocásticos se modela la evolución temporal de la secuencia de espectros obtenida de la señal de voz mediante un HMM con el fin de contemplar estocásticamente las diversas fuentes de variabilidad de la señal, mientras que las técnicas de comparación de patrones se basan simplemente en secuencias de espectros de los patrones de referencia y de la palabra a reconocer.

3. De los métodos de reconocimiento utilizados en este trabajo el que permite implementar los mejores sistemas de reconocimiento de palaras aisladas monolocutor es DTW.

4. A pesar de que los DHMM no superan a los DTW en la tasa de reconocimiento, los DHMM son muy utilizados principalmente en el reconocimiento de voz contínua ya

que el tiempo de cómputo de la etapa de comparación es mucho más rápido comparado con DTW ya que el dicccionario DHMM se hace offline que es la etapa que demanda más tiempo de cómputo.

5. Las curvas ROC de todos los clasificadores implementados en este proyecto demuestran que la mayoria de ellos funciona de manera aceptable excepto los sistemas que utilizan Espectrogramas-DTW distancia euclidiana o Espectrogramas-DHMM ya que sus tasas de reconocimiento son muy bajas. Sin embargo los sistemas que presentan mejor rendimiento están por encima del 90 % de eficiencia.

6. De los métodos de extracción de características usando DHMM, el que permite la construcción del mejor sistema de reconocimiento de palabras aisladas monolocutor es MFCC y el que permite la construcción del peor sistema es Espectrogramas.

7. De los métodos de extracción de características usando DTW, el que permite la construcción del mejor sistema de reconocimiento de palabras aisladas monolocutor es LPC y el que permite la construcción del peor sistema es Espectrogramas.

8. Al usar Espectrogramas como método de extracción de características es recomendable utilizar la distancia coseno, este método es fácil de implementar y se obtienen resultados aceptables, su desventaja es que su rendimiento es inferior comparado con MFCC y LPC, solo permite reconocimiento monolocutor.

9. Al usar MFCC como método de extracción de características es recomendable utilizar la distancia euclidiana, la principal ventaja de los MFCC es que permite reconocimiento multilocutor [Rabiner and Schafer, 1978], sus resultados en el reconocimiento de voz monolocutor es por encima del 90 % de eficiencia.

10. Al usar LPC como método de extracción de características es recomendable utilizar la distancia de Itakura, la principal ventaja de los LPC es que se son fáciles de implementar y la obtención de los coeficientes es rápida si se utiliza el algoritmo de Levinson-Durbin. Los resultados obtenidos con LPC están por encima del 92 % de eficiencia, solo permite reconocimiento monolocutor.

11. Los objetivos planteados para este proyecto se cumplieron de manera satisfactoria.

7.2. Trabajos Futuros

1. En este trabajo solo se realizaron codebooks utilizando el algoritmo de agrupamiento K-medias sin embargo existen varios algoritmos más eficientes que se pueden implementar, como son los algoritmos basado en teoría de grafos, por ejemplo el Grafo de Vencindad Relativa (RNG por sus siglas en inglés), el Grafo de Gabriel (GG por sus siglas en inglés), etc.

2. Implementar Modelos Ocultos de Markov Contínuos (CHMM) para el reconocimiento de voz ya que los resultados obtenidos usando DHMM presentan altas tasas de positivos falsos debido principalmente a la etapa de VQ.

3. Implementar el diccionario de palabras utilizando una base de datos (por ejemplo mysql) con el fin de que el vocabulario de palabras sea más amplio y poder implementar algún algoritmo de indexamiento a la base.

4. Actualmente la etapa de reconocimiento compara la palabra a reconocer con todas las palabras almacenadas en el diccionario, la desventaja de esto es que para vocabularios grandes se vuelve muy tardado, para ello se puede implementar algoritmos para indexamiento, por ejemplo el algoritmo Fixed Queries Array.

5. Utilizar las grabaciones de la base de datos de voz TIMIT para poder comparar los resultados obtenidos con diferentes grupos de trabajo del todo el mundo.

6. Implementar un sistema de reconocimiento de voz contínua usando HMM, para aplicaciones de Dictado automático, es decir, permitir a los usuarios dictar a la computadora y esta escriba lo que el usuario le dicta.

Apéndice A

Procesamiento de la señal de voz en el dominio del tiempo

Código fuente de algoritmos implementados en Java.

A.1. Energía de tiempo corto

Método que calcula la energía de tiempo corto de una señal de audio aplicando la ventana de Hamming. Recibe como parámetros:

sennal: la señal de audio $x(n)$.

En: el arreglo en el cual se almacenarán los valores de la raíz de la energía de tiempo corto.

N: tamaño de la ventana utilizada.

```
public static void EnergiaTiempoCorto(int sennal[], int En[], int N){
 int t = sennal.length;
 double wn[] = new double[N];
 ventanaHamming(wn, N);
 long aux = 0;
 for (int i = 0; i < t; i++) {
      En[i] = 0;
      aux = 0;
      for (int j = 0; j < N; j++) {
           if ((i + j) < t) {\\
           aux += (sennal[i + j] * sennal[i + j]) * wn[j];//ventana de Hamming
           }
       }
       En[i] = (int) Math.sqrt(aux);
  }
}
```

A.2. Ventana de Hamming

Método que calcula la ventana de Hamming. Recibe como parámetros:

wn: arreglo de doubles donde se guardan los coeficientes de la ventana de Hamming.

N: tamaño de la ventana.

```
public static void ventanaHamming(double wn[], int N) {
        for (int i = 0; i < N; i++) {
            res[i] = 0.54 - 0.46 * Math.cos(6.2832 * i / N);
        }
    }
```

A.3. Régimen de cruces por cero de tiempo corto

Método que calcula el régimen de cruces por cero de tiempo corto de una señal de audio aplicando la ventana de Rectangular. Recibe como parámetros:

sennal: la señal de audio $x(n)$.

Zn: el arreglo en el cual se almacenarán los valores de los cruces por cero.

N: tamaño de la ventana utilizada.

```
 public static void CrucesXcero(int sennal[], int Zn[], int N) {
     int t = sennal.length;
     for (int i = 0; i < t; i++) {
        Zn[i] = 0;
        for (int j = 0; j < N - 1; j++) {
            if ((i + j + 1) < t) {
                if (sennal[i + j] < 0 && sennal[i + j + 1] >= 0 || sennal[i + j] >= 0 && sennal[i + j + 1] < 0) {
                      Zn[i] += 1;
                }
            }
        }
     }
 }
```

A.4. Entropía de tiempo corto

Método que calcula la entropía de tiempo corto de una señal de audio aplicando la ventana de Rectangular. Recibe como parámetros:

sennal: la señal de audio $x(n)$.

H: el arreglo en el cual se almacenarán los valores de la raíz de la energía de tiempo corto.

N: tamaño de la ventana utilizada.

```
public static void Entropia(int sennal[], double H[], int N) {
        double cons = 0.5 * Math.log(6.2832), prom;
        for (int i = 0; i < sennal.length; i++) {
            prom = 0.0;
            double delta2 = 0.0;
            H[i] = 0.0;
            for (int j = 0; j < N; j++) {
                if ((i + j) < sennal.length) {
                    prom += sennal[i + j];
                }
            }
            prom /= N;
            for (int j = 0; j < N; j++) {
                if ((i + j) < sennal.length) {
                    delta2 += ((sennal[i + j] - prom) * (sennal[i + j] - prom));
                }
            }
            delta2 /= (N - 1);
            double n= 1.0;
            H[i] = n*cons + 0.5 * Math.log(delta2);
        }
}
```

A.5. Autocorrelación modificada

Método para calcular la autocorrelación modificada usando el algoritmo de Blankenship. Recibe como parámetros:

x: señal de voz $x(m)$.

N: tamaño de la señal - (k+1).

k: valor de autocorrelación a calcular, $\hat{R}_n(k)$.

```
public static double blankenship(double x[], int N, int k) {
        int t, ti;
        double b_k = 0, c_k = 0, s = 0;
        int i, j;
        determina_q_a_b(k, N);
        //System.out.println((2*q*k+a*k+b)+"  q=  "+q+" b=  "+b+" a=  "+a);

        for (j = 0; j < q; j++) {
            s = 0;
            t = 2 * j * k;
            for (i = 1; i <= k; i++) {
                ti = t + i;
                s += x[ti + k] * (x[ti] + x[ti + 2 * k]);
            }
            b_k += s;
        }
        t = 2 * q * k;
        if (a == 0) {
            for (i = 1; i <= b; i++) {
                ti = t + i;
                c_k += x[ti] * x[ti + k];
            }
```

```
        } else {
            for (i = 1; i <= b; i++) {
                ti = t + i;
                c_k += x[ti + k] * (x[ti] + x[ti + 2 * k]);
            }
            for (i = b + 1; i <= k; i++) {
                ti = t + i;
                c_k += x[ti] * x[ti + k];
            }
        }
        return b_k + c_k;
}

static int q, a, b;
```

/**

* Función para determinar los valores de q, a y b.

* k .- Valor de la autocorrelacion a calcular.

* N .- Tamaño de la señal - (k+1).

**/

```
public static void determina_q_a_b(int k, int N) {
        q = 0;
        if (k != 0) {
            q = N / (2 * k);
        } else {
            //System.out.println("determina_q_a_b: k VALE CERO!!!\n");
        }
        if ((2 * q + 1) * k <= N && k != 0) {
            a = 1;
        } else {
            a = 0;
        }
        b = N - (2 * q * k) - a * k;
}
```

Apéndice B

Extracción de características

Código fuente de algoritmos implementados en Java.

B.1. Espectrogramas

Método que obtiene el espectrograma de una palabra. Recibe de parámetros:

x: Señal de voz $x(n)$.

Espectrograma: Matriz donde se almacenará el espectrograma (18 bandas, 8000Hz).

N: Tamaño de la ventana.

delta: Avance de la ventana.

```
public static void Espectograma(int x[], double Espectrograma[][], int N, int delta) {
        double wn[] = new double[N];
        ventanaHamming(wn, tamV);
        int con = 0;
        double potencia[] = new double[N];
        double stfaR[] = new double[N];
        double stfaI[] = new double[N];
        for (int i = 0; i < x.length; i += delta) {
            for (int j = i, k = 0; j < (i + N); j++, k++) {
                if (j < x.length) {
                    stfaR[k] = x[j] * wn[k];
                    stfaI[k] = 0.0;
                } else {
                    stfaR[k] = 0.0;
                    stfaI[k] = 0.0;
                }
            }
            funciones.fft1(stfaR, stfaI, N, -1);
            for (int j = 0; j < N; j++) {
                potencia[j] = stfaR[j] * stfaR[j] + stfaI[j] * stfaI[j];
            }
```

```
            for (int j = 0; j < (N / 2); j++) {
                int f = j * (8000) / (N);
                Espectrograma[con][bandaBark(f)] = Espectrograma[con][bandaBark(f)] + potencia[j];
            }
            con++;
        }
}
```

/**

* Función para determinar a que banda pertenece la frecuencia f.

* f .- Frecuencia en Hertz.

**/

```
public static int bandaBark(int f) {
        if (f < 100) {
            return 0;
        } else if (f < 200) {
            return 1;
        } else if (f < 300) {
            return 2;
        } else if (f < 400) {
            return 3;
        } else if (f < 510) {
            return 4;
        } else if (f < 630) {
            return 5;
        } else if (f < 770) {
            return 6;
        } else if (f < 920) {
            return 7;
        } else if (f < 1080) {
            return 8;
        } else if (f < 1270) {
            return 9;
        } else if (f < 1480) {
            return 10;
        } else if (f < 1720) {
            return 11;
        } else if (f < 2000) {
            return 12;
        } else if (f < 2320) {
            return 13;
        } else if (f < 2700) {
            return 14;
        } else if (f < 3150) {
            return 15;
        } else if (f < 3700) {
            return 16;
        } else {
            return 17;
        }
}
```

/**

* Función para determinar la DFT.

* real[] .- Parte real de la DFT.

* imaginaria[] .- Parte imaginaria de la DFT.

* npts .- Tamaño de señal.

* Direccion.- (-1) DFT directa, (1) DFT inversa.

**/

```
public static void fft1(double real[], double imaginaria[], int npts, int Direccion) {
        double fh[] = new double[2 * (npts + 1)];
        double fact = 1.0;
        int j;
        if (Direccion == -1) {
            fact = 1.0 / (double) (npts);
        }

        for (j = 0; j < 2 * npts; j += 2) {
            fh[j + 1] = real[j / 2];
            fh[j + 2] = imaginaria[j / 2];
        }

        four1(fh, npts, Direccion);

        for (j = 0; j < 2 * npts; j += 2) {
            real[j / 2] = fact * fh[j + 1];
            imaginaria[j / 2] = fact * fh[j + 2];
        }
}
```

/**

* Función para determinar la DFT en un solo arreglo.

* data[] .- Parte real de la DFT (impares), parte imaginaria (pares).

* npts .- Tamaño de señal.

* isign.- (-1) DFT directa, (1) DFT inversa.

**/

```
private static void four1(double data[], int nn, int isign) {
        //
        // TRANSFORMADA DE FOURIER
        //
        int n, mmax, m, j, istep, i;
        double wtemp, wr, wpr, wpi, wi, theta;
        double tempr, tempi, tt;

        n = nn << 1;
        j = 1;
        for (i = 1; i < n; i += 2) {
            if (j > i) {
                tt = data[j];
                data[j] = data[i];
                data[i] = tt;
                tt = data[j + 1];
                data[j + 1] = data[i + 1];
                data[i + 1] = tt;
            }
            m = n >> 1;
```

```
            while (m >= 2 && j > m) {
                j -= m;
                m >>= 1;
            }
            j += m;
        }
        mmax = 2;
        while (n > mmax) {
            istep = 2 * mmax;
            theta = 6.28318530717959 / (isign * mmax);
            wtemp = Math.sin(0.5 * theta);
            wpr = -2.0 * wtemp * wtemp;
            wpi = Math.sin(theta);
            wr = 1.0;
            wi = 0.0;
            for (m = 1; m < mmax; m += 2) {
                for (i = m; i <= n; i += istep) {
                    j = i + mmax;
                    tempr = wr * data[j] - wi * data[j + 1];
                    tempi = wr * data[j + 1] + wi * data[j];
                    data[j] = data[i] - tempr;
                    data[j + 1] = data[i + 1] - tempi;
                    data[i] += tempr;
                    data[i + 1] += tempi;
                }
                wr = (wtemp = wr) * wpr - wi * wpi + wr;
                wi = wi * wpr + wtemp * wpi + wi;
            }
            mmax = istep;
        }
}
```

B.2. MFCC

Método que obtiene la secuencia de vectores característicos MFCC de una palabra.

Recibe de parámetros:

x: Señal de voz $x(n)$.

MFCC: Matriz donde se almacenará los coefientes cepstrales de Mel (16 filtros, $f_s = 8000Hz$).

N: Tamaño de la ventana.

delta: Avance de la ventana.

```
static void MFCC(int[] x, double[][] MFCC, int N, int delta) {
        double wn[] = new double[N];
        double Xprima[] = new double[MFCC[0].length];
        ventanaHamming(wn, N);
        int con = 0;
        double potencia[] = new double[N];
        double stfaR[] = new double[N];
        double stfaI[] = new double[N];
```

```
        for (int i = 0; i < x.length; i += delta) {
            for (int j = i, k = 0; j < (i + N); j++, k++) {
                if (j < x.length) {
                    stfaR[k] = x[j] * wn[k];
                    stfaI[k] = 0.0;
                } else {
                    stfaR[k] = 0.0;
                    stfaI[k] = 0.0;
                }
            }
            funciones.fft1(stfaR, stfaI, N, -1);
            for (int j = 0; j < N; j++) {
                potencia[j] = stfaR[j] * stfaR[j] + stfaI[j] * stfaI[j];
            }
            double fMELmax = 2595 * Math.log10((4000 / 700) + 1);
            double fMELmin = 0.0;
            double incrMEL = (fMELmax - fMELmin) / (16.0);//15 M
            int posi[] = new int[16];
            for (int j = 0; j < 16; j++) {
                double fMEL = fMELmin + j * incrMEL;
                posi[j] = (int) ((((Math.pow(10.0, fMEL / 2595) - 1) * 700.0) * N) / 8000.0);
            }
            for (int j = 0; j < 15; j++) {
                int tam = posi[j + 1] - posi[j];
                double res[] = new double[tam];
                ventanaTriangulo(res, tam);
                double suma = 0.0;
                for (int k = posi[j], l = 0; k < posi[j + 1]; k++, l++) {
                    suma += (potencia[k] * res[l]);
                }
                Xprima[j] = Math.log(suma);
            }
            for (int j = 0; j < 15; j++) {
                double suma = 0;
                for (int k = 0; k < 15; k++) {
                    double arg = ((j * 3.1416) / 15.0) * (1.0 * k - 0.5);
                    suma += (Xprima[k] * Math.cos(arg));
                }
                MFCC[con][j] = suma;
            }
            con++;
        }
}
```

/**

* Función para determinar un filtro triangular de Mel.

* wn[] .- Filtro triangular.

* tamV .- Ancho del filtro.

**/

```
private static void ventanaTriangulo(double wn[], int tamV) {
        double N = 1.0 * tamV;
        for (int i = 0; i < tamV; i++) {
            if (i <= tamV / 2.0) {
                ven[i] = (2.0 / tamV) * i;
            } else {
                ven[i] = 1 + (-2.0 / tamV) * (i - tamV / 2.0);
```

```
            }
        }
}
```

B.3. LPC

Método que obtiene la secuencia de vectores característicos LPC de una palabra.

Recibe de parámetros:

x: Señal de voz.

LPC: Matriz donde se almacenará los coefientes LPC (16 coeficientes).

N: Tamaño de la ventana.

delta: Avance de la ventana.

```
static void LPC(int[] x, double[][] LPC, int N, int delta) {
        int p = 16;
        double R[] = new double[p + 1];
        double Frame[] = new double[N + 1];
        double FramePre[] = new double[N + 1];
        double E[] = new double[p];
        double alfa[] = new double[p + 1];
        double alfaN[] = new double[p + 1];
        double ki = 0;
        int con = 0;
        for (int i = 0; i < x.length; i += delta) {
            for (int j = i, k = 1; j < (i + N); j++, k++) {
                if (j < x.length) {
                    Frame[k] = x[j];
                }
            }
            preenfasis(Frame, 0.98, FramePre);
            for (int j = 0; j <= p; j++) {
                R[j] = blankenship(FramePre, N - (j + 1), j);
            }
            E[0] = R[0];
            for (int j = 1; j <= p; j++) {
                double sum = 0.0;
                for (int k = 1; k <= (j - 1); k++) {
                    sum = sum + (alfa[k] * R[(j - k)]);
                }
                ki = (R[j] - sum) / E[j - 1];
                alfaN[j] = ki;
                for (int k = 1; k <= (j - 1); k++) {
                    alfaN[k] = alfa[k] - ki * alfa[j - k];
                }
                if (j < p) {
                    E[j] = (1 - (ki * ki)) * E[j - 1];
                }
                for (int k = 1; k <= j; k++) {
                    alfa[k] = alfaN[k];
                }
            }
            for (int j = 1; j <= p; j++) {
```

```
                LPC[con][j - 1] = alfa[j];
            }
            con++;
        }
}
```

```
/**
* Función para aplicar pre-énfasis a la señal de audio.
* Frame[] .- Señal de audio.
* FramePre[] .- Señal de audio pre-énfatizada.
* omg .- Nivel de pre-énfatización.
**/

static void preenfasis(double[] Frame, double omg, double[] FramePre) {
        for (int i = 1; i < Frame.length; i++) {
            FramePre[i] = Frame[i] - omg * Frame[i];
        }
}
```

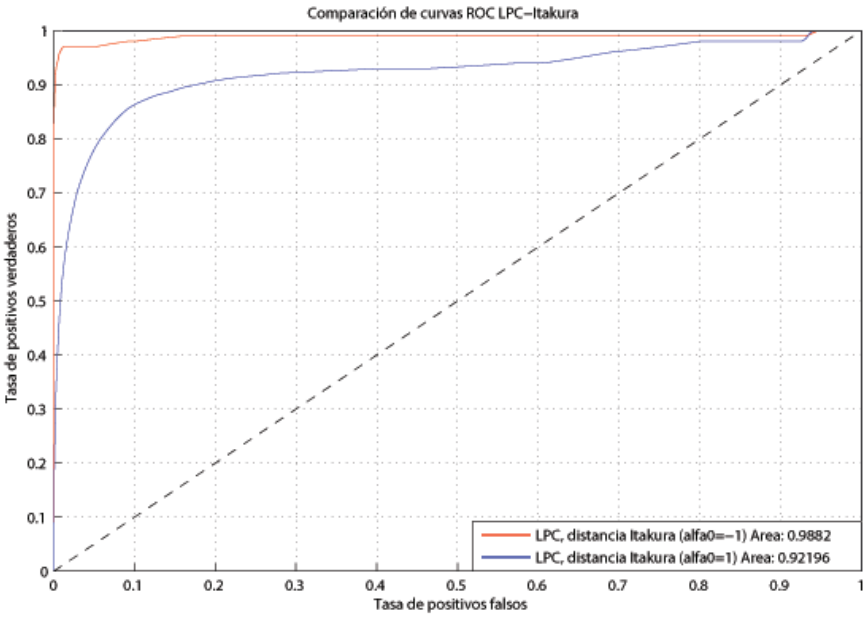

Figura B.1: Posibles valores de α_0 para la distancia de Itakura

B.3.1. Distancia de Itakura para LPC

Algunos autores definen $\alpha_0 = 1$ [Rabiner and Schafer, 1978, pág. 498-500] y otros definen $\alpha_0 = -1$ [Iosu, 1999]. En la Figura B.1 se muestran las curvas ROC de un sistema usando la distancia de Itakura con $\alpha_0 = 1$ y $\alpha_0 = -1$, donde se aprecia que funciona mejor la distancia de Itakura si se define $\alpha_0 = -1$.

Apéndice C

Técnicas de reconocimiento de voz

Código fuente de algoritmos implementados en Java.

C.1. DTW

C.1.1. DTW distancia coseno y euclidiana

Método que implementa el algoritmo DTW. Recibe como parámetros:

resul: Matriz de la palabra a reconocer.

res2: Matriz de la palabra en el diccionario.

Regresa la distancia acumulada entre ambas matrices, la bandera *metodo* = 1 distancia euclidiana de lo contrario distancia coseno.

```
static metodo=1;
public static double Dynamic_Time_Warping(double[][] resul, double[][] res2) {
        int k = resul.length, l = res2.length;
        double matriz[][] = new double[k][l];
        for (int i = 0; i < k; i++) {
            if (i == 0) {
                matriz[i][0] = distancia(resul[i], res2[0]);
            } else {
                matriz[i][0] = matriz[i - 1][0] + distancia(resul[i], res2[0]);
            }
        }
        for (int i = 0; i < l; i++) {
            if (i == 0) {
                matriz[0][i] = distancia(resul[0], res2[i]);
            } else {
                matriz[0][i] = matriz[0][i - 1] + distancia(resul[0], res2[i]);
            }
        }
        for (int i = 1; i < k; i++) {
            for (int j = 1; j < l; j++) {
```

```
                matriz[i][j] = Distancia_Optima(resul[i], res2[j], matriz[i - 1][j - 1],
                matriz[i - 1][j], matriz[i][j - 1]);
            }
        }
        double res = matriz[k - 1][l - 1] / (l + k);
        return res;
}

static double Distancia_Optima(double a[], double b[], double Daa,
double Dab, double Dba) {
        double val1, val2, val3, minima = 0;
        double distancia = distancia(a, b);
        val1 = Daa + 2.0 * distancia;
        val2 = Dab + distancia;
        val3 = Dba + distancia;
        minima = val1;
        if (val2 < minima) {
            minima = val2;
        }
        if (val3 < minima) {
            minima = val3;
        }

        return minima;
}

public static double distancia(double v1[], double v2[]) {
        if (metodo == 1) {
            double d = 0.0;
            for (int i = 0; i < v1.length; i++) {
                d += (v1[i] - v2[i]) * (v1[i] - v2[i]);
            }
            return Math.sqrt(d);
        } else {
            double mult = 0.0, sumaA = 0, sumaB = 0, divisor = 0;
            for (int i = 0; i < v1.length; i++) {
                mult += (v1[i] * v2[i]);
                sumaA += (v1[i] * v1[i]);
                sumaB += (v2[i] * v2[i]);
            }
            divisor = sumaA * sumaB;
            divisor = Math.sqrt(divisor);
            if (divisor == 0.0) {
                return 0;
            } else {
                return (1.0 - Math.abs(mult / divisor));
            }
        }
}
```

C.1.2. DTW distancia de Itakura

Método que implementa el algoritmo DTW para la distancia de Itakura. Recibe como parámetros:

resul: Matriz de la palabra a reconocer.

res: Matriz de la palabra en el diccionario.

R: Arreglo de matrices de autocorrelación.

Regresa la distancia acumulada entre ambas matrices, distancia de Itakura.

```
static double Dynamic_Time_Warping_Itakura(double[][] resul,
double[][] res2, double R[][][]) {
        int k = resul.length, l = res2.length;
        double matriz[][] = new double[k][l];
        for (int i = 0; i < k; i++) {
            if (i == 0) {
                matriz[i][0] = distanciaItakura(resul[i], res2[0], R[i]);
            } else {
                matriz[i][0] = matriz[i - 1][0] + distanciaItakura(resul[i], res2[0], R[i]);
            }
        }
        for (int i = 0; i < l; i++) {
            if (i == 0) {
                matriz[0][i] = distanciaItakura(resul[0], res2[i], R[0]);
            } else {
                matriz[0][i] = matriz[0][i - 1] + distanciaItakura(resul[0], res2[i], R[0]);
            }
        }
        for (int i = 1; i < k; i++) {
            for (int j = 1; j < l; j++) {
                matriz[i][j] = Distancia_Optima_Itakura(resul[i], res2[j],
                matriz[i - 1][j - 1], matriz[i - 1][j], matriz[i][j - 1], R[i]);
            }
        }
        double res = matriz[k - 1][l - 1] / (l + k);
        return res;
}

private static double distanciaItakura(double[] a, double[] b,
double R[][]) {
        double ag[][] = new double[1][a.length + 1];
        double bg[][] = new double[1][b.length + 1];
        for (int i = 0; i < a.length + 1; i++) {
            if (i == 0) {
                ag[0][i] = -1;
                bg[0][i] = -1;
            } else {
                ag[0][i] = a[i - 1];
                bg[0][i] = b[i - 1];
            }
        }
        double agT[][] = transpuesta(ag);
        double bgT[][] = transpuesta(bg);
        double numerador[][] = multiplica(ag, R);
        numerador = multiplica(numerador, agT);
        double denominador[][] = multiplica(bg, R);
        denominador = multiplica(denominador, bgT);
        double dis = Math.abs(Math.log10(numerador[0][0] / denominador[0][0]));
        return dis;
}

static double Distancia_Optima_Itakura(double a[], double b[],
double Daa, double Dab, double Dba, double R[][]) {
        double val1, val2, val3, minima = 0;
        double distancia = distanciaItakura(a, b, R);
        val1 = Daa + 2.0 * distancia;
        val2 = Dab + distancia;
        val3 = Dba + distancia;
        minima = val1;
```

```
        if (val2 < minima) {
            minima = val2;
        }
        if (val3 < minima) {
            minima = val3;
        }

        return minima;
}

static public double[][] transpuesta(double origen[][]) {
        int nr = origen.length, nc = origen[0].length, i, j;

        double res[][] = new double[nc][nr];

        for (i = 0; i < nr; i++) {
            for (j = 0; j < nc; j++) {
                res[j][i] = origen[i][j];
            }
        }

        return res;

}

public static double[][] multiplica(double A[][], double B[][]) {

        int nr = A.length, nc = B[0].length, m = B.length;
        int i, j, k;

        double res[][] = new double[nr][nc];

        for (i = 0; i < nr; i++) {
            for (j = 0; j < nc; j++) {
                res[i][j] = 0;

                for (k = 0; k < m; k++) {
                    res[i][j] += A[i][k] * B[k][j];
                }
            }
        }
        return res;
}
```

C.2. DHMM

C.2.1. Algoritmo Forward escalado

Método que implementa el algoritmo Forward escalado. Recibe como parámetros:

A: Matriz A del DHMM.

B: Matriz B del DHMM.

PI: vector PI del DHMM.

O: secuencia de observaciones.

Regresa un arreglos de dos objetos, la matriz de evaluación y el vector c_t de escalamiento.

```
static public Object[] Algoritmo_Forward_Escalado(double A[][], double B[][],
double PI[], int O[]) {
        Object res[] = new Object[2];
        int T = O.length;
        int N = A.length;
        double alfa_gorro[][] = new double[N][T];
        double alfa_barra[][] = new double[N][T];
        double c_t[] = new double[T];
        double suma = 0;
        for (int i = 0; i < N; i++) {
            alfa_barra[i][0] = PI[i] * B[i][O[0]];
            suma += alfa_barra[i][0];
        }
        if (suma == 0.0) {
            suma = 1.0;
        }
        c_t[0] = 1.0 / suma;
        for (int i = 0; i < N; i++) {
            alfa_gorro[i][0] = c_t[0] * alfa_barra[i][0];
        }
        for (int t = 1; t < T; t++) {
            for (int j = 0; j < N; j++) {
                suma = 0.0;
                for (int i = 0; i < N; i++) {
                    suma += alfa_gorro[i][t - 1] * A[i][j] * B[j][O[t]];
                }
                alfa_barra[j][t] = suma;
            }
            suma = 0.0;
            for (int i = 0; i < N; i++) {
                suma += alfa_barra[i][t];
            }
            if (suma == 0.0) {
                suma = 1.0;
            }
            c_t[t] = 1.0 / suma;
            for (int j = 0; j < N; j++) {
                alfa_gorro[j][t] = c_t[t] * alfa_barra[j][t];
            }
        }
        res[0] = alfa_gorro;
        res[1] = c_t;
        return res;
}

public static double terminacionForward(double[][] alfa) {
        double sum = 0.0;
        int pos = alfa[0].length - 1;
        for (int j = 0; j < alfa.length; j++) {
            sum += alfa[j][pos];
        }
        return sum;
}
```

C.2.2. Algoritmo Backward Escalado

Método que implementa el algoritmo de Backward escalado. Recibe como parámetros: A: Matriz A del DHMM.

B: Matriz B del DHMM.

O: secuencia de observaciones.

c_t: vector de escalamiento.

Regresa la matriz de evaluación.

```
static public double[][] Algoritmo_Backward_Escalado(double A[][]
,double B[][], int O[], double c_t[]) {
  int T = O.length;
  int N = A.length;
  double beta_barra[][] = new double[N][T];
  double beta_gorro[][] = new double[N][T];
  for (int i = 0; i < N; i++) {
     beta_barra[i][T - 1] = 1.0;
     beta_gorro[i][T - 1] = beta_barra[i][T - 1] * c_t[T - 1];
  }
  for (int t = T - 2; t >= 0; t--) {
    for (int i = 0; i < N; i++) {
       double sum = 0.0;
       for (int j = 0; j < N; j++) {
         sum += (A[i][j] * B[j][O[t + 1]] * beta_gorro[j][t + 1]);
       }
       beta_barra[i][t] = sum;
     }
     for (int i = 0; i < N; i++) {
       beta_gorro[i][t] = c_t[t] * beta_barra[i][t];
     }
   }
   return beta_gorro;
}

public static double terminacionBackward(double[][] beta,
double B[][], double PI[], int O[]) {
        double sum = 0.0;
        for (int i = 0; i < PI.length; i++) {
            sum += PI[i] * B[i][O[0]] * beta[i][0];
        }
        return sum;
}
```

C.2.3. Algoritmo de Viterbi Escalado

Método que implementa el algoritmo de Viterbi escalado. Recibe como parámetros:

A: Matriz A del DHMM.

B: Matriz B del DHMM.

PI: vector PI del DHMM.

O: secuencia de observaciones.

Regresa el vector de evaluación.

```
public static double[] Viterbi_Esc(double A[][], double B[][], double PI[], int O[]) {
        int T = O.length;
        int N = A.length;
```

```
        double delta[][] = new double[N][T];
        int Psi[][] = new int[N][T];
        double qT_ast[] = new double[T + 1];

        for (int i = 0; i < N; i++) {
            delta[i][0] = Math.log10(PI[i]) + Math.log10(B[i][O[0]]);
            Psi[i][0] = 0;
        }

        for (int t = 1; t < T; t++) {
            for (int j = 0; j < N; j++) {
                double max = delta[0][t - 1] + Math.log10(A[0][j]);
                int argmax = 0;
                for (int i = 0; i < N; i++) {
                    if ((delta[i][t - 1] + Math.log10(A[i][j])) > max) {
                        max = delta[i][t - 1] + Math.log10(A[i][j]);
                        argmax = i;
                    }
                }
                delta[j][t] = max + Math.log10(B[j][O[t]]);
                Psi[j][t] = argmax;
            }
        }

        double max = delta[0][T - 1];
        int argmax = 0;
        for (int i = 0; i < N; i++) {
            if (delta[i][T - 1] > max) {
                max = delta[i][T - 1];
                argmax = i;
            }
        }
        qT_ast[T] = max;
        qT_ast[T - 1] = argmax;
        for (int t = T - 2; t >= 0; t--) {
            qT_ast[t] = Psi[(int) (qT_ast[t + 1])][t + 1];
        }
        return qT_ast;
}
```

C.2.4. Algoritmo Baum-Welch múltiples secuencias de observaciones

Método que implementa el algoritmo de Baum-Welch múltiples secuencias de observaciones. Recibe como parámetros:

A: Matriz A del DHMM.

B: Matriz B del DHMM.

PI: vector PI del DHMM.

O: secuencias de observaciones.

```
public static void Bamwech_E_M_Multiple_Escalado(double[][] A,
double[][] B, double[] PI, int[][] O) {
        int I = A.length;
        int numObservaciones = O.length;
        int K = B[0].length;
        double error = 1e-10;
```

```
inicializarDiag(A, 2);
PI[0] = 1;
for (int i = 1; i < I; i++) {
    PI[i] = 0;
}
int cambios = 10;
while (cambios != 0) {
    cambios = 0;
    Object resAlfaG[];
    double betaG[][][] = new double[numObservaciones][][];
    double alfaG[][][] = new double[numObservaciones][][];
    double c_t[][] = new double[numObservaciones][];
    for (int l = 0; l < numObservaciones; l++) {
        resAlfaG = Algoritmo_Forward_Escalado(A, B, PI, O[l]);
        alfaG[l] = (double[][]) resAlfaG[0];
        c_t[l] = (double[]) resAlfaG[1];
        betaG[l] = Algoritmo_Backward_Escalado(A, B, O[l], c_t[l]);
    }
    for (int i = 0; i < I; i++) {
        for (int j = 0; j < I; j++) {
            double Sprod = 0;
            double suma = 0;
            for (int k = 0; k < numObservaciones; k++) {
                for (int t = 0; t < (O[k].length - 1); t++) {
                    Sprod += alfaG[k][i][t] * A[i][j] * B[j][O[k][t + 1]] * betaG[k][j][t + 1];
                    suma += alfaG[k][i][t] * betaG[k][i][t] * (1.0 / c_t[k][t]);
                }
            }
            double res;
            if (suma == 0) {
                res = 0.0;
            } else {
                res = Sprod / suma;
            }
            if (Math.abs(res - A[i][j]) > error) {
                cambios++;
            }
            A[i][j] = res;
        }
    }
    for (int i = 0; i < I; i++) {
        for (int l = 0; l < K; l++) {
            double suma = 0.0;
            double sumT = 0.0;
            for (int k = 0; k < numObservaciones; k++) {
                for (int t = 0; t < (O[k].length - 1); t++) {
                    double prod = alfaG[k][i][t] * betaG[k][i][t] * (1.0 / c_t[k][t]);
                    if (O[k][t] == l) {
                        suma += prod;
                    }
                    sumT += prod;
                }
            }
            double res;
            if (sumT == 0.0) {
                res = 0.0;
            } else {
                res = suma / sumT;
            }
            if (Math.abs(res - B[i][l]) > error) {
                cambios++;
            }
            B[i][l] = res;
```

```
                }
            }
        }
}

public static void inicializarDiag(double[][] A, int delta) {
        int I = A.length;
        int J = A[0].length;
        for (int i = 0; i < I; i++) {
            for (int j = i; j < (i + delta); j++) {
                if (j < J) {
                    A[i][j] = Math.random();
                }
            }
            double suma = 0.0;
            for (int j = 0; j < J; j++) {
                suma += A[i][j];
            }
            for (int j = 0; j < J; j++) {
                A[i][j] = A[i][j] / suma;
            }
        }
}

public static void inicializar(double[][] A) {
        int I = A.length;
        int J = A[0].length;
        for (int i = 0; i < I; i++) {
            for (int j = 0; j < J; j++) {
                A[i][j] = Math.random();
            }
            //sumo sus valores
            double suma = 0.0;
            for (int j = 0; j < J; j++) {
                suma += A[i][j];
            }
            //Dandole valores que sumen 1.
            for (int j = 0; j < J; j++) {
                A[i][j] = A[i][j] / suma;
            }
        }
}
```

C.2.5. Algoritmo floor smoothing

Método que implementa el algoritmo floor smoothing. Recibe como parámetros:

B: Matriz del DHMM.

UMBRAL: η umbral de floor smoothing.

```
public static void floor_smoothing(double[][] B, double UMBRAL) {
        for (int i = 0; i < B.length; i++) {
            for (int j = 0; j < B[0].length; j++) {
                if (B[i][j] < UMBRAL) {
                    B[i][j] = UMBRAL;
                }
            }
            double suma = 0.0;
            for (int j = 0; j < B[i].length; j++) {
```

```
                suma += B[i][j];
            }
            for (int j = 0; j < B[i].length; j++) {
                B[i][j] = B[i][j] / suma;
            }
        }
}
```

Referencias

[Álvarez, 2012] Álvarez, A. (2012). Notas: Historia de los sistemas de reconocimiento automático del habla. URL: http://tamarisco.datsi.fi.upm.es/ASIGNATURAS/FRAV/apuntes/historia_rv.pdf. Consulta: Abril 2013.

[Barton et al., 1999] Barton, C., Felipe, I., Dhiraj, M., and Avery, W. (1999). Shazan. URL: http://www.shazam.com/. Consulta: Abril 2013.

[Bernal et al., 2012] Bernal, J., Gómez, P., and Bobadilla, J. (2012). Una visión práctica en el uso de la transformada de fourier como herramienta para el análisis espectral de la voz. URL: http://stel.ub.edu/labfon/sites/default/files/EFE-X-JBernal_PGomez_JBobadilla-FFT_una_vision_practica_herramienta_para_el-_analisis_espectral_de_la_voz.pdf. Consulta: Abril 2013.

[Calderón, 2006] Calderón, F. (2006). Notas de procesamiento digital de señales. URL: http://lc.fie.umich.mx/~calderon/senales/senales.pdf. Consulta: Abril 2013.

[Camarena, 2011] Camarena, J. A. (2011). Notas de síntesis y reconocimiento de voz. URL: http://lc.fie.umich.mx/~camarena/cursos.html. Consulta: Abril 2013.

[CMU, 2008] CMU (2008). Project listen. a reading tutor that listens. URL: http://www.cs.cmu.edu/~listen/index.html. Consulta: Abril 2013.

[Cowling and Sitte, 2002] Cowling, M. and Sitte, R. (2002). Recognition of environmental sounds using speech recognition techniques. *The International Series in Engineering and Computer Science*, 703:31–46.

[Fawcett, 2004] Fawcett, T. (2004). Roc graphs: Notes and practical considerations for researchers. URL: http://binf.gmu.edu/mmasso/ROC101.pdf. Consulta: Abril 2013.

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

[Ibe, 2008] Ibe, O. C. (2008). *Markov Processes for Stochastic Modeling.* Academic Press, University of Massachusetts, Lowell Massachusetts, 1 edition.

[Iosu, 1999] Iosu, S. (1999). Análisis de predicción lineal (lp). URL: http://www.euskalnet.net/iosus/speech/lp.html. Consulta: Abril 2013.

[Muller, 2007] Muller, M. (2007). *Information Retrieval for Music and Motion.* Springer, Englewood Cliffs, New Jersey.

[Oropeza, 2006] Oropeza, I. J. L. (2006). Algoritmos y métodos para el reconocimiento de voz en español mediante sílabas. *Computación y Sistemas*, 9(3):270–286.

[Pascual et al., 2007] Pascual, D., Pla, F., and Sánchez, S. (2007). Algoritmos de agrupamiento. URL: http://marmota.dlsi.uji.es/WebBIB/papers/2007/1_Pascual-MIA-2007.pdf. Consulta: Abril 2013.

[Play, 2009] Play, G. (2009). Voicelink para android versión 1.1. URL: https://play.google.com/store/apps/details?id=com.navee.android.voicelink2. Consulta: Abril 2013.

[Play, 2011] Play, G. (2011). Búsqueda por voz para android versión 2.1.4. URL: https://play.google.com/store/apps/details?id=com.google.android.voicesearch. Consulta: Abril 2013.

[Puertas, 2000] Puertas, J. I. (2000). Robustez en reconocimiento fonético de voz para aplicaciones telefónicas. URL: http://oa.upm.es/657/1/JOSE_IGNACIO_PUERTAS_TERA.pdf. Consulta: Abril 2013.

[Rabiner, 1989] Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257–286.

[Rabiner and Juang, 1993] Rabiner, L. R. and Juang, B. H. (1993). *Fundamentals of Speech Recognition*. Prentice-Hall, Englewood Cliffs, N. J.

[Rabiner and Schafer, 1978] Rabiner, L. R. and Schafer, R. W. (1978). *Digital Processing of Speech Signals*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

[Serrano, 2012] Serrano, E. (2012). Reconocimiento de voz en windows 7. URL: http://ideasabogados.com/2012/reconocimiento-de-voz-windows/. Consulta: Abril 2013.

[Sigurdsson et al., 2006] Sigurdsson, S., Petersen, K. B., and Lehn-Schioler, T. (2006). Mel frequency cepstral coefficients: An evaluation of robustness of mp3 encoded music. In *Proceedings of the International Symposium on Music Information Retrieval*.

[Software, 2012] Software, S. M. (2012). Voice translator para android versión 1.3. URL: https://play.google.com/store/apps/details?id=com.smartmobilesoftware.voicetranslatorfree. Consulta: Abril 2013.

Glosario

ANN *Artificial Neural Networks*-Redes Neuronales Artificiales

CHMM *Continuous Hidden Markov Models*-Modelos Continuos Ocultos de Markov

CMU *Carnegie Mellon University*-Universidad de Carnegie Mellon

DCT *Discrete Cosine Transform*-Transformada Coseno Discreta

DFT *Discrete Fourier Transform*-Transformada Discreta de Fourier

DHMM *Discrete Hidden Markov Models*-Modelos Discretos Ocultos de Markov

DTW *Dynamic Time Warping*-Doblado Dinámico en Tiempo

FFT *Fast Fourier Transform*-Transformada Rápida de Fourier

GG *Gabriel Graph*-Grafo de Gabriel

HMM *Hidden Markov Models*-Modelos Ocultos de Markov

IBM *International Business Machines*

IDFT *Inverse Discrete Fourier Transform*-Transformada Discreta Inversa de Fourier

LISTEN *Literacy Innovation that Speech Technology Enables*-Alfabetización Innovación que Permite la Tecnología de Voz

LPC *Linear Prediction Coefficients*-Coeficientes de Predicción Lineal

LTI *Linear Time-Invariant*-Lineal e Invariante en el Tiempo

LVQ *Learning Vector Quantization*-Aprendizaje de Cuantificación Vectorial

MFCC *Mel Frequency Cepstral Coefficients*-Coeficientes Cepstrales en las Frecuencias de Mel

PARCOR *PARtial CORrelation*-CORrelación PARcial

PDF *Probability Distribution Function*-Función de Distribución de Probabilidades

RNG *Relative Neighborhood Graph*-Grafo de Vencindad Relativa

ROC *Receiver Operating Characteristic*-Característica de Operación de Receptor

SOM *Self-Organizing Maps*-Auto-Organización de Mapas

VQ *Vector Quantization*-Cuantización Vectorial

WAV *WAVEform audio file format*-Formato de Archivo de Audio WAVE