SISTEMA DE RECONOCIMIENTO DE FONEMAS BASADO EN MFCCS Y PARÁMETROS ARTICULATORIOS

ALBERTO PATIÑO SAUCEDO

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES
BUCARAMANGA
2015

SISTEMA DE RECONOCIMIENTO DE FONEMAS BASADO EN MFCCS Y PARÁMETROS ARTICULATORIOS

ALBERTO PATIÑO SAUCEDO

Trabajo de Grado para optar al título de
Ingeniero Electrónico

Director
FRANKLIN ALEXANDER SEPÚLVEDA, PH.D

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES
BUCARAMANGA
2015

A Mamicacha y Rafa

Índice general

Índice de figuras

Índice de anexos

Resumen

TÍTULO: SISTEMA DE RECONOCIMIENTO DE FONEMAS BASADO EN MFCCS Y PARÁMETROS ARTICULATORIOS[1]

AUTOR: ALBERTO PATIÑO SAUCEDO [2]

PALABRAS CLAVE: Reconocimiento de fonemas, Parámetros articulatorios, Coeficientes Cepstrales en la escala de Mel, Modelos Ocultos de Markov

Con la llegada de la era digital, la comprensión del lenguaje hablado por parte de los sistemas computacionales ha visto un desarrollo notable, pero en el caso del reconocimiento de fonemas aún se sigue investigando para lograr un reconocimiento cada vez más preciso. En este trabajo de grado se muestra que la información contenida en los datos articulatorios puede ayudar a mejorar las tasas de reconocimiento del habla a nivel fonético. Para ello se han construido dos sistemas de reconocimiento de fonemas dependientes del hablante: un sistema base que parametriza la señal de voz en forma de Coeficientes Cepstrales en la escala de Mel (MFCCs por sus siglas en inglés) y otro que combina los parámetros MFCCs con los datos de articulografía electromagnética (EMA, por sus siglas en inglés). Los datos de entrenamiento y validación fueron tomados de la base de datos MOCHA-TIMIT. De la implementación de los sistemas, se observa una mejoría en la precisión del sistema base cuando se le agregan los datos articulatorios de un 11 %, llegando a tenerse una tasa de precisión alrededor del 70 %. Estos resultados demuestran que los datos articulatorios poseen información útil que adicionada a las señales acústicas parametrizadas pueden utilizarse para mejorar el rendimiento de los sistemas de reconocimiento de fonemas.

[1] Trabajo de grado

[2] Facultad de Ingenierías Fisico Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones. Director: Franklin Alexander Sepúlveda, Ph.D

Abstract

TITLE: PHONE RECOGNITION BASED ON MFCCS AND ARTICULATORY PARAMETERS[1]

AUTHOR: ALBERTO PATIÑO SAUCEDO [2]

KEY WORDS: Phone recognition, Articulatory parameters, Mel-Cepstrum coeficcients, Hidden Markov Models

With the arrival of the digital age, the understanding of speech by computer systems has seen a remarkable development, but in the case of phone recognition there is still research to do in order to achieve a more precise recognition. In this research work it is shown that the information included in the articulatory data can help to improve speech recognition rates at the phonetic level. In order to achieve this, two speaker-dependent phoneme recognition systems have been built: a baseline system that codes the voice signal in the form of Mel Frequency Cepstral Coefficientes (MFCCs) and a second system that combines both MFCCs and articulatory (EMA) data. The training and testing data were taken from the MOCHA-TIMIT database. From the implementation of both systems, it has been observed an improvement of about 11 % in the accuracy rate of the baseline system when articulatory data is added, reaching around 70 %. These scores prove that articulatory data hold useful information that being added to the feature vectors of the acoustic signals are able to improve the performance of phoneme recognition systems.

[1]Degree project

[2]Faculty of Physical Mechanical Engineerings. Department of Electric, Electronic and Telecommunications Engineerings. Advisor: Franklin Alexander Sepúlveda, Ph.D

1. Introducción

1.1. Motivación

Con la llegada de la era digital, los sistemas computarizados se han desarrollado y propagado al punto de ser parte indistinguible de la actividad humana, ya sea como herramientas o como entretención. Esto ha llevado a incrementar las expectativas de una interacción amistosa con tales sistemas, lo cual ha promovido el diseño de aplicaciones con características cada vez más 'humanas', es decir, con la capacidad de entender y replicar la habilidad que más distingue a nuestra especie: el lenguaje.

En consecuencia, la comprensión del lenguaje hablado ha visto un desarrollo notable con la aparición de sistemas de reconocimiento automático de la voz (ASR, por sus siglas en inglés); entre las que se incluyen control por comandos de voz, sistemas de diálogo para personas con impedimentos auditivos y fonoaudiológicos, traducción automática, etc. [10].

El reconocimiento automático de la voz ha sido objeto de intensa investigación por más de cuatro décadas, alcanzando resultados notables; pero mientras el reconocimiento de dígitos ha llegado a tasas de 99.6 % [13], no se puede decir lo mismo del reconocimiento de fonemas, para el cual las mejores tasas se sitúan alrededor del 80 % [1]. El reconocimiento de fonemas es una tarea de relevante importancia que podría mejorar el desempeño de otros sistemas de procesamiento de señales de voz tales como el mismo reconocimiento de voz (ASR) [12], la verificación del hablante [7], y el aprendizaje de un segundo idioma [30], entre otros.

Una de las ventajas del reconocimiento de fonemas es su versatilidad, ya que permite conocer las características fonéticas de la voz, independiente del vocabulario, pudiéndose adaptar a diferentes idiomas, en contraste con el reconocimiento de voz enfocado a palabras o frases aisladas. Esta versatilidad se ve reflejada en un

amplio rango de aplicaciones entre las cuales están los sistemas de reconocimiento automático de voz de amplio vocabulario (LVASR) [16], detección de palabras clave [27], reconocimiento de idioma [33] y aplicaciones de detección de acordes musicales [6], entre otros. La capacidad de reconocer fonemas con una alta precisión se constituye entonces en un problema fundamental en la rama del procesamiento natural del lenguaje.

Dentro de las posibles aplicaciones del reconocimiento de fonemas, se destaca su aplicación a la tarea de verificación del hablante. Una de las cuestiones por resolver está en definir qué tanta influencia puede tener la descripción fonética de la voz de cada individuo en dicha verificación, tal como se plantea en [33]. Es sabido que diversos factores como la edad, el género, extracción social y región de procedencia actúan como diferenciadores en la forma de hablar de las personas.

Hasta donde se conoce, no existen bases de datos *(corpus)* de señales de voz debidamente segmentadas a nivel fonético para el caso del castellano en su versión colombiana, lo cual genera dificultades. Ello provoca que generalmente se usen bases de datos en inglés como la TIMIT y la MOCHA-TIMIT [15]. Esta aparente dificultad, sin embargo, se convierte en una ventaja al ser el inglés un idioma con mayor variabilidad fonética que el español, lo que permitiría una mayor diferenciación entre muestras de voz similares [11].

1.2. Conceptos básicos en reconocimiento de fonemas

Para entender el funcionamiento de un sistema de reconocimiento de fonemas es necesario hacer una revisión de los conceptos básicos relacionados con este campo. En ésta sección se explica brevemente el concepto de la señal de voz y el proceso de producción y comprensión de la misma desde un punto de vista anatómico. También se hace una introducción al concepto de fonema y se revisan los esfuerzos y técnicas que han sido utilizados para el reconocimiento de fonemas.

1.2.1. La señal de voz

Una de las características que distingue a los seres humanos de otras especies es la capacidad de transmitir sonidos con significado abstracto, llamado voz. La voz es una señal acústica, es decir, una vibración longitudinal del aire que se puede representar tanto en el dominio del tiempo como en el de frecuencia. Tales sonidos son parte fundamental del lenguaje, con el cual es posible manifestar y comunicar los pensamientos.

La producción de la voz es un proceso inherente a toda persona sana que se aprende desde temprana edad y que permite la conversión de pensamientos en sonidos inteligibles. Si bien a simple vista parece un procedimiento sencillo, una mirada más profunda al proceso de producción de voz revela un complicado mecanismo. En la producción de voz intervienen tres subsistemas anatómicos que trabajan armónicamente: aparato respiratorio (conformado por los pulmones y el diafragma), fonatorio (conformado por la laringe y las cuerdas vocales) y articulatorio (conformado por la lengua, los labios, la mandíbula y el velo) (figura 1.1). El proceso es el siguiente: El aire es expelido de los pulmones a través de la tráquea. El flujo de aire causa una vibración de las cuerdas vocales, que actúan como un primer filtro. El aparato articulatorio actúa como un segundo filtro, generando distintos sonidos según la disposición del tracto vocal [5].

La comprensión del mensaje que transporta la voz es un proceso igualmente complejo donde interviene el aparato auditivo, conformado por el oído externo, el medio y el interno. Las ondas sonoras son captadas por el oído externo (la oreja). El oído medio transforma la energía sonora en mecánica y el interno hace la conversión definitiva de la energía mecánica a impulsos eléctricos que llegan al cerebro para su análisis [17].

La señal acústica es una onda longitudinal, es decir, vibra en la misma dirección de su propagación. Tal vibración puede ser captada en un rango de 20 a 20 KHz. El oído humano responde al sonido como variaciones de la presión del aire circundante. Este hecho es aprovechado también por los micrófonos para convertir la onda acústica en una señal eléctrica por medio de transductores de presión a energía eléctrica. De esta forma, es posible captar la señal acústica y almacenarla para su posterior análisis en sistemas computacionales. Además de la señal acústica, es posible registrar información acerca de la producción y articulación de la voz por medio de técnicas como resonancia magnética, rayos X e imagen ultrasónica [28]

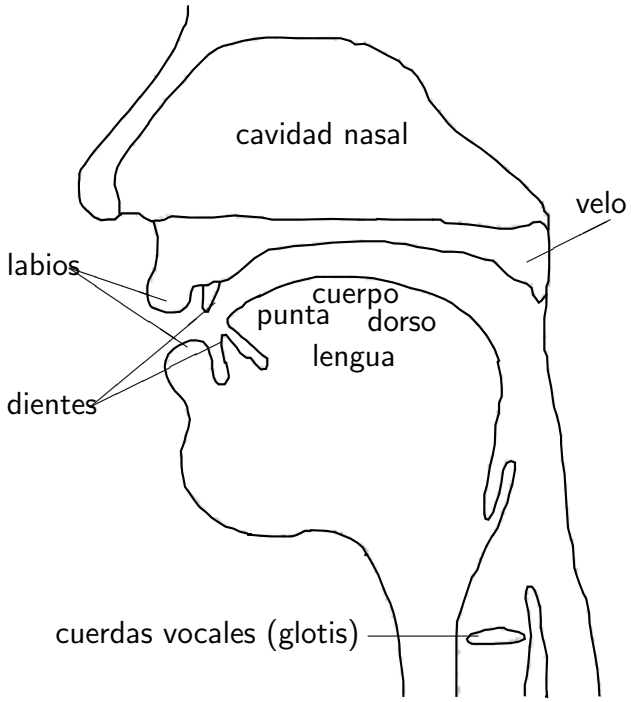

Figura 1.1: Anatomía de la producción de voz. Adaptada de [28].

ó por medio de sensores colocados en puntos estratégicos del aparato articulatorio. Este último método se conoce como Articulografía Electromagnética (EMA por sus siglas en inglés), del cual se hablará más adelante.

1.2.2. Fonemas

La producción de voz implica no sólo la parte mecánica mencionada anteriormente, sino también un trabajo previo por parte del cerebro. Es en la mente donde se generan los pensamientos a ser transmitidos por medio del habla. Las palabras a su vez están constituidas por fonemas, que son definidas como las unidades teóricas básicas que otorgan un significado lingüístico a la señal de voz [5]. Si se observa una señal acústica en el tiempo, los fonemas son las unidades básicas en que dicha señal se puede subdividir (ver figura 1.2).

El sistema estándar para representar los fonemas se conoce como Alfabeto Fonético Internacional (IPA). Por ejemplo la palabra 'choza' está compuesta por 4 fonemas, /ʧ/, /o/, /s/ (/θ/ en España) y /a/. Los fonemas consonánticos se clasifican según el punto y modo de articulación y las vocales se clasifican según el modo (posición de la lengua) y la apertura de la cavidad bucal (ver tabla 1.1) [11].

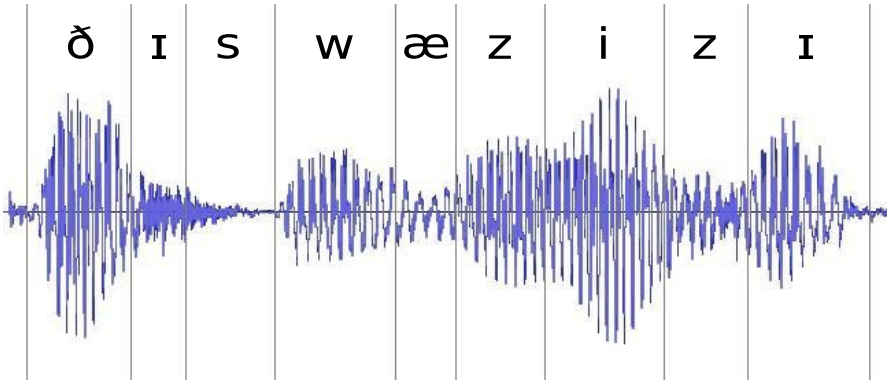

Figura 1.2: Señal de voz de que representa la frase “This was easy” dividida en sus fonemas.

	Labial	Labiodental	Dental	Alveolar	Palatal	Velar	Glotal
Plosiva	p b			t d		k g	ʔ
Nasal	m			n		ŋ	
Fricativa		f v	θ ð	s z	ʃ ʒ		h
Retroflexiva				r			
Lateral				l			
Fluida	w				j		

Cuadro 1.1: Consonantes en inglés distribuidas en punto (columnas) y modo (filas) de articulación. Basado en la tabla IPA [9].

El punto de articulación se refiere al lugar del tracto vocal donde se produce la articulación, produciéndose normalmente por el contacto entre un articulador fijo (como el velo) y uno móvil (como la lengua o los labios). El modo de articulación tiene que ver con la forma en que se expulsa el aire al pronunciarse los fonemas [11].

1.2.3. Algunas técnicas de reconocimiento de fonemas y trabajos previos

Si bien la representación fonética del lenguaje está plenamente estandarizada, cuando se hace un análisis más detallado de la señal de voz se encuentra que la variedad física de los sistemas respiratorio, fonatorio y articulatorio de los seres humanos se traduce en una variabilidad en los fonemas, lo que hace más difícil su identificación por medio de algoritmos. Por ello se debe recurrir a métodos de extracción de parámetros óptimos de la señal acústica para ser entrenados por modelos estadísticos que permitan obtener una estimación de la sucesión de fonemas más

probables en un segmento de voz, para el caso de reconocimiento de fonemas.

El primer paso para el reconocimiento de fonemas es entonces, obtener una representación adecuada de la señal acústica por medio de vectores característicos de la misma. Esto implica hacer un preprocesamiento de la señal en el dominio de la frecuencia para reducir el ruido y eliminar frecuencias indeseadas. También es común implementar un sistema de detección de activación de la voz (VAD), con el fin de determinar el comienzo y final exacto de la señal de habla, descartando lo que esté por fuera de este intervalo como ruido. Una vez la señal está preprocesada se procede a extraer de la misma los vectores característicos. Las técnicas actuales divergen entre extracción por Coeficientes de Mel (MFCCs) o extracción por coeficientes predicción Lineal (LPF) [19]. En este trabajo se utilizará la técnica de MFCCs porque se basa en la representación en frecuencia de la señal de voz y se apoya en en hecho de que la percepción de la frecuencia no sigue un patrón lineal. Más adelante se describirá en detalle dicha técnica.

El segundo paso en el reconocimiento de fonemas es el entrenamiento por medio de modelos estadísticos. En cuanto al reconocimiento de fonemas bajo la base del TIMIT, las tasas de reconocimiento han ido incrementando conforme se van optimizando las técnicas de clasificación, pudiéndose ver la evolución desde el primer trabajo de Lee y Hon en 1989, basado en HMMs, con una precisión del 66,08 % [12], pasando por Robinson, 1994, basado en RNNs, con una precisión del 73,9 % [22], Schwarz, quien en 2006 alcanzó una precisión del 78,52 % [20] hasta el trabajo de Mohamed en 2011, que alcanzó tasas del 79.3 % usando DBNs *(Deep Belief Networks)* [1].

1.3. Planteamiento del problema

En la actualidad, las mejores tasas en la tarea de reconocimiento del fonemas se sitúan alrededor del 80 % [1], por lo cual se requiere incrementar la investigación alrededor de esta tarea. Sin embargo, se necesita un sistema base sobre el cual poder realizar futuras mejoras a modo de resultados de investigación.

En aras de mejorar el desempeño de dichos sistemas, una de las opciones consiste en mejorar la representación de la señal de voz agregando información adicional a la misma. En particular los datos articulatorios (EMA) se erigen como una alternativa para desempeñar esta labor, debido a que dichos datos contienen información sobre

el mecanismo de producción de voz que no se puede obtener directamente de la señal acústica.

Se plantea entonces el diseño de un sistema de reconocimiento de fonemas, independiente de vocabulario, que utilice además de los datos acústicos, la información articulatoria proveniente de datos EMA. Dicho sistema será entrenado y evaluado sobre la base de datos MOCHA-TIMIT. El criterio de evaluación será la tasa de precisión en el reconocimiento de fonemas en una muestra representativa extraída de esta base de datos.

En resumen, se busca establecer si la incorporación de datos articulatorios mejorará la tasa de reconocimiento de fonemas al compararse el sistema planteado con uno donde no se utilice esta información; por lo cual se debe hacer una evaluación para verificar si el sistema planteado puede mejorar las tasas existentes en el reconocimiento de fonemas. Para realizar esta labor se plantea:

1. El diseño de un sistema de reconocimiento de fonemas usando una plataforma o motor de entrenamiento de modelos acústicos ampliamente reconocido y probado por la comunidad científica, por ejemplo HTK [8]. La señal de voz para este sistema se codifica en MFCCs.

2. El diseño de un sistema de reconocimiento de fonemas en el mismo motor de entrenamiento utilizando información de la señal de voz codificada en MFCCs y parámetros articulatorios.

2. Reconocimiento de fonemas basado en Modelos Ocultos de Markov

La tarea del reconocimiento de voz usando la técnica de *Modelos Ocultos de Markov (HMMs)* se puede dividir en dos grandes pasos: El primero se ocupa de la extracción de vectores característicos de la señal de habla, mientras que en el segundo se aplica la técnica de HMMs a dichos vectores con el fin de obtener el modelo de la señal acústica que representa los fonemas.

2.1. Extracción de características con MFCCs

El primer paso para construir un sistema de reconocimiento de fonemas consiste en extraer información relevante contenida en vectores característicos. Esto con el fin de obtener un modelo adecuado de la señal de habla, de forma tal que se puedan usar menos recursos computacionales que si se tuviera en cuenta la señal de habla sin procesar. Existen varias opciones para realizar la extracción, siendo las más utilizadas las técnicas de Predicción Lineal (LPC) y Coeficientes Cepstrales en Frecuencia de Mel (MFCCs) [21].

Otros valores se pueden añadir a los vectores característicos tales como medidas de la energía, coeficientes delta y de aceleración. Los coeficientes delta representan una aproximación de la derivada de los coeficientes obtenidos ya sea por LPC o MFCCs, mientras que los coeficientes de aceleración se aproximan a la segunda derivada de dichos coeficientes [21]. En el marco de este trabajo se utilizará la técnica de MFCCs para obtener los vectores característicos de la señal de voz y además se considerarán los parámetros articulatorios dados por los datos EMA como

coeficientes adicionales de la misma.

La extracción de vectores característicos por MFCCs consiste principalmente en cuatro etapas. Primero se hace un preprocesado de la señal, donde se reduce el ruido y se hace la detección de la activación de la voz (VAD). El segundo paso es dividir la señal de voz en pequeñas secciones superpuestas, de forma que cada sección represente una señal de voz quasi-estacionaria. El tercer paso es obtener un vector característico para cada sección, aplicando la técnica de MFCCs. Como último paso se hace el postprocesado de esta información, consistente en normalizar los datos obtenidos.

2.1.1. Preprocesado de la señal de voz

Se le hace preprocesado a la señal de voz para convertirla en una señal que pueda ser tratada de una manera adecuada por los sistemas de reconocimiento. Para ello se utilizan técnicas de reducción del ruido, preénfasis y detección de la activación de la Voz (VAD por sus siglas en inglés). Mediante un filtro pasa bajas se pueden eliminar los componentes espectrales más allá de los 8Khz, por ejemplo. Un filtro utilizado muy comúnmente es el de preénfasis, que "nivela" el espectro de la señal de voz. Esto se hace con el fin de mantener la energía constante en todo el ancho de banda (Figura 2.2). El filtro de preénfasis mayormente utilizado es filtro FIR que se describe a continuación:

$$H(z) = 1 - 0{,}97z^{-1} \tag{2.1}$$

La respuesta en frecuencia de este filtro puede ser observada en la figura 2.1.

Luego de la reducción del ruido y del preénfasis se aplica un algoritmo de VAD. Esto se hace para detectar los puntos de inicio y fin de la señal de voz. Las partes de la señal acústica por fuera de los intervalos de activación de la voz son tomados como silencios o ruidos de fondo. El algoritmo de VAD parece sencillo de crear a simple vista, pero en la práctica su implementación presenta dificultades y la creación de un algoritmo eficiente para resolver este problema aún es materia de investigación. La dificultad principal estriba en que en señales ruidosas el ruido tiene una energía tal que puede ser confundida como señal de voz.

Algunos de los algoritmos de VAD más robustos utilizan información de la

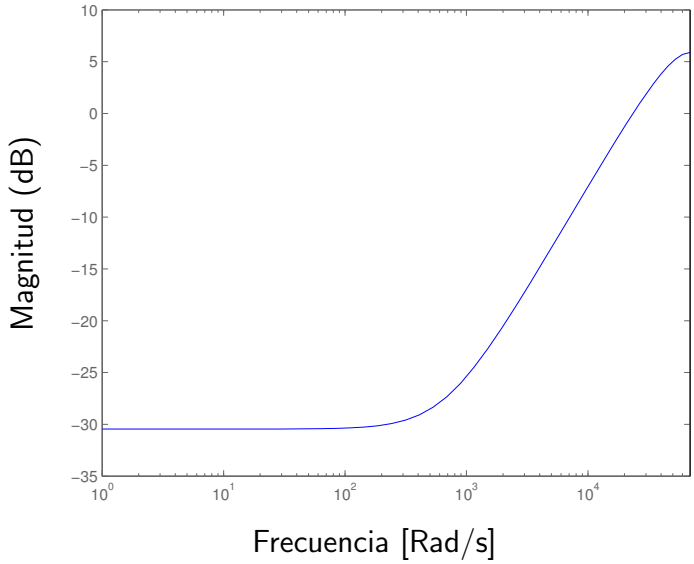

Figura 2.1: Filtro FIR de preénfasis.

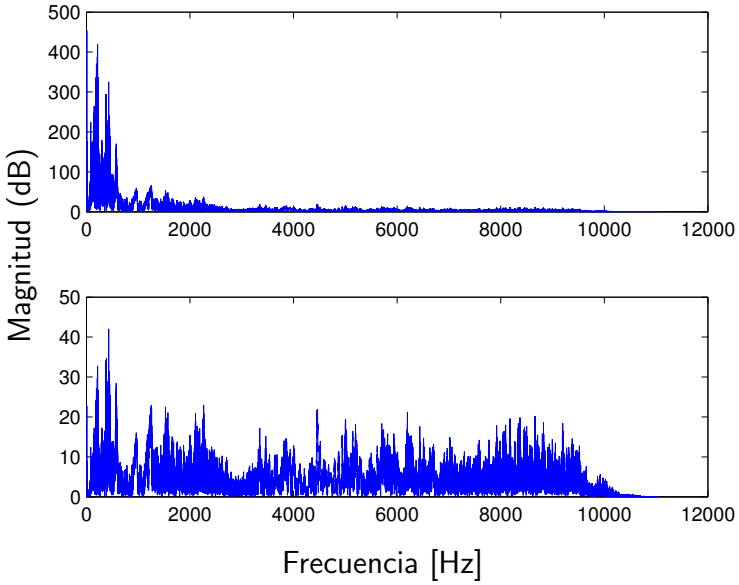

Figura 2.2: Amplitud del espectro de una señal de voz sin preénfasis (arriba) y con preénfasis (abajo).

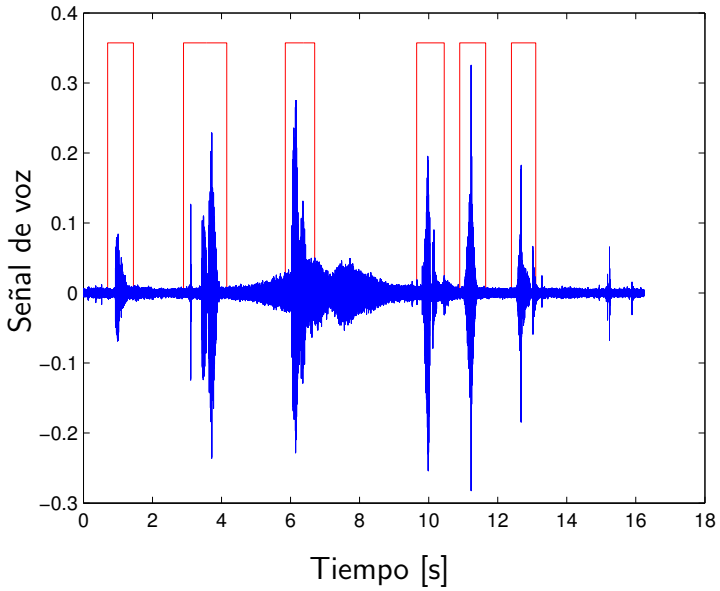

Figura 2.3: Detección de la activación de la voz para el dictado de una serie de números.

energía, el espectro de frecuencias, la relación señal a ruido y medidas de allanamiento del espectro. Un algoritmo de VAD que utiliza estos conceptos se presenta en los Anexos a este trabajo. Un resultado típico de la implementación de este algoritmo se puede ver en la figura 2.3.

2.1.2. División de la señal de voz en secciones

Luego del preprocesado, se divide la señal en secciones superpuestas. Cada sección tiene una de longitud de K muestras, con muestras adyacentes separadas P muestras, siendo $K > P$. En el presente trabajo, se utiliza una frecuencia de muestreo de 16 KHz. Los valores de K y P son de 400 y 160 respectivamente. Esto corresponde a secciones de 25 ms, separadas 10 ms. Escogiéndose secciones de 25 ms se puede asumir que cada muestra es casi estacionaria.

A cada sección obtenida se le aplica una ventana para reducir la discontinuidad entre bloques, en un proceso denominado *ventaneo*. Una ventana comúnmente utilizada es la del tipo *Hamming*, dada por [32]:

$$s(k) = 0{,}54 - 0{,}46\cos(\frac{2\pi k}{K-1}), \tag{2.2}$$

donde K es el tamaño del bloque original, k el k-ésimo elemento del mismo y $s(k)$ es el bloque discreto obtenido.

2.1.3. Extracción de los coeficientes cepstrales en la escala de Mel

La técnica de coeficientes cepstrales en la escala de Mel (MFCCs)[24] es un método ampliamente utilizado para obtener los vectores característicos de la señal de voz. Dicha técnica se inspira en el funcionamiento del órgano más importante que interviene en la audición humana, la cóclea o caracol.

El método se aplica a cada bloque obtenido en la sección anterior y consta de las siguientes etapas:

1. Se calcula el periodograma (densidad espectral de potencia).
2. Se aplica un banco de R filtros al espectro de potencia y se suman las energías de cada filtro.
3. Se toma el logaritmo de las energías del banco de filtros para formar un vector de longitud R.
4. Se calcula la DCT (Transformación discreta del coseno) del vector.

2.1.3.1. Densidad espectral de potencia

La densidad espectral o periodograma es una representación de la energía de cada frecuencia. El cálculo de esto se hace motivado por el funcionamiento de la cóclea, que vibra en diferentes puntos de acuerdo a la frecuencia del sonido que entra al oído. Dependiendo de la localización de la vibración, ciertas vellosidades de la cóclea estimulan los nervios que son los que envían al cerebro la información sobre la frecuencia escuchada [32].

La densidad espectral de potencia se calcula en forma discreta aplicando la Transformada discreta de Fourier a cada i-ésimo bloque $s_i(k)$ y luego se calcula la potencia de esta transformada.

La Transformada discreta de Fourier (DFT) del bloque $s_i(k)$ se define de la forma [2]

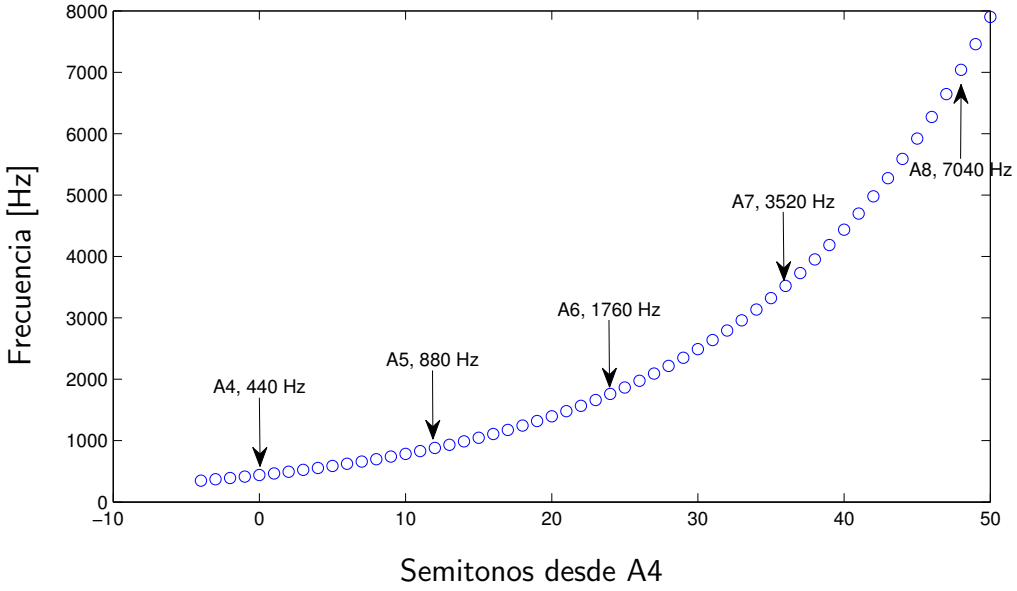

Figura 2.4: Relación entre el semitono percibido desde A4 (La 440 Hz) y la frecuencia.

$$S_i(n) = \sum_{k=1}^{K} s_i(k) e^{-j2\pi kn/K} \quad 1 \leq n \leq N \tag{2.3}$$

La densidad espectral de potencia se calcula así [2]:

$$P_i(n) = \frac{1}{K} \left| S_i(n) \right|^2 , \tag{2.4}$$

donde K es el tamaño de cada bloque y N es tamaño de la DFT.

2.1.3.2. Aplicación del banco de filtros en la escala Mel

El periodograma calculado aún contiene información que no es necesaria para el reconocimiento de voz. Por lo general, la cóclea no puede discernir la diferencia entre dos frecuencias cercanamente espaciadas. Este fenómeno se hace más pronunciado a medida que la frecuencia se hace mayor. Para ilustrar mejor este efecto, considérese la relación entre las notas de la escala musical y la frecuencia de las mismas (figura 2.4): la distancia frecuencial entre dos semitonos adyacentes aumenta con el incremento de la frecuencia generando un comportamiento exponencial.

Por lo anterior, se hace uso de la escala de Mel, que se aproxima más al comportamiento del oído humano que una escala no lineal. La conversión de frecuencias de

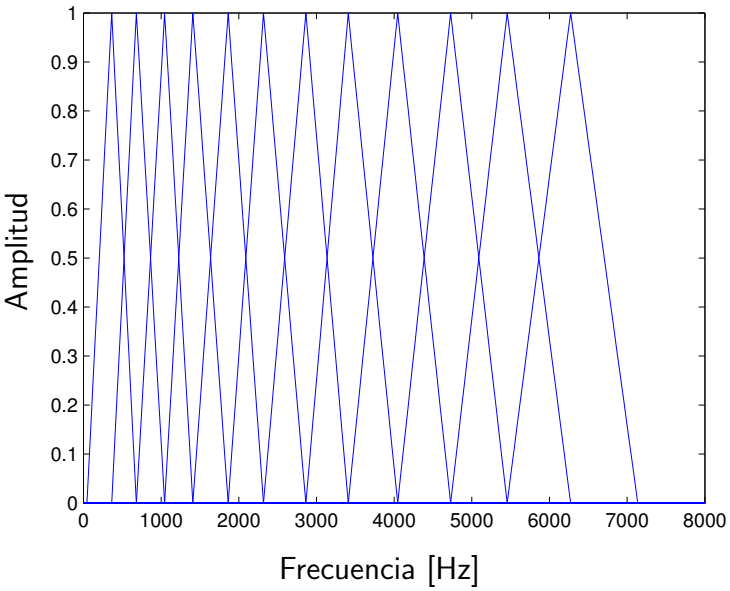

Figura 2.5: Banco de 12 filtros MFCC.

la escala normal f a la escala Mel está dada por [32]

$$M(f) = 1125 \ln(1 + f/700), \tag{2.5}$$

Con esta escala se obtiene el banco de filtros que se le aplicará al periodograma, de acuerdo a la siguiente expresión [32]

$$H_m(k) = \begin{cases} 0 & k < f(m-1) \\ \frac{k - f(m-1)}{f(m) - f(m-1)} & f(m-1) \leq k \leq f(m) \\ \frac{f(m+1) - k}{f(m+1) - f(m)} & f(m) \leq k \leq f(m+1) \\ 0 & k > f(m+1) \end{cases} \tag{2.6}$$

La figura 2.5 muestra un banco de 12 filtros espaciados según la frecuencia de Mel. Nótese que la relación del ancho de banda es proporcional a la frecuencia central. Para cada filtro se calcula la sumatoria de energía.

Una vez obtenidas la energías de cada filtro (un vector de R elementos, siendo R el número de filtros aplicados) se procede a obtener el logaritmo de la energía

asociada a cada filtro. Como paso final se calcula la DCT (Transformada discreta del coseno) de los logaritmos de las energías. Esto se hace para decorrelacionar las energías (recuérdese que los bloques de voz calculados se traslapan entre sí)[32].

De otra parte, se ha observado un mayor rendimiento en los sistemas de reconocimiento de voz si se calculan no sólo los coeficientes MFCCs sino también sus primera y segunda derivadas. Esto se debe a la señal de habla también contiene información de la dinámica del sistema articulatorio [18]. Dicho de otra forma, La velocidad y aceleración de los componentes del aparato fonatorio también se ve reflejada en la señal de voz. Para calcular el coeficiente delta se utiliza [26]:

$$d_t = \frac{\sum_{n=1}^{N} n(c_{t+n} - c_{t-n})}{2\sum_{n=1}^{N} n^2} \tag{2.7}$$

2.1.4. Postprocesado de la señal de voz

Una vez calculados los coeficientes, es necesario normalizar la información, es decir, hacer que los vectores tengan promedio cero y desviación estándar uniforme. Esto es importante para su implementación en la técnica de modelos ocultos de Markov [5]. Para ello primero se calcula el vector promedio $f_{\vec{\mu}}(n)$ de la matriz de coeficientes característicos $x(n;m)$:

$$f_{\vec{\mu}}(n) = \frac{1}{M}\sum_{m=0}^{M-1} x(n;m) \tag{2.8}$$

Con $f_{\vec{\mu}}(n)$ se calculan los vectores característicos normalizados, de la siguiente forma

$$f(n;m) = x(n;m) - f_{\vec{\mu}}(n) \tag{2.9}$$

2.2. Entrenamiento y reconocimiento con Modelos ocultos de Markov

Los sistemas de reconocimiento de voz constan de una serie de modelos estadísticos que representan los diferentes sonidos a ser reconocidos, en este caso los fonemas. Al tener la señal de voz una estructura temporal, esta se puede codificar como una secuencia de vectores espectrales que abarcan todo el rango de frecuencias de audio, los coeficientes de Mel. Es en esta circunstancia donde los Modelos Ocultos de Markov pueden ser utilizados, para construir dichos modelos a partir de los vectores característicos de la señal de voz [14].

2.2.1. Reconocimiento de modelos aislados

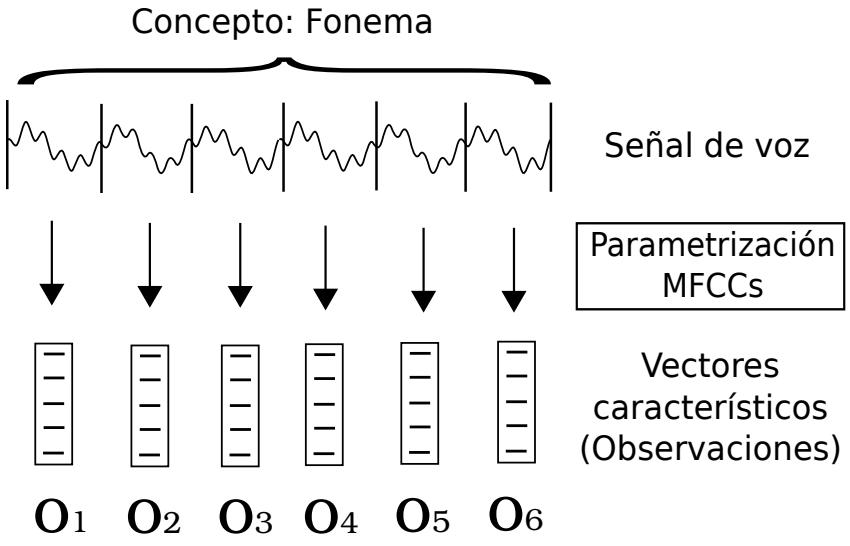

Figura 2.6: Modelo de Markov representado como una máquina de estados generadora de una secuencia de observación.

Si se representa la señal de voz como una serie de vectores característicos u *observaciones* $\mathbf{O}$ (ver figura 2.7), definidas como

$$\mathbf{O} = \mathbf{o_1}, \mathbf{o_2}, ..., \mathbf{o_T} \tag{2.10}$$

donde $\mathbf{o_t}$ es el vector característico de la voz observado en un tiempo t, el reconocimiento de cada estado de voz, en este caso cada fonema, se puede considerar

como la solución a

$$\arg \max_i \{P(w_i|\mathbf{O})\} \tag{2.11}$$

donde $\mathbf{w_i}$ es el i-ésimo fonema del vocabulario. Este problema no es computable directamente, pero usando la regla de Bayes se tiene [25]:

$$P(w_i|\mathbf{O}) = \frac{P(\mathbf{O}|w_i)P(w_i)}{P(\mathbf{O})} \tag{2.12}$$

En palabras, dada una serie de observaciones $\mathbf{O}$, el problema del reconocimiento de fonemas se limita a determinar para cuál i-ésimo fonema de un vocabulario dado la probabilidad de ocurrencia es máxima. Dado $P(w_i)$ por defecto, el fonema más probable depende sólo de la probabilidad $P(\mathbf{O}|w_i)$, es decir la probalidad de ocurrencia de una serie de observaciones dado un fonema w_i. Esto se puede expresar de la siguiente manera

$$P(\mathbf{o_1}, \mathbf{o_2}, \mathbf{o_3}, ...|w_i) \tag{2.13}$$

La anterior expresión no se puede calcular directamente debido a la dimensionalidad de la secuencia $\mathbf{O}$. Sin embargo, si se asume un modelo paramétrico de la producción de voz como los Modelos Ocultos de Markov, la estimación de $P(\mathbf{O}|w_i)$ se puede reemplazar por un problema mucho más simple, el cual consiste en estimar los parámetros de los HMMs asociados a las observaciones $\mathbf{O}$.

2.2.2. Arquitectura de un reconocedor basado en HMMs

Un modelo de Markov es una máquina de estados finita, que cambia su estado cada unidad de tiempo. Se asume que la secuencia de observaciones $\mathbf{O}$ es generada por esta máquina de estados (ver figura 2.7).

Cada tiempo t en el que se ingresa a un estado j, un vector característico $\mathbf{o_t}$ se genera, de acuerdo a la densidad de probabilidad $b_j(\mathbf{o_t})$. Además, la transición del estado i al j también es probabilística y está dada por la probabilidad discreta a_{ij}.

La probabilidad conjunta de que $\mathbf{O}$ está generado por el modelo M moviéndose a través de la secuencia de estados X se calcula simplemente como el producto de las probabilidades de transición y las probabilidades de salida

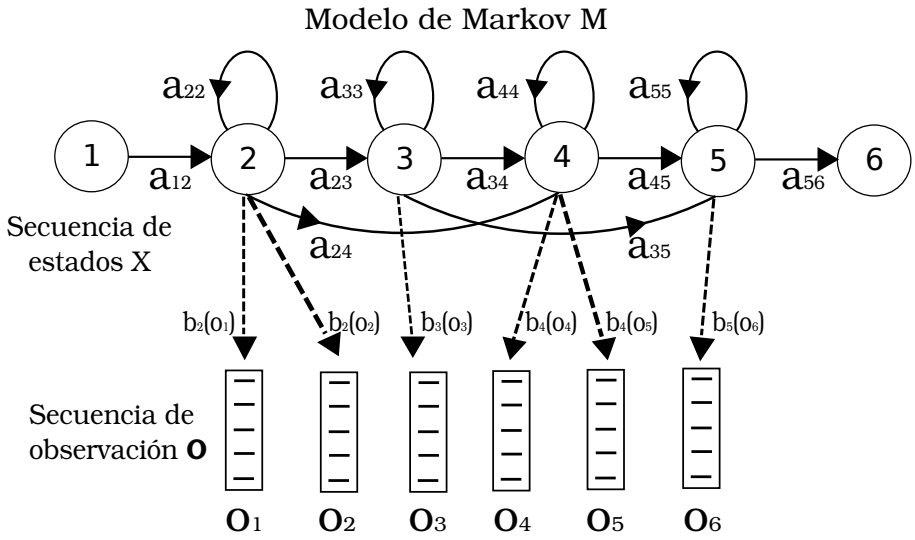

Figura 2.7: Reconocimiento de modelos aislados. Adaptada del *HTKBook* [25].

$$P(\mathbf{O}, X|M) = a_{12}b_2(\mathbf{o_1})a_{22}b_2(\mathbf{o_2})a_{23}b_3(\mathbf{o_3})... \tag{2.14}$$

No obstante, en la práctica, sólo la secuencia de observación **O** se conoce, mientras la secuencia de estados subyacente es desconocida. Por esto se les llama a los modelos *Ocultos*.

Dado que X es desconocido, la probabilidad requerida se computa sumando todos los posibles estados $X = x(1), x(2), x(3), ..., x(t)$ esto es:

$$P(\mathbf{O}|M) = \sum_X a_{x(0)x(1)} \prod_{t=1}^{T} b_{x(t)}\mathbf{o_t} a_{x(t)x(t+1)} \tag{2.15}$$

donde $x(0)$ es el modelo del estado inicial y $x(T+1)$ es el modelo de salida.

Si bien la computación directa de esta ecuación no es manejable de forma directa, existen métodos recursivos que permiten que ambas cantidades sean calculadas eficientemente [14]. Asumiéndose que $P(\mathbf{O}|w_i) = P(\mathbf{O}|M_i)$, donde M_i es la secuencia de modelos de Markov generados, se tienen todos los elementos para resolver el problema del reconocimiento.

Todo esto se hace bajo la suposición de que se conocen los parámetros a_{ij} y $b_j(\mathbf{o_t})$ para cada modelo M_i. Dado un conjunto de muestras de entrenamiento correspon-

dientes a un modelo en particular, los parámetros de ese modelo se pueden calcular automáticamente por un método robusto de re-estimación.

Con un número de muestras suficiente de cada fonema, se puede construir un modelo de Markov oculto que modele todas las fuentes de variabilidad inherentes al habla. La siguiente figura ilustra el proceso de entrenamiento y reconocimiento por medio de HMMs.

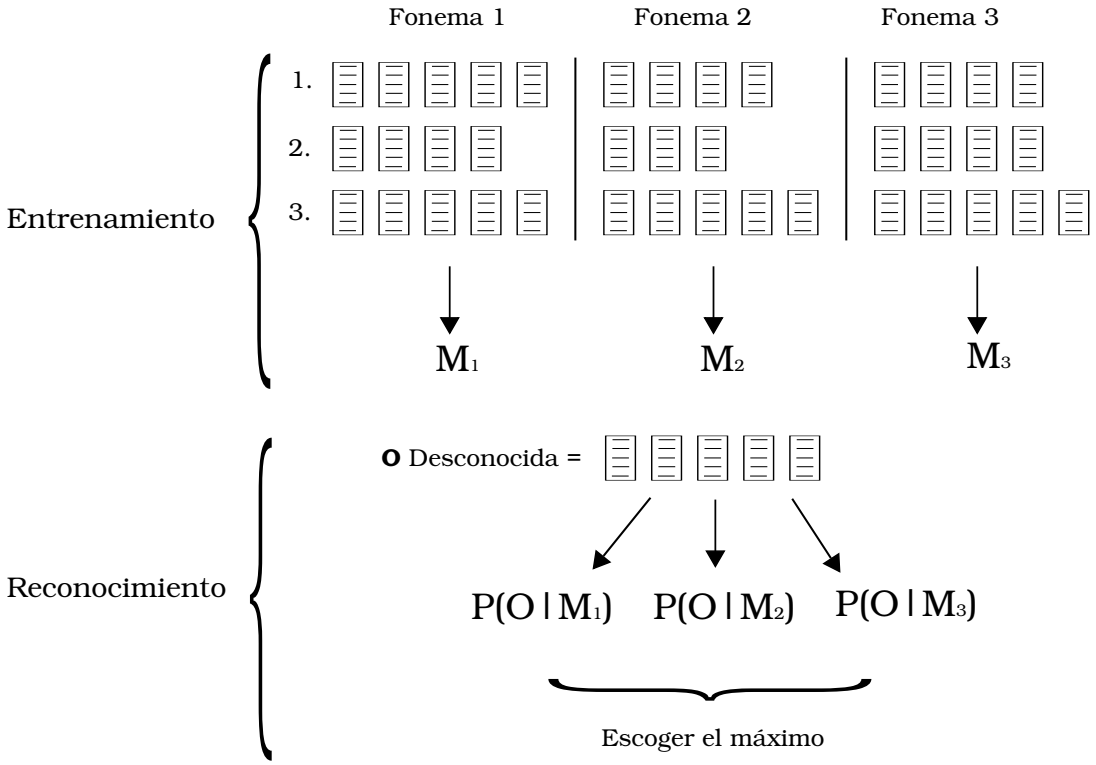

Figura 2.8: Ilustración del proceso de entrenamiento y reconocimiento de 3 fonemas, con 3 muestras para cada fonema.

3. Desarrollo del sistema en HTK

3.1. La herramienta HTK

HTK es una herramienta para la construcción y manipulación de Modelos Ocultos de Markov. Fue desarrollada en el Departamento de Ingeniería de la Universidad de Cambridge (CUED) y es utilizada principalmente para la investigación en reconocimiento de voz, aunque también ha sido usada en otras ramas de la investigación tales como síntesis de la voz, reconocimiento de caracteres y secuenciamiento de ADN [8].

Consiste en una serie de módulos de librerías cuyo código fuente está en lenguaje C, diseñadas para ser ejecutadas en forma de comandos de línea. Estas herramientas proveen facilidades para el análisis del habla, el entrenamiento de HMMs, además del análisis de resultados

La versión estable de HTK es la 3.4.1 y se puede obtener de forma gratuita en el sitio web http://htk.eng.cam.ac.uk/ previo registro y aceptación de una licencia. Para ejecutarse una aplicación específica de HTK se proporciona el nombre de la herramienta seguida de los argumentos principales que se pueden configurar con argumentos opcionales. Por ejemplo el siguiente comando invocaría a la herramienta ficticia *HFic*:

```
HFic -T 1 -f 14.8 -a -s texto archivo1 archivo2
```

Esta herramienta tiene dos argumentos principales llamadas archivo1 y archivo2 y cuatro opciones, denotadas por un guión seguido de una letra. Los valores de las opciones -T, -f y -s son un número entero, uno flotante y una cadena de caracteres (*string*) respectivamente. la opción -a se usa como bandera para activar o desactivar alguna característica de la herramienta y por tanto no tiene un valor definido.

El estilo por líneas de comando, si bien parece anticuado comparado con las interfaces gráficas modernas, es muy útil para escribir comandos en terminales *(shells)* que controlen la ejecución de HTK, lo cual facilita la construcción de sistemas complejos, la experimentación y la documentación de los detalles.

A continuación se detallan brevemente algunas de las herramientas de HTK para la preparación, el entrenamiento, reconocimiento y validación de datos (figura 3.1).

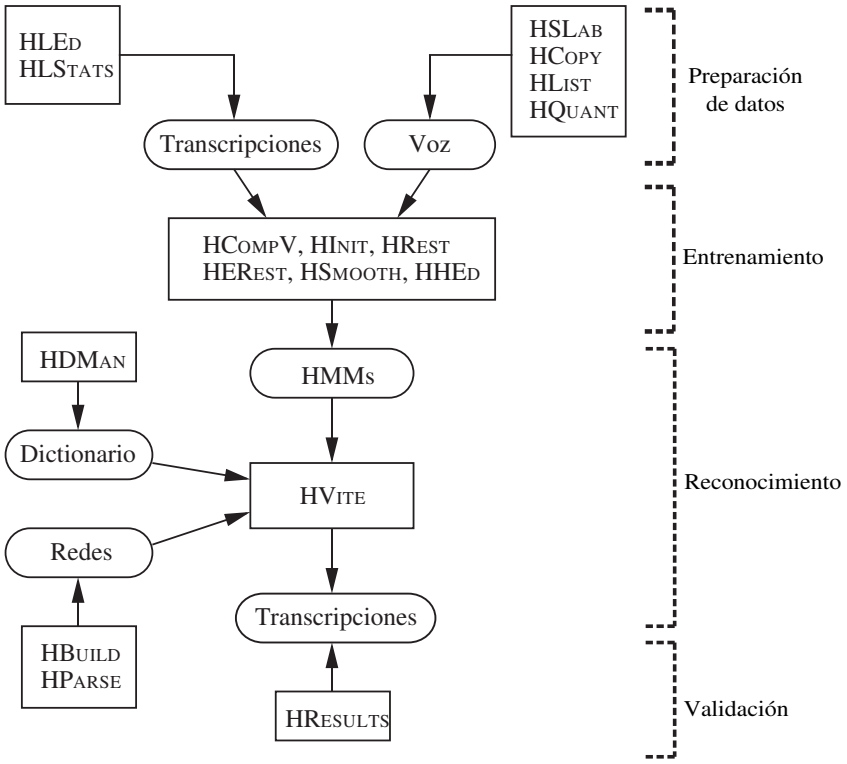

Figura 3.1: Herramientas de HTK. Adaptada del *HTKBook* [25].

- **HSLab:** Grabado y visualización de archivos de voz.
- **HCopy:** Parametrización de datos. Convierte archivos entre diversos formatos.
- **Hlist:** Visualización del contenido de los archivos.
- **HQuant:** Cuantización de los archivos de entrada.
- **HLEd:** Edición de etiquetas, realizando transformaciones en los archivos etiquetados.
- **HLStat:** Cómputo de estadísticas para el análisis de los datos de entrada.
- **HCompV:** Cálculo del promedio y la covarianza de un conjunto de datos.

- **HInit:** Estimación de parámetros iniciales del HMM.
- **HRest:** Re-estimación de parámetros con el algoritmo de *Baum-Welch*.
- **HERest:** Re-estimación de parámetros con una versión embebida del algoritmo de Baum-Welch
- **HSmooth:** Suavizado *(smoothing)* de las probabilidades discretas de los modelos.
- **HHEd:** Manipulación de definiciones de HMMs.
- **HEAdapt:** Adaptación de modelos para distintos hablantes.
- **HDMan:** Preparación de un diccionario fonético desde una o más fuentes.
- **HBuild:** Convierte archivos de entrada que representan modelos linguísticos al formato HTK.
- **HParse:** Computa archivos de sintaxis (gramática).
- **Hvite:** Reconocimiento de palabras con al algoritmo de *Viterbi*.
- **HResults:** Análisis de los resultados del sistema.

3.2. La base de datos MOCHA-TIMIT

La base de datos MOCHA-TIMIT[1] consiste en datos articulatorios y acústicos grabados en paralelo. La información articulatoria consiste en grabaciones de señales de articulografía electromagnética (EMA), electro-glotografía (EPG) y electro-palatografía. Los datos EMA son señales de 14 canales que indican las coordenadas x y y de 7 sensores posicionados en el velo, dorso, cuerpo y punta de la lengua, labios superior e inferior y el incisivo inferior [4] (ver imagen 3.2). La frecuencia de muestreo de los datos EMA es de 500 Hz, mientras que la misma para los datos acústicos es de 16 KHz.

Esta base de datos aún está en proceso de desarrollo, por lo que sólo tiene dos hablantes para los cuales los archivos han sido debidamente corregidos y se ha realizado un proceso de etiquetado de los segmentos fonéticos de manera exhaustiva y manual [23]. El hablante masculino es llamado *msak0* y el femenino *fsew0*.

[1] Alan Wrench. Queen Margaret University College

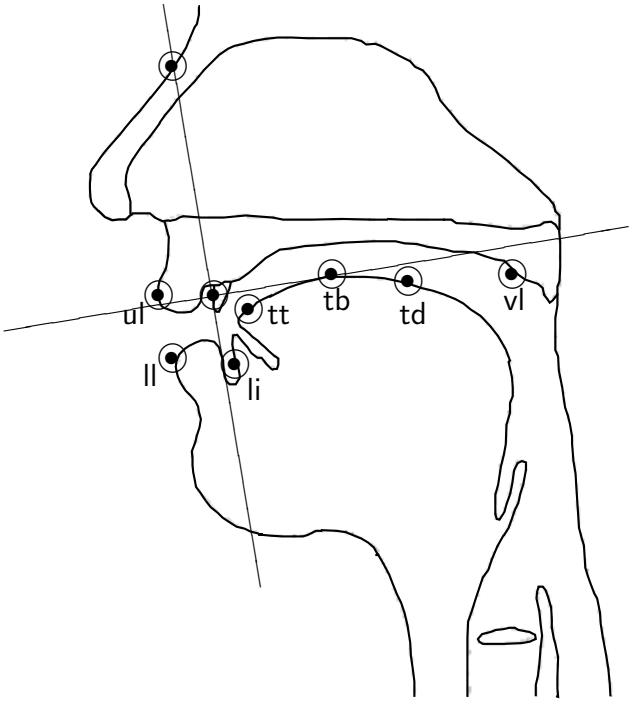

Figura 3.2: Posición de los contactos EMA en la base de datos MOCHA-TIMIT. tt, lengua (punta); tb, lengua (cuerpo); td, lengua (dorso) ; li, incisivo inferior; ll, labio inferior; ul, labio superior; vl, velo. Tomada de [28].

Las 460 frases de esta base de datos fueron diseñadas para proveer un material diverso fonéticamente, con el fin de maximizar su uso con propósitos de investigación. Se puede descargar libremente para uso no comercial accediendo al enlace http://www.cstr.ed.ac.uk/research/projects/artic/mocha.html. En el mismo sitio se pueden encontrar archivos de hablantes adicionales (*ffes0, mjjn0, fjmw0*) que incluyen etiquetas fonéticas; sin embargo los archivos de dichas etiquetas están aún sin corregir. En la sección anexa a este documento *Experimentos con hablantes adicionales* se muestran los resultados de la implementación de un sistema de reconocimiento de fonemas para dos de estos hablantes.

3.3. Desarrollo del sistema con MFCCs (Sistema de referencia)

En esta sección se explica el funcionamiento del sistema construido para el reconocimiento de fonemas usando HTK y la base de datos MOCHA-TIMIT para el entrenamiento y la validación. El proceso detallado se puede consultar en el documento anexo *Reconocimiento de Fonemas con HTK*.

3.3.1. Preparación de los datos

El sistema base a construir trabaja con una base de datos de 460 frases en forma de audio (WAV) y de articulografía electromagnética (EMA) para los hablantes. Cada segmento de voz está etiquetado en el tiempo en forma de archivos de etiqueta *(label files)*. Para el caso particular de dos hablantes, se tiene un total de 2760 archivos a procesar, los cuales deben procesarse para ser leídos por HTK de una forma práctica.

3.3.1.1. Datos de voz

Un paso previo importante para construir el sistema de reconocimiento de fonemas es preparar los archivos de audio de forma que puedan ser leídos por HTK sin inconvenientes. Existe una discrepancia entre los encabezados *(headers)* de los archivos WAV descargados de la base de datos MOCHA-TIMIT y los que pueden ser leídos por HTK, ya que esta última sólo acepta archivos de audio con encabezado en formato RIFF *(Resource Interchange File Format)*. El encabezado de los archivos de audio originales de la base de datos MOCHA-TIMIT no sigue este formato. Por este motivo se hace necesario modificar dicho encabezado haciendo uso de herramientas de edición de audio. Con este fin, en este proyecto se utilizó en un principio la herramienta Audacity, que además permitía la visualización de los archivos de audio con fines de análisis (Ver figura 3.3). Posteriormente se utilizaron los comandos de Matlab *wavread* y *wavwrite* por cuestiones de tipo práctico.

3.3.1.2. Label Files

Es conveniente unir los 460 *Label Files* de cada hablante en un sólo archivo con formato MLF *(Master Label File)* para su mejor manipulación por la herramienta HTK. Para ello el autor de este trabajo ha creado un algoritmo en Python que realiza este proceso.

Un archivo MLF típico donde se han unido los contenidos de dos *label files* es el siguiente:

```
#!MLF!#
"*/001.lab"
```

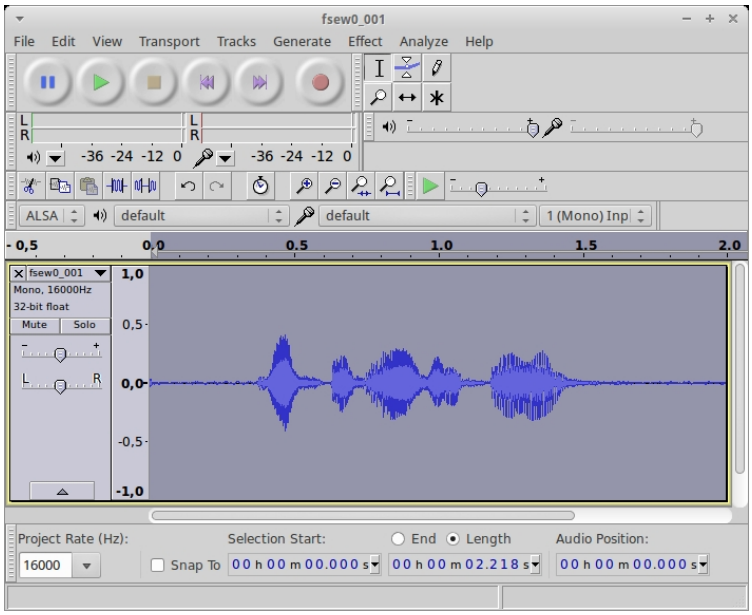

Figura 3.3: Señal de habla visualizada en Audacity.

```
0 2500000 sil
2500000 2800000 breath
2800000 3500000 sil
3500000 4200000 dh
4200000 5000000 i
6400000 6800000 @
6800000 7500000 z
7500000 9200000 ii
16700000 19100000 sil
.
"*/002.lab"
0 2800000 sil
2800000 4600000 breath
4600000 6200000 sil
6200000 6900000 i
12400000 13400000 oo
13400000 15200000 s
15200000 16900000 ei
16900000 18600000 f
18600000 22100000 sil
.
```

3.3.1.3. Creación del diccionario fonético

El presente trabajo está basado en un sistema de reconocimiento de fonemas aislados en HTK. En este caso, los modelos a entrenarse son los fonemas correspondientes a una selección de frases pronunciada por los hablantes *fsew0* y *mska0* de la base de datos MOCHA-TIMIT.

HTK necesita la lista de los modelos a entrenar. Como los modelos son los mismos fonemas, la lista de modelos es simplemente una lista de los fonemas. De igual forma, HTK requiere un diccionario fonético para hacer la correspondencia entre las palabras a entrenar y su respectivo modelo. Como en el caso del reconocimiento de fonemas las palabras son los mismos fonemas, entonces las palabras se corresponden con sí mismas.

Para la creación de los archivos de texto que contienen la lista de fonemas y el diccionario fonético, el autor de este trabajo realizó un script de Python que toma el archivo MLF de las etiquetas de los fonemas para las 460 frases y obtiene de él ambos archivos.

El archivo con la lista de fonemas tiene la siguiente forma:

```
iy
aa
@@
ch
ei
...
```

El diccionario fonético tiene la siguiente forma:

```
iy    iy
aa    aa
@@    @@
ch    ch
ei    ei
...
```

3.3.1.4. Creación de la gramática

HTK dispone de herramientas que pueden ayudar a mejorar la tasa de reconocimiento teniendo en cuenta patrones en la gramática del sistema, es decir la forma en que están organizados los modelos a reconocer, en este caso los fonemas. Se observó que todos los *label files* tienen la sintaxis representada en la figura 3.4.

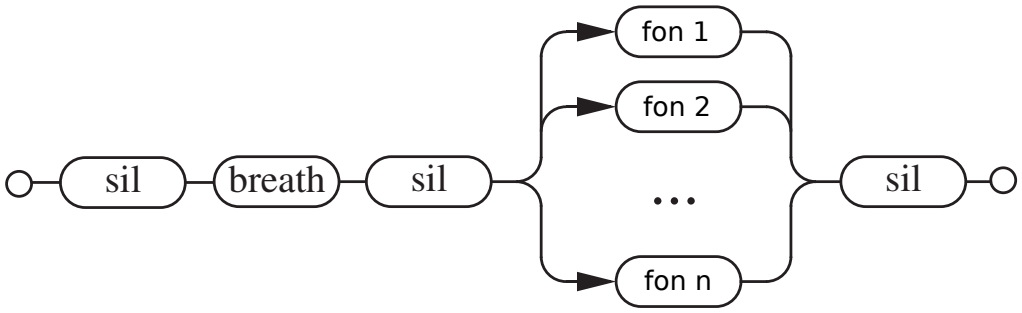

Figura 3.4: Sintaxis de los *label files.*

Debido a esta estructura presente en todos los *label files*, se puede forzar a HTK a hacer el reconocimiento siguiendo este patrón. Para ello se debe crear un archivo de gramática, que contiene lo siguiente:

```
$phone= iy |
aa | @@ | ch | ei | ai | ii | zh | p | ng | sh | th | i@ | uh | @ |
dh | oi | ow | eir | jh | a | oo | b | e | d | g | f | i | h | k |
 m | l | o | n | uu | s | r | u | t | w | v | y | ou | z;
( sil breath sil <$phone> sil )
```

A continuación se convierte el archivo anterior al formato HTK con el comando *HParse*. El archivo obtenido se utilizará en el paso reconocimiento para forzar a HTK a reconocer los fonemas siguiendo la gramática dada.

3.3.2. Obtención de los MFCCs

Para obtener los vectores característicos de la señal de voz (MFCCs) se hace uso de la herramienta *HCopy*. Esta requiere de una lista de archivos de entrada (los datos WAV) y los archivos de salida (los datos MFC) codificados en un archivo SCP *(script file*. También se necesita un archivo de configuración donde se especifiquen

las propiedades de la conversión de un tipo de dato a otro. Un esquema del proceso se puede ver en la figura 3.5. El archivo de configuración utilizado para el sistema de reconocimiento de fonemas con sólo datos de voz es el siguiente (la función de cada línea se presenta en forma de comentarios):

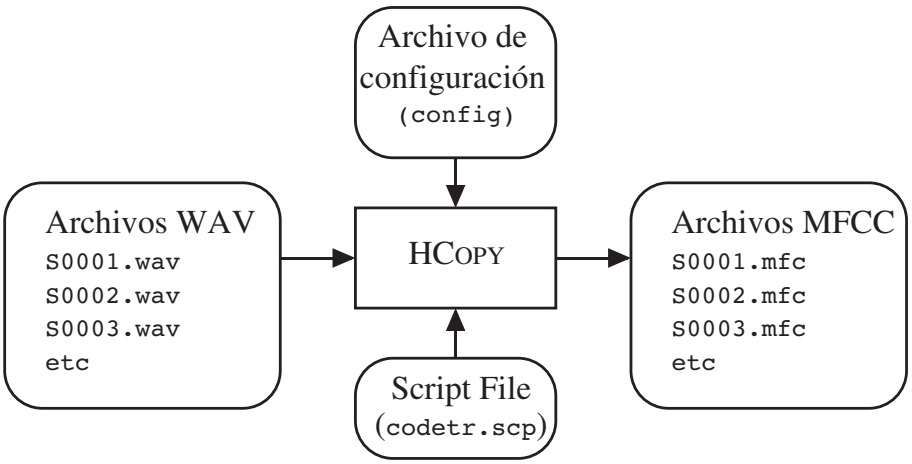

Figura 3.5: Obteniendo los vectores MFCC con HCopy. Adaptada del *HTKBook* [25].

```
SOURCEKIND = WAVEFORM      # Tipo del archivo fuente: Forma de onda
SOURCEFORMAT = WAV         # Formato de fuente: Wav
SOURCERATE = 625           # Periodo de muestreo de la entrada: 1/16 KHz
TARGETKIND = MFCC_0_D_A    # Tipo del archivo de salida: MFCC
TARGETRATE = 100000.0      # Periodo de muestreo de cada bloque: 10ms
WINDOWSIZE = 250000.0      # Tamaño de cada bloque: 25ms
USEHAMMING = T             # Se usa la ventana (Hamming window)
PREEMCOEF = 0.97           # Coeficiente del filtro de preénfasis
NUMCHANS = 20              # Número de canales
CEPLIFTER = 22             # Coeficiente del Lifter (elevador)
NUMCEPS = 12               # Número de filtros de mel a calcular
```

Aquí se puede ver que acorde con las especificaciones de los archivos de audio de la base de datos MOCHA-TIMIT, la frecuencia de muestreo del audio de entrada es de 16KHz. También se hace uso de preénfasis y *Hamming Window*. El tamaño de los bloques escogido es de 25ms, tomándose cada 10ms. Se escoge una frecuencia de 100 Hz con el fin de poder posteriormente sincronizar los MFCCs con los datos EMA. En cuanto al ancho de ventana de 25 ms, este en un valor típico en los sistemas de

reconocimiento de voz, al ser un tiempo que mantiene el balance entre un bloque casi estacionario, pero con un número de muestras suficiente.

El *TARGETKIND* indica el tipo de archivo de salida y puede tener las siguientes opciones, que se indican anteponiendo un guión bajo a cada opción después del tipo de archivo de salida, en este caso MFCC [25]:

```
A  Se agregan los coeficientes de aceleración
C  La forma externa está comprimida
D  Se agregan los coeficientes delta
E  Se agrega el logaritmo de la energía
K  Se agrega suma de verificación a la forma externa
N  Se agrega el logaritmo absoluto de la energía
T  Se agregann los coeficientes de tercer diferencial
V  Se agrega el índice VQ
Z  Se substrae la media cepstral
0  Se agrega el primer coeficiente cepstral, C0
```

En el archivo de configuración anterior: MFCC_0_D_A, al archivo de audio se le calculan los coeficientes MFCCs, teniéndose en cuenta el primer coeficiente, y los coeficientes delta y aceleración. Como el número de filtros de mel es 12, si a esto se le agrega el primer coeficiente daría 13. Con los 13 coeficientes de Mel, el cual es un valor típico [25], se calculan los coeficientes delta y aceleración para un total de 39 coeficientes.

3.3.3. Entrenamiento

3.3.3.1. Creación del modelo inicial

Para la creación de la definición del HMM inicial ese necesario crear primero un modelo prototipo. Este se almacena en un archivo de texto y su función es describir la topología del HMM, indicándose el tamaño de los vectores característicos y el número de estados escogido. Para el sistema correspondiente a este trabajo se toma un tamaño de vector de 39 y HMMs de 5 estados.

Una vez definido el modelo prototipo adecuado, el HMM se puede inicializar usándose la herramienta *HInit*. El principio básico de *HInit* depende del concepto

de un HMM como generador de vectores de habla. Cada muestra entrenada se puede ver como la salida de un HMM cuyos parámetros están por estimarse. Así, si el estado que genera cada vector de los datos de entrenamiento se conoce, las varianzas y promedios se pueden estimar promediando todos los vectores asociados a cada estado. Para ello *HInit* utiliza el algoritmo de *Viterbi* [25]. En el caso del reconocimiento de fonemas, se inicializa un modelo distinto para cada fonema con *HInit*.

Luego de inicializar los HMMs para cada fonema, se utiliza la herramienta *HRest*, cuya operación es similar a la de *HInit* sólo que ya se espera que los modelos que entrena ya estén inicializados. *HRest* calcula la probabilidad de cada estado en cada tiempo usando un algorimo *Forward-Backward*.

El siguiente paso consiste en crear un archivo de tipo MMF *(Master Macro File)* cuyo formato es similar al del MLF y contiene las definiciones de todos los modelos. Esto se hace para evitar tener un gran número de archivos de definición para dichos modelos. Para ello se utiliza la herramienta de edición de modelos de HTK, llamada *HHed*. Para más detalles acerca de la creación del modelo inicial refiérase al documento anexo a este trabajo *Reconocimiento de Fonemas con HTK*.

3.3.3.2. Reestimación de los modelos

La reestimación de los modelos se hace con la herramienta *HERest*, que utiliza el algoritmo de reestimación de *Baum-Welch* [25]. Esta toma como entradas el modelo inicial, la lista de fonemas, la transcripción de las frases a nivel fonético y la ubicación de los vectores característicos para reestimar los promedios y las varianzas de cada modelo. Luego de hacerse la reestimación por primera vez, se repite el proceso, esta vez tomando el modelo obtenido de la reestimación anterior.

3.3.3.3. Enlazando estados

En los pasos anteriores se ha generado un modelo de markov de 5 estados para cada fonema, así como un modelo para los silencios *(breath y sil)*. El siguiente paso consiste en añadir transiciones extra de los estados 2 al 4 y de los estados 4 al 2 en los modelos de los silencios. La idea es hacer el modelo más robusto permitiendo a los estados individuales absorber los varios tipos de silencio dentro de los datos de entrenamiento (ver figura 3.7).

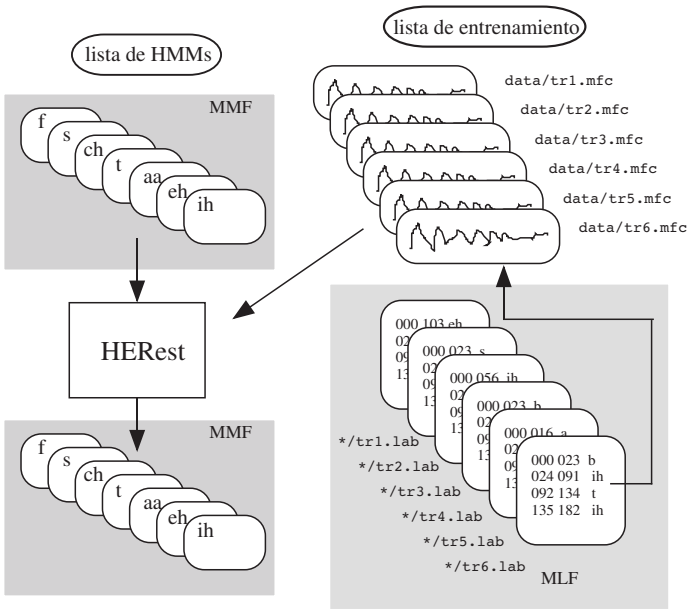

Figura 3.6: Procesamiento de archivos con HERest. Adaptada del *HTKBook* [25].

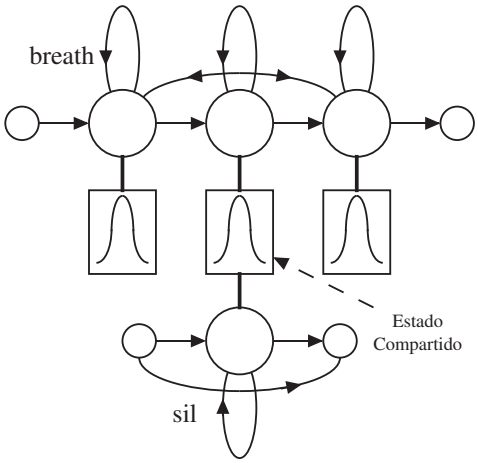

Figura 3.7: Enlazando estados en los modelos de silencio. Adaptada del *HTKBook* [25].

El enlace de estados se realiza con la herramienta *HHed* y se alterna con la reestimación de los modelos hecha por la herramienta *HHRest*. Un esquema de todo el proceso de entrenamiento de fonemas en HTK se proporciona en la figura 3.8.

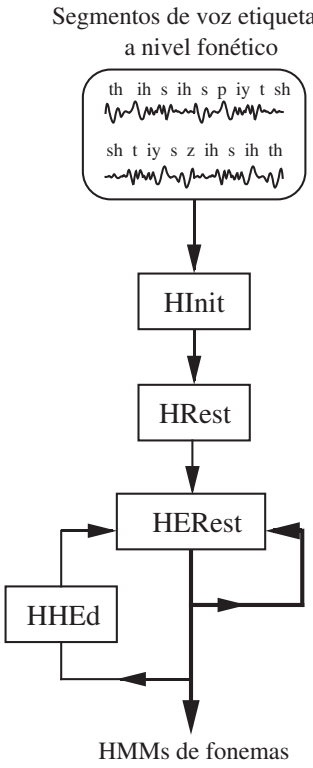

Figura 3.8: Entrenamiento de HMMs con HTK. Adaptada del *HTKBook* [25].

3.3.4. Reconocimiento

La herramienta *HVite* usa el algoritmo de *Viterbi* para el reconocimiento de los modelos definidos en los pasos anteriores. *HVite*, para el sistema diseñado en este trabajo, carga el archivo de definición de la gramática y la última definición HMM estimada para generar un archivo en formato MLF con los resultados del reconocimiento. Este archivo es el que se compara con el MLF de referencia con el fin de evaluarse cuantitativamente el rendimiento del sistema.

3.3.5. Evaluación de resultados

Para evaluar la tasa de reconocimiento del sistema se utiliza un archivo de referencia, en este caso se toma el archivo MLF que contiene los label files que vienen

etiquetados desde la base de datos del MOCHA-TIMIT. Este se compara con el MLF generado por HVite mediante el uso de la herramienta *HResults*. La salida de la herramienta *HResults* tiene esta forma:

```
====================== HTK Results Analysis =======================
  Date: Thu Jan 22 09:42:12 2015
  Ref : words.mlf
  Rec : recout.mlf
------------------------ Overall Results --------------------------
SENT: %Correct=0.00 [H=0, S=457, N=457]
WORD: %Corr=78.17, Acc=68.81 [H=12354, D=943, S=2508, I=1478, N=15805]
===================================================================
```

Como no se está haciendo reconocimiento a nivel de oraciones, la tasa de reconocimiento SENT es 0 %.

El resultado de WORD es el que interesa y se interpreta así:

- **N:** Número total de fonemas, 15805
- **H:** Número de fonemas reconocidos correctamente, 12354
- **D:** Número de errores por omisión, 934
- **S:** Número de errores por sustitución, 2508
- **I:** Número de errores por inserción, 1478
- **%Corr:** Tasa de reconocimiento ignorando los errores de inserción, H/N = 78.17 %
- **Acc:** Tasa de reconocimiento tomando los errores de inserción ,(H-I)/N 68.81 %

3.4. Añadiendo la información articulatoria

La adición de datos articulatorios al sistema descrito en las secciones anteriores supone el uso de diversas herramientas que permiten la lectura y procesado de dichos datos, y el uso de los mismos como componentes adicionales de los vectores característicos del habla.

3.4.1. Preparación de los datos articulatorios

Para el procesado de los datos articulatorios, se hace uso de un conjunto de rutinas hechas para MATLAB llamadas *Voicebox* y *EMATools*. Voicebox es una herramienta para el procesamiento de voz desarrollada principalmente por Mike Brookes del Departamento de Ingeniería Eléctrica y Electrónica del Imperial College de Londres [29] mientras que EMAtools fue desarrollada por Alan Wrench, del Centro de Investigación para las Tecnologías del habla de la Universidad de Edinburgo (se trata del mismo investigador que lideró la recopilación de la base de datos MOCHA-TIMIT) [15]. Ambos *toolboxes* están disponibles libremente bajo los términos de Licencia Pública GNU. Las rutinas utilizadas en el presente trabajo fueron:

De EMATools:

- **readest.m:** Lee los datos articulartorios.
- **sprintsi.m:** Rutina auxiliar utilizada por readest.m

De Voicebox:

- **writehtk.m:** Escribe archivos para que puedan ser leídos por HTK

Con el uso de MATLAB y un *script* que facilite la lectura de los 460 archivos EMA para cada hablante, es posible convertir la información articulatoria a un formato que HTK pueda leer. Este script debe tener en cuenta además la normalización particular que se le debe realizar a los datos EMA. Ello debido a la naturaleza misma del proceso de toma de muestras. En el CD-ROM anexo a este trabajo se pueden consultar los *scripts* utilizados para la lectura y procesado de datos EMA, en la carpeta *scripts_matlab*.

Específicamente, se codificaron los archivos EMA en formato el MFCC utilizado por HTK. Es importante aclarar que con esto no se hace el proceso de obtención de los coeficientes de Mel para los datos articulatorios, sino que tales datos se codifican directamente en el formato usado por HTK para almacenar los MFCCs, sin aplicar cambio alguno a la información relevante. Esto se hace para poder unir los datos EMA a los Coeficientes MFCC obtenidos de la señal de audio, considerando los coeficientes vectores EMA como elementos adicionales de los vectores característicos para cada bloque de voz.

3.4.2. Unión de los datos articulatorios con los MFCCs

Para unir la información articulatoria con la información de los vectores característicos obtenidos de las muestras de audio, se hace uso de un módulo de la herramienta Edinburgh Speech Tools llamado *ch_track*. Este módulo se ejecuta bajo el formato de línea de comandos y está diseñado para la manipulación de archivos de pista *(track files)*. Entre los diversos formatos que es capaz de procesar está el formato MFCC de HTK.

La unión de los datos articulatorios con los MFCCs obtenidos se realiza entonces utilizando *ch_track* configurado para unir las pistas de entrada en paralelo, siendo las entradas el archivo MFCC obtenido con HCopy y el MFCC obtenido con *writehtk.m*. Se observó que hay una discrepancia en el número de muestras de los archivos de audio y los datos EMA; ello debido a que los archivos de audio tienen algunas centésimas de segundo adicional de silencio. Esto se soluciona configurando la ejecución de *ch_track* para que limite el tamaño del archivo combinado al tamaño del archivo de entrada más pequeño.

El resutado es un conjunto de datos en formato MFCC con un tamaño de vector que corresponde a la suma de los tamaños de los vectores de entrada. En este proyecto se trabajó con 39 vectores de MFCC de audio y 14 de parámetros articulatorios, por lo que el tamaño del vector del MFCC combinado es de 53. Este número se debe tener en cuenta para definir el archivo de configuración y el modelo prototipo para la definición del modelo HMM inicial. Dicho esto, la reestimación de modelos, el enlace de estados, el reconocimiento de los modelos y el análisis de los datos se hace de la misma forma que se describió en la sección anterior.

4. Análisis de Resultados

4.1. Metodología utilizada

Con el fin de evaluar el rendimiento de los sistemas de reconocimiento de habla, y en particular de fonemas, uno de los métodos más utilizados consiste en separar los datos disponibles en un conjunto de datos de entrenamiento y otro conjunto, más pequeño, de datos de prueba. Esto se hace para tener una implementación más cercana a una situación real, donde la señal de habla puesta a prueba no siempre coincide con los datos usados en el entrenamiento [32]. Por supuesto, para hacer esta implementación debe contarse con un gran número de datos disponibles. La Base de datos MOCHA-TIMIT, con 460 frases etiquetadas en aproximandamente 16000 fonemas para dos hablantes de ambos sexos, se constituye un conjunto de datos suficiente para la evaluación del rendimiento del sistema construido en el marco de este trabajo.

Existen tres tipos de error en los sistemas de reconocimiento del habla [32]:

- **Errores por sustitución (S):** Un fonema incorrecto sustituye a uno correcto en la frase reconocida
- **Errores por omisión (D):** Un fonema correcto se omite en la frase reconocida
- **Errores por inserción (I):** Una palabra extra se agrega en la frase reconocida

Por ejemplo, si se considera la novena frase de la base de datos MOCHA-TIMIT *"Where were you while we were away?"* para el hablante femenino en el sistema combinado de MFCCs y datos EMA, se tienen dos descripciones fonéticas: la correcta que corresponde al etiquetado y la reconocida por el sistema, a la cual se

le evaluará el rendimiento. En el siguiente recuadro se hace una comparación para ambas descripciones, donde los errores en la frase reconocida se indican con sus respectivas letras.

REFERENCIA	PRUEBA (ERROR)	
sil	sil	
breath	breath	
sil	sil	
w	w	
eir	ou	(S)
	l	(I)
w	w	
@@	ou	(S)
	i	(I)
y	y	
uu	iy	(S)
w	w	
ai		(D)
l	l	
w	w	
ii	ii	
w	w	
@@	oo	(S)
r		(D)
@		(D)
w	w	
ei	ai	(S)
	iy	(I)
sil	sil	

De aquí se observa que en la frase reconocida hay 5 errores por sustitución, 3 por omisión y 3 por inserción.

Una de las formas de evaluación más ampliamente usadas para los sistemas de reconocimiento de fonemas es la tasa de error fonético (PER por sus siglas en inglés) [3] [32]. Esta mide la diferencia entre la secuencia de fonemas reconocidos con la secuencia correcta y se calcula sumando el total de errores sobre el número

de fonemas de la secuencia correcta (N). Lo anterior se expresa con la fórmula

$$PER = 100 * \frac{I + S + D}{N} \tag{4.1}$$

La precisión del sistema (A por *accuracy*) de reconocimiento se computa como:

$$A = 100 - PER \tag{4.2}$$

Existe otra medida que sólo tiene en cuenta el número de fonemas correctos identificados por el reconocedor. Esta se calcula sin tenerse en cuenta los errores por inserción, de la siguiente forma:

$$C = 100 * \frac{N - S - D}{N} \tag{4.3}$$

Para la serie de fonemas reconocidos del ejemplo anterior, la PER es del 52.38 %, la precisión es del 47.62 % y el porcentaje de fonemas correcto es del 61.90 %.

Se plantea hacer un análisis de 4 sistemas desarrollados: con y sin datos articulatorios para ambos hablantes, masculino y femenino. El indicador a usar para evaluar los sistemas construidos es la precisión (A). Los datos de entrenamiento usados corresponden al 80 % del total de frases y los datos de validación corresponden al 20 % restante, tal como se hizo en el sistema implementado por [31]. Las frases específicas de cada conjunto de datos es elegida de manera aleatoria. En total se harán 15 entrenamientos para cada sistema.

La hipótesis que se plantea es que la precisión mejora en los sistemas que hacen uso de los parámetros articulatorios. Para ello basta con demostrar que los promedios de precisión están lo suficientemente alejados entre sí, si se asume una varianza pequeña. El test de hipótesis se hace con una prueba paramétrica que compara dos muestras, en este caso dos vectores de 15 elementos por cada hablante que contienen las tasas de precisión para los dos tipos de sistema. La prueba paramétrica a utilizarse es la función *ttest* de Matlab.

Si la función retorna el valor 0 al tener como entradas los dos vectores de prueba, se asume que los datos en estos vectores provienen de muestras aleatorias de una distribución normal con el mismo promedio. En tal caso la hipótesis planteada no

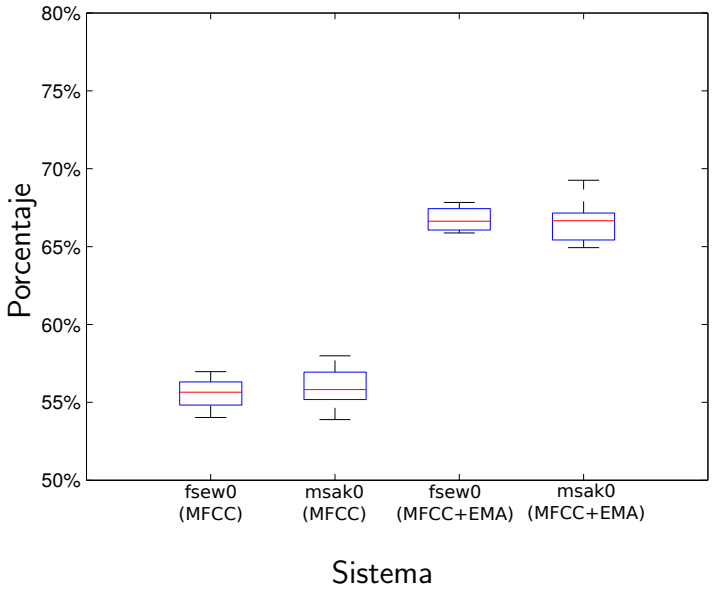

Figura 4.1: Boxplot de la tasa de precisión (A) para los cuatro sistemas.

se cumpliría. Por otro lado, si la función retorna el valor 1, se asume que los dos vectores de entrada provienen de poblaciones con promedios distintos. En este caso se cumple la hipótesis. Esta prueba se aplica una vez por cada hablante.

4.2. Resultados

Las tasas globales de precisión (%A) y porcentajes de fonemas correcto (%C) para los dos hablantes se indican en las tablas 4.1 y 4.2.

Las mejoras en las tasas de reconocimiento se calculan se pueden observar en la tabla 4.3.

El test de hipótesis se cumple ya que la tasa de precisión promedio aumenta al agregarse los parámetros articulatorios para ambos hablantes. Al aplicarse la función *ttest* a los conjuntos de datos, se obtiene el valor esperado de 1, que indica que las medias están lo suficientemente alejadas.

La mayor tasa de precisión obtenida fue de un 69.26 % para la segunda prueba del sistema combinado (hablante *msak0*).

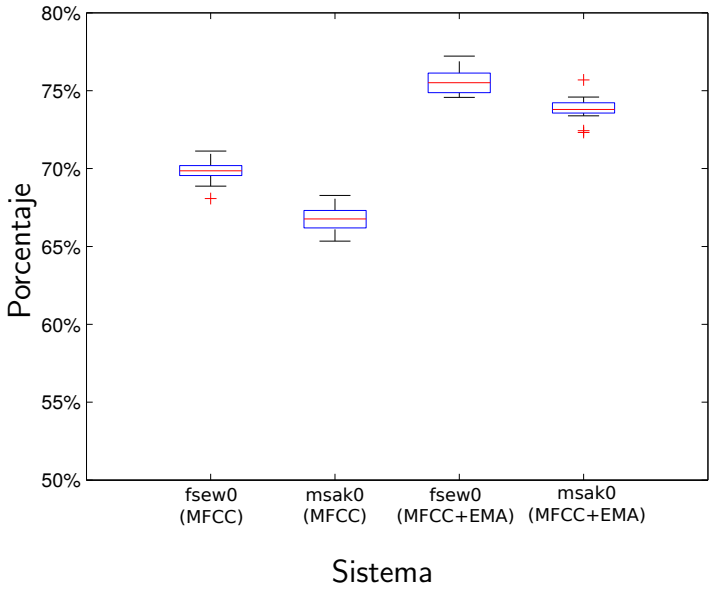

Figura 4.2: Boxplot del porcentaje de fonemas correctos sobre el total (C) para los cuatro sistemas.

	fsew0 (wav)		**fsew0 (combinado)**	
Ent.	**A**	**C**	**A**	**C**
1	54.77 %	69.43 %	67.26 %	76.20 %
2	55.65 %	69.62 %	67.83 %	75.50 %
3	56.93 %	69.79 %	65.88 %	74.57 %
4	55.65 %	70.07 %	66.26 %	75.91 %
5	54.27 %	70.21 %	66.63 %	74.73 %
6	54.97 %	69.96 %	66.05 %	74.86 %
7	56.61 %	70.77 %	66.31 %	75.67 %
8	54.28 %	68.87 %	67.49 %	74.92 %
9	55.74 %	70.67 %	67.69 %	77.22 %
10	54.03 %	68.07 %	65.95 %	75.51 %
11	56.37 %	69.83 %	67.68 %	76.24 %
12	56.10 %	69.86 %	65.87 %	74.83 %
13	56.97 %	71.12 %	67.13 %	75.78 %
14	55.14 %	69.53 %	67.03 %	76.23 %
15	56.13 %	70.13 %	66.12 %	75.43 %
PROM	**55.57 %**	**69.86 %**	**66.75 %**	**75.57 %**
STD	0.9665 %	0.7494 %	0.7295 %	0.7261 %

Cuadro 4.1: Resultados para el hablante femenino *(fsew0)*

	msak0 (wav)		**msak0 (combinado)**	
Ent.	**A**	**C**	**A**	**C**
1	54.28 %	66.17 %	66.52 %	74.19 %
2	57.36 %	68.27 %	69.26 %	75.69 %
3	57.01 %	66.52 %	65.11 %	73.58 %
4	53.89 %	65.34 %	67.97 %	74.15 %
5	57.60 %	67.83 %	64.93 %	72.32 %
6	56.28 %	67.05 %	65.99 %	73.79 %
7	55.24 %	66.13 %	66.88 %	73.67 %
8	56.29 %	67.23 %	65.37 %	72.43 %
9	55.75 %	66.23 %	67.00 %	73.87 %
10	56.74 %	67.62 %	65.10 %	73.39 %
11	57.99 %	67.34 %	65.58 %	73.56 %
12	55.56 %	66.81 %	67.21 %	74.23 %
13	55.82 %	66.16 %	67.66 %	74.59 %
14	55.12 %	66.77 %	66.66 %	73.64 %
15	55.16 %	66.53 %	66.75 %	74.40 %
PROM	**56.01 %**	**66.80 %**	**66.53 %**	**73.83 %**
STD	1.1909 %	0.7699 %	1.2199 %	0.8189 %

Cuadro 4.2: Resultados para el hablante masculino *(msak0)*

	fsew0	**msak0**
A	11.17 %	10.53 %
C	5.71 %	7.03 %

Cuadro 4.3: Mejoras en las tasas de precisión y porcentaje correcto de fonemas para ambos hablantes

5. Conclusiones

Se logró construir un sistema de reconocimiento de fonemas basado en MFCCs, con el uso de las herramientas de HTK, que presenta una precisión de aproximadamente 55.6 % para el hablante femenino y 56 % para el masculino. Tal sistema utilizó 39 coeficientes cepstrales y modelos de Markov de 5 estados.

Adicionalmente, se logró añadir la información articulatoria a los vectores característicos de la señal de voz obtenidos para el caso anterior e implementarse un sistema combinado que presentó una precisión de aproximadamente 66.8 % para ambos hablantes. La tasa de precisión mejoró un 11.1 % para el hablante femenino y un 10.5 % para el masculino. En el marco de este trabajo se generaron scripts de UNIX *(shell scripts)*, Python y Matlab que pueden ser de utilidad para experimentos futuros en el campo del reconocimiento de fonemas.

A partir de la evaluación del desempeño de los sistemas de reconocimiento desarrollados, se puede inferir que los datos articulatorios contienen información útil que adicionada a las señales acústicas parametrizadas pueden utilizarse para mejorar el rendimiento de los sistemas de reconocimiento de fonemas.

Referencias

[1] A. Mohamed, G. D. y. G. H. Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing* (2011). 13, 18

[2] A. Oppenheim, R. S. *Tratamiento de señales en tiempo discreto.* Pearson Education, 2011. 24, 25

[3] Antal, M. Toward a simple phoneme based speech recognition system. *Studia Universitatis Babes, Bolyai, Informatica 2* (2007), 41. 49

[4] E. Uraga, T. H. Automatic speech recognition experiments with articulatory data. *Interspeech* (2006). 34

[5] Ejnarsson, M. N. . M. Speech recognition using hidden markov models. Master's thesis, Department of Telecommunications and Signal Processing, Blekinge Institute of Technology, 2002. 15, 16, 27

[6] Fujihara, H. . G. M. Three techniques for improving automatic synchronization between music and lyrics: Fricative detection, filler model, and novel feature vectors for vocal activity detection. *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (2008). 14

[7] Furui, S. 50 years of progress in speech and speaker recognition research. *Proceedings of ECTI Transactions on Computer and Information Technology* (2005). 13

[8] HTK. Página web principal [en línea]. *http://htk.eng.cam.ac.uk/*. 19, 32

[9] International Phonetic Association (IPA). Página web principal [en línea]. *https://www.internationalphoneticassociation.org/*. 17

[10] J. Baker, L. D. . O. D. M. Research developments and directions in speech recognition and understanding. *IEEE Signal Processing Magazine 26* (2009), 75–80. 13

[11] Kent, R. *Acoustic analysis of speech*. Singular, 2002. 14, 16, 17

[12] Lee, K. . H. H. Speaker-independent phone recognition using hidden markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing* (1989), 1642–1648. 13, 18

[13] Li, J. *Soft Margin Estimation for Automatic Speech Recognition*. PhD thesis, 2008. Georgia Institute of Technology, School of Electrical and Computer Engineering. 13

[14] M. Gales, S. Y. The application of hidden markov models in speech recognition. *Foundations and Trends in Signal Processing 1* (2007). 28, 30

[15] MOCHA-TIMIT Database. Página web principal [en línea]. *http://www.cstr.ed.ac.uk/research/projects/artic/mocha.html*. 14, 46

[16] Morris, J. . F.-L. E. Conditional random fields for integrating local discriminative classifiers. *IEEE Transactions on Audio, Speech, and Language Processing* (2008). 14

[17] Musiek, F. E. *The Auditory System: Anatomy, Physiology, and Clinical Correlates*. Paperback, 2006. 15

[18] O. Ichikawa, T. Fukuda, M. N. Dynamic features in the linear-logarithmic hybrid domain for automatic speech recognition in a reverberant environment. *IEEE Journal of Selected Topics in Signal Processing 4* (2010). 27

[19] O'Shaughnessy, D. Automatic speech recognition: History, methods and challenges. *Pattern Recognition* (2008). 18

[20] P. Schwarz, P. M. . J. C. Hierarchical structures of neural networks for phone recognition. *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing* (2006). 18

[21] R. Lawrence, J. B.-H. *Fundamentals of Speech Recognition*. Prentice Hall, 1993. 20

[22] Robinson, T. An application of recurrent nets to phone probability estimation. *IEEE Transactions on Neural Networks* (1994). 18

[23] RSingampalli, P. J. . V. Statistical identification of articulation constraints in the production of speech. *Speech Communication 51* (2009). 34

[24] S. Davis, P. M. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing 28* (1980). 24

[25] S. Young, G. Evermann, M. G. *The HTK Book. Revised for HTK Version 3.4.* Cambridge University Engineering Department, 2006. 8, 29, 30, 33, 40, 41, 42, 43, 44

[26] Schafer, R. *Homomorphic Systems and Cepstrum Analysis of Speech.* Springer, 2008. 27

[27] Schwarz, P. *Phone recognition based on long temporal context.* PhD thesis, 2008. Brno University of Technology, Faculty of Information Technology. 14

[28] Sepúlveda, F. *Estimation of Articulatory Parameters from the Acoustic Speech Signal.* PhD thesis, 2012. Departamento de Ingenierías Eléctrica, Electrónica y Computación. Universidad Nacional de Colombia. 8, 15, 16, 35

[29] Voicebok Toolkit. Página web principal [en línea]. *http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html*. 46

[30] Witt, S. M. . Y. S. J. Phone-level pronunciation scoring and assessment for interactive language learning. *Speech communication* (2000), 95–108. 13

[31] Wrench, A., and Richmond, K. Continuous speech recognition using articulatory data. 50

[32] X. Huang, A. Acero, . H. H. *Spoken Language Processing: A guide to theory, algorithm, and system development.* Prentice Hall, 2011. 23, 24, 26, 27, 48, 49

[33] Zissman, M. Comparison of :four approaches to automatic language identification of telephone speech. *lEEE Transactions On Speech And Audio Processing* (1996). 14

Anexo A: Experimentos con hablantes adicionales

Si bien las etiquetas fonéticas para los hablantes *fjmw0* y *mjjn0* no han sido corregidas aún, se realizó la implementación del sistema de reconocimiento de fonemas para estos hablantes. Los resultados muestran unos indicadores de rendimiento apropiados para el hablante *mjjn0*, mientras que los mismos son bajos para el hablante *fjmw0*. Los resultados pueden analizarse en las siguientes tablas.

	fjmw0 (wav)		**fjmw0 (combinado)**	
Ent.	**A**	**C**	**A**	**C**
1	41.71 %	59.29 %	42.03 %	49.88 %
2	41.92 %	58.31 %	39.51 %	47.71 %
3	39.67 %	57.80 %	41.24 %	51.30 %
4	41.21 %	59.69 %	39.92 %	49.50 %
5	40.53 %	56.16 %	39.26 %	49.07 %
6	41.05 %	59.09 %	41.22 %	49.65 %
7	43.00 %	59.21 %	38.84 %	47.44 %
8	39.86 %	56.46 %	39.37 %	47.06 %
9	38.87 %	54.93 %	40.35 %	48.84 %
10	39.91 %	57.65 %	36.03 %	44.86 %
11	41.80 %	58.14 %	39.02 %	48.69 %
12	40.90 %	57.82 %	38.01 %	47.60 %
13	38.94 %	57.90 %	40.50 %	48.83 %
14	41.16 %	60.11 %	39.86 %	54.86 %
15	41.77 %	57.08 %	40.78 %	50.16 %
PROM	**40.82 %**	**57.82 %**	**39.73 %**	**49.03 %**
VAR	1.1783 %	1.4133 %	1.4703 %	2.2285 %

Cuadro 1: Resultados para el hablante *(fjmw0)*

	mjjn0 (wav)		**mjjn0 (combinado)**	
Ent.	**A**	**C**	**A**	**C**
1	62.32 %	73.12 %	65.85 %	73.99 %
2	62.40 %	73.34 %	66.81 %	74.52 %
3	61.67 %	72.88 %	65.42 %	73.86 %
4	62.13 %	72.81 %	64.35 %	72.83 %
5	63.64 %	73.38 %	66.81 %	75.27 %
6	63.61 %	73.28 %	64.95 %	73.73 %
7	62.02 %	73.08 %	64.44 %	73.57 %
8	60.34 %	72.71 %	65.87 %	74.37 %
9	62.94 %	72.32 %	65.76 %	74.69 %
10	62.30 %	72.84 %	64.18 %	73.28 %
11	61.84 %	72.77 %	65.80 %	74.01 %
12	62.65 %	72.84 %	66.48 %	74.71 %
13	63.62 %	73.63 %	66.02 %	74.07 %
14	61.07 %	72.45 %	65.54 %	73.19 %
15	62.47 %	73.05 %	65.34 %	74.20 %
PROM	**62.33 %**	**72.97 %**	**65.57 %**	**74.02 %**
STD	0.9211 %	0.3531 %	0.8247 %	0.6470 %

Cuadro 2: Resultados para el hablante *(mjjn0)*

	fjmw0	**mjjn0**
A	-1.09 %	3.24 %
C	-8.79 %	1.05 %

Cuadro 3: Mejoras en las tasas de precisión y porcentaje correcto de fonemas para ambos hablantes

Anexo B: Guía para el reconocimiento de fonemas con HTK

Proceso para construir un sistema de reconocimiento de fonemas con HTK:

Probado en Ubuntu 14.10. 32 bits

Los códigos citados se encuentran en el enlace de GitHub:
https://github.com/albertigno/Phone-Recognition-Mocha-TImit-Ema

PARTE I: DESARROLLO DE UN SISTEMA DE RECONOCIMIENTO DE FONEMAS EN HTK (SÓLO AUDIO)

1. INSTALACIÓN PROGRAMAS

1.1 Programas que se asumen instalados

Python, Matlab

1.2 HTK

Para instalar HTK en linux hay que registrarse en la página oficial y descargar el paquete.

Registro: http://htk.eng.cam.ac.uk/register.shtml
Descarga (Versión 3.4.1): http://htk.eng.cam.ac.uk/ftp/software/HTK-3.4.1.tar.gz

El paquete se descomprime en la ubicación que el usuario desee, y se abre una

terminal en esa ubicación, en la carpeta htk. Se ejecutan los siguientes comandos:

./configure
make all
sudo make install

Nota 1: Esto instalará los programas en la ubicación por defecto, "/usr/local/bin"

Si se quiere instalar en otra ubicación, el primer comando cambia:

./configure --prefix=/path/to/your/installation

donde "/path/to/your/installation" es la carpeta de instalacion definida por el usuario.

Nota 2: En caso de que se trate de un computador de 64 bits, el orden de los comandos sería:

linux32 bash
sudo apt-get install libx11-dev
./configure
make all
sudo make install

Prueba

Para probar si funciona se puede ejecutar cualquiera de los programas de HTK, por ejemplo:

Hinit

Una prueba más elaborada consiste en desargar los ejemplos de la página oficial.

ejemplos: http://htk.eng.cam.ac.uk/ftp/software/HTK-samples-3.4.1.tar.gz

Se descomprime en la ubicación deseada y se abre una terminal en la carpeta HTKDemo

Luego se ejecuta el siguiente comando, que realiza un test de reconocimiento

./runDemo configs/monPlainM1S1.dcf

los resultados deberían coincidir con lo siguiente:

On the training set:
---------- Overall Results ----------
SENT: %Correct=0.00 [H=0, S=7, N=7]
WORD: %Corr=77.63, Acc=74.89 [H=170, D=37, S=12, I=6, N=219]
==========

On the test set:
---------- Overall Results ----------
SENT: %Correct=0.00 [H=0, S=3, N=3]
WORD: %Corr=63.91, Acc=59.40 [H=85, D=35, S=13, I=6, N=133]
==========

1.3 Rutinas de Matlab (Tomadas del paquete voicebox)

Para leer y procesar los datos articulatorios se hace uso de un conjunto de rutinas de Matlab (readest.m, sprintsi.m, writehtk.m), que se pueden descargar en el enlace de gitHub.

1.4 Edinburgh Speech Tools

Prerrequisitos: tener instalados gcc (compilador de C) y g++ (compilador de C++).

Se ejecuta en un terminal:

sudo apt-get install speech-tools

2. PREPARACIÓN DE LOS DATOS

2.1 Descarga Bases de datos

Descarga de la base de datos MOCHA-TIMIT. Ir a:
http://data.cstr.ed.ac.uk/mocha/

Y descargar los siguientes archivos:
msak0_v1.1.tar.gz (Hablante masculino)
fsew0_v1.1.tar.gz (Hablante femenino)
mocha-timit.txt Transcripción de las 460 frases

Descarga de los label files. Ir a:
http://personal.ee.surrey.ac.uk/Personal/P.Jackson/Dansa/Mocha/
Y descargar:
msak0 (label files hablante masculino)
fsew0 (label files hablante femenino)

Descomprimir el contenido de los tar.gz.

Organizar los archivos en carpetas para cada hablante: Se necesitan los .wav (audios), .lb (label files), .ema (datos articulatorios).

2.2 Preparación de los archivos .wav y los datos articulatorios

El encabezado (*header*) de los archivos de audio descargados no puede ser leído directamente por HTK. De igual forma, HTK no es capaz de leer los datos EMA directamente. Por ello se debe hacer un proceso previo para convertir los archivos de la base de datos a un formato aceptable para HTK.

El programa principal para procesar los datos es "leer.m". Antes de ejecutarse se deben configurar las ubicaciones de los archivos de la base de datos, ya que esto varía según el usuario. Esto se hace modificando las siguientes líneas de código:

```
% Carpeta donde se ubican la base de datos MOCHA_timit
MOCHA_folder = '/home/alberto/Documents/HTK/Emas/';

% carpeta donde se ubican las rutinas
root_folder =
'/home/alberto/Desktop/HTK/Matlab_Routines/Read_EMA_Data';

% carpeta donde se desea guardar los datos ema convertidos a formato mfc
MFC_folder = '/home/alberto/Documents/HTK/Matlab_MFCC/msak0/';

% carpeta donde se desea guardar los archivos de audio
WAV_folder = '/home/alberto/Documents/HTK/Audios/msak0/';

% Seleccionar hablante
speaker = 'msak0';
```

Nota 3: El directorio de la base de datos MOCHA-timit debe contener las carpetas /fsew0 y /msak0 que a su vez deben contener los 460 archivos .ema, .wav y .lb.

Una vez configurado lo anterior, se ejecuta el programa leer.m en matlab. Esto genera los archivos ema convertidos al formato MFCC y los archivos wav con el encabezado (header) corregido, de forma que pueden ya ser procesados directamente por HTK.

2.3 Master Label Files

Es conveniente unir los 460 label files de cada hablante en un sólo archivo con formato .mlf (master label file) para su mejor manipulación por la herramienta HTK. Para ello el autor de este trabajo ha creado un algoritmo en Python que hace este proceso. El script se puede descargar como *"lb2mlf.py"* en el enlace de GitHub.

La primera línea del código indica el tipo de archivo que se va a manipular:
Debe modificarse de acuerdo al hablante: *"fsew0_ "* si es femenino o *"mska0_ "* si es masculino.

Para hacer este proceso debe colocarse el script de python en la carpeta donde están ubicados los label files (formato .lb), abrir la ventana de comandos en esa carpeta y ejecutar:

python lb2mlf.py

Esto crea el archivo '*words.mlf*' dentro de la misma carpeta de ejecución. Tal archivo contiene todos los label files en un formato que puede ser leído por HTK.

2.4 Listas de entrenamiento y prueba

De las 460 frases de la base MOCHA-TIMIT, un 80% se usarán para entrenamiento y el 20% para prueba, escogidas al azar y sin repetirse ninguna frase entre una lista y otra.

Ambas listas se dan en formato SCP (script file) y se crean con el script de python *list_ gen.py.* Al abrirse la carpeta donde esta ubicado este script, si se

ejecuta:

python lb2mlf.py

Se crearán dos archivos: train.scp (lista de entrenamiento) y test.scp (lista de prueba), los cuales tienen la siguiente forma:

./mfccs/008.mfc
./mfccs/310.mfc
./mfccs/205.mfc
./mfccs/357.mfc
./mfccs/377.mfc
... etc

3. OBTENER LOS VECTORES CARACTERÍSTICOS

A partir de ahora se hará realizará la guía con el hablante femenino fsew0.
Con esto se convierten los archivos wav al formato mfcc

3.1 Crear el archivo de configuración

Dentro de la carpeta donde se desarrollará el programa se crea el archivo de configuración, en esta guía lo llamamos 'config', de la siguiente forma:

```
SOURCEKIND = WAVEFORM      # Tipo del archivo fuente: Forma de onda
SOURCEFORMAT = WAV         # Formato de fuente: Wav
SOURCERATE = 625           # Periodo de muestreo de la entrada: 1/16
                           # KHz
TARGETKIND = MFCC_0_D_A    # Tipo del archivo de salida: MFCC
TARGETRATE = 100000.0      # Periodo de muestreo de cada bloque: 10ms
WINDOWSIZE = 250000.0      # Tamaño de cada bloque: 25ms
USEHAMMING = T             # Se usa la ventana (Hamming window)
PREEMCOEF = 0.97           # Coeficiente del filtro de preénfasis
NUMCHANS = 20              # Número de canales
CEPLIFTER = 22             # Coeficiente del Lifter
NUMCEPS = 12               # Número de filtros de mel a calcular
```

Observaciones:

Cada unidad de tiempo en el archivo de configuración equivale a 100ns =1e-7s.

Por ejemplo la tasa de muestreo de los archivos de audio es 1/16KHz = 625e-7 s = 625 unidades.

El targetkind puede tener las siguientes opciones:

A	Se agregan los coeficientes de aceleración
C	La forma externa está comprimida
D	Se agregan los coeficientes delta
E	Se agrega el logaritmo de la energía
K	Se agrega suma de verificación a la forma externa
N	Se agrega el logaritmo absoluto de la energía
T	Se agregann los coeficientes de tercer diferencial
V	Se agrega el índice VQ
Z	Se substrae la media cepstral
0	Se agrega el primer coeficiente cepstral, C0

En el archivo config anterior: MFCC_0_D_A, al archivo de audio se le calculan los coeficientes MFCC, teniéndose en cuenta el primer coeficiente, y los coeficientes delta y aceleración.

3.2 Crear la lista de ubicación de los archivos

Los archivos de texto que contienen listas de archivos son denominados *script files* y por convención tienen formato scp. Es necesario crear uno archivo scp para indicarle a HTK la ubicación de todos los archivos de entrada (.wav) y los de salida (.mfc). Tal archivo se ha llamado codetrain.scp y tiene la siguiente forma (primeras cinco líneas y linea final):

```
./sounds/fsew0_001.wav ./mfccs/001.mfc
./sounds/fsew0_002.wav ./mfccs/002.mfc
./sounds/fsew0_003.wav ./mfccs/003.mfc
./sounds/fsew0_004.wav ./mfccs/004.mfc
./sounds/fsew0_005.wav ./mfccs/005.mfc
...
./sounds/fsew0_460.wav ./mfccs/460.mfc
```

Observaciones: En este caso, la carpeta donde están ubicados los archivos de audio es /sounds y la carpeta donde estarán los archivos de salida es /mfccs. El programa que realizará la transformación, HCopy, buscará estas ubicaciones dentro de la carpeta madre, a la que llamamos en esta guía /HTK.

3.2 Ejecutar HCopy

El programa HCopy copia uno o más archivos a una salida designada, con la opción de poderse convertir los datos a una forma parametrizada (para ello se utiliza el archivo de configuración).

Opciones utilizadas:

-T *Opción de trazo (muestra el proceso desarrollado en la consola)*
-C *define el archivo de configuración*
-S *define el scrip file o la serie de archivos de entrada y de salida*

Se ejecuta el siguiente comando:

HCopy -T 1 -C config -S codetrain.scp

Debe aparecer en la consola:

```
./sounds/fsew0_001.wav -> ./mfccs/001.mfc
./sounds/fsew0_002.wav -> ./mfccs/002.mfc
./sounds/fsew0_003.wav -> ./mfccs/003.mfc
./sounds/fsew0_004.wav -> ./mfccs/004.mfc
./sounds/fsew0_005.wav -> ./mfccs/005.mfc
...
./sounds/fsew0_460.wav -> ./mfccs/460.mfc
```

Con esto los vectores característicos contenidos en los archivos .mfc están creados en la carpeta /mfccs.

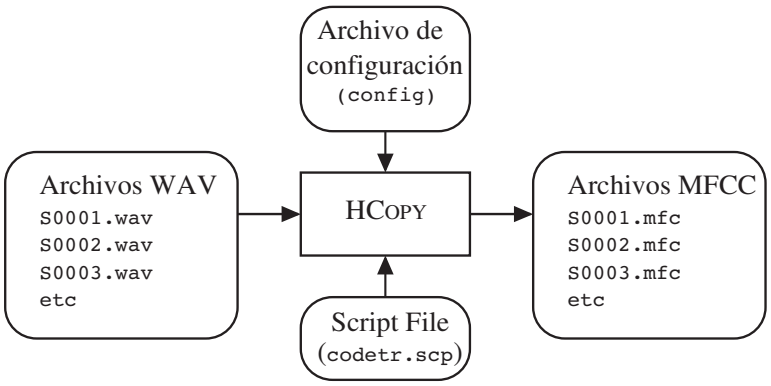

Imagen 1. Uso de HCopy

4. ENTRENAMIENTO

4.1 Creación de la lista de fonemas y el diccionario

Antes de proceder con este paso es conveniente crear las carpetas donde se alojarán los modelos de markov que se van a entrenar. Estas van desde /hmm0 hasta /hmm24. En total 25 carpetas.

HTK necesita la lista de los modelos a entrenar. Los modelos son los fonemas, por tanto la lista de modelos es simplemente una lista de los fonemas.

De igual forma, HTK requiere un diccionario fonético, para hacer la correspondencia entre las palabras a entrenar y su respectivo modelo. Como en el caso del reconocimiento de fonemas las palabras son los mismos fonemas, entonces las palabras se corresponden con sí mismas.

Para la creación de la lista de fonemas y el diccionario, el autor de este trabajo hizo un script de python que toma el archivo words.mlf (el master label file) y obtiene de él la lista de fonemas. El nombre del script es *"mlf2phones.py"* y se puede descargar en el link de gitHub. Para ejecutarlo debe ubicarse en la misma carpeta donde esté words.mlf, abrirse una perminal y ejecutarse:

python mlf2phones.py

Esto crea los archivos *monophones0* y *dictionary.*

El archivo monophones0 tiene la siguiente forma:

iy
aa
@@
ch
ei
...

El archivo dictionary tiene la siguiente forma:

iy	iy
aa	aa
@@	@@
ch	ch
ei	ei

...

Ambos archivos deben guardarse en la carpeta raíz donde se está realizando el sistema, en este caso /HTK.

4.2 Creación del modelo inicial

Para la creación de la definición del HMM inicial ese necesario crear primero un modelo prototipo. Este se almacena en un archivo de texto llamado 'proto' y su función es describir la topología del HMM, indicándose el tamaño de los vectores característicos y el número de estados escogido.

La longitud del vector establecida es de 39 debido a que los vectores parametrizados son 13 y a estos se añaden los 13 coeficientes delta y los 13 de aceleración. El archivo proto debe crearse en la carpeta principal del entrenamiento y contiene lo siguiente:

```
~o <VecSize> 39 <MFCC_0_D_A>
~h "proto"
<BeginHMM>
  <NumStates> 5
  <State> 2
    <Mean> 39
      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    <Variance> 39
      1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
  <State> 3
    <Mean> 39
      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    <Variance> 39
      1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
  <State> 4
    <Mean> 39
      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    <Variance> 39
      1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
  <TransP> 5
  0.0 1.0 0.0 0.0 0.0
  0.0 0.6 0.4 0.0 0.0
  0.0 0.0 0.6 0.4 0.0
  0.0 0.0 0.0 0.7 0.3
  0.0 0.0 0.0 0.0 0.0
<EndHMM>
```

También se necesita un segundo archivo de configuración, que se llamará *"config2"* que contiene lo siguiente:

```
TARGETKIND = MFCC_0_D_A
TARGETRATE = 100000.0
WINDOWSIZE = 250000.0
USEHAMMING = T
PREEMCOEF = 0.97
NUMCHANS = 20
CEPLIFTER = 22
NUMCEPS = 12
```

Una vez definido el modelo prototipo, el HMM se puede inicializar usándose la herramienta Hinit, inicializándose un modelo distinto para cada fonema:

```
HInit -l iy -o iy -v 0.0001 -C config2 -S train.scp -I words.mlf -M hmm0  proto
HInit -l aa -o aa -v 0.0001 -C config2 -S train.scp -I words.mlf -M hmm0  proto
HInit -l th -o th -v 0.0001 -C config2 -S train.scp -I words.mlf -M hmm0  proto
... etc
```

Opciones utilizadas:

-l *nombre del modelo*
-o *nombre del archivo de salida*
-v *varianza mínima*
-C *define el archivo de configuración*
-S *define el scrip file o la serie de archivos de entrada y de salida*
-I *carga del archivo MLF*
-M *directorio de salida del HMM*

Luego de inicializar los HMMs para cada fonema, se utiliza la herramienta HRest, cuya operación es similar a la de HInit sólo que ya se espera que los modelos que entrena ya estén inicializados.

```
HRest -S train.scp -H hmm0/iy -M hmm1 -l iy -I words.mlf hmm0/iy
HRest -S train.scp -H hmm0/aa -M hmm1 -l aa -I words.mlf hmm0/aa
HRest -S train.scp -H hmm0/th -M hmm1 -l th -I words.mlf hmm0/th
... etc
```

Opciones utilizadas:

-l *nombre del modelo*

-I *carga del archivo MLF*
-H Ubicación de la definición del HMM de entrada
-M *directorio de salida del HMM*

Dados los modelos de cada fonema almacenados en la carpeta /hmm1, se debe construir en la misma carpeta un Master Macro File (MMF) cuyo formato es similar al Master Label File (MLF) y contiene una lista con las definiciones de todos los modelos. Esto se hace para evitar tener un gran número de archivos de definición para los mismos.

Para ello se utiliza la herramienta de edición de modelos de HTK, llamada HHed. El nombre del archivo creado es *"hmmdefs"*. La ejecucion es como sigue:

HHEd -w hmm1/hmmdefs -d hmm1 co.hed monophones0

Opciones utilizadas:
-w *archivo de salida*
-d *directorio de los archivos de entrada*

Nota: *"co.hed"* es un archivo requierido por HHed, que sin embargo no realiza ningun efecto en el comando anterior. Se crea como un archivo en blanco.

4.3 Reestimación de los modelos (Entrenamiento 1)

Los modelos de entrada ubicados en la carpeta /hmm1 son reestimados usando la herramienta de reestimación HERest invocada como sigue:

HERest -T 1 -C config2 -I words.mlf -t 250.0 150.0 1000.0 -S train.scp -H hmm1/hmmdefs -M hmm2 monophones0

Opciones utilizadas:
-A *Imprime los argumentos del programa en la línea de comandos*
-D *Muestra los archivos de configuración*
-T *Opción de trazo*
-C *Carga el archivo de configuración*
-I *Carga el Master Label File (.mlf)*
-t *El primer valor configura el umbral de podado (pruning) de los datos para el entrenamiento.*
-S *Carga del script file (ubicaciones de los archivos MFCC)*
-H *Ubicación de las definiciones del HMM*

-M Ubicación de los modelos de salida

La salida de este comando en la terminal debe mostar lo siguiente:

```
HTK Configuration Parameters[8]
Module/Tool     Parameter            Value
#               NUMCEPS                 12
#               CEPLIFTER               22
#               NUMCHANS                20
#               PREEMCOEF          0.970000
#               USEHAMMING              TRUE
#               WINDOWSIZE     250000.000000
#               TARGETRATE     100000.000000
#               TARGETKIND          MFCC_0_D_A

HERest  ML Updating: Transitions Means Variances

System is PLAIN
46 Logical/46 Physical Models Loaded, VecSize=39
2 MMF input files
Pruning-On[250.0 150.0 1000.0]
        Processing Data: 001.mfc; Label 001.lab
        Utterance prob per frame = -7.592732e+01
        Processing Data: 002.mfc; Label 002.lab
        Utterance prob per frame = -7.572837e+01
        Processing Data: 003.mfc; Label 003.lab
        Utterance prob per frame = -7.680112e+01
        Processing Data: 004.mfc; Label 004.lab
        Utterance prob per frame = -7.747089e+01
        Processing Data: 005.mfc; Label 005.lab
        Utterance prob per frame = -7.349487e+01
        ...
        Processing Data: 460.mfc; Label 460.lab
        Utterance prob per frame = -7.753304e+01
        Saving hmm's to dir hmm1
        Reestimation complete - average log prob per frame = -7.863106e+01
        - total frames seen = 1.822420e+05

HTK Configuration Parameters[8]
Module/Tool     Parameter            Value
                NUMCEPS              12
                CEPLIFTER            22
                NUMCHANS             20
                PREEMCOEF            0.970000
                USEHAMMING           TRUE
```

WINDOWSIZE	250000.000000
TARGETRATE	100000.000000
TARGETKIND	MFCC_0_D_A

La ejecución de este programa ubica los modelos reestimados (archivos hmmdefs y macros en la carpeta /hmm1).

Luego se hace el mismo procedimiento 2 veces más, ejecutando los comandos:

HERest -T 1 -C config2 -I words.mlf -t 250.0 150.0 1000.0 -S train.scp -H hmm2/hmmdefs -M hmm3 monophones0

HERest -T 1 -C config2 -I words.mlf -t 250.0 150.0 1000.0 -S train.scp -H hmm3/hmmdefs -M hmm4 monophones0

Hasta este punto se han hecho 3 re-estimaciones del modelo inicial, que se ubican en las carpetas /hmm2, /hmm3 y /hmm4.

4.4 Enlazando estados (Entrenamiento 2)

El paso anterior ha generado un modelo de markov de 5 estados para cada fonema, así como un modelo para los silencios (breath y sil). El siguiente paso consiste en añadir transiciones extra de los estados 2 al 4 y de los estados 4 al 2 en los modelos de los silencios. La idea es hacer el modelo más robusto permitiendo a los estados individuales absorber los varios tipos de silencio dentro de los datos de entrenamiento (ver figura).

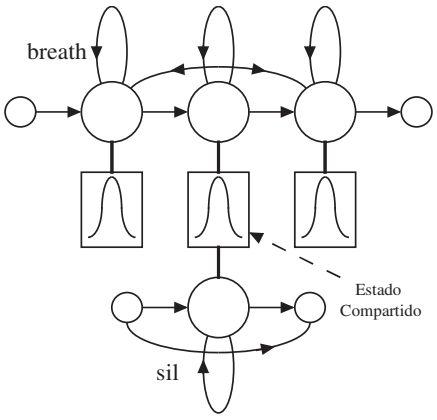

Imagen 2: Enlazando estados en los modelos de silencio

Para hacer esto se crea en la carpeta /HTK el archivo “sil.hed” conteniendo lo siguiente:

```
AT 2 4 0.2 {breath.transP}
AT 4 2 0.2 {breath.transP}
AT 1 3 0.3 {sil.transP}
TI silst {breath.state[3],sil.state[2]}
```

Y luego se ejecuta en una consola el programa HHed (las opciones utilizadas son las mismas que para el programa HERest):

HHEd -T 1 -H hmm4/hmmdefs -M hmm5 sil.hed monophones0

La salida en la consola de comando es:

```
HHEd
 46/46 Models Loaded [5 states max, 1 mixes max]

AT 2 4 0.200 {}
 Adding transitions to transP
 AT: 1 transP matrices adjusted

AT 4 2 0.200 {}
 Adding transitions to transP
 AT: 1 transP matrices adjusted

AT 1 3 0.300 {}
 Adding transitions to transP
 AT: 1 transP matrices adjusted

TI silst {}
 Tie items

Saving new HMM files ...
Edit Complete
```

Esto crea en la carpeta /hmm5 los modelos ajustados al enlace entre los estados 2 y 4.

Luego se hace el reestimamiento 3 veces más:

HERest -T 1 -C config2 -I words.mlf -t 250.0 150.0 1000.0 -S train.scp -H

hmm5/hmmdefs -M hmm6 monophones0

HERest -T 1 -C config2 -I words.mlf -t 250.0 150.0 1000.0 -S train.scp -H hmm6/hmmdefs -M hmm7 monophones0

HERest -T 1 -C config2 -I words.mlf -t 250.0 150.0 1000.0 -S train.scp -H hmm7/hmmdefs -M hmm8 monophones0

A partir de este punto se alternan los enlaces entre los estados 2 y 4 con los reestimamientos. Para ello se crean 3 archivos más, llamados "*MU2.hed", "MU4.hed" y "MU8.hed",* ubicados en /HTK.

MU2.hed -> MU 2 {*.state[2-4].mix}

MU4.hed -> MU 4 {*.state[2-4].mix}

MU8.hed -> MU 8 {*.state[2-4].mix}

El resto del entrenamiento ocupará las carpetas hasta /hmm24 y se ejecuta con los siguientes comandos:

HHEd -T 1 -H hmm8/hmmdefs -M hmm9 MU2.hed monophones0

HERest -T 1 -C config2 -I words.mlf -t 250.0 150.0 1000.0 -S train.scp -H hmm9/hmmdefs -M hmm10 monophones0

HERest -T 1 -C config2 -I words.mlf -t 250.0 150.0 1000.0 -S train.scp -H hmm10/hmmdefs -M hmm11 monophones0

HERest -T 1 -C config2 -I words.mlf -t 250.0 150.0 1000.0 -S train.scp -H hmm11/hmmdefs -M hmm12 monophones0

HHEd -T 1 -H hmm12/hmmdefs -M hmm13 MU4.hed monophones0

HERest -T 1 -C config2 -I words.mlf -t 250.0 150.0 1000.0 -S train.scp -H hmm13/hmmdefs -M hmm14 monophones0

HERest -T 1 -C config2 -I words.mlf -t 250.0 150.0 1000.0 -S train.scp -H hmm14/hmmdefs -M hmm15 monophones0

HERest -T 1 -C config2 -I words.mlf -t 250.0 150.0 1000.0 -S train.scp -H hmm15/hmmdefs -M hmm16 monophones0

HHEd -T 1 -H hmm16/hmmdefs -M hmm17 MU8.hed monophones0

HERest -T 1 -C config2 -I words.mlf -t 250.0 150.0 1000.0 -S train.scp -H hmm17/hmmdefs -M hmm18 monophones0

HERest -T 1 -C config2 -I words.mlf -t 250.0 150.0 1000.0 -S train.scp -H hmm18/hmmdefs -M hmm19 monophones0

HERest -T 1 -C config2 -I words.mlf -t 250.0 150.0 1000.0 -S train.scp -H hmm19/hmmdefs -M hmm20 monophones0

HERest -T 1 -C config2 -I words.mlf -t 250.0 150.0 1000.0 -S train.scp -H hmm20/hmmdefs -M hmm21 monophones0

HERest -T 1 -C config2 -I words.mlf -t 250.0 150.0 1000.0 -S train.scp -H hmm21/hmmdefs -M hmm22 monophones0

HERest -T 1 -C config2 -I words.mlf -t 250.0 150.0 1000.0 -S train.scp -H hmm22/hmmdefs -M hmm23 monophones0

HERest -T 1 -C config2 -I words.mlf -t 250.0 150.0 1000.0 -S train.scp -H hmm23/hmmdefs -M hmm24 monophones0

En este punto ya está hecho el entrenamiento de los fonemas y sólo hace falta evaluar los resultados.

5. RECONOCIMIENTO Y EVALUACIÓN DE RESULTADOS

5.1 Creando un archivo de gramática

HTK dispone de herramientas que pueden ayudar a mejorar la tasa de reconocimiento teniendo en cuenta patrones en la gramática del sistema, es decir la forma en que están organizados los modelos a reconocer, en este caso los fonemas. Se observó que todos los label files tienen la siguiente forma:

sil
breath
sil
fonema 1
fonema 2
...
fonema n
sil

Debido a esta estructura presente en todos los label files, se puede forzar a HTK a hacer el reconocimiento siguiendo este patrón: sil-breath-sil- <fonemas>-sil. Para ello se debe crear un archivo de gramática, que será llamado "gram" que contenga lo siguiente:

```
$phone= iy |
aa | @@ | ch | ei | ai | ii | zh | p | ng | sh | th | i@ | uh | @ | dh | oi | ow | eir | jh | a |
oo | b | e | d | g | f | i | h |  k | m | l | o | n | uu | s | r | u | t | w | v | y | ou | z;
( sil breath sil <$phone> sil )
```

A continuación se convierte el archivo anterior al formato HTK con el comando Hparse:

HParse gram wdnet

Esto crea el archivo "wdnet" que se utilizará en el paso del reconocimiento para forzar a HTK a reconocer los fonemas siguiendo la gramática dada.

5.2 Reconocimiento con HVite

HVite es la herramienta de HTK que se usa para el reconocimiento de modelos (palabras o fonemas). Utiliza el algoritmo de Viterbi. En este sistema se debe ejecutar:

```
HVite -H hmm24/hmmdefs -C config2 -S train.scp -l '*' -i recout.mlf -w  wdnet
-p 0.0 -s 5.0 dictionary monophones0
```

Opciones usadas:

- *-H* *Definción del modelo de entrada (macros y hmmdefs)*
- *-C* *Archivo de configuración*
- *-S* *Script file con la ubicación de los MFCCs*
- *-l* *Especificación de directorio donde se ubica el Master Labe File de salida*
- *-i* *Nombre del Master Label File de salida con el reconocimiento*
- *-w* *Nombre del archivo de gramática*
- *-p* *Logaritmo de la probabilidad de inserción de palabras*
- *-s* *Factor de la gramática*

La salida de este comando es el archivo *"recout.mlf"* que contiene el resultado del reconocimiento del sistema. El contenido de este archivo debe tener la siguiente

forma:

```
#!MLF!#
"'*'/001.rec"
<Label File reconocido de la primera frase>
.
"'*'/002.rec"
<Label File reconocido de la segunda frase>
.
"'*'/003.rec"
<Label File reconocido de la tercera frase>
...
.
"'*'/460.rec"
<Label File reconocido de la última frase>
```

Este archivo debe compararse con un MLF de referencia con el fin de evaluar cuantitativamente el rendimiento del sistema.

5.2 Evaluación de resultados

Para evaluar la tasa de reconocimiento del sistema se utiliza un archivo de referencia, en este caso se toma el archivo "words.mlf" que contiene los label files que vienen etiquetados desde la base de datos del Mocha-Timit. Este se compara con "recout.mlf" ejecutándose el programa HResults:

HResults -I words.mlf monophones0 recout.mlf

La salida aparece en consola como:

```
==================HTK Results
Analysis====================
  Date: Mon Feb  2 23:27:31 2015
  Ref : words.mlf
  Rec : recout.mlf
---------------------- Overall Results -------------------------
SENT: %Correct=0.00 [H=0, S=92, N=92]
WORD: %Corr=69.43, Acc=54.77 [H=2112, D=193, S=737, I=446, N=3042]
================================================================
==
```

Como no se está haciendo reconocimiento a nivel de oraciones, la tasa de

reconociminento SENT es 0%.

El resultado de WORD es el que interesa y se interpreta así:

- N: Número total de fonemas, 3042
- H: Número de fonemas reconocidos correctamente, 2112
- D: Número de errores por omisión, 193
- S: Número de errores por sustitución, 737
- I: Número de errores por inserción, 446
- %Corr: Tasa de reconocimiento: 69.43%
- Acc: Precisión (Tomando errores de inserción): 54.77%

PARTE 2: AÑADIENDO LA INFORMACIÓN ARTICULATORIA

1. CREACIÓN DE LOS VECTORES CARACTERÍSTICOS

Para unir la información articulatoria con la información de los vectores característicos obtenidos de las muestras de audio, se hace uso de un módulo de la herramienta Edinburgh Speech Tools llamado "ch_track". Ver info en el siguiente enlace: http://www.cstr.ed.ac.uk/projects/speech_tools/manual-1.2.0/x737.htm

Por ejemplo, para unir los datos articulatorios codificados en formato MFCC y los verdaderos vectores caracteríticos de la primera frase, los cuales están en los archivos 001e.mfc y 001w.mfc respectivamente, se ejecuta el siguiente comando en la carpeta donde estén ubicados los archivos:

ch_track 001e.mfc 001w.mfc -pc first -o 001.mfc -otype htk_mfcc -S 0.01

donde la opción -pc first indica que se combinan los vectores en paralelo, con el tamaño acotado al primer archivo (001e.mfc), el argumento de la opción -o indica la el archivo de salida, el de -otype indica el tipo de archivo de salida, en este caso MFCC de HTK y el argumento de -S indica el periodo de muestreo en segundos.

HTK provee una herramienta para visualizar el contenido de los archivos, llamada HList. Se hará uso de ella para mostrar la forma de los arvhivos de entrada y de salida del comando ch_track ejecutado anteriormente. Hlist se ejecuta de la siguiente forma:

HList -h -e 1 archivo.mfc

La opción -h imprime el encabezado del archivo y la opción -e 0 imprime el primer vector del archivo "archivo.mfc"

MFCC audio, 220 bloques de tamaño 39:

```
---------------------------------- Source: 001w.mfc ----------------------------------
  Sample Bytes:  156      Sample Kind:   MFCC_D_A_K_0
  Num Comps:     39       Sample Period: 10000.0 us
  Num Samples:   220      File Format:   HTK
------------------------------------ Samples: 0->0 -----------------------------------
0:    -13.167  -3.588  11.727  -8.182   3.945   5.937  -3.274  -9.861   1.061   1.842
        1.467  11.936  67.465   1.155   2.771  -1.376   1.810   1.241  -0.784   0.122
        0.705  -0.334  -1.479  -0.541  -0.493  -1.948   0.130  -0.128  -0.297   0.258
       -0.724  -0.148  -0.068   0.376  -0.020   0.226  -0.085  -0.647  -0.043
---------------------------------------- END -----------------------------------
```

Datos EMA codificados como MFCC, 194 bloques de tamaño 14

```
---------------------------------- Source: 001e.mfc ----------------------------------
  Sample Bytes:  56       Sample Kind:   MFCC
  Num Comps:     14       Sample Period: 10000.0 us
  Num Samples:   194      File Format:   HTK
------------------------------------ Samples: 0->0 -----------------------------------
0:     0.882   0.383   0.815   0.140   0.734   0.294  -0.361  -0.011   0.309  -0.454
       0.745  -0.718  -1.432  -0.023
---------------------------------------- END -----------------------------------
```

Datos combinados, 193 bloques de tamaño 14+39=53

```
---------------------------------- Source: 001.mfc ----------------------------------
  Sample Bytes:  212      Sample Kind:   MFCC
  Num Comps:     53       Sample Period: 10000.0 us
  Num Samples:   193      File Format:   HTK
------------------------------------ Samples: 0->0 -----------------------------------
0:     0.800   0.430   0.744   0.164   0.735   0.306  -0.380   0.011   0.305  -0.420
       0.723  -0.682  -1.464   0.075 -12.184   2.169   9.141  -3.020  11.159   6.449
      -5.046  -9.703   0.602  -5.612   2.181  14.287  61.857   1.675   2.754  -2.026
       2.846   0.377  -1.439  -0.383   1.961  -0.396  -1.680  -1.196  -2.305  -2.357
       0.046  -0.485  -0.257  -0.020  -1.165  -0.125  -0.057   0.316   0.046   0.753
       0.077  -0.745   0.217
---------------------------------------- END -----------------------------------
```

Debe ejecutarse ch_track para las 360 frases.

2. ENTRENAMIENTO Y RECONOCIMIENTO

Los pasos para el entrenamiento y reconocimiento son los mismos que se realizaron en la parte 1, con la excepción de que el archivo de configuración de entrenamiento "*config2*" se modifica tomando esta simple forma:

```
TARGETKIND = MFCC
TARGETRATE = 100000.0
PREEMCOEF = 0.97
```

El archivo *"proto"* se modifica también para un tamaño de vector de 53:

```
  ~o <VecSize> 53 <MFCC>
~h "proto"
<BeginHMM>
  <NumStates> 5
  <State> 2
    <Mean> 53
      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
    <Variance> 53
      1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
 <State> 3
    <Mean> 53
      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
    <Variance> 53
      1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
 <State> 4
    <Mean> 53
      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
    <Variance> 53
      1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
 <TransP> 5
  0.0 1.0 0.0 0.0 0.0
  0.0 0.6 0.4 0.0 0.0
  0.0 0.0 0.6 0.4 0.0
  0.0 0.0 0.0 0.7 0.3
```

0.0 0.0 0.0 0.0 0.0
<EndHMM>

El resultado de implementar el sistema combinado mejora la tasa de reconocimiento, como se puede ver:

```
==================== HTK Results Analysis
====================
  Date: Mon Feb  2 23:28:30 2015
  Ref : words.mlf
  Rec : recout.mlf
------------------------ Overall Results --------------------------
SENT: %Correct=0.00 [H=0, S=92, N=92]
WORD: %Corr=76.20, Acc=67.26 [H=2446, D=205, S=559, I=287, N=3210]
```