SIMULACIÓN DE VOCES A TRAVÉS DE UN CONVERSOR TEXTO-VOZ BASADO EN MODELOS OCULTOS DE MARKOV

PROYECTO FIN DE CARRERA

INGENIERÍA DE TELECOMUNICACIÓN

MARZO 2012

AUTOR: DIEGO MOLINA MIRAVALLES

DIRECTOR: DR. EDUARDO LLEIDA SOLANO

DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA Y COMUNICACIONES

ÁREA DE TEORÍA DE LA SEÑAL Y COMUNICACIONES

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

Universidad Zaragoza

Agradecimientos

A Eduardo, por su dedicación y paciencia.
También a Antonio y Kike, por echarme una mano.
A mis amigos/as, por estar siempre ahí.
A Laura, por estar día tras día apoyándome.
A mis padres, por sus consejos y ánimos.

SIMULACIÓN DE VOCES A TRAVÉS DE UN CONVERSOR TEXTO-VOZ BASADO EN MODELOS OCULTOS DE MARKOV

RESUMEN

Una parte importante de los sistemas de inteligencia ambiental la constituye el interfaz hombre-máquina, y dentro de éste la síntesis de voz. La síntesis de voz consiste en la producción artificial de voz humana. Los principales retos de los conversores texto-voz son la producción de una voz artificial inteligible y natural, la completa automatización del proceso y que el texto necesario para la síntesis no provenga de una modificación del lenguaje original.

A lo largo de este proyecto se ha puesto en marcha un sistema completo de conversión texto-voz de última generación basado en la síntesis de voz por modelos ocultos de Markov. Para llevarlo a cabo se han empleado algoritmos de adaptación de modelos acústicos, concretamente Maximum A Posteriori y Maximum Likelihood Linear Regression. Estos algoritmos permiten obtener una voz sintetizada a partir de pocas muestras de voz y no fonéticamente balanceadas del locutor deseado, pues utilizan como base otros registros que sí están fonéticamente balanceados entrenados previamente para la síntesis.

Para realizar este proceso de conversión texto-voz se ha elaborado una base de datos, tanto de un locutor genérico como del locutor a adaptar, y su representación escrita. Se ha realizado un proceso de entrenamiento, consistente en la elaboración de los modelos acústicos empleados en la síntesis, aplicando distintos algoritmos para el cálculo de los modelos. Finalmente se han aplicado los algoritmos adaptativos descritos anteriormente. Una vez obtenidos los modelos acústicos se ha procedido a generar voz artificial siguiendo el modelo digital de producción del habla, excitación más filtro.

El resultado del proceso es una voz artificial que busca asemejarse a la voz original, semejanza que se ha evaluado mediante programación dinámica. Por último, se ha elaborado una aplicación web que, sirviéndose del sistema de síntesis elaborado, servirá para crear un banco de voces de los usuarios que la empleen.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y alcance del proyecto.

1.1 Planteamiento del problema y motivación.

El uso del lenguaje y del habla es el principal medio de comunicación empleado por el ser humano. Sumado esto al marco de la sociedad de la información en la que vivimos, parece lógico establecer una fusión entre tecnología y habla. Se pueden diferenciar dos ramas en las tecnologías aplicadas al habla: el reconocimiento y la síntesis de voz.

El reconocimiento del habla consiste en un proceso de clasificación de patrones, que permite procesar una voz emitida por un locutor y reconocer la información en ella contenida. El reconocimiento es empleado en aplicaciones tan diversas como: autoaprendizaje de idiomas, sistemas de seguridad, dictado de documentos o información telefónica sin operador.

La síntesis de voz, por su parte, consiste en la generación de una voz artificial a partir de un texto introducido por el usuario. Los sistemas de síntesis abren también un amplio abanico de posibilidades y distintas aplicaciones relacionadas con:

- Telefonía: servicio de teleoperadores automáticos.
- Tratamiento de discapacidades del habla.

- Ayuda a la docencia.
- Lectura de documentos.

Desde el Grupo de Tecnologías de las Comunicaciones (GTC) de la Universidad de Zaragoza se han venido desarrollando sistemas de síntesis basados en la concatenación de difonemas. Este proyecto nace movido por la necesidad de establecer las bases de un sistema de síntesis por modelos ocultos de Markov para la lengua castellana. La principal ventaja que ofrece este tipo síntesis frente a los sistemas de concatenación de difonemas es, que la cantidad de locuciones necesarias para obtener buenos resultados es mucho menor sin que esto influya considerablemente en la calidad de la voz generada artificialmente.

1.2 Síntesis de voz. Antecedentes.

El mecanismo de producción de la voz en los seres humanos consiste en la emisión de un flujo de aire desde los pulmones con la intensidad adecuada para hacer vibrar las cuerdas vocales. Las partes del cuerpo humano implicadas en este proceso son: pulmones, tracto vocal y cuerdas vocales.

Los sonidos producidos por la voz humana pueden ser clasificados como sonoros o sordos. Los sonidos sonoros son producidos mediante la vibración de las cuerdas vocales, mientras que los sordos provienen de las contracciones de alguna sección del tracto vocal. A nivel de análisis de señal, los sonidos sonoros presentan una marcada periodicidad, mientras que los sonidos sordos son de naturaleza aperiódica.

La síntesis del habla consiste en generar una voz a partir de una representación escrita; el término empleado en inglés resulta muy esclarecedor: *Text-To-Speech* (de texto a voz).

Existen diversos tipos de síntesis, entre los que destacan:

- Síntesis concatenativa por selección de unidades: se parte de una base de datos que se segmenta en frases, palabras, sílabas y fonemas. Para producir la voz se concatenan estos segmentos pregrabados. Se obtiene una voz muy

natural pero para ello es necesaria una base de datos de voz y texto muy amplia, lo que en algunos casos la hace inviable. Es la técnica empleada por el GTC.

- Síntesis por formantes: la voz se genera a partir de un tren de pulsos introducido en un banco de filtros. Cada uno de los filtros modela una de las resonancias del tracto vocal. Se consigue una voz muy inteligible pero también demasiado robótica.

- Síntesis articulatoria: se obtiene la voz a partir de parámetros relacionados con la forma del tracto vocal y las distintas articulaciones bucales. No se han alcanzado niveles elevados de calidad todavía.

- Síntesis mediante modelos ocultos de Markov (HMMs:Hidden Markov Models): objeto de estudio en este proyecto. Analiza las propiedades estadísticas de las señales de audio y genera unos modelos a partir de los cuales, mediante un sistema de excitación más filtro, se obtiene la voz artificial. Ha sido más empleado para reconocimiento que para síntesis.

Las principales características de los sistemas basados en HMMs son:

- Requieren menos cantidad de memoria para funcionar.

- Proporcionan una voz más artificial que la síntesis concatenativa, pero más inteligible.

- Permiten cambiar características de la voz o el estilo de locución.

- Resultan muy útiles y sencillas para la adaptación de voces a partir de muy pocas frases.

El proceso de síntesis mediante modelos ocultos de Markov consta de tres etapas bien diferenciadas, que se muestran en la Figura 1.1.

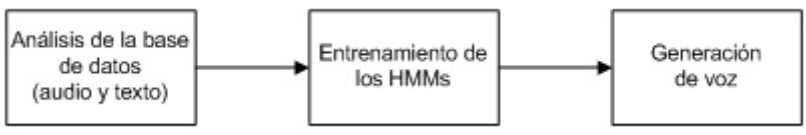

Figura 1.1. Etapas del proceso de síntesis mediante HMMs.

En los próximos capítulos se expondrán en detalle cada una de estas partes, pero para tener una visión general, a continuación se expone una breve explicación de cada una de ellas:

- Obtención de las señales de audio: se graba al locutor a adaptar diciendo unas frases adecuadamente elegidas.
- Análisis de la base de datos: partiendo de los ficheros de audio y sus correspondientes transcripciones de texto, se obtienen los parámetros espectrales y la frecuencia fundamental.
- Entrenamiento de los HMMs: con los datos obtenidos en la etapa anterior se generan unos modelos iniciales que paso a paso se irán mejorando mediante algoritmos recursivos hasta obtener los modelos finales.
- Generación de voz artificial: en base a un texto introducido por el usuario, se genera una voz artificial siguiendo el modelo digital de producción del habla, excitación más filtro.

1.3 Objetivos.

El principal objetivo de este proyecto es establecer las bases de un sistema de síntesis mediante HMMs para la lengua castellana. La síntesis se realizará mediante métodos de adaptación, que permiten generar una voz artificial bastante semejante a la original a partir de muy pocas muestras de audio del locutor a adaptar. Por otro lado, se plantea también un análisis de similitud de voces y el desarrollo de una aplicación web para la adquisición de un banco de voces.

El análisis de similitud entre las voces originales y las voces artificiales generadas se realiza mediante un alineamiento temporal a través de un algoritmo de programación dinámica.

La aplicación web desarrollada permitirá obtener una amplia base de datos de locutores y modelos que ayuden a la mejora del método de síntesis empleado. Esta aplicación contará con un interfaz en el que, mediante un nombre de identificación y una contraseña, se permitirá al usuario generar su propia voz artificial.

Por último, esta memoria pretende servir de guía de uso del sístema de síntesis propuesto y facilitar futuros trabajos e investigaciones del GTC.

1.4 Estructura de la memoria.

Tras este breve capítulo de introducción el resto de la memoria se estructura de la siguiente forma:

- En el Capítulo 2 se desarrolla una exposición teórica sobre las síntesis mediante HMMs. Comienza con una explicación sobre los datos necesarios para la síntesis, centrándose después en los tipos de modelos y los parámetros de salida de éstos. A continuación, se explican los conceptos teóricos que están detrás de todo el proceso de entrenamiento de los modelos, desde la creación de los modelos iniciales a la adaptación a locutor. Se finaliza el capítulo con los fundamentos teóricos de la generación de voz artificial.

- En los capítulos 3 y 4 se explican en detalle todos los pasos que envuelven el proceso de síntesis. El Capítulo 3 se centra en el proceso de entrenamiento de los modelos con el programa *HTS*. Se explican los distintos pasos, las funciones empleadas, los ficheros necesarios a la entrada y los producidos a la salida.

  En el Capítulo 4 se expone el sistema de generación de voz *vivoSinte* y la herramienta en la que éste se basa, *hts_engine*. Se explican los distintos pasos seguidos para la generación y los procedimientos empleados por el sistema *vivoSinte*.

- En el Capítulo 5 se presenta el estudio de la distorsión existente entre señales originales y sintetizadas. Por otro lado se expone la aplicación Web desarrollada.

- En el Capítulo 6 se analizan los resultados, se exponen las conclusiones del proyecto y se proponen líneas futuras de trabajo.

Capítulo 2

Síntesis mediante modelos ocultos de Markov.

Los modelos ocultos de Markov (HMMs:Hidden Markov Models [19][3]) son procesos doblemente estocásticos, formados por una secuencia de estados, que no es observable y de ahí la denominación de ocultos, y por la salida de cada uno de ellos. Los HMMs pueden ser discretos o continuos, los valores de salida de cada estado de los modelos discretos pertenecen a un conjunto finito de posibles valores, mientras que en los modelos continuos el dominio es infinito. En el Apéndice A se lleva a cabo una explicación detallada de la estadística de los HMMs.

Para la síntesis de voz se emplean modelos continuos, puesto que las características de las señales de voz toman valores continuos. Estas características tendrán una función densidad de probabilidad también continua, normalmente Gaussiana.

El proceso de síntesis consta de dos partes fundamentales. La Figura 2.1 muestra un pequeño esquema de este proceso.

La primera parte consiste en la obtención de los HMMs para las tres características principales que definen una señal de voz: la frecuencia fundamental, el espectro y la duración. Como el objetivo es generar una voz artificial lo más parecida posible a la original, con un número limitado de frases del locutor, el proceso de entrenamiento se divide en dos partes. Por un lado, se generan unos

modelos preliminares a partir del procesado de unas señales de voz base y sus correspondientes transcripciones en texto. Este proceso consta de cuatro fases fundamentales:

- Elaboración de unos modelos iniciales.
- Mejora de los modelos iniciales mediante el algoritmo de Viterbi.
- Refinamiento de los modelos aplicando el algoritmo de Baum-Welch.
- Técnicas de agrupamiento o clustering empleando árboles de decisión.

Una vez obtenidos estos modelos preliminares, se les aplicará una fase de adaptación en la que se generarán unos modelos finales adaptados al locutor que nos ocupe.

Tras obtener los modelos adaptados, la segunda parte consiste en la generación de una voz artificial a partir de estos modelos y un texto introducido por el usuario. El objetivo final es obtener una voz artificial lo más parecida posible a la voz original. La generación de la voz se realiza mediante el modelo digital de producción del habla, excitación más filtro.

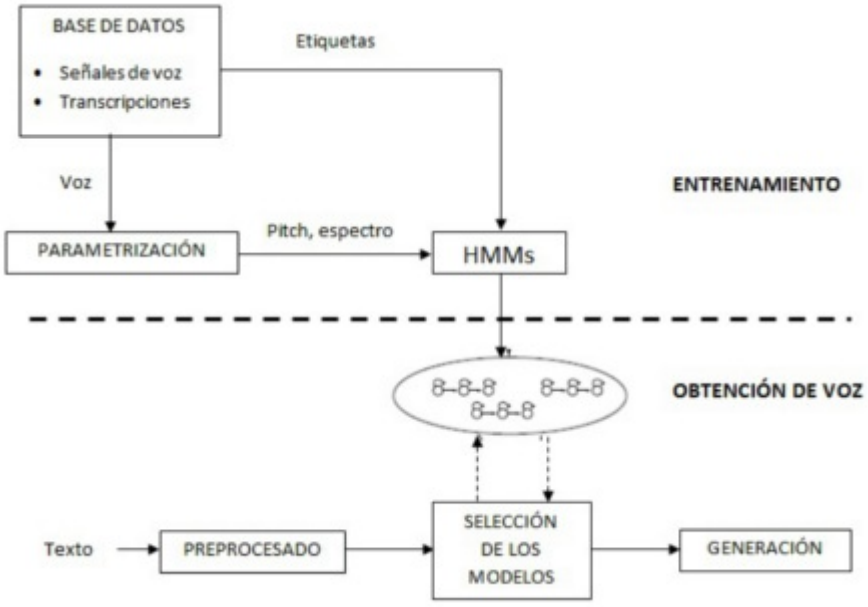

Figura 2.1. Esquema del proceso de síntesis mediante HMMs.

2.1 Datos necesarios para el entrenamiento.

El proceso de entrenamiento necesita de una base de datos bien elaborada, compuesta por un conjunto de señales de voz y sus correspondientes representaciones escritas en el idioma en el que se desee entrenar, tanto para la voz base como para la voz a adaptar. Mediante un análisis exhaustivo de esta base de datos, el entrenamiento da como resultado los modelos que son necesarios para la síntesis.

Para poder realizar el entrenamiento es necesario un procesado previo de las representaciones escritas, que consiste en el segmentado y etiquetado de estas oraciones. Una vez realizado éste, tendremos una base de datos de etiquetas que nos darán información sobre la sucesión, posición dentro de la frase, duración y la entonación con que se pronuncia cada fonema en la grabación.

Por último, las señales de voz también se someten un preprocesado en el que se extrae cierta información relacionada con la frecuencia fundamental (en inglés pitch), el espectro y la duración.

2.1.1 Frecuencia fundamental.

La frecuencia fundamental o pitch, estrictamente hablando, está relacionada con la percepción del tono. En el dominio temporal, se asocia el pitch con el periodo de la señal de voz, mientras que en el dominio de la frecuencia, el pitch es la separación frecuencial entre dos formantes consecutivos en sonidos con cierta sonoridad, y por tanto, periodicidad. Se denominan formantes a los picos que aparecen en el espectro sonoro de las vocales, independientemente del tono.

La frecuencia fundamental de la voz del ser humano oscila entre 70 y 400 Hz, siendo más baja para los hombres (voz más grave), que para las mujeres (voz más aguda). Por tanto, se trata de un parámetro característico de cada locutor, que permite diferenciar y reconocer su voz, y está vinculado con la frecuencia fundamental de vibración de las cuerdas vocales. Aquí se debe distinguir entre sonidos sonoros, que poseen cierta periodicidad y cuyo pitch tomará valores dentro del rango 70-400 Hz, y los sonidos sordos que son aperiódicos, lo que imposibilita

determinar su pitch.

Por tanto, el pitch constituye un importante parámetro para la síntesis de voz, y como tal, debe ser considerado y entrenado por algún tipo de HMM. Para realizar dicho entrenamiento, en primer lugar deberá ser extraído de las señales de voz de la base de datos.

Para obtener el pitch a partir de la señal de voz se recurre a un método de correlación cruzada normalizada [17][15].

La función de autocorrelación aplicada en este método es la siguiente:

$$r_x(m) = \frac{1}{L}\sum_{m=0}^{L-1} x(n)x(n-m) \tag{2.1}$$

El periodo de pitch (τ) es la longitud, en número de muestras, que separa el máximo de correlación (m=0) y el segundo máximo. Una vez obtenido el periodo, el pitch se obtiene facilmente calculando el inverso, $f_o = 1/\tau$.

2.1.2 Información espectral.

Toda la información espectral se obtiene mediante una parametrización de las señales de voz. Existen diversos métodos para realizar esta parametrización. El empleado en este proyecto es el conocido como Coeficientes Cepstrum en la escala de Mel, MFCCs (del inglés, Mel-frequency cepstral coefficients).

Con esta parametrización se consigue un espectro cuya resolución frecuencial es similar a la del oído humano, que posee alta resolución a frecuencias altas. Esta característica hace que el método de los MFCCs sea utilizado tanto en síntesis como en reconocimiento de voz.

El cepstrum de la señal tiene información sobre la variación del espectro, y se calcula a partir de la señal de voz, trama a trama, siendo una trama un segmento de la señal original de una longitud determinada. En cada trama se aplica el análisis mel-cepstral, obteniendo sus coeficientes mel-cepstrales [4]. El modelo espectral,

$H(e^{j\omega})$, se representa mediante M coeficientes mel-cepstrales:

$$H(z) = exp \sum_{m=0}^{M} \widetilde{c}(m)\widetilde{z}^{-m} \tag{2.2}$$

donde

$$\widetilde{z}^{-1} = \frac{z^{-1} - \alpha}{1 - \alpha z^{-1}}, \quad |\alpha| < 1 \tag{2.3}$$

es una transformación bilineal que aproxima el plano Z en escala de Mel en función del plano Z en escala lineal.

La fase característica para está transformación viene dada por la expresión:

$$\widetilde{\omega} = arctan \frac{(1-\alpha^2)sen(\omega)}{(1+\alpha^2)cos(\omega) - 2\alpha} \tag{2.4}$$

donde el parámetro α sirve para ajustar la aproximación entre las dos escalas, en función de la frecuencia de muestreo. Por ejemplo, para una frecuencia de muestreo de 10 KHz, con $\alpha = 0.35$ se consigue una buena aproximación a la escala de Mel.

Para obtener una estimación no sesgada se utiliza el siguiente criterio, minimizando respecto a $\{\widetilde{c}(m)\}_{m=0}^{M}$.

$$E = \frac{1}{2\pi} \int_{-\pi}^{\pi} \{exp\, R(\omega) - R(\omega) - 1\}\, d\omega \tag{2.5}$$

donde

$$R(\omega) = log I_N(\omega) - log|H(e^{j\omega})|^2 \tag{2.6}$$

e I_N representa el periodograma modificado [10], que permite estimar espectralmente el proceso $x(n)$, el cual es considerado estacionario dentro de cada trama. El parámetro N representa el número de muestras de la trama. Para conseguir una buena estimación espectral se necesita un número de muestras suficientemente elevado, ya que I_N es asintóticamente insesgado.

Para facilitar la minimización de 2.2 se aplica una transformación sobre 2.5 de

modo que el factor de ganancia K quede fuera de *H(z)*,

$$H(z) = exp \sum_{m=0}^{M} b(m)\phi_m(z) = K \cdot D(z) \tag{2.7}$$

donde

$$K = exp\ b(0) \tag{2.8}$$

$$D(z) = exp \sum_{m=1}^{M} b(m)\phi_m(z) \tag{2.9}$$

y

$$b(m) = \begin{cases} \widetilde{c}(m), & m = M; \\ \widetilde{c}(m) - \alpha b(m+1), & 1 \leq m < M; \end{cases} \tag{2.10}$$

$$\phi_m(z) = \begin{cases} 1, & m = 0; \\ \frac{(1-\alpha^2)z^{-1}}{1-\alpha z^{-1}} \widetilde{z}^{-(m-1)}, & m \geq 1; \end{cases} \tag{2.11}$$

Y, al ser H(z) un sistema de fase mínima, minimizar E es equivalente a minimizar

$$\epsilon = \frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{I_N(\omega)}{|D(e^{j\omega})|^2}\ d\omega \tag{2.12}$$

con respecto a $\underline{b} = [b(1), b(2), ..., b(M)]^T$.

Al ser ϵ convexo, según [25], la minimización de 2.12 se puede llevar a cabo eficientemente aplicando el método de Newton-Raphson.

2.2 Modelos de las características de la voz empleados en síntesis.

Los tres elementos necesarios para realizar la síntesis son el pitch, el espectro y la duración. En esta sección se exponen los HMMs que los modelan.

2.2.1 Modelo para el pitch.

El patrón del pitch está compuesto por valores continuos en la región de sonidos sonoros y de un símbolo discreto en la región de sonidos sordos como se puede ver

en la Figura 2.2.

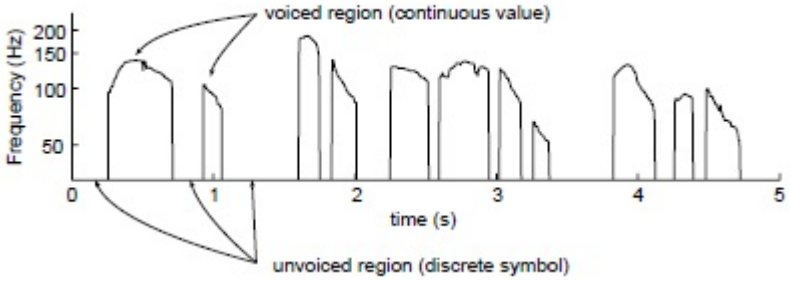

Figura 2.2. Ejemplo de patrón de F0, extraído de[24].

En la caracterización del pitch aparecen dos procesos estocásticos para los observables de los HMMs: la existencia o no de pitch, y en caso de existir que valor toma. Para modelarlos correctamente es necesario un híbrido entre un HMM discreto y otro continuo. Para esto, las probabilidades de salida de los estados se definen mediante una distribución multi-espacial de probabilidad (MSD [21][24]). A continuación se detalla esta distribución de probabilidad y se demuestra su utilidad para modelar el pitch.

Se considera un espacio muestral, Ω, el cual es la unión de G subespacios:

$$(\Omega = \bigcup_{g=1}^{G} \Omega_g) \tag{2.13}$$

Cada subespacio Ω_g es un espacio de dimensión real n_g ($\Omega_g = \Re^{n_g}$) con n_g = 0,1,...,m, que no tiene por qué ser la misma para todos los subespacios. Cada subespacio tiene una función densidad de probabilidad (f.d.p.), $\mathcal{N}_g(\underline{x})$, $\underline{x} \in \Re^{n_g}$ y su propia probabilidad, $P(\Omega_g) = \omega_g$, donde $\sum_{g=1}^{G} \omega_g = 1$.

Cada realización E, está representada por una variable aleatoria $\underline{\mathbf{o}}$ compuesta por una variable aleatoria continua $\underline{x} \in \Re^n$ y un conjunto de índices $\mathcal{X}$. De manera que, $\underline{o} = (\mathcal{X}, \underline{x})$, indica que la salida $\underline{x}$ está estadísticamente definida por los subespacios indicados en $\mathcal{X}$. Finalmente la probabilidad de tener una observación

$\underline{o}$ viene definida por la expresión:

$$b(\underline{o}) = \sum_{g \in S(\underline{o})} \omega_g \mathcal{N}_g(\mathcal{V}(\underline{o})), \text{ donde } \mathcal{V}(\underline{o}) = \underline{x},\ S(\underline{o}) = \mathcal{X} \tag{2.14}$$

Para el caso que nos ocupa y como se ha apuntado anteriormente, se tendrá un valor continuo en la región de sonido sonoro y ningún valor en la región de sonido sordo. Esto se puede modelar asumiendo que el valor de pitch para un sonido sordo corresponde a un espacio cero-dimensional y para un sonido sonoro corresponderá a un espacio unidimensional, esto es $n_G = 0$ y $n_g = 1$ (g = 1,2,...,G-1) respectivamente.

Para el subespacio $n_G = 0$, se impone $\mathcal{N}_G(\underline{x}) = 1$ por simplicidad. Por otro lado, para el subespacio $n_g = 1$, $\mathcal{N}_g(\underline{x})$ es una f.d.p. Gaussiana.

$$S(\underline{o}) = \begin{cases} \{1, 2, ..., G-1\}, & \text{(sonidos sonoros)} \\ \{G\}, & \text{(sonidos sordos)} \end{cases} \tag{2.15}$$

Así pues, la estadística de los observables de los estados queda definida según:

$$b(\underline{o}) = P(sordo)\mathcal{N}(o|sordo) + P(sonoro)\mathcal{N}(o|sonoro) =$$

$$= \omega_G \mathcal{N}_G(*) + \omega_g \mathcal{N}_g(f_o) = \omega_G + \omega_g \mathcal{N}_g(f_o) \tag{2.16}$$

De esta formulación se puede extraer la conclusión de que para el caso de sonido sordo sólo nos interesa la aparición de dicho evento, mientras que para un sonido sonoro nos interesan tanto su aparición como su valor, de ahí $\mathcal{N}_g(f_o$. Las probabilidades de que un sonido sea sordo o sonoro son ω_G y ω_g, respectivamente.

Por tanto, en estos modelos, los observables de salida de los estados son escalares, con el valor del pitch cuando se refiere a un sonido sonoro y ningún valor cuando se trata de un sonido sordo.

Un posible modelo para los estados y observables de salida para el pitch se muestra en la Figura 2.3

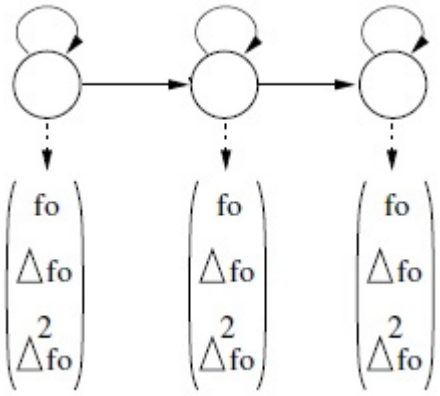

Figura 2.3. Modelo de HMMs para la frecuencia fundamental.

2.2.2 Modelo para el espectro.

El modelo asociado al espectro genera 3 vectores, los coeficientes estáticos y dos tipos de coeficientes dinámicos:

- Coeficientes estáticos ($\underline{c}$): relacionados con los coeficientes mel-cepstrales.
- Coeficientes dinámicos de primer orden ($\underline{\Delta}_c$): primera derivada de los coeficientes estáticos.
- Coeficientes dinámicos de segundo orden ($\underline{\Delta}_c^2$): segunda derivada de los coeficientes estáticos.

Estos coeficientes dinámicos modelan la evolución temporal de los coeficientes estáticos: la información sobre si el próximo valor será mayor o menor que el actual y su tendencia a crecer o decrecer. En la Figura 2.4 se muestra un posible modelo para estos estados y los observables a la salida.

En cuanto a la estadística de los observables, éstos son más sencillos de caracterizar que los correspondientes al pitch. Los coeficientes a la salida de cada estado son vectores en $\Re^n$ y se pueden caracterizar de manera sencilla mediante n mezclas. La función densidad de probabilidad que caracteriza a estos coeficientes para el estado *i-ésimo* se corresponde con la expresión 2.17, que consiste en la ponderación de N funciones de densidad de probabilidad, con su media $\underline{\mu}_{in}$ y

matriz de covarianza $\underline{\underline{\Sigma}}_{in}$[19].

$$b_i(\underline{o}) = \sum_{n=1}^{N} c_{in} \mathcal{N}(o; \underline{\mu}_{in}, \underline{\underline{\Sigma}}_{in}) \tag{2.17}$$

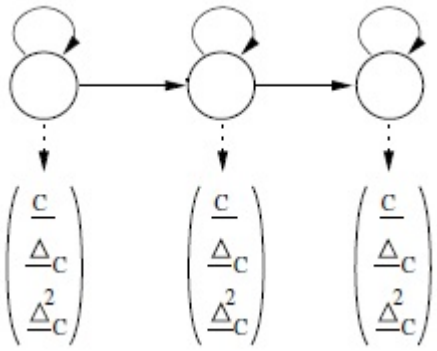

Figura 2.4. Modelo de HMMs para el espectro.

2.2.3 Modelo para la duración.

Estos modelos hacen referencia a la duración de cada fonema. Los observables se emitirán una vez por fonema, en vez de trama a trama como sucedía en los modelos anteriores. Por esta razón, este modelo constará de un único estado.

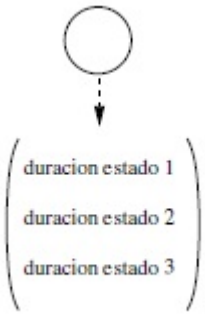

Figura 2.5. Modelo de HMMs para la duración.

En la Figura 2.5 se muestra un posible modelo para la duración, con un estado que emite un vector de dimensión igual al número de estados que tienen los modelos del pitch y el espectro, donde cada componente indica la duración de cada uno

de los estados de estos modelos. El observable a la salida del estado tendrá una estadística Gaussiana multivariable con una media y varianza determinadas.

Cabe destacar que tanto los modelos del pitch como los del espectro están formados por 5 estados cada uno, mientras que el modelo para la duración está compuesto por un único estado.

2.3 Entrenamiento de los modelos.

2.3.1 Modelos iniciales.

En la primera fase del entrenamiento se busca obtener unos modelos iniciales que sirvan de primera aproximación a los modelos definitivos. Estos modelos son poco precisos, pero suponen una base sobre la que edificar los modelos finales. Para obtener los modelos iniciales, se emplean las señales de voz y las representaciones escritas que componen la base de datos, tanto de la voz base como de la que se desea adaptar. Con estos datos se realiza una estimación inicial de los parámetros de cada modelo λ (pitch, espectro, y duración). La estimación de los modelos consiste en un cálculo de medias, varianzas y matrices de covarianza. En posteriores etapas se realizan reestimaciones sobre estos valores para precisar los modelos finales.

El modelo obtenido en esta etapa procede de la segmentación temporal de los parámetros de voz. Los parámetros de los estados de los HMMs se obtienen de la partición de los fonemas en partes iguales, como se muestra en la Figura 2.6. Esta partición equitativa no resulta ser la óptima, como puede observarse en el estado 3 de la Figura 2.6. Para que la partición fuese más precisa las fronteras deberían estar mejor situadas (por ejemplo, la frontera marcada con * podría estar más a la derecha).

2.3.2 Mejora de los modelos iniciales mediante el algoritmo de Viterbi.

Como hemos visto en la sección anterior, se requiere mejorar las fronteras entre estados, para ello se deben ajustar los límites de los estados del modelo inicial. Para

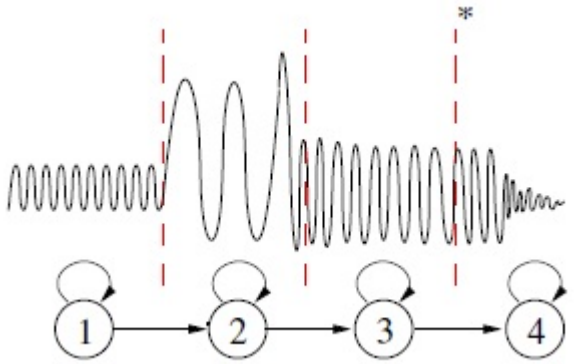

Figura 2.6. Segmentación en partes iguales de un fonema.

este propósito se emplean algoritmos iterativos sobre los parámetros asociados a cada fonema, teniendo en cuenta el modelo inicial generado con anterioridad.

El objetivo es mejorar el modelo inicial, λ_o, y determinar la secuencia de estados $S = (s_1, s_2, ..., s_T)$ que mejor se adecúa a la secuencia de tramas del fonema caso de estudio, $\underline{\underline{O}} = (\underline{o}_1, \underline{o}_2, ..., \underline{o}_T)$. Esta relación entre estados y tramas se puede visualizar en la Figura 2.7. Una vez ajustada esta relación, se vuelven a calcular las medias y matrices de covarianza de los observables de salida de cada estado. Para realizar todo este proceso se emplea de forma recursiva el algoritmo de Viterbi [2][25].

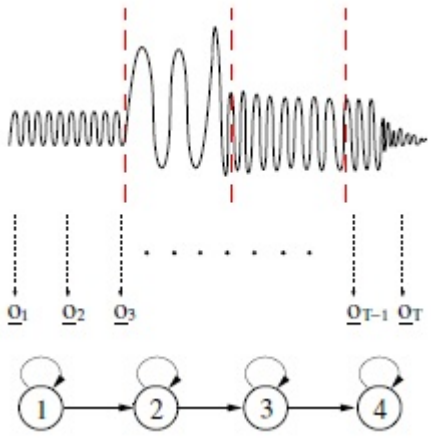

Figura 2.7. Relación entre estados y tramas.

El algoritmo de Viterbi fue propuesto en 1967 como un método de decodificación de códigos convolucionales, se trata de una solución recursiva al problema de estimar la secuencia finita de estados en tiempo discreto de un

proceso de Markov. Recordamos que un proceso de Markov es aquel en el que la probabilidad de estar en el estado x_{k+1} en el tiempo *k+1*, depende sólo del estado x_k en el tiempo k:

$$P(x_{k+1}|x_0, x_1, ..., x_k) = P(x_{k+1}|x_k) \tag{2.18}$$

Aplicando el algoritmo de Viterbi se busca la secuencia de estados para la cual $P(S|\underline{\underline{O}}, \lambda)$ es máxima, que es equivalente a maximizar $P(S, \underline{\underline{O}}|\lambda)$, puesto que

$$P(S|\underline{\underline{O}}, \lambda) = P(\underline{\underline{O}}|\lambda)P(S, \underline{\underline{O}}|\lambda) \tag{2.19}$$

Las probabilidades de transición entre estados, en nuestro caso, son entre estados consecutivos y restringen la secuencia de estados obtenida.

El algoritmo de Viterbi segmenta cada secuencia de entrenamiento $\underline{\underline{O}}$ usando un procedimiento de alineación de estados que resulta de maximizar

$$\phi_N(T) = \max_i \phi_i(T)a_{iN} \tag{2.20}$$

para $1 < i < N$ donde

$$\phi_j(t) = [\max_i \phi_i(t-1)a_{ij}]b_j(\underline{o}_t) \tag{2.21}$$

con condiciones iniciales dadas por

$$\phi_1(1) = 1 \tag{2.22}$$

$$\phi_j(1) = a_{1j}b_j(\underline{o}_1) \tag{2.23}$$

para $1 < j < N$.

2.3.3 Refinamiento de los modelos aplicando el algoritmo de Baum-Welch.

Al aplicar el algoritmo de Viterbi, se obtiene un modelo que se aproxima mejor estadísticamente, a la voz original, que el modelo inicial. Sin embargo, el algoritmo de Viterbi emplea únicamente una secuencia de estados para realizar las estimaciones, mientras que el algoritmo de Baum-Welch [7][13][19] considera todos los posibles alineamientos entre estados y observaciones, no solamente la mejor de estas posibilidades, como es el caso del algoritmo de Viterbi. En vez de asignar cada vector de observables a un estado específico, cada observación es asignada a cada estado proporcionalmente a la probabilidad de permanecer en él.

A continuación se detalla el algoritmo. $L_j(t)$ denota la probabilidad de estar en el estado j en el tiempo t. Las medias y covarianzas se calculan como sigue

$$\widehat{\mu}_j = \frac{\sum_{t=1}^{T} L_j(t)\underline{o}_t}{\sum_{t=1}^{T} L_j(t)} \tag{2.24}$$

$$\widehat{\Sigma}_j = \frac{\sum_{t=1}^{T} L_j(t)(\underline{o}_t - \mu_j)(\underline{o}_t - \mu_j)'}{\sum_{t=1}^{T} L_j(t)} \tag{2.25}$$

donde los sumatorios en los denominadores se emplean para normalizar.

Por supuesto para aplicar las ecuaciones 2.24 y 2.25 se debe calcular anteriormente la probabilidad $L_j(t)$, lo cual se logra eficientemente empleando el algoritmo "forward-backward"que se detalla a continuación.

La probabilidad forward $\alpha_j(t)$ (probabilidad de observar los primeros t vectores y estar en el estado j en el tiempo t) para el modelo M con N estados se define como

$$\alpha_j(t) = P(\underline{o}_1, \underline{o}_2, ..., \underline{o}_t, x(t) = j|M) \tag{2.26}$$

Esta probabilidad puede ser calculada eficientemente mediante

$$\alpha_j(t) = \left[\sum_{i=2}^{N-1} \alpha_i(t-1)a_{ij}\right] b_j(\underline{o}_t) \tag{2.27}$$

con condiciones iniciales

$$\alpha_1(1) = 1 \tag{2.28}$$

$$\alpha_j(1) = a_{1j} b_j(\underline{o}_1) \tag{2.29}$$

para $1 < j < N$, siendo la condición final

$$\alpha_N(T) = \sum_{i=2}^{N-1} \alpha_i(T)\alpha_{iN} \tag{2.30}$$

La probabilidad backward $\beta_j(t)$ se define como

$$\beta_j(t) = P(\underline{o}_{t+1}, \underline{o}_{t+2}, ..., \underline{o}_T | x(t) = j, M) \tag{2.31}$$

y se calcula como sigue

$$\beta_j(t) = \sum_{j=2}^{N-1} a_{ij} b_j(\underline{o}_{t+1}) \beta_j(t+1) \tag{2.32}$$

con condición inicial

$$\beta_i(T) = a_{iN} \tag{2.33}$$

para $1 < i < N$, siendo la condición final

$$\beta_1(1) = \sum_{j=2}^{N-1} a_{1j} b_j(\underline{o}_1) \beta_j(1) \tag{2.34}$$

por definición

$$P(\underline{\underline{O}}|M) = \alpha_N(T) \tag{2.35}$$

$$\alpha_j(t)\beta_j(t) = P(\underline{\underline{O}}, x(t) = j|M) \tag{2.36}$$

Por último, para calcular la probabilidad $L_j(t)$

$$L_j(t) = P(x(t) = j|\underline{\underline{O}}, M) \tag{2.37}$$

$$= \frac{P(\underline{\underline{O}}, x(t) = j|M)}{P(\underline{\underline{O}}|M)} \tag{2.38}$$

$$= \frac{1}{P}\alpha_j(t)\beta_j(t) \tag{2.39}$$

donde P = $P(\underline{\underline{O}}|M)$.

2.3.4 Técnicas de clustering empleando árboles de decisión.

El siguiente paso en el entrenamiento consiste en emplear los mismos algoritmos (Viterbi y Baum-Welch) pero considerando los fonemas con información contextual. En este tipo de fonemas, se conoce la posición que ocupa el fonema dentro de la frase y cuáles son los fonemas que le acompañan; además puede incluir información de la entonación a la hora de pronunciarlo. Por tanto, son de gran utilidad a la hora de optimizar los modelos en el proceso de entrenamiento.

La cantidad de información que contienen los fonemas contextuales hace que el tamaño de la base de datos sea insuficiente para hacer un uso eficiente de sus características. Para resolver este problema se realiza una agrupación entre los estados de los distintos modelos. De esta manera los datos que se utilizan en la estimación de los parámetros de algún modelo, son empleados para realizar esta misma estimación en un modelo diferente. Este proceso de agrupamiento se conoce como "clustering" y se lleva a cabo con los árboles de decisión construidos mediante la técnica MDL (Minimum Description Length [16]) que permite determinar el árbol óptimo.

La elaboración de estos árboles consiste en la formulación de múltiples preguntas referidas al contexto. En cada nodo se formula la pregunta y de ese nodo saldrán dos ramas, en función de que la respuesta sea afirmativa o negativa, que conducirán a otro nodo y otra pregunta. Así se crearán los árboles para cada uno de los modelos. Por ejemplo, si nos fijamos en la Figura 2.8, la pregunta "R-fricativa?" se realiza en todos los primeros estados de los modelos, en unos modelos la respuesta será afirmativa y en otros negativa, produciendose la agrupación y separación de éstos.

Las preguntas se seleccionan mediante el criterio MDL, que permite escoger la pregunta óptima entre todas las posibles. En este criterio se define un modelo

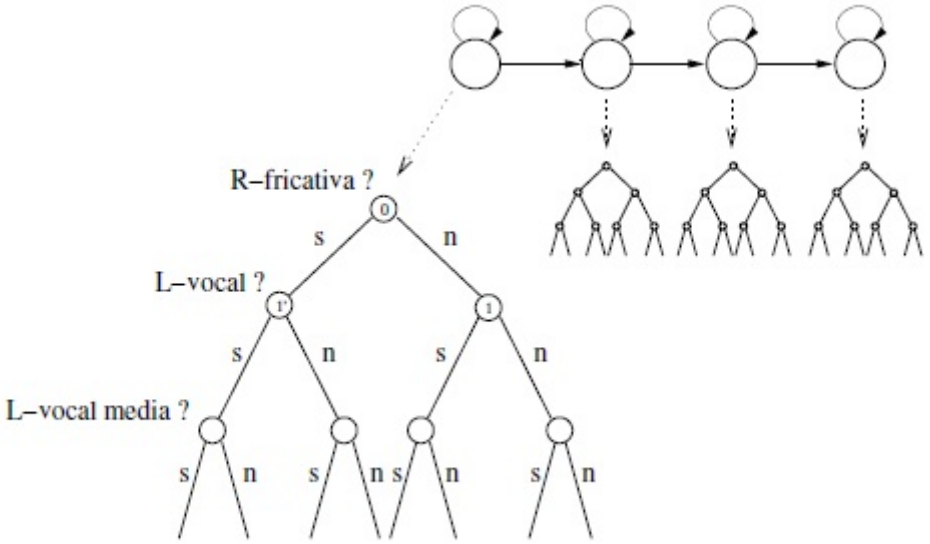

Figura 2.8. Ejemplo de árbol de decisión.

como el conjunto de nodos finales en un árbol de decisión, como se muestra en la Figura 2.9. Para cada nodo se calcula un parámetro conocido como DL (Description Length) y el modelo con la menor DL será seleccionado como el modelo óptimo. Una de las características del método MDL es, que una vez se ha llegado a un óptimo de DL se puede parar el procedimiento, es decir, sólo se seguirán realizando divisiones en nodos mientras el parámetro DL siga mejorando.

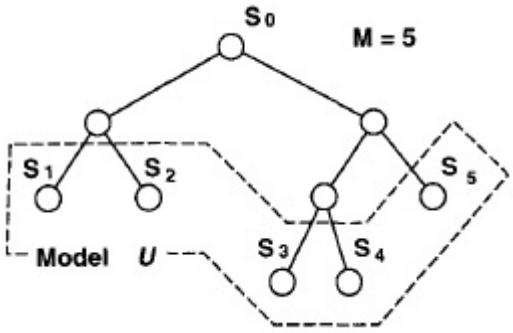

Figura 2.9. Modelo (conjunto de nodos) en un árbol de decisión.

Al final del proceso se tiene un conjunto de ligaduras de nodos o estados, consiguiendo asi reducir el número de parámetros a estimar.

Por otro lado, como los factores asociados al contexto afectan de forma independiente a los parámetros espectrales, pitch y duración, se genera un árbol de decisión para cada uno de los modelos. Una vez generados estos árboles, se vuelven

a realizar reestimaciones sobre los modelos empleando los algoritmos expuestos anteriormente (Viterbi y Baum-Welch) que permitan refinar y mejorar los modelos.

2.4 Técnicas de adaptación.

La adaptación es una técnica empleada para generar modelos acústicos de un locutor del que se dispone poco material para poder realizar el entrenamiento. Para ello, se emplea un modelo generalista, creado a partir de una voz base, y sobre sus modelos se actualizan los parámetros para adaptar al nuevo locutor. De esta manera, se consigue un modelo final monolocutor, utilizando poco material en forma de señales de audio y texto.

Existen varias técnicas para llevar a cabo la adaptación, entre ellas destacan el algoritmo MAP (del inglés, Maximum A Posteriori) y el algoritmo MLLR (Maximum Likelihood Linear Regression). A continuación se describen brevemente ambos métodos. El Apéndice C contiene una exposición más detallada de los procesos matématicos que están detrás de estos algoritmos.

2.4.1 Maximum Likelihood Linear Regression.

El algoritmo MLLR [12][18] toma los datos para la adaptación del nuevo locutor y realiza una actualización de medias y covarianzas de los modelos del locutor independiente (voz base) hasta maximizar la probabilidad de la adaptación. El resto de parámetros no se actualizan hasta que se ha conseguido el objetivo anterior.

Para llevar esto a cabo, se realiza una transformación lineal:

$$\widehat{\underline{\mu}}_{jc} = \underline{\underline{W_c}}\, \underline{\mu}_{jc} \tag{2.40}$$

$$\widehat{\Sigma}_{jc} = \underline{\underline{H}}\, \Sigma_{jc}\, \underline{\underline{H}}^T \tag{2.41}$$

donde $\underline{\mu}_{jc}$ es el vector de medias de la *c-ésima* Gaussiana en el estado *j-ésimo*, $\underline{\underline{W_c}}$ es la matriz de transformación lineal de medias, Σjc la matriz de covarianzas

de la *c-ésima* Gaussiana en el estado *j-ésimo* y $\underline{\underline{H}}$ la matriz de transformación de covarianzas.

El algoritmo MLLR se puede emplear de dos formas distintas. La primera de ellas, es la llamada MLLR global, en la que se adaptan las medias y varianzas en todos los estados de forma global. La segunda forma consiste en dividir las Gaussianas de cada estado en clases de regresión[1] y adaptar cada una de estas clases por separado. Si se disponen de pocos datos para llevar a cabo la adaptación se emplea el método global, mientras que si se aumenta la cantidad de datos la segunda forma de utilizar el algoritmo es la que se aplica.

Para obtener las clases de regresión es necesario generar un árbol de clases de regresión (del inglés, regression class tree)[5]. Un árbol de clases de regresión está formado por una jerarquía de clases de regresión y un grupo de clases base[2], que también pueden ser clases de regresión. Las componentes se agrupan en función de su cercanía en el espacio acústico, sin tener en cuenta a que fonema pertenecen. La Figura 2.10 muestra un ejemplo de un árbol de clases de regresión. En el árbol los nodos terminales se correponden con las clases principales, cada Gaussiana del modelo pertenecerá a una de ellas.

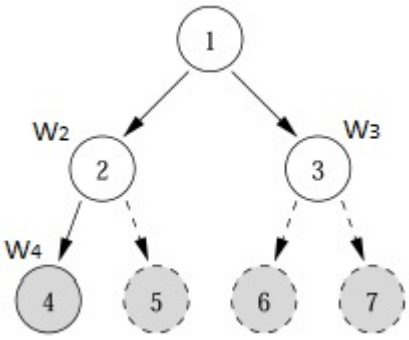

Figura 2.10. Ejemplo de árbol de clases de regresión.

Las flechas y líneas discontinuas indican que no se tienen suficientes datos como para generar una matriz de transformación, mientras que las flechas y líneas continuas se corresponden con nodos terminales que sí poseen suficiente

[1] Clase de regresión: agrupación de parámetros de un modelo.

[2] Clase base: mínima agrupación de parámetros que pueden ser transformados independientemente.

información para llevar a cabo la transformación. En el ejemplo de la Figura 2.10, se generarían transformaciones (W_n) para los nodos 2,3 y 4. Para los casos sin suficientes datos, las clases estarán asociadas a la matriz de transformación del nodo superior.

Así pues esta transformación de cada clase principal de regresión se lleva a cabo de la siguiente manera:

$$\left\{ \begin{array}{l} W_2 \rightarrow \{C_5\} \\ W_3 \rightarrow \{C_6, C_7\} \\ W_4 \rightarrow \{C_4\} \end{array} \right\} \tag{2.42}$$

2.4.2 Maximum A Posteriori.

Este algoritmo de adaptación [6] puede ser visto como una adaptación Bayesiana. Para emplearlo es necesario partir de un modelo inicial, que en el caso que nos ocupa será el del locutor independiente. Combinando este modelo inicial con la nueva información obtenida del locutor al que se quiere adaptar, mediante un factor de adaptación, se estima el parámetro adaptado final.

Matemáticamente, el proceso de adaptación se basa en la siguiente fórmula

$$\widehat{\underline{\mu}}_{jc} = \frac{N_{jc}}{N_{jc} + \tau} \overline{\underline{\mu}}_{jc} + \frac{\tau}{N_{jc} + \tau} \underline{\mu}_{jc} \tag{2.43}$$

donde $\underline{\mu}_{jc}$ es la media del modelo independiente para el estado *j-esimo* y *c-ésima* Gaussiana, $\overline{\underline{\mu}}_{jc}$ la media de los datos adaptados, N_{jc} la probabilidad de ocupación de los datos de adaptación y τ el peso de la información a priori (modelo inicial).

La principal desventaja de este método de adaptación es que requiere una cantidad de datos mayor que MLLR, esto es debido a que se realiza la adaptación de cada componente del modelo. Por contra, cuanta más información se posee, el algoritmo MAP resulta más efectivo que el MLLR.

2.5 Generación de voz.

Una vez finalizado el entrenamiento y adaptación, la síntesis finaliza con la generación de la voz haciendo uso de los modelos generados previamente. El proceso de producción de voz a partir del texto a generar, normalmente introducido por el usuario, y de los modelos obtenidos en el entrenamiento, consta de las siguientes etapas:

1. Normalización del texto, que consiste en la conversión de abreviaturas, fechas, números, etc., en su correspondiente representación fonética.

2. División del texto en unidades prosódicas, marcando la entonación, duración o velocidad de los fonemas.

3. Asignación de los modelos. A partir de la secuencia de fonemas y la representación prosódica, se escoge el modelo generado en el entrenamiento que mejor se adecúa a cada fonema en ese contexto. Entendiendo por contexto la situación dentro de la frase, es decir, si el fonema se encuentra al principio o al final de la oración, entre qué fonemas se encuentra situado, etc.

4. Generación de la secuencia de modelos asociada al contexto. Para esta tarea se emplean árboles de decisión, generados también durante el entrenamiento.

5. Finalmente, el sistema genera la voz asociada a la secuencia de HMMs empleando el modelo digital de producción del habla, excitación más filtro.

2.5.1 Obtención de los parámetros acústicos a partir de los HMM.

Los parámetros acústicos se obtienen mediante la transformacíon de la secuencia de HMMs en una secuencia de tramas con información sobre el pitch y el espectro para cada instante.

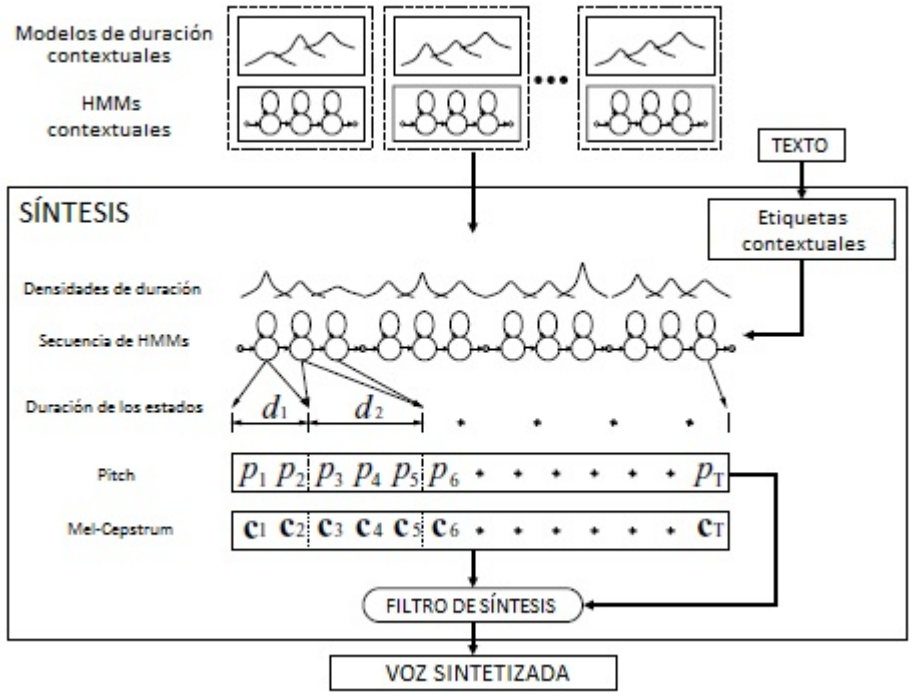

Figura 2.11. Esquema del proceso de generación de voz.

En primer lugar se determina la duración de los modelos de la frase a sintetizar, es decir, de cada uno de los estados que componen los modelos[19]. De esta forma, para una duración T del modelo, el objetivo es obtener la secuencia de estados, $S = (s_1, s_2, ..., s_T)$, que maximiza

$$ln(P(S|\lambda, T)) = \sum_{k=1}^{K} ln(p_k(d_k)) \tag{2.44}$$

con la restricción

$$T = \sum_{k=1}^{K} d_k \tag{2.45}$$

donde $p_k(d_k)$ es la probabilidad de que el estado k dure d_k, siendo K el número de estados del modelo λ. Cada probabilidad $p_k(d_k)$ se modela como una Gaussiana de media $\xi(k)$ y varianza $\sigma^2(k)$. La secuencia $\left\{(d_k)_{k=1}^{K}\right\}$ que maximiza 2.44 se obtiene

$$d_k = \xi(k) + \rho \cdot \sigma^2(k) \tag{2.46}$$

$$\rho = \frac{T - \sum_{k=1}^{K} \xi(k)}{\sum_{k=1}^{K} \sigma^2(k)} \tag{2.47}$$

Se observa en 2.47 que el parámetro ρ se encuentra relacionado con la duración T. Esto facilitará controlar la duración mediante este parámetro en vez de directamente a través de T . Por ejemplo si $\rho = 0$ las duraciones que se asignarán a cada estado serán las duraciones medias obtenidas en el entrenamiento.

El siguiente paso a dar consiste en imponer la duración de los estados de cada modelo y asignarles el número de tramas correspondiente. Está asignación de duración se aplica a los modelos para el pitch y los coeficientes espectrales y se traduce en que el sistema permanecerá en un estado tantas tramas como duración se le haya impuesto. En la Figura 2.12 se observa esta asignación, donde para mayor claridad se muestran las tramas sin solapar.

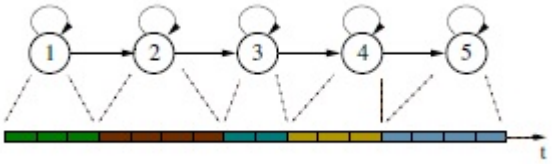

Figura 2.12. Asignación de duración y tramas a cada estado de un modelo .

Una vez obtenida la secuencia de estados y asignación de tramas, hay que obtener los parámetros asociados al pitch y al espectro [9] [24]. A continuación se detalla el proceso para determinar los coeficientes espectrales, el proceso para el pitch es análogo.

Este proceso busca encontrar la observación $\underline{\underline{O}} = (\underline{o}_1^T, \underline{o}_2^T, ..., \underline{o}_T^T)^T$ que maximice su probabilidad para una secuencia de estados, $S = \{(s_1, i_1), (s_2, i_2)..., (s_T, i_T)\}$, y un modelo, λ. Para ello se debe maximizar $P(\underline{\underline{O}}|S, \lambda)$ respecto a $\underline{\underline{O}}$. El vector de parámetros $\underline{o}_t$ consiste en un vector de características estáticas (coeficientes cepstrales) $\underline{c}_t = [\underline{c}_t(1), \underline{c}_t(2), ..., \underline{c}_t(M)]^T$ y vectores de características dinámicas

$$\Delta \underline{c}_t = \sum_{\tau=-L_-^1}^{L_+^1} \omega^{(1)}(\tau) c_{t+\tau} \tag{2.48}$$

$$\Delta^2 \underline{c}_t = \sum_{\tau=-L_-^2}^{L_+^2} \omega^{(2)}(\tau) c_{t+\tau} \tag{2.49}$$

de modo que $\underline{o}_t = \left[\underline{c}_t^T, \Delta \underline{c}_t^T, \Delta^2 \underline{c}_t^T\right]$.

A continuación se muestra el proceso de maximización para un estado s y una sola mezcla i. El logaritmo de $P(\underline{\underline{O}}|S,\lambda)$ puede expresarse

$$lnP(\underline{\underline{O}}|S,\lambda) = -\frac{1}{2}\underline{\underline{O}}^T \underline{\underline{U}}^{-1} \underline{\underline{O}} + \underline{\underline{O}}^T \underline{\underline{U}}^{-1} \underline{\underline{M}} + K \tag{2.50}$$

donde

$$\underline{\underline{U}}^{-1} = diag\left[\underline{\underline{U}}_{s_1,i_1}^{-1}, \underline{\underline{U}}_{s_2,i_2}^{-1}, ..., \underline{\underline{U}}_{s_T,i_T}^{-1}\right] \tag{2.51}$$

$$\underline{\underline{M}} = \left[\underline{\mu}_{s_1,i_1}^T, \underline{\mu}_{s_2,i_2}^T, ..., \underline{\mu}_{s_T,i_T}^T\right]^T \tag{2.52}$$

siendo $\underline{\mu}_{s_t,i_t}$ y $\underline{\underline{U}}_{s_t,i_t}$ el vector de medias de dimensión 3M×1 y la matriz de covarianza 3M×3M respectivamente.

Es fácil observar que la solución trivial se alcanza cuando $\underline{\underline{O}} = \underline{\underline{M}}$, esto es sin tener en cuenta 2.48 y 2.49, ambas restricciones se pueden expresar de forma matricial

$$\underline{\underline{O}} = \underline{\underline{W}} \cdot \underline{\underline{C}} \tag{2.53}$$

donde

$$\underline{\underline{C}} = [\underline{c}_1, \underline{c}_2, ..., \underline{c}_t]^T \tag{2.54}$$

$$\underline{\underline{W}} = [\underline{\omega}_1, \underline{\omega}_2, ..., \underline{\omega}_t]^T \tag{2.55}$$

$$\underline{\omega}_t = \left[\underline{\underline{\omega}}_t^0, \underline{\underline{\omega}}_t^1, \underline{\underline{\omega}}_t^2\right] \tag{2.56}$$

Donde cada $\underline{\underline{\omega}}_t^n$ tiene la siguiente forma

$$\underline{\underline{\omega}}_t^{(n)} = \left[\underbrace{0_{NxN}}_{primero}, ..., 0_{NxN}, \underbrace{\omega^{(n)}(-L_{inf}^{(n)})\underline{\underline{I}}_{NxN}}_{t-L_{inf}^{(n)}\ esimo}, ..., \underbrace{\omega^{(n)}(0)\underline{\underline{I}}_{NxN}}_{t\ esimo}, ..., \underbrace{\omega^{(n)}(L_{sup}^{(n)})\underline{\underline{I}}_{NxN}}_{t+L_{sup}^{(n)}\ esimo}, 0_{NxN}, ..., \underbrace{0_{NxN}}_{T\ esimo}\right]^T$$

Conforme a 2.53, maximzar $P(\underline{\underline{O}}|S,\lambda)$ respecto a $\underline{\underline{O}}$ es equivalente a hacerlo

respecto a $\underline{\underline{C}}$, por tanto

$$\frac{\partial P(\underline{\underline{W}}\,\underline{\underline{C}}|S,\lambda)}{\partial \underline{\underline{C}}} = 0 \tag{2.57}$$

obteniéndose el siguiente sistema de ecuaciones

$$\underline{\underline{W}}^T\underline{\underline{U}}^{-1}\underline{\underline{W}}\,\underline{\underline{C}} = \underline{\underline{W}}^T\underline{\underline{U}}^{-1}\underline{\underline{M}}^T \tag{2.58}$$

A partir de 2.58 se pueden obtener los coeficientes espectrales que maximizan $P(\underline{\underline{O}}|S,\lambda)$. Para resolver el sistema se necesitan un número de operaciones del orden de T^3N^3, para reducir este número de operaciones se recurre a la descomposición de Cholesky.

Anteriormente se ha indicado que el proceso para calcular los parámetros del pitch sería análogo, sin embargo, hay que puntualizar que para que así sea se debe distinguir entre sonidos sordos y sonoros, cuyas peculiaridades se han explicado previamente. Mediante árboles de decisión se realiza esta separación para después aplicar este proceso para los sonidos sonoros.

Existen otros dos métodos para la obtención de los coeficientes espectrales y del pitch que consisten en maximizar $P(\underline{\underline{O}}|S,\lambda)$ respecto a $\underline{\underline{O}}$ y a S, o en maximizar $P(\underline{\underline{O}}|\lambda)$ respecto a $\underline{\underline{O}}$. Ambos métodos vienen detallados en [24].

2.5.2 Obtención de voz a partir de los parámetros acústicos.

Una vez obtenidos los parámetros, el siguiente paso consiste en generar señales de voz a partir de ellos. El modelo empleado para producir dichas señales se representa en la Figura 2.13.

El proceso se puede dividir en dos sistemas bien diferenciados: la excitación y el filtrado. En la parte de excitación se introduce un tren de impulsos cuya separación temporal es igual al periodo del pitch ($\tau = 1/f_o$) y un ruido, con ello se pretende simular la existencia o no de pitch (sonidos sonoros o sordos) respectivamente. En la parte correspondiente al filtrado se genera la voz a partir de la excitación, empleando un filtro que modela el tracto vocal del locutor.

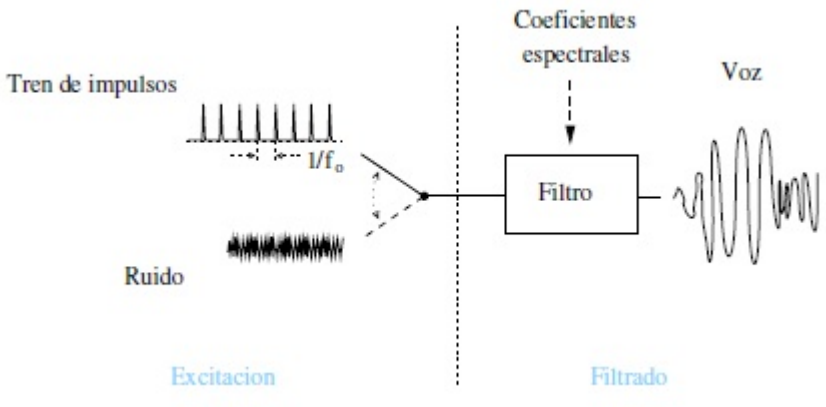

Figura 2.13. Modelo de generación de voz sintetizada.

A continuación se expone más en detalle la generación de este filtro de síntesis mediante coeficientes espectrales. Para la realización de este filtro se emplea el método MLSA (Mel Log Spectrum Approximation [9][4]).

La expresión para la función de transferencia del filtro MLSA es

$$H(z) = e^{F(z)} \tag{2.59}$$

donde $F(z)$ es la función de transferencia del filtro "básico", si éste resulta estable, el filtro MLSA será estable y de fase mínima.

Idealmente está función de transferencia toma la forma

$$F(z) = \sum_{n=0}^{N} \widetilde{c}(n) z^{-n} \tag{2.60}$$

donde $\widetilde{c}(n)$ representa los coeficentes del cepstrum en la escala de Mel y z representa el plano Z correspondiente a la escala de Mel. El logaritmo de la función de transferencia del filtro MLSA se expresa

$$ln|H(z)| = \sum_{n=0}^{N} \widetilde{c}(n) cos(n\widetilde{\omega}) \tag{2.61}$$

El filtro MLSA presenta la mejor aproximación para la envolvente del espectro en magnitudes logarítmicas. Ésto se debe a que en el proceso de obtención de los coeficientes del cepstrum, a partir de las señales de voz (explicado en la página 11),

se persigue que la expresión 2.2 se aproxime lo máximo posible al espectro de la propia señal de voz.

Sin embargo, este filtro resulta irrealizable en la práctica y por tanto se realiza una aproximación mediante una función de transferencia racional empleando la aproximación de Padé [9]. Mediante esta aproximación se consigue que $|H(z)|$ sea lo más semejante posible a la envolvente del espectro en la escala de Mel.

Capítulo 3

Entrenamiento de los modelos ocultos de Markov.

HTS (HMM-based Speech Synthesis System [22]) es un sistema de síntesis basado en HMMs que modifica el sistema de reconocimiento HTK (Hidden Markov Model Toolkit). HTK consiste en un conjunto de aplicaciones y librerias que son utilizadas para reconocimiento de voz, con la herramienta HTS se introducen modificaciones que permiten emplearla para desarrollar sistemas de síntesis.

HTS no cuenta con una herramienta para analizar texto, para ello se sirve de otras aplicaciones como Festival u Open JTalk. En este proyecto se emplea Festival, que nos permitirá transformar los ficheros de transcripciones para realizar el entrenamiento de los modelos.

A lo largo de este capítulo se explica el proceso de entrenamiento: el tratamiento y transformación de los archivos de audio y texto, y la obtención de los modelos para los parámetros espectrales, pitch y duración tanto para la voz base como para la voz adaptada final. Al final del proceso quedarán una serie de ficheros con los modelos, árboles de decisión y ventanas que nos permitirán generar la voz artificial.

3.1 Datos iniciales.

Una base de datos completa y equilibrada es muy importante para poder realizar correctamente el proceso de entrenamiento de los HMMs. Dicha base de

datos consta de archivos de audio y sus correspondientes transcripciones escritas. El entrenamiento adaptativo que se va a realizar en este proyecto requiere de dos bases de datos bien diferenciadas: voz base y voz a adaptar.

Por un lado, el conjunto de datos para la voz base. En nuestro caso disponemos de señales sonoras y transcripciones de varios locutores extraidos de la base de datos Albayzin. En total, 84 locutores con 25, 50 y 200 locuciones dependiendo del caso. La base de datos Albayzin nos garantiza un conjunto de locuciones fonéticamente balanceadas, ya que los contextos que se consideran relevantes en el castellano (contextos que ocurren al menos el 10 % de las apariciones del sonido) están presentes al menos 4 veces.

Por otro lado, para realizar la adaptación se han elegido 25 frases del conjunto de Albayzin, seleccionadas de uno de los locutores empleados para la voz base. Con estas frases se procede a grabar la voz del locutor con la ayuda de un micrófono.

El formato de los archivos de audio es *.raw*, caracterizado por una frecuencia de muestreo de 16kHz, 16 bits por muestra y un solo canal. En cuanto a las transcripciones, éstas sufren un proceso de transformación para poder ser finalmente empleadas en el entrenamiento. Inicialmente se tienen las frases almacenadas en archivos de texto y serán transformadas al formato empleado por Festival (sistema TTS multilingüe), utilizado en este proyecto, *.utt*. En próximas secciones se entrará más en detalle en esta transformación.

Otros elementos importantes en los datos iniciales son los ficheros *questions_qst.hed* y *questions_utt_qst.hed*. El primero contiene un conjunto de posibles preguntas que se emplean para la elaboración de los árboles de decisión y el segundo se emplea para la elaboración de las etiquetas a partir de los ficheros *.utt*. Estas preguntas son dependientes del idioma, es decir, no se pueden utilizar los mismos archivos para generar voces en inglés que en castellano. A continuación aparece una pequeña muestra del archivo *questions_qst.hed*.

$QS \quad "LL\_b" \quad \{b\hat{}*\}$

$QS \quad "LL\_Nasal" \quad \{m\hat{}*, n\hat{}*, ny\hat{}*\}$

$QS \quad "Pos\_C - Word\_in\_C - Phrase(Bw) == 8" \quad \{* + 8\&*\}$

La estructura de estas preguntas es la siguiente: QS "name" {*condition*},

"name" identifica a la pregunta dentro del fichero y "condition" representa la condición que debe cumplirse para que esa pregunta tenga resultado positivo. Por ejemplo, la primera pregunta será cierta si se cumple la condición bˆ∗, es decir, si el fonema previo al anterior es /*b*/. La segunda tendrá resultado positivo si estamos ante una frase de tipo "*LL_Nasal*" y la tercera si la palabra actual ocupa la octava posición en la frase empezando a contar desde el final. La sintaxis empleada viene explicada en el Apéndice B.

3.2 Preparación de los datos.

3.2.1 Señales de audio

Para realizar el entrenamiento no se emplean las señales de audio en formato *.raw*, sino que se realiza una transformación para obtener, por un lado, los coeficientes cepstrum en las frecuencias de Mel (MFCC) y por otro lado, el pitch. Durante el proceso de transformación se generan una serie de archivos, siempre uno por cada oración contenida en la base de datos. La preparación de los datos se realiza tanto para la voz base como para la voz a adaptar.

El primer paso para obtener los MFCCs es transformar la señal de audio empleando una ventana deslizante. Se pueden emplear distintos tipos de ventana (Blackman, Hamming o Hanning), siendo la de Hamming la empleada por defecto. De esta transformación se obtienen ficheros con extensión *.mgc* que contienen un total de 25 coeficientes cepstrum para cada trama, con sus correspondientes coeficientes dinámicos.

Una vez obtenidos los parámetros cepstrum, el siguiente paso a dar es obtener el pitch. Para cada señal de voz se produce un fichero con extensión *.lf0* que contiene los valores de pitch ($\log(f_o)$) para cada trama de la señal. Dependiendo del valor que tome $\log(f_o)$ estaremos ante un sonido sordo o sonoro.

El último paso de este proceso consiste en agrupar en un mismo fichero los coeficientes cepstrum y los valores de pitch. Para ello, se genera un fichero con extensión *.cmp* que contendrá el pitch y los coeficientes espectrales, junto a sus

respectivos coeficientes dinámicos, como se muestra en la Figura 3.1. Una vez generados estos ficheros, uno para cada oración, la parte que corresponde a las señales de audio está lista para ser entrenada.

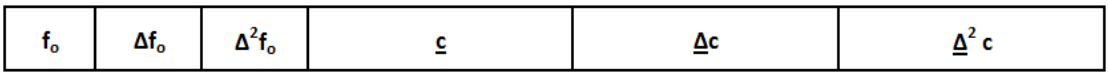

Figura 3.1. Distribución de un archivo .cmp para una trama.

3.2.2 Ficheros de texto

La situación con los ficheros de texto es análoga a los archivos de audio, se deben procesar para poder emplearlos en el entrenamiento de los HMMs.

Se parte de unos archivos en formato *.txt*, cada uno de estos ficheros contiene una de las frases de la base de datos, que se corresponden con un fichero de audio. Las oraciones en estos ficheros no incluyen tildes, signos de puntuación, ni mayúsculas y el caracter $\tilde{n}$ se sustituye por $\sim n$.

La primera transformación consiste en pasar de formato *.txt* a *.utt*. Este formato es el empleado por *Festival* para representar las oraciones mediante texto. El proceso de transformación consta de diversos pasos. En primer lugar, se divide la oración en unidades más sencillas, en este caso palabras. A continuación, se realiza una identificación de los tipos de palabra, separando fechas, números, años, etc. Por último, se trata la prosodia, pronunciación, entonación y duración de cada unidad.

Un ejemplo del comando a emplear para generar un archivo *.utt* a partir de una frase es el siguiente:

```
festival --language spanish -b "(utt.save (SynthText \"Hola mundo\") \"prueba.utt\")"
```

Una vez obtenidos los ficheros *.utt*, el siguiente paso de la preparación de los datos de texto es la elaboración de los archivos de etiquetas *.lab*. Cada línea de estos ficheros representa un fonema y toda la información asociada a él (más detalles sobre el formato en el Apéndice B). Se generan dos archivos *.lab* por cada fichero

.txt. El primero de ellos contiene toda la información relacionada con el contexto del fonema como es la duración, los fonemas que le preceden y suceden o la posición dentro de la frase. Estas son las etiquetas denominadas *full*. El segundo fichero generado contiene sólo los fonemas y su duración correspondiente, son las etiquetas *mono*.

Por último se generan diversos ficheros, estos son:

- 2 Master Label Files (MLF) que contienen los directorios donde se encuentran las etiquetas *mono* y *full*..
- 2 listas que aglutinan todos los fonemas de la base de datos (*mono.list*) y todos los fonemas en versión extendida con información contextual (*full.list*).
- 2 ficheros *.scp*: *train.scp* y *adapt.scp*. Contienen la lista de ficheros con extensión *.cmp* (path absoluto más nombre del fichero) que serán empleados en el entrenamiento de la voz base y de la adaptada respectivamente.

3.3 Entrenamiento de los HMMs con HTS para la voz base.

El proceso de entrenamiento de la voz base sigue diversas etapas como se muestra en la Figura 3.2. A continuación se detalla cada parte de este entrenamiento explicando la labor que realizan cada una de las funciones de *HTK/HTS*.

3.3.1 Obtención de los modelos iniciales.

El primer paso del entrenamiento consiste en la elaboración de los modelos iniciales a partir de los datos preparados previamente.

En primer lugar, se calcula la media y varianza global de los parámetros de voz del locutor independiente. Esta tarea la lleva a cabo la función *HCompV*, que toma como ficheros de entrada los archivos con extensión *.cmp* y genera un fichero denominado *init.mmf* que contiene los valores de las varianzas globales. Estos

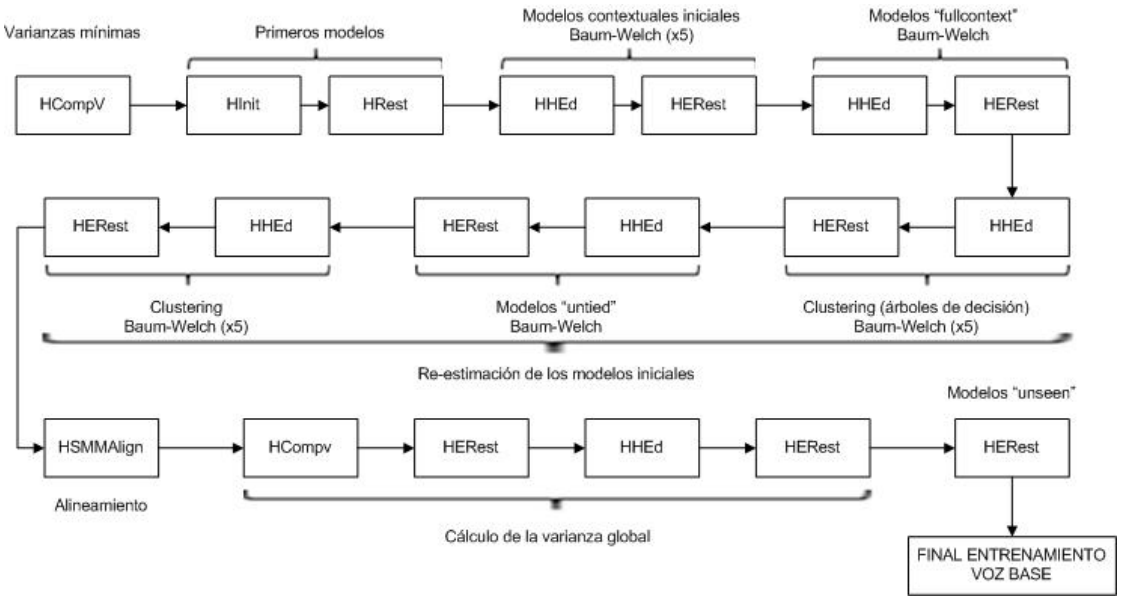

Figura 3.2. Esquema del proceso de entrenamiento de la voz base.

valores sirven de punto de partida para iterar sobre ellos durante el entrenamiento de los modelos.

El siguiente paso consiste en elaborar los primeros modelos, para ello se hace uso de la función *HInit*. Las principales etapas (Figura 3.3) en la estimación de estos primeros modelos son: segmentación uniforme y alineamiento de Viterbi. Con la segmentación uniforme cada aparición del fonema se segmenta en partes iguales (cinco partes correspondientes con cada estado de los modelos) y se realiza una primera estimación en la que cada estado recibe sus parámetros estadísticos de salida (medias, varianzas y covarianzas). Las varianzas de estos primeros estados tienen como cota mínima las calculadas en el paso anterior y que se encuentran en el fichero *init.mmf*.

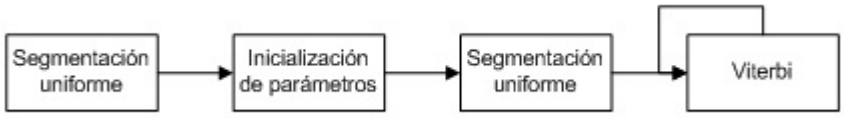

Figura 3.3. Esquema de la función HInit.

Una vez realizada la segmentación uniforme se aplica el algoritmo de Viterbi. El objetivo es mejorar la estimación obtenida en el paso anterior y elaborar la matriz de probabilidades de transición entre estados. Se leen las secuencias de vectores

$\underline{\underline{O}} = (\underline{o}_1, \underline{o}_2, ..., \underline{o}_T)$ de todas las realizaciones del fonema sobre el que se va a aplicar el algoritmo. Se aplica Viterbi en cada lectura, obteniendo la secuencia de estados más probable. Las probabilidades de transición se calculan a partir del número de veces que se visita un estado durante el proceso de alineamiento (modelo de estados "left-to-right"). El proceso termina una vez realizado un número de iteraciones predeterminado o hasta que no se obtiene una mejora significativa tras la anterior iteración.

Por último, se emplea la función *HRest*, que aplica el algoritmo de Baum-Welch para reestimar los modelos iniciales de los fonemas aislados (sin información contextual). El funcionamiento es similar al de *HInit*, pero en vez de partir del prototipo se parte ya de los modelos iniciales generados anteriormente y en vez del algoritmo de Viterbi se utiliza el de Baum-Welch. Mientras Viterbi realiza una asociación entre observaciones y estados más severa, Baum-Welch considera que una observación puede proceder de varios estados diferentes con una probabilidad determinada. Esta diferencia se muestra en la Figura 3.4.

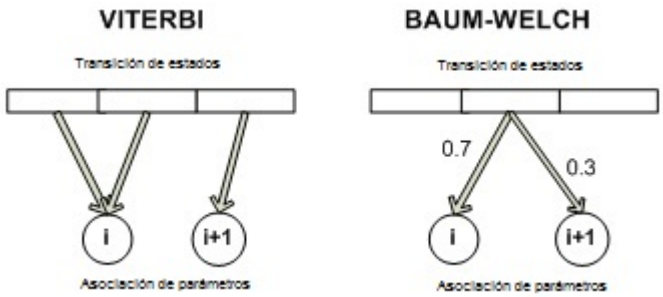

Figura 3.4. Relación observación/estado en los algoritmos de Viterbi y Baum-Welch.

3.3.2 Modelos contextuales.

En la etapa anterior se han elaborado los modelos para los fonemas aislados, en los siguientes pasos se emplean los fonemas con información contextual para elaborar los modelos. Este paso es esencial ya que se consigue una mejora sustancial en los modelos, consiguiéndose que fonemas que se pronuncian de manera distinta no sean tratados por igual, como sucedía en la etapa anterior.

El primer paso consiste en aplicar la función *HHEd*, que permite manipular conjuntos de modelos y realizar nuevas agrupaciones de éstos. Esta función emplea el fichero *full.list* (que contiene la información contextual de todos los fonemas de la base de datos) para asignar los modelos iniciales al fonema. En primera instancia a distintos contextos del mismo fonema se les asignará el modelo inicial de dicho fonema. Por ejemplo para los siguientes contextos del fonema "@":

$$D - @ + s \ // \ k - @ + m \ // \ n - @ + k$$

A todos ellos se les asignará el modelo del fonema "@", con lo cual existe en este punto redundancia de modelos: distintos fonemas contextuales con el mismo modelo. Esta redundancia se reduce mediante un comando de la función *HHEd*, que agrupa todas las versiones del fonema contextual "phone" y se referencian mediante el macro *T_phone*. El comando es el siguiente:

$$TI \ T\_phone \ \{* - phone + *.transP\}$$

El último paso para obtener los modelos contextuales es aplicar el algoritmo de Baum-Welch, pero esta vez a través de la función *HERest*. Aplicando el algoritmo de Baum-Welch de forma iterativa se consigue mejorar los modelos contextuales hasta el punto que fonemas que se pronuncian de manera distinta no sean tratados por igual.

La siguiente etapa del entrenamiento consiste en diversas iteraciones del algoritmo de Baum-Welch empleando una técnica de agrupamiento o "clustering" para mejorar los modelos contextuales.

3.3.3 Re-estimación de los modelos.

Como se mencionó en el Capítulo 2, para que los modelos contextuales que se han obtenido fuesen realmente buenos haría falta una cantidad de datos y señales de voz mucho mayor de la que tenemos. Para solucionar este problema se introdujo el concepto de "clustering". El clustering consiste en establecer ligaduras

entre los modelos que tengan estados parecidos entre sí. Por ejemplo, se podría considerar que los modelos "N-a+s" y "f-a+n" comparten los parámetros de los estados centrales del fonema 'a'. Como ésta, se pueden encontrar otras formas de ligar modelos.

Para realizar estas ligaduras se hace uso de los árboles de decisión, éstos se crean mediante la función *HHEd*. Se generan distintos árboles de decisión para los parámetros espectrales, pitch y duración; ya que el contexto no afecta de igual manera a cada uno de ellos y por tanto, los modelos se agrupan de manera independiente.

Una vez elaborados los árboles de decisión se aplican diversas iteraciones del algoritmo de Baum-Welch a través de la función *HERest*. Durante esta etapa se analiza la frase completa, se ajustan las tramas entre los estados y se reestiman los parámetros de los HMMs. El uso de estos modelos con ligaduras proporciona mayor riqueza y por tanto mejora el proceso, ya que se tiene mayor información en cada estimación. Si nos fijamos en el ejemplo anterior, "N-a+s" y "f-a+n", considerando la ligadura en el estado central: al estimar una realización que contenga la secuencia "-a+s", se tendrán en cuenta todas las realizaciones en las que aparezca tanto "N-a+s" como "f-a+n", lo que resulta, por tanto, en un aumento de los datos a emplear en cada estimación.

Como se aprecia en el esquema de la Figura 3.2 el proceso de reestimación comprende tres etapas en las que se ejecutan *HHEd* y *HERest*. La primera de esas etapas es la explicada anteriormente, creación de los árboles de decisión y clustering.

La segunda etapa consiste en deshacer las ligaduras de los modelos contextuales con *HHEd* y aplicar de nuevo el algoritmo de Baum-Welch sobre estos modelos sin ligaduras mediante *HERest*.

Finalmente, se vuelven a realizar ligaduras y se aplica de nuevo el algoritmo de Baum-Welch. En este punto se generan los modelos de duración ya que se considera que los modelos obtenidos para los parámetros espectrales y el pitch son los suficientemente buenos como para poder obtener los de duración.

3.3.4 Cálculo de la varianza global

Uno de los inconvenientes de este tipo de entrenamiento es que produce "sobresuavizado" (del inglés, over-smoothing) en las características estáticas, lo cual genera un efecto de voz sorda. Para solucionar este problema se aplica el cálculo de la varianza global [20], la cual está incorrelada con este efeco de sobresuavizado. Para llevar a cabo este proceso se emplean las funciones *HCompV*, *HERest* y *HHEd*.

Una vez realizado este paso, finaliza el entrenamiento de la voz base, quedando todo listo para la adaptación.

3.4 Entrenamiento de los HMMs con HTS para la voz adaptada.

La etapa de adaptación se puede dividir en dos tramos: la generación de los árboles de clases de regresión y la ejecución de los algoritmo de adaptación MLLR y MAP. Previamente se requiere una generación de modelos, estos pasos se muestran en la Figura 3.5.

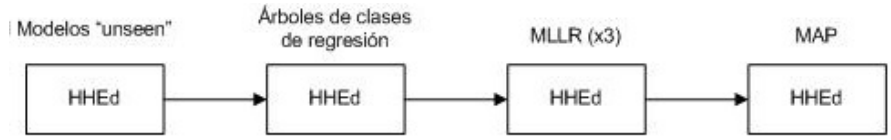

Figura 3.5. Esquema del proceso de entrenamiento de la voz adaptada.

3.4.1 Árboles de clases de regresión.

En el capítulo 2 se ha explicado el concepto de árbol de clases de regresión, básicamente consiste en agrupar las componentes en función de su proximidad en el espacio sonoro, así estas componentes pueden ser tratadas de forma parecida.

Llegados a este punto hay que destacar que, aunque los árboles de clases de regresión son empleados exclusivamente para la adaptación, como herramienta para aplicar el algoritmo MLLR, éstos se generan a partir de la base de datos del

locutor independiente, es decir, de la voz base. Por tanto son independientes de la voz a adaptar y de cualquier nuevo locutor que se introduzca. Se han incluido en esta sección porque se considera que van intimamente ligados al algoritmo MLLR y por tanto, del proceso de adaptación.

Para generar estos árboles se emplea la herramienta *HHEd* que genera un árbol de clases de regresión binario y etiqueta cada componente [25]. En la Figura 3.6 se muestra un ejemplo de clases de regresión y un árbol de clases de regresión generado por *HHEd*. En el árbol se puede observar que se emplea las cuatro clases del macro "*baseclass_4.base*".

CLASE DE REGRESIÓN

```
~b ‘‘baseclass_4.base’’
<MMFIDMASK> CUED_WSJ*
<PARAMETERS> MIXBASE
<NUMCLASSES> 4
  <CLASS> 1  {(one,sil).state[2-4].mix[1-12]}
  <CLASS> 2  {two.state[2-4].mix[1-12]}
  <CLASS> 3  {three.state[2-4].mix[1-12]}
  <CLASS> 4  {four.state[2-4].mix[1-12]}
```

ÁRBOL DE CLASES DE REGRESIÓN

```
~r "regtree_4.tree"
<BASECLASS>~b "baseclass_4.base"
<NODE> 1 2 2 3
<NODE> 2 2 4 5
<NODE> 3 2 6 7
<TNODE> 4 1 1
<TNODE> 5 1 2
<TNODE> 6 1 3
<TNODE> 7 1 4
```

Figura 3.6. Ejemplo de clases de regresión y árbol de clases de regresión.

El proceso seguido para la generación de los árboles se puede resumir en estos cinco puntos:

1. Selección de un nodo terminal a partir del cual ramificar.
2. Cálculo de medias y varianzas de las mezclas en ese nodo.
3. Creación de dos nodos hijo con la misma media que el padre y una fracción de la varianza.
4. Asignación de una componente a cada hijo aplicando una medida Euclidea.
5. Se recalcula la media del nodo hijo de acuerdo a la componente asignada.

Una vez finalizado este proceso habríamos generado los modelos para la voz base, se trata de un proceso costoso en tiempo debido a la gran cantidad de cálculos y el tamaño de la base de datos [23]. En nuestro caso el proceso dura en torno a

las 12 horas, por eso es importante generar la voz base una única vez y almacenar estos modelos para poder realizar la adaptación rapidamente.

3.4.2 Entrenamiento adaptativo: MLLR + MAP.

La última etapa del entrenamiento consiste en el empleo de los algoritmos de adaptación MLLR y MAP. Como ya han sido explicados en detalle en el Capítulo 2, a continuación se expone brevemente el funcionamiento de éstos y el uso que hacen de los árboles de clases de regresión.

El siguiente paso en el proceso de adaptación consiste en la aplicación del algoritmo MLLR (Maximum Likelihood Linear Regression). Para ejecutar este algoritmo se emplea la función *HERest*. Se leen los árboles de clases de regresión de arriba a abajo, es decir, se comienza en el nodo raiz y se va recorriendo el árbol generando transformadas sólo en aquellos nodos que cumplen las siguientes premisas:

- Que tengan suficiente información.
- Que sean nodos terminales o que tengan nodos hijo sin suficiente información.

El algoritmo se ejecuta de forma iterativa, hasta tres veces en nuestro caso, para ir refinando los resultados.

El último paso del proceso de adaptación consiste en aplicar el algoritmo MAP (Maximum A Posteriori) para mejorar la estimación y los modelos obtenidos con el algoritmo MLLR. Sólo se realiza una ejecución de este algoritmo.

3.5 Conversión de los modelos a formato hts_engine.

Una vez generados los modelos para el locutor adaptado se procede a convertirlos al formato empleado por la herramienta de síntesis *hts_engine*. Para llevar a cabo esta transformación se emplea la función *HHEd*, que sirve para

editar archivos. El resultado de esta conversión son los siguientes ficheros, que se emplearán posteriormente para la generación de voz:

- Modelos contextuales: lf0.pdf, dur.pdf y mgc.pdf.
- Árboles de decisión: tree-lf0.inf, tree-dur.inf y tree-mgc.inf.
- Varianza global: gv-lf0.pdf, tree-gv-lf0.inf, gv-mgc.pdf, tree-gv-mgc.inf y gv-switch.inf.
- Ventanas: lf0.win y mgc.win.
- Filro paso bajo: lpf.pdf, tree-lpf.inf y lpf.win.

Capítulo 4

Generación de voz.

El software *hts_engine* es una herramienta empleada para generar voces a partir de HMMs que han sido entrenados mediante el sistema de síntesis HTS.

En la primera parte de este capítulo se explica el proceso de generación de voz empleando la herramienta *hts_engine*, partiendo de la transformación de la frase a sintetizar al formato empleado por HTS para el tratamiento de texto y la posterior asignación de modelos.

En la segunda parte se introduce el sistema empleado por el Grupo de Tecnologías de las Comunicaciones, conocido como *vivoSinte*. Este sistema tiene como base el programa *hts_engine* adaptándolo a las necesidades del idioma castellano.

4.1 Generación de voz con HTS.

El proceso de generación de voz artificial mediante HMMs sigue varias etapas, las cuales se pueden apreciar en la Figura 4.1. El primer paso consiste en la elaboración de las etiquetas, las cuales se generan a partir del texto introducido para ser sintetizado. Posteriormente se seleccionan los modelos que mejor se adaptan al texto introducido, para finalmente generar la señal de audio.

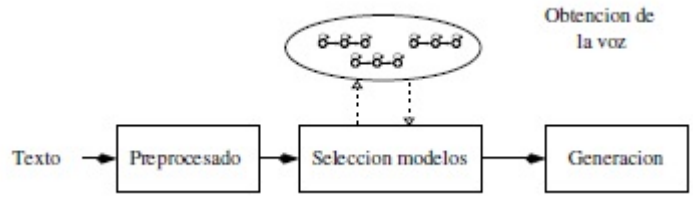

Figura 4.1. Esquema del proceso de generación de voz.

4.1.1 Procesado del texto a sintetizar.

El primer paso es la transformación del texto al formato que emplea HTS, este formato es el mismo que el empleado en el proceso de entrenamiento, en la Figura 4.2 se muestra un esquema simplificado de los pasos a dar.

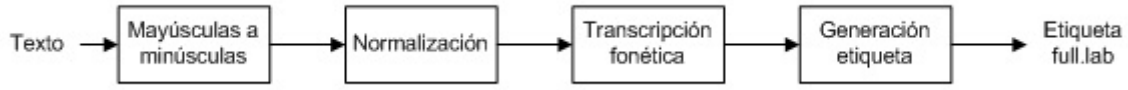

Figura 4.2. Esquema del proceso de transformación del texto a sintetizar.

Para llegar a este formato "etiqueta", primero hay que realizar una normalización del texto, convirtiendo las mayúsculas en minúsculas y realizando la transcripción fonética. Esta transcripción fonética se debe ajustar al formato empleado en HTS para representar los fonemas. Finalmente se generan las etiquetas con información contextual, es decir, información sobre los fonemas anteriores y posteriores al fonema objeto de estudio. A continuación se muestra un ejemplo simplificado de la etiqueta para la palabra "hola" (los detalles sobre el formato se pueden encontrar en el Apéndice B).

$xx\hat{}\#-o1+l=a@1\_1/A:0\_0\_0/B:1-1-1...$

$\#\hat{}o1-l+a=\#@1\_2/A:1\_1\_1/B:0-0-2...$

$o1\hat{}l-a+\#=xx@2\_1/A:1\_1\_1/B:0-0-2...$

$l\hat{}a-\#+xx=xx@xx\_xx/A:0\_0\_2/B:xx-xx-xx...$

Se puede observar fácilmente la información contextual que genera este formato de etiqueta. Por ejemplo centrándonos en la tercera línea, la etiqueta indica que el fonema /l/ va precedido por el fonema /o1/ y le sigue el fonema /a/.

4.1.2 Selección de modelos.

En esta etapa se desea asociar a cada fonema uno de los modelos generados durante el entrenamiento. Esta asociación se realiza a través de los árboles de decisión, nunca a nivel de modelo sino a nivel de estado. El modelo del fonema se construye estado a estado a partir del árbol de decisión, asignando los estados que se adapten mejor a cada fonema. El proceso consiste en recorrer el árbol hasta llegar a un nodo final que asocie un estado del modelo al fonema contextual. El proceso se repite hasta que se tienen todos los estados que componen el modelo.

La clasificación y asignación de modelos la lleva a cabo el programa *hts_engine*. Para poder entender mejor el proceso, a continuación se expone el formato de los ficheros que contienen los modelos. Estos ficheros contienen la información estadística de cada modelo, es decir, de los nodos finales de los árboles de decisión. Los archivos a los que se hace referencia tienen extensión *.pdf*, existiendo uno por cada tipo de parámetro (espectro, pitch y duración).

Los ficheros de modelos están compuestos por una cabecera, con información sobre el número de parámetros y nodos finales de cada estado, y una parte de datos. Un ejemplo de cabecera para cada *.pdf* podría ser:

mcp.pdf $\Rightarrow$ 75 293 306 264 208 254

lf0.pdf $\Rightarrow$ 3 315 279 208 297 233

dur.pdf $\Rightarrow$ 5 367

Centrándonos en la primera cifra de la cabecera, ésta indica el número de observables de salida que emite cada modelo. Cada estado del modelo del espectro emitirá 75 valores, que se corresponden con 25 valores de cada uno de sus tres vectores de salida ($\underline{c}$, $\Delta_{\underline{c}}$ y $\Delta^2_{\underline{c}}$). El pitch por su parte emitirá 3 parámetros (lf0, Δ_{lf0} y Δ^2_{lf0}) y el modelo de duración 5 valores, uno por estado, que indican el tiempo durante el cual se deben emitir los observables de los otros modelos. El resto de cifras de la cabecera se corresponden con el número de nodos finales que tienen los árboles de decisión para cada uno de los estados de los modelos (cinco estados espectro y pitch, y un estado el modelo de duración).

En la Figura 4.3 se muestra la estructura que sigue la parte de datos de los

ficheros *mcp.pdf* y *lf0.pdf* para uno de los cinco estados de los modelos.

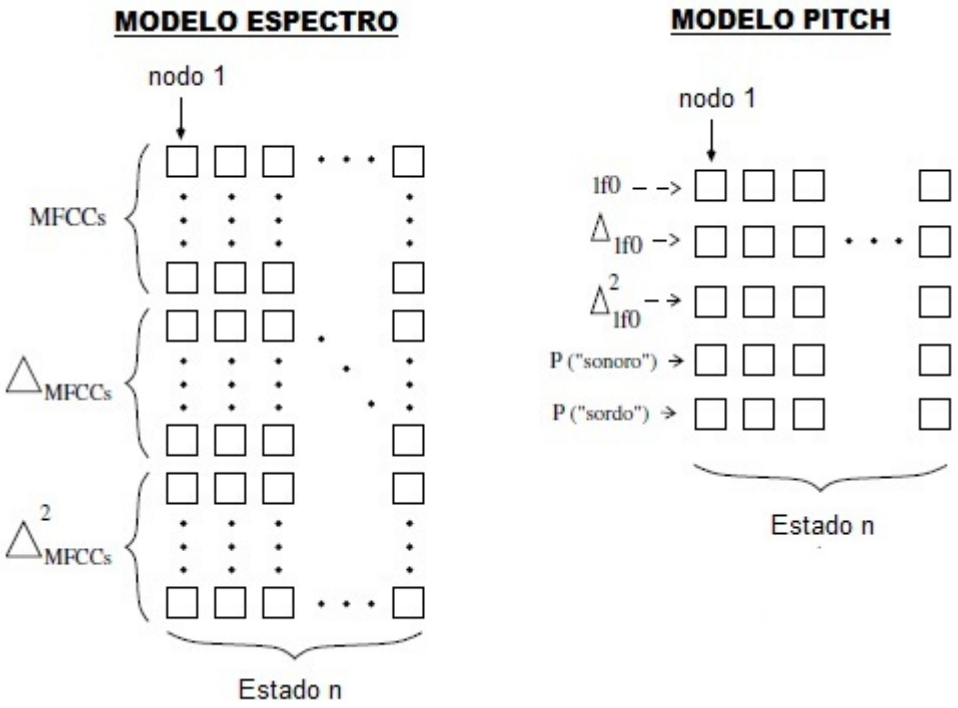

Figura 4.3. Estructura de los ficheros mcp.pdf y lf0.pdf.

Para los modelos del espectro cada columna representa la estadística de salida de cada nodo terminal del árbol de decisión. El modelo del pitch inicialmente sigue la misma estructura y añade información sobre la probabilidad de que el sonido sea sonoro o sordo.

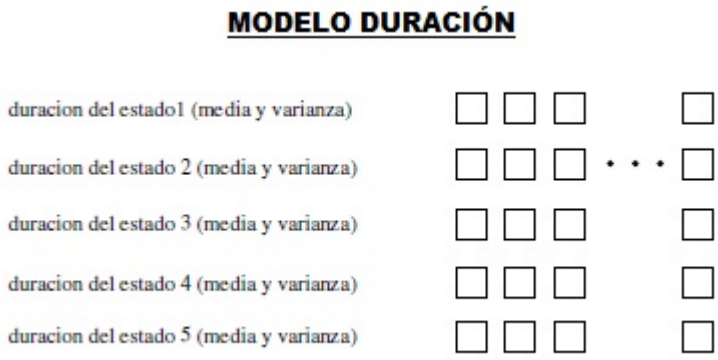

Figura 4.4. Estructura del fichero dur.pdf.

El fichero *dur.pdf* tiene una estructura distinta a los dos anteriores. Como se muestra en la Figura 4.4 no es necesario indexar a nivel de estado ya que este

modelo sólo posee uno. Cada fila representa la duración de cada uno de los cinco estados de los modelos del espectro y el pitch, mientras que las columnas indican el nodo terminal de los árboles de decisión.

4.2 Sistema vivoSinte.

El programa *vivoSinte* es un sistema de generación de voz desarrollado por el Grupo de Tecnologías de las Comunicaciones de la Universidad de Zaragoza basado en la síntesis por HMMs. Hace uso de la herramienta *hts_engine* para asignar y clasificar los modelos a los diferentes fonemas. Dispone de herramientas y funciones que transforman el texto a ser sintetizado al formato empleado por *hts_engine*, tarea que se realiza previamente a la asignación de modelos. A lo largo de esta sección se va a ir detallando todo este proceso.

4.2.1 Inicialización del sistema.

Previamente a la introducción del texto a sintetizar por parte del usuario, se realiza la inicialización del sistema. Esta tarea la lleva a cabo el módulo *initialize_vivoSinte*.

1. Se dispone de una lista con los nombres de los locutores disponibles (archivo *.txt*) a partir de la cual se redirecciona al programa a cada uno de ellos. A su vez, cada locutor dispone de otro archivo *.txt* donde vienen detalladas las rutas en las que se encuentran todos los ficheros obtenidos en el entrenamiento.

2. Se establecen una serie de parámetros para lleva a cabo la generación de voz, entre ellos destacan: la frecuencia de muestreo, el periodo de trama, la tasa de bit, un parámetro de postfiltrado, la elocidad de la locución o el volumen. Algunos de estos parámetros se emplean también durante el proceso de entrenamiento y por tanto deben coincidir los valores para que la generación de voz sea correcta.

3. Se inicializa *hts_engine* que reserva espacio en memoria para las variables que intervienen en el proceso. Se crea una variable, *engine*, que contendrá información de los modelos (número de observables de salida, estados o árboles de decisión asociados), los parámetros anteriormente nombrados y campos inicializados para posteriormente ser modificados durante el proceso de asignación y clasificación.

   Este proceso se repite para cada locutor, almacenando en memoria sus modelos para tener rápido acceso a ellos a la hora de sintetizar y no tener que cargarlos cada vez que se desee cambiar de locutor.

4. El usuario introduce el nombre del locutor cuya voz desea utilizar y la frase a sintetizar. Si el locutor introducido ya se encuentra cargado en memoria, se seleccionan sus modelos y se procede a generar la voz artificial. Si por el contrario el locutor no se encuentra cargado, se repite el paso 3 y se actualiza la lista de locutores.

   En la Figura 4.5 se muestra una captura de pantalla con la adquisición del locutor y el texto a sintetizar, en este caso para un locutor no conocido previamente.

```
Los speakers cargados son:
'rajoy'
'hombre'
Introduzca el speaker y el texto a sintetizar con el siguiente formato:
#SPEAKER TEXTO
#diego prueba de nuevo locutor
speaker: 'diego' texto: 'prueba de nuevo locutor'
locutor nuevo cargado!
```

Figura 4.5. Captura de pantalla de la adquisición de locutor y texto a sintetizar.

4.2.2 Procesado del texto a sintetizar.

Una vez el usuario ha introducido el texto a sintetizar se debe realizar un procesado previo antes de la asignación de modelos. El módulo *synthesize_text* realiza tanto este procesado como la generación de voz. El procesado se realiza en

tres pasos bien diferenciados: tratamiento de caracteres especiales, normalización del texto y creación de etiquetas.

1. Se realiza un tratamiento de caracteres especiales, pasando su formato al empleado por HTS. Entre estos caracteres especiales están las vocales con diéresis o la letra "ñ". A continuación se muestra un ejemplo del caracter original y el que resulta de la transformación:

$$\tilde{n} \Rightarrow \sim n$$

$$\ddot{u} \Rightarrow : u$$

2. Se normaliza y clasifica el texto introducido. Esta normalización consiste en identificar los signos de puntuación (puntos suspensivos, comas, signos de interrogación...), contar las letras, sílabas y palabras que componen la frase y finalmente en caso de ser necesario dividir la frase en subfrases. La división se lleva a cabo cuando la oración contiene un punto o una coma, ya que estos signos de puntuación implican un cambio de entonación y una pausa a la hora de pronunciar la frase.

3. Se crea el fichero de etiquetas en su versión extendida con información contextual, fichero que se ha denominado *full.lab*. El módulo *crea_HTSlab* es el encargado de realizar esta tarea. Partiendo de la normalización previa, realiza una última transformación de la oración pasando todos los caracteres a minúsculas. A continuación se procede a la transcripción fonética de la frase, siempre con el formato de fonemas empleado en HTS. En este punto, se crea una variable denominada *discurso* que contiene información sobre el texto a sintetizar. Un ejemplo de esta variable para el texto "Buenos días" se muestra en la Figura 4.6. En ella se pueden observar los campos: número de frases, palabras, sílabas y la transcripción fonética.

4.2.3 Generación de voz.

El sistema ya dispone de todas las herramientas para asignar los modelos y proceder a la generación de voz. Mediante el módulo *exec_HTS_engine* se realiza la

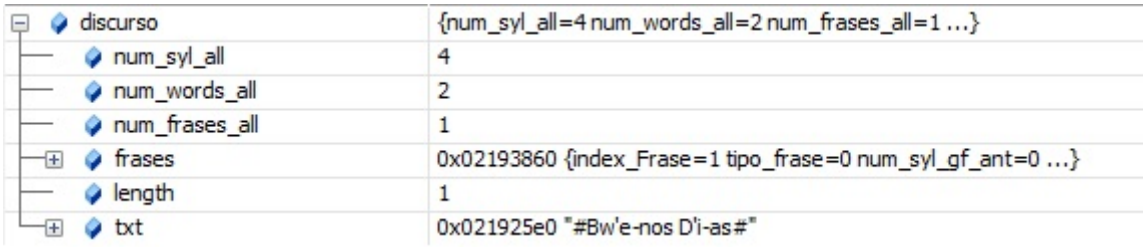

Figura 4.6. Ejemplo de la variable discurso para la frase "Buenos días".

asignación de modelos llamando a *hts_engine* con todos los parámetros y variables de entrada y salida que éste necesita.

Entre todas estas variables destaca *engine*, una estructura con una serie de punteros que contienen toda la información sobre los modelos y los parámetros empleados en el proceso de síntesis. Los campos que componen esta estructura son:

- Global: parámetros del proceso como la frecuencia de muestreo, periodo de trama, volumen o velocidad de locución.
- Audio: configuración del audio en formato *.wav*.
- MS: información de los ficheros que contienen los modelos: número de estados, árboles de decisión y ventanas.
- Label: contenido del fichero *.lab* con la frase a sintetizar.
- SSS: secuencia de estados.
- PSS: modelos asignados.
- GSS: parámetros de la voz generada.

Una vez se hace la llamada a *hts_engine* el proceso de generación de voz sigue las siguientes etapas:

1. Obtención de los modelos que mejor se adaptan a las características de los fonemas contextuales. Estos modelos se escogen haciendo uso de los árboles de decisión asociados.

2. Generación de los parámetros acústicos. El criterio empleado es la maximización de la probabilidad de las observaciones conocida la secuencia de estados [24]. Esto implica que en primer lugar se genera la duración de cada estado y a continuación se obtienen los observables más probables atendiendo a las funciones densidad de probabilidad.

3. Generación de la voz artificial. El sistema de producción de voz es el basado en filtro más excitación. Como se detalla en la Sección 2.5.2., como excitación se emplea un tren de impulsos a la frecuencia del pitch para sonidos sonoros y un ruido Gaussiano para los sonidos sordos. El filtro empleado sigue el método MLSA, cuyos coeficientes espectrales son los cepstrum en la escala de Mel. Este filtro de síntesis simula el tracto vocal del locutor.

Este proceso da como resultado un archivo de audio con extensión *.wav*, que contiene la frase generada con la voz deseada. *WAV* es un formato de audio sin compresión, admite archivos mono y estéreo a diferentes resoluciones y frecuencias de muestreo, y es empleado para la reproducción en PC. El archivo de salida en este caso es mono, con frecuencia de muestreo 16kHz y 32 bits.

Capítulo 5

Resultados y aplicaciones.

5.1 Resultados.

Con el objetivo de establecer una relación entre la voz original y la voz sintetizada se recurre a un cálculo de la distorsión. Este cálculo se realiza mediante una técnica de programación dinámica empleando el algoritmo Dynamic Time Warping (DTW [11][14]) con la distancia de Itakura [1].

La señal de voz se segmenta en tramas de 10 ms cada una. Con esta segmentación, una palabra se compondrá de docenas de tramas dependiendo de la velocidad de la locución. Una vez segmentadas las señales original y sintetizada surje un problema, las dos señales pueden no tener la misma longitud y por tanto tener distinto número de tramas.

Para salvar esta diferencia en el número de tramas se recurre al algoritmo DTW. Este método es ampliamente utilizado en reconocimiento del habla. El algoritmo realiza la asignación entre tramas de las dos señales mediante un reescalado del eje temporal.

Supongamos que tenemos dos señales X y W con seis y ocho tramas respectivamente. Una posible asignación de tramas sería:

$$P = \{(0,0),(1,1),(2,2),(3,2),(4,2),(5,3),(6,4),(7,4)\}$$

En la Figura 5.1 se muestra gráficamente esta asignación.

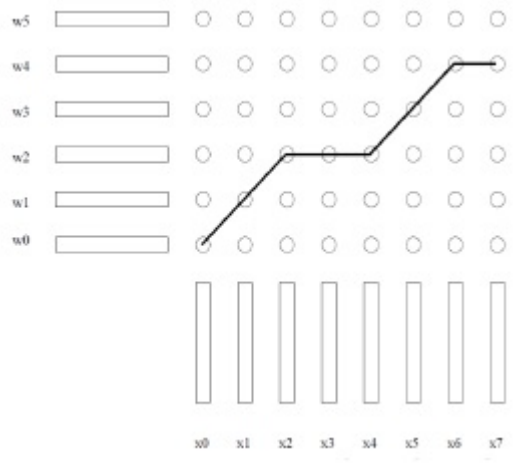

Figura 5.1. Posible asignación entre pares de tramas de dos señales.

Evidentemente esta asignación no puede realizarse aleatoriamente, sino que se realiza mediante un cálculo de distancia entre tramas. Existen diversos métodos para calcular esta distancia, como la distancia euclidea o la distancia cosenoidal, sin embargo en este proyecto se empleará el método de la distancia de Itakura.

El empleo del método de Itakura está muy extendido en aplicaciones de procesado del habla, ya que indica la distorsión entre los espectros de las dos señales de voz a comparar. La expresión para calcular esta distancia es:

$$d_I = ln \int_{-\pi}^{\pi} \frac{|A_w(\omega)|^2}{|A_x(\omega)|^2} \frac{d\omega}{2\pi} \tag{5.1}$$

Por tanto, juntando ambos algoritmo (DTW e Itakura) la asignación de tramas se realiza como sigue:

$$\delta(i,j) = d_I(\underline{x}_i, \underline{w}_j) + min \begin{cases} \delta(i, j-1) \\ \delta(i-1, j-1) \\ \delta(i-1, j) \end{cases} \tag{5.2}$$

Mediante este algoritmo se asegura que se alcanza el punto (i,j) a través del camino óptimo partiendo del punto (0,0) y que la distancia acumulada $\delta(i,j)$ es la mínima entre todas las posibles entre (0,0) e (i,j).

Por último, una vez finalizado el proceso de asignación de tramas y calculado el camino de mínimo coste, la distancia acumulada $\delta(T_w - 1, T_x - 1))$ es la distancia

entre las dos señales, D(W,X).

La comparación de las voces originales y sintetizadas se divide en dos partes:

- Frases contenidas en la base de datos de entrenamiento.
- Frases no contenidas en la base de datos de entrenamiento.

Esto se hace así por la siguiente razón: comprobar la degradación que se produce al sintetizar frases que han sido incluidas en el proceso de entrenamiento frente a frases que no han formado parte de él. Al sintetizar frases que han servido para entrenar los HMMs, las ligaduras entre fonemas y el contexto son conocidas.

A continuación se muestran dos figuras con la curva de mínimo coste que resulta de aplicar el algoritmo DTW. La Figura 5.2 presenta las gráficas resultantes para tres frases que han sido utilizadas en el proceso de entrenamiento, mientras que la Figura 5.3 muestra las gráficas para tres frases no contenidas en la base de datos de entrenamiento. En el eje X se representan las tramas de la señal sintetizada y en el eje Y las tramas de la señal original. El valor del coste mínimo se obtiene en la esquina inferior derecha de las figuras.

Para comparar los costes mínimos obtenidos para los dos tipos de señales, representamos estos valores mediante dos histogramas (Figura 5.4). El histograma de la izquierda se corresponde con las señales de voz empleadas en el entrenamiento y el histograma de la derecha con las señales de voz no contenidas en la base de datos de entrenamiento.

Los valores medios y las desviaciones típicas de los costes mínimos para las señales contenidas y no contenidas en la base de datos de entrenamiento se muestran en la tabla 5.1. Los resultados obtenidos son mejores para las señales empleadas en la fase de entrenamiento, esto es debido a que los fonemas y contextos son ya conocidos. Para mejorar los resultados para las frases no contenidas en la base de datos habría que ampliar la base de datos de entrenamiento, lo que implicaría un aumento en los fonemas y contextos entrenados.

	Señales entrenadas	Señales no entrenadas
Media	2,8176	3,2106
Desviación típica	0,1276	0,0974

Tabla 5.1. Valores medios y desviaciones típicas de los costes mínimos.

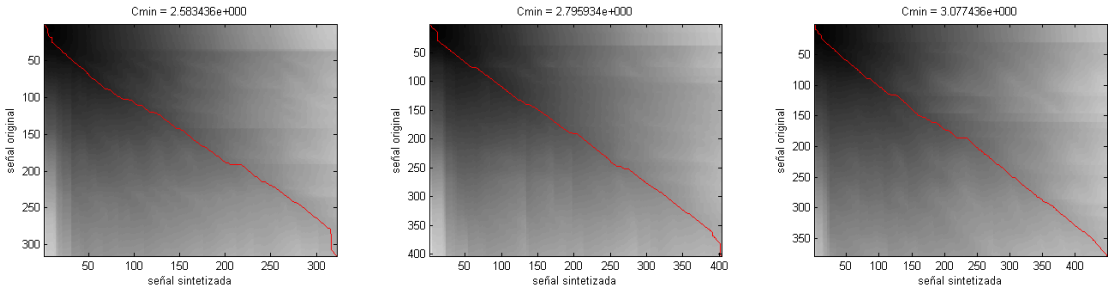

Figura 5.2. Curva de coste mínimo para 3 señales de voz empleadas durante el entrenamiento.

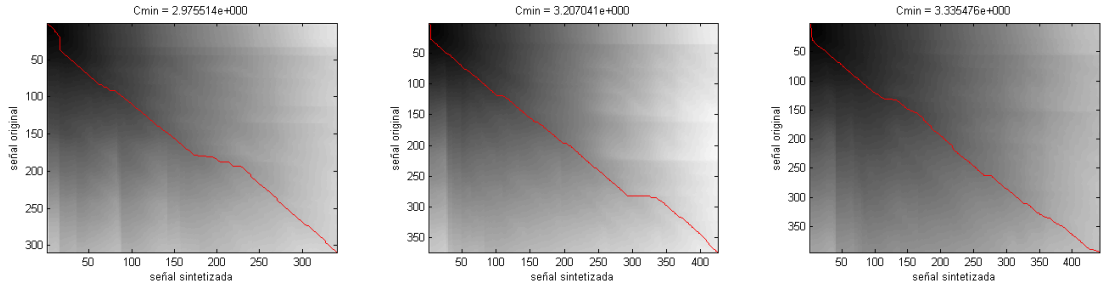

Figura 5.3. Curva de coste mínimo para 3 señales de voz no empleadas durante el entrenamiento.

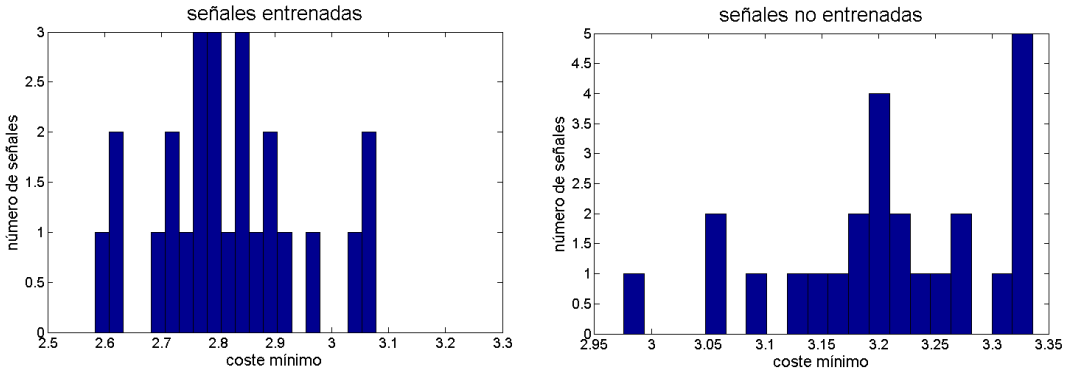

Figura 5.4. Histograma de los costes mínimos.

5.2 Aplicaciones.

Uno de los objetivos de este proyecto es la elaboración de una aplicación Web que permita realizar el proceso de síntesis. La aplicación comprende desde el proceso de grabación de la voz hasta la generación de voz artificial mediante HMMs adaptada a la voz original. La Figura 5.5 muestra las etapas de este proceso.

Figura 5.5. Etapas para la generación de voz en la aplicación Web.

Entrando en detalle en cada etapa:

1. Login: cada usuario dispondrá de un nombre de identificación y una contraseña. En este proceso de identificación el sistema buscará si el usuario se encuentra en su base de datos. Una vez identificado, cada usuario dispondrá de un sistema de carpetas y ficheros independientes del resto.

2. Grabación y transmisión del audio: el usuario debe grabar una serie de frases para realizar la síntesis. Estas frases se muestran por pantalla y mediante dos botones (REC y STOP) el locutor controlará la grabación, Figura 5.6.

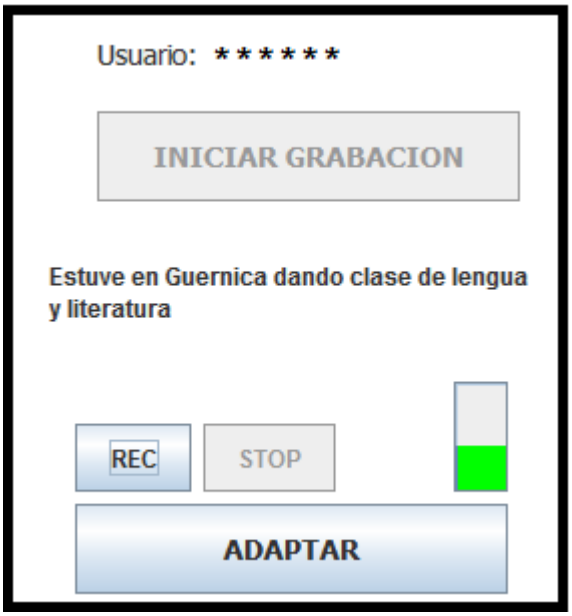

Figura 5.6. Captura de pantalla del proceso de grabación.

La transmisión de los ficheros de audio desde el ordenador del usuario al servidor donde se encuentran todos los ficheros y programas necesarios para el entrenamiento de los HMMs se realiza mediante una conexión TCP(Transmission Control Protocol) que proporciona una transmisión segura y libre de errores. El paquete recibido contiene una cabecera que lo identifica y los datos, que se almacenan en formato *.raw*.

3. Entrenamiento de los HMMs: se sigue el proceso descrito en el Capítulo 3 para la adaptación de la voz recibida. Los ficheros correspondientes a la voz base se encuentran almacenados y por tanto no se repite esta parte del proceso de entrenamiento, reduciendo considerablemente el tiempo empleado en este proceso.

4. Transmisión de los ficheros de HMMs: el proceso de entrenamiento se realiza bajo un sistema operativo basado en UNIX, mientras que la generación de voz funciona sobre WINDOWS. Por tanto, surje la necesidad de realizar la transmisión de los ficheros con los HMMs de un sistema al otro. Para realizar esta transmisión se debe configurar un servicio SSH (Secure SHell) que permita la recepción de ficheros mediante el comando de copia SCP (Secure CoPy).

   Al finalizar el proceso de entrenamiento, el sistema UNIX lanza el comando SCP mediante el que se envía una carpeta, con el nombre del usuario, que contiene todos los ficheros de HMMs necesarios para generar la voz artificial.

5. Generación de la voz artificial: una vez recibidos los ficheros con los HMMs se puede proceder a generar la voz artificial. El proceso es el que ha sido detallado en el Capítulo 4 y la herramienta empleada sigue siendo *vivoSinte*.

Capítulo 6

Conclusiones y líneas futuras.

En este proyecto se ha desarrollado un sistema de síntesis del habla basado en HMMs. El proceso comienza con la grabación de la voz del locutor, continúa con el entrenamiento de los HMMs obtenidos a partir de ésta, y finaliza con la generación de una voz artificial que busca asemejarse lo más posible a la original.

El trabajo llevado a cabo se ha dividido en las siguientes fases. En primer lugar se ha efectuado una fase de documentación, en la que se han consultado las referencias y bibliografía asociada a la problemática de los HMMs y la síntesis de voz. A continuación, se llevó a cabo una etapa de familiarización con el software empleado, siguiendo demostraciones obtenidas de la web de HTS (*http://hts.sp.nitech.ac.jp/*). Posteriormente, se ha realizado la generación de una voz en castellano, tanto sin métodos de adaptación como con ellos, durante este proceso se actualizó el sistema *vivoSinte* a las nuevas versiones de los programas de entrenamiento. Para evaluar la similitud entre voces originales y artificiales se ha realizado un análisis de distorsión. Por último, se ha elaborado una aplicación web que permita emplear la herramienta de síntesis a distintos usuarios.

Para comprobar la similitud entre las voces originales y las sintetizadas se ha realizado un estudio de la distorsión mediante un método de programación dinámica. Este estudio se ha realizado tanto sobre frases empleadas en el proceso de entrenamiento como con frases que no estaban contenidas en esa base de datos. Con esta división se pretende ilustrar cómo el sistema sintetiza con mayor precisión

oraciones cuyos fonemas y contextos son conocidos de antemano. En los valores de coste mínimo obtenidos para ambos casos podemos constatar esta diferencia. Para frases empleadas en la fase de entremamiento los valores obtenidos para los costes mínimos son, en media, un 14 % menores que para las no empleadas. Para reducir esta diferencia se debería aumentar el tamaño de la base de datos, lo que implicaría un aumento de los contextos y fonemas. Por otro lado, el aumento de la base de datos influiría en la cantidad de memoria empleada en el proceso y en la velocidad de aplicación del algoritmo de adaptación.

De cara a futuras investigaciones sobre este sistema de síntesis proponemos la mejora del proceso de generación de voz, centrándose en parámetros de la locución como pueden ser la entonación y la prosodia. Mediante la mejora de estos dos parámetros se podría obtener una voz sintetizada mucho más natural, acercándola por tanto a la voz original.

Otra posible línea de investigación haría referencia al empleo de este sistema para tratamiento de patologías del habla. En este proyecto se ha sintetizado la voz del Presidente del Gobierno, Mariano Rajoy. El objetivo era que este sistema fuera capaz de aprender la patología de su voz (seseo) y sintetizarla posteriormente. Como resultado, el sistema es capaz de aprenderla y generar la voz sintetizada con la misma patología. La posible línea de investigación propuesta serviría para, una vez aprendida la patología por el sistema, mediante técnicas de procesado de señal corregir dicho problema de la voz.

Por último, se propone emplear la aplicación Web desarrollada en este proyecto para crear un banco de voces, con o sin patología. Para este último caso, este banco de voces podría ser de utilidad, por ejemplo, en casos de pérdida de las facultades del habla.

Bibliografía

[1] G Chen. Enhanced Itakura measure incorporating masking properties of human auditory system. *Signal Processing*, 83(7):1445–1456, July 2003.

[2] G. David Forney. The Viterbi Algorithm. pages 302–309, 1972.

[3] I. Folgueira. Sintesis de voz mediante Modelos Ocultos de Markov. 2008.

[4] T. Fukada, K. Tokuda, Ta. Kobayashi, and S. Imait. An adaptive algorithm for mel-cepstral analysis of speech. *Systems Research*, pages 137–140, 1992.

[5] M.J.F. Gales. The generation and use of regression class trees for MLLR adaptation. *Engineering*, (August), 1996.

[6] J.L. Gauvain and C.H. Lee. Maximum a Posteriori Estimation for Multivariate Gaussian Mixture Observations of Markov Chains. *Audio*, 2(2), 1994.

[7] R. Hsiao, Y.C. Tam, and T. Schultz. Generalized Baum-Welch algorithm for discriminative training on large vocabulary continuous speech recognition system. *Language*, pages 3769–3772, 2009.

[8] X. Huang, A. Acero, and H.W. Hon. *Spoken language processing*, volume 1. Prentice Hall PTR New Jersey, 2001.

[9] S. Imai. Cepstral Analysis Synthesis on the Mel Frequenciy Scale. pages 93–96, 1983.

[10] S Imai. Adaptive mel cepstral analysis based on UELS method. In *The IEEE Adaptive Systems for Signal Processing Communications and Control Symposium*, pages 304–309, 2000.

[11] M. Lama, P; Namburu. Speech Recognition with Dynamic Time Warping using MATLAB. *cs.uccs.edu*, 36(I):41–48, 1988.

[12] C.J. Leggetter and P.C. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. pages 171–185, 1995.

[13] M. Oudelha and R.N. Ainon. HMM Parameters Estimation Using Hybrid Baum Â Welch Genetic Algorithm. *Training*, (l):542–545.

[14] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, February 1978.

[15] S.A. Samad, A. Hussain, and L.K. Fah. Pitch detection of speech signals using the cross-correlation technique. In *TENCON 2000. Proceedings*, volume 1, pages 283–286. IEEE, 2000.

[16] K. Shinoda and T. Watanabe. MDL-based context-dependent subword modeling for speech recognition.

[17] D. Talkin. A robust algorithm for pitch tracking (RAPT). *Speech coding and synthesis*, 495:518, 1995.

[18] M. Tamura, T. Masuko, K. Tokuda, and T. Kobayashi. Adaptation of pitch and spectrum for HMM-based speech synthesis using MLLR. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 2, pages 805–808. IEEE, 2001.

[19] P. Taylor. Text-to-Speech Synthesis.

[20] T. Toda and S. Young. Trajectory training considering global variance for HMM-based speech synthesis. In *Acoustics, Speech and Signal Processing,*

2009. ICASSP 2009. IEEE International Conference on, pages 4025–4028. IEEE, 2009.

[21] K. Tokuda, T. Masuko, N. Miyazaki, and T. Kobayashi. Multi-space probability distribution HMM. *IEICE Transactions on Information and Systems E series D*, 85(3):455–464, 2002.

[22] K. Tokuda, H. Zen, and A.W. Black. An HMM-based speech synthesis system applied to English. In *Speech Synthesis, 2002. Proceedings of 2002 IEEE Workshop on*, pages 227–230. IEEE, 2002.

[23] J. Yamagishi, H. Zen, T. Toda, and K. Tokuda. Speaker-Independent HMM-based Speech Synthesis System â HTS-2007 System for the Blizzard Challenge 2007. *System*, 2007.

[24] T. Yoshimura. Simultaneous modeling of phonetic and prosodic parameters, and characteristic conversion for HMM-Based Text-to-Speech systems. *Distribution*, (January), 2002.

[25] S. Young, M. Gales, X. Liu, and P. Woodland. The HTK Book. *Memory*, (July 2000), 2006.

Apéndice A

Modelos ocultos de Markov.

Los modelos ocultos de Markov (HMMs:Hidden Markov Models) siguen una estadística Markoviana. Se representan mediante un conjunto de estados que definen su evolución. En las cadenas de primer orden, expuestas en la próxima sección, cada estado tiene únicamente una salida.

En el caso de los HMMs, estos pueden tener varios valores de salida en cada estado, lo que implica la aparición de una nueva variable aleatoria. Esto puede verse como un proceso estocástico doble: por una lado las transiciones entre estados y por otro la salida de cada estado. Un HMM es una cadena de Markov donde los observables de salida son una variable aleatoria generada de acuerdo a una función probabilística asociada a cada estado [8].

A.1 Cadena de Markov de primer orden.

Se considera una cadena de Markov de primer orden con 'M' estados posibles. Al ser una cadena de primer orden, cada estado tendrá una salida y por tanto el conjunto del sistema tendrá 'M' posibles salidas. Se puede definir una secuencia de observación, X, que consiste en un conjunto de variables aleatorias procedentes de la evolución de la cadena durante 'n' instantes de tiempo.

$$X = (x_1, x_2, ..., x_n) \tag{A.1}$$

El alfabeto de cada una de las variables aleatorias es: $\Theta = \{o_1, o_2, ..., o_M\}$ y la secuencia de estados asociada se define como $S = \{s_1, s_2, ..., s_n\}$. Aclarar que en este último caso el subíndice, 'n', no se refiere al número del estado sino al instante de tiempo.

La probabilidad de tener una observación X es

$$P(X) = P(x_1, x_2, ..., x_n) = P(x_1) \prod_{i=2}^{n} P(x_i|x_1, x_2, ..., x_{i-1}) \tag{A.2}$$

desglosándola,

$$P(X) = P(x_1)P(x_2|x_1)P(x_3|x_1, x_2)...(x_n|x_1, x_2, ..., x_{n-1}) \tag{A.3}$$

Se puede asumir que la probabilidad de estar en un estado depende exclusivamente del estado inmediatamente anterior, esto se puede expresar

$$P(x_i|x_1, x_2, ..., x_{i-1}) = P(x_i|x_{i-1}) \tag{A.4}$$

de acuerdo a esta relación podemos reescribir A.3 de la siguiente forma

$$P(X) = P(x_1)P(x_2|x_1)P(x_3|x_2)...(x_n|x_{n-1}) \tag{A.5}$$

como cada estado sólo puede tener una salida, finalmente

$$P(X) = P(x_1)P(x_2|x_1)P(x_3|x_2)...(x_n|x_{n-1}) =$$

$$= P(s_1)P(s_2|s_1)P(s_3|s_2)...(s_n|s_{n-1}) = P(S) \Rightarrow$$

$$\Rightarrow P(X) = P(S) \tag{A.6}$$

Por tanto, de acuerdo a la expresión A.6 existe una relación uno a uno entre la secuencia de observación X y la secuencia de estados de Markov.

A.2 Modelos ocultos de Markov (HMM).

En los HMMs cada estado puede tener más de una salida posible, en general hablaremos de 'N' posibles salidas. Así pues, consideramos de nuevo una cadena de Markov de 'M' estados posibles, cada uno de ellos puede tener 'N' salidas posibles. De nuevo denominamos a la secuencia de observación $X = (x_1, x_2, ..., x_n)$, al alfabeto $\Theta = \{o_1, o_2, ..., o_N\}$ y a la secuencia de estados $S = \{s_1, s_2, ..., s_n\}$.

En estos modelos no existe correspondencia uno a uno entre la secuencia de observación y la secuencia de estados, es decir

$$P(X) \neq P(S)$$

$$P(x_1, x_2, ..., x_n) \neq P(s_1, s_2, ..., s_n) \tag{A.7}$$

Por tanto, generalmente no se puede determinar la secuencia de estados para una secuencia de observación dada, es decir, la secuencia de estados no es observable, es oculta. Esta es la razón por la que se denominan modelos ocultos de Markov (del inglés, Hidden Markov Models).

Formalmente, un HMM viene definido por:

- $\underline{\underline{A}} = \{a_{ij}\}$ donde $a_{ij} = P(s_t = j | s_{t-1} = i)$ con $1 \leq$ i $\leq$ M y $1 \leq$ j $\leq$ M - Matriz de probabilidades de transición de estados.
- $\underline{\underline{B}} = \{b_i(o_{k_t})\} = P(x_t = o_k | s_t = i)$ donde $1 \leq$ i $\leq$ M y $1 \leq$ k $\leq$ N - Matriz de densidades de probabilidad para la salida de los estados.
- $\pi = \{\neq_i\} =$ P(s_0=i) donde $1 \leq$ i $\leq$ M - Distribución de probabilidad de los estados iniciales

La notación empleada para referirse a un HMM es $\lambda = (\underline{\underline{A}}, \underline{\underline{B}}, \pi)$.

En la Figura A.1 se muestra un ejemplo de un diagrama de estados de un HMM, con las probabilidades de transición entre estados y los observables de salida de cada estado.

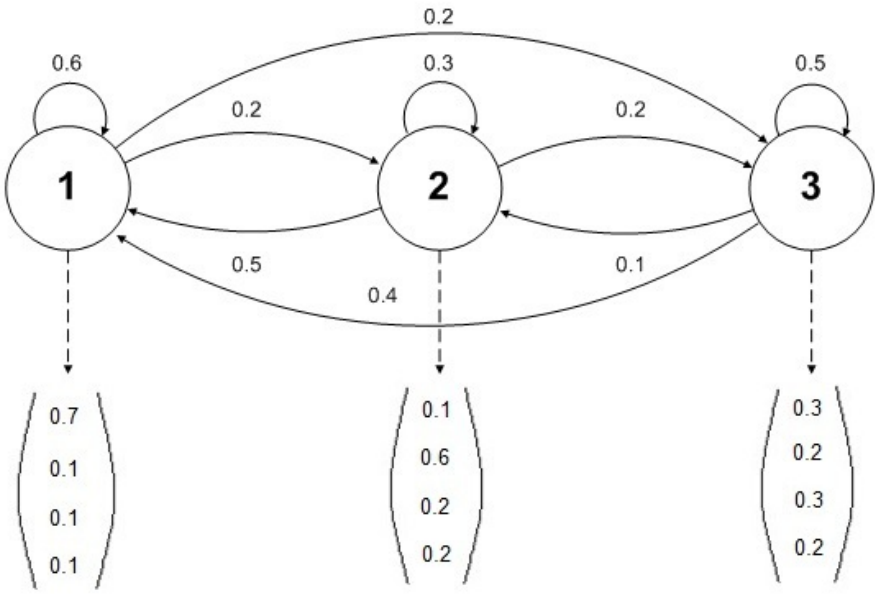

Figura A.1. Diagrama de estados de un HMM.

Para profundizar más en los conceptos matemáticos que comprenden la resolución de problemas de reconocimiento de voz con HMMs se puede acudir a [25][8].

Apéndice B

Formato de las etiquetas.

El propósito de este apéndice es detallar el formato empleado en la elaboración de los ficheros de etiquetas. En concreto de las etiquetas con información contextual que a lo largo de la memoria se denominan *full.lab*. Estos archivos tienen gran importancia en el proceso de síntesis, pues contienen toda la información sobre las frases a entrenar o a sintetizar.

Cada línea de un fichero *full.lab* sigue la siguiente estructura:

$t_0 \quad t_1$

$p_0 \,\hat{}\, p_1 - p_2 + p_3 = p_4 \;@\; p_5 \;\_\; p_6$

$/A : a_0 \;\_\; a_1 \;\_\; a_2$

$/B : b_0 - b_1 - b_2 \;@\; b_3 - b_4 \;\&\; b_5 - b_6 \;\#\; b_7 - b_8 \;\$\; b_9 - b_{10} \;!\; b_{11} - b_{12} \;;\; b_{13} - b_{14} \mid b_{15}$

$/C : c_0 + c_1 + c_2$

$/D : d_0 \;\_\; d_1$

$/E : e_0 + e_1 \;@\; e_2 + e_3 \;\&\; e_4 + e_5 \;\#\; e_6 + e_7$

$/F : f_0 \;\_\; f_1$

$/G : g_0 \;\_\; g_1$

$/H : h_0 = h_1 \,\hat{}\, h_2 = h_3 \mid h_4$

$/I : i_0 \;\_\; i_1$

$/J : j_0 + j_1 - j_2$

A continuación se va a detallar cada componente.

Duración de cada fonema

t_0: instante de comienzo del fonema en estudio.

t_1: instante de finalización del fonema en estudio.

Las unidades empleadas para expresar estos tiempos son centenares de picosegundo.

Fonemas

p_0: fonema anterior al fonema previo.

p_1: fonema previo al fonema objeto de estudio.

p_2: fonema a estudiar.

p_3: fonema posterior al fonema objeto de estudio.

p_4: fonema siguiente al fonema posterior.

p_5: posición del fonema en estudio en la sílaba actual contando desde el principio de ésta (forward).

p_6: posición del fonema en estudio en la sílaba actual contando desde el final de ésta (backward).

Sílabas

a_0: toma valor 1 si la sílaba previa es "stressed", valor 0 en caso contrario.

a_1: toma valor 1 si la sílaba previa es tónica, valor 0 en caso contrario.

a_2: número de fonemas que contiene la sílaba previa.

b_0: toma valor 1 si la sílaba en estudio es "stressed", valor 0 en caso contrario.

b_1: toma valor 1 si la sílaba en estudio es tónica, valor 0 en caso contrario.

b_2: número de fonemas que contiene la sílaba objeto de estudio.

b_3: posición que ocupa la sílaba actual en la palabra (forward).

b_4: posición que ocupa la sílaba actual en la palabra (baclward).

b_5: posición que ocupa la sílaba actual en la frase (forward).

b_6: posición que ocupa la sílaba actual en la frase (backward).

b_7: número de sílabas "stressed" antes de la actual en toda la frase.

b_8: número de sílabas "stressed" después de la actual en toda la frase.

b_9: número de sílabas tónicas antes de la actual en toda la frase.

b_{10}: número de sílabas tónicas después de la actual en toda la frase.

b_{11}: número de sílabas desde la la "stressed" anterior más próxima hasta la actual.

b_{12}: número de sílabas desde la actual hasta la "stressed" posterior más próxima.

b_{13}: número de sílabas desde la la tónica anterior más próxima hasta la actual.

b_{14}: número de sílabas desde la actual hasta la tónica posterior más próxima.

b_{15}: nombre de la vocal en la sílaba actual.

c_0: toma valor 1 si la sílaba posterior es "stressed", valor 0 en caso contrario.

c_1: toma valor 1 si la sílaba posterior es tónica, valor 0 en caso contrario.

c_2: número de fonemas de la siguiente sílaba.

Palabras

d_0: tipo de palabra (gpos) de la palabra previa a la objeto de estudio.

d_1: número de sílabas de la palabra previa a la objeto de estudio.

e_0: gpos de la palabra en estudio.

e_1: número de sílabas de la palabra en estudio.

e_2: posición de la palabra en estudio dentro de la frase (forward).

e_3: posición de la palabra en estudio dentro de la frase (backward).

e_4: número de palabras de tipo "content" previas a la actual.

e_5: número de palabras de tipo "content" posteriores a la actual.

e_6: número de palabras desde la "content" más próxima hasta la actual.

e_7: número de palabras desde la actual hasta la "content" más próxima.

f_0: gpos de la palabra siguiente a la objeto de estudio.

f_1: número de sílabas de la palabra posterior a la objeto de estudio.

Grupo fónico

g_0: número de sílabas del grupo fónico previo al objeto de estudio.

g_1: número de palabras del grupo fónico previo al objeto de estudio.

h_0: número de sílabas del grupo fónico en estudio.

h_1: número de palabras del grupo fónico en estudio.

h_2: posición del grupo fónico actual en la frase (forward).

h_3: posición del grupo fónico actual en la frase (backward).

h_4: TOBI o entonación de la sílaba en estudio.

i_0: número de sílabas del siguiente grupo fónico.

i_1: número de palabras del siguiente grupo fónico.

Frase completa

j_0: número de sílabas de la oración.

j_1: número de palabras de la oración.

j_2: número de frases en toda la oración.

Se emplea el término oración como conjunto de frases. Por ejemplo, "Buenas tardes, ¿va todo bien?", constituye una oración con dos frases separadas por comas.

Toda la información de este Apéndice ha sido extraida del fichero *label-full.awk*. Este fichero se emplea en la parte de entrenamiento relacionada con la elaboración de las etiquetas detallada en la sección 3.3.2.

Apéndice C

Algoritmos de adaptación.

En este apéndice se realiza una exposición matemática de los algoritmos de adaptación empleados en este proyecto. Consta de dos secciones, la primera dedicada al algoritmo Maximum Likelihood Linear Regression (MLLR) y la segunda dedicada al algoritmo Maximum A Posteriori (MAP).

C.1 Maximum Likelihood Linear Regression.

El algoritmo de adaptación MLLR [12][18] realiza una adaptación lineal mediante una serie de transformaciones que reducen la diferencia entre el modelo para el locutor base y el locutor a adaptar. Esta transformación lineal se aplica a las Gaussianas del locutor deseado de la siguiente forma

$$\underline{\widehat{\mu}}_{jc} = \underline{\underline{W_c}}\, \underline{\mu}_{jc} \tag{C.1}$$

donde $\underline{\underline{W_c}}$ representa la matriz de transformación lineal de las medias de orden $n \times (n+1)$ y $\underline{\mu}_{jc}$ es el vector de medias de la c-ésima Gaussiana en el estado *j-ésimo* y presenta la siguiente estructura

$$\underline{\mu}_{jc} = [\omega\ \mu_{j1}\ ...\ \mu_{jC}]' \tag{C.2}$$

se puede observar que se ha incluido un término de offset, ω, que de manera

estándar suele tomar valor 1.

Para estimar las matrices de regresión, MLLR maximiza la verosimilitud de los modelos adaptados al locutor cuya voz se desea adaptar. En el método de obtención de esta matriz, que se desarrolla a continuación, sólo se consideran distribuciones de covarianzas diagonales.

Se asume que los datos de adaptación, $\underline{\underline{O}}$, es una serie de T observaciones

$$\underline{\underline{O}} = (\underline{o}_1, \underline{o}_2, ..., \underline{o}_T) \tag{C.3}$$

La distribución de probabilidad de cada Gaussiana de parámetros Θ, para la secuencia de observables $\underline{\underline{O}}$ y una matriz de variables ocultas $\underline{\underline{Z}}$, se define como

$$p(\underline{\underline{O}}, \underline{\underline{Z}}|\Theta) = \frac{1}{2\pi^{D/2}|\Sigma_c|^{1/2}} exp\{-\frac{1}{2}(\underline{\underline{O}} - \underline{\widehat{\mu}}_c)^T \Sigma_c^{-1} (\underline{\underline{O}} - \underline{\widehat{\mu}}_c)\} \tag{C.4}$$

donde $\underline{\mu}_c$ representa la media y Σ_c la covarianza de la Gaussiana. Se define una función auxiliar, $Q(\Theta|\Theta^{(k)})$, que se maximiza respecto a $\underline{\underline{W}}$:

$$Q(\Theta|\Theta^{(k)}) = E[log\, p(\underline{\underline{O}}, \underline{\underline{Z}}|\Theta)|\underline{\underline{O}}, |\Theta^{(k)}] \tag{C.5}$$

De acuerdo a las ecuaciones C.1 y C.4, C.5 se puede expresar

$$Q(\Theta|\Theta^{(k)}) = \sum_n \sum_c \left\langle \delta_{Z_{n,c}} \right\rangle^k [-\frac{1}{2} log(2\pi) - \frac{1}{2} log|\Sigma_c| \frac{1}{2} (\underline{o}_n - \underline{\underline{W_c}}\, \underline{\mu}_c)^T \Sigma_c^{-1(k)} (\underline{o}_t - \underline{\underline{W_c}}\, \underline{\mu}_c)] \tag{C.6}$$

Los parámetros del modelo que maximizan $Q(\Theta|\Theta^{(k)})$, maximizan igualmente la verosimilitud con los nuevos datos, por tanto, derivando Q respecto de $\underline{\underline{W}}$ e igualando a cero se obtiene el máximo de la función:

$$\frac{\partial Q(\Theta|\Theta^{(k)})}{\partial \underline{\underline{W_c}}} = \sum_n \sum_c \left\langle \delta_{Z_{n,c}} \right\rangle^k [\Sigma_c^{-1} (\underline{\underline{W_c}} (2\underline{\mu}_c\, \underline{\mu}_c)) - 2\Sigma_c^{-1} \underline{o}_n \underline{\mu}_c^T] = 0 \tag{C.7}$$

por tanto

$$\sum_{n,c} \left\langle \delta_{Z_{n,c}} \right\rangle^k \Sigma_c^{-1} \underline{o}_n \underline{\mu}_c^T = \sum_{n,c} \left\langle \delta_{Z_{n,c}} \right\rangle^k [\Sigma_c^{-1} \underline{\underline{W_c}}\, \underline{\mu}_c\, \underline{\mu}_c^T] \tag{C.8}$$

Esta última formula se trata de una ecuación matricial, se definen las matrices resultantes a ambos lados de la igualdad como $\underline{\underline{Z}}$ e $\underline{\underline{Y}}$, por tanto $\underline{\underline{Z}} = \underline{\underline{Y}}$. La dimensión de ambas matrices es $n \times (n+1)$. Para poder manejar más fácilmente $\underline{\underline{Y}}$ se realiza una manipulación:

$$\underline{\underline{Y}} = \sum_c \underline{\underline{V^c}}\, \underline{\underline{W_c}}\, \underline{\underline{D^{(c)}}} \tag{C.9}$$

siendo

$$\underline{\underline{V^c}} = \sum_c \left\langle \delta_{Z_{n,c}} \right\rangle^k \Sigma_c^{-1} \tag{C.10}$$

$$\underline{\underline{D^{(c)}}} = \underline{\mu}_c\, \underline{\mu}_c^T \tag{C.11}$$

A continuación se obtienen las componentes de cada una de la matriz $\underline{\underline{Y}}$, a los que se denomina y_{ij}, en función de las compontentes de las matrices $\underline{\underline{V^c}}$, $\underline{\underline{W_c}}$ y $\underline{\underline{D^{(c)}}}$.

$$y_{ij} = \sum_{p=1}^{n} \sum_{q=1}^{n+1} w_{pq} [\sum_c v_{ip}^{(c)} d_{qj}^{(c)}] \tag{C.12}$$

Si todas las matrices de covarianza son diagonales y $\underline{\underline{D}}$ es simétrico:

$$\sum_c v_{ip}^{(c)} d_{qj}^{(c)} = \left\{ \begin{array}{ll} \sum_c v_{ip}^{(c)} d_{qj}^{(c)} & \text{cuando i = p} \\ 0 & \text{cuando i} \neq \text{p} \end{array} \right\} \tag{C.13}$$

De modo que,

$$y_{ij} = \sum_{q=1}^{n+1} w_{iq} g_{jq}^{(i)}, \tag{C.14}$$

donde $g_{jq}^{(i)}$ son los elementos de la matriz $\underline{\underline{G^{(i)}}}$ de orden $(n+1)\times(n+1)$ definidos como:

$$g_{jq}^{(i)} = \sum_c v_{ii}^{(c)} d_{jq}^{(c)} \tag{C.15}$$

Como se apuntó anteriormente, $\underline{\underline{Z}} = \underline{\underline{Y}}$ y por tanto $z_{ij} = y_{ij}$.

Se debe apuntar que tanto z_{ij} como $g_{jq}^{(i)}$ no dependen de $\underline{\underline{W_c}}$, y por tanto ambas se pueden obtener únicamente a partir de los vectores observados y los parámetros del modelo inicial. Extrayendo las componentes de $\underline{\underline{W_c}}$ del sistema de ecuaciones

se obtiene:

$$\underline{w}_i^T = (\underline{\underline{G}}^{(i)})^{-1}\, \underline{w}_i^T \tag{C.16}$$

donde $\underline{w}_i$ y $\underline{z}_i$ son las filas *i-ésimas* de $\underline{\underline{W_c}}$ y $\underline{\underline{Z}}$, respectivamente. Estas ecuaciones pueden resolverse por métodos como la descomposición LU o la eliminación de Gauss-Jordan y asi calcular $\underline{\underline{W_c}}$ fila a fila.

Con $\underline{\underline{W_c}}$ calculada y empleando C.1 se pueden obtener los nuevos valores de las medias.

C.2 Maximum A Posteriori.

La técnica de adaptación MAP [6] se emplea para ajustar los parámetros del modelo del locutor independiente a los datos del locutor a adaptar. En términos matemáticos, se pretende maximizar la probabilidad a posteriori del modelo θ a los datos del locutor al que se quiere adaptar, x:

$$\theta = arg\, \max_{\theta} g(\theta|x) = arg\, \max_{\theta} f(x|\theta)g(\theta) \tag{C.17}$$

Las fórmulas empleadas en el proceso de re-estimación de medias, desviaciones estándar, pesos, probabilidades de transición y probabilidades iniciales se exponen a continuación.

$$\widehat{\mu}_{jk} = \frac{\tau_{jk}^m \mu_{jk}^m + \sum_{t=1}^T c_{jkt} x_t}{\tau_{jk}^m + \sum_{t=1}^T c_{jkt}} \tag{C.18}$$

$$\widehat{\sigma}_{jk} = \frac{U_{jk}^\gamma + S_{jk}^\gamma + \tau_{jk}^m (\mu_{jk}^m - \widehat{\mu}_{jk})(\mu_{jk}^m - \widehat{\mu}_{jk})^T}{\alpha_{jk}^\gamma - p - 1 + c_{jk}} \tag{C.19}$$

$$\widehat{w}_{jk} = \frac{\gamma_{jk} - 1 + \sum_{t=1}^T c_{jkt}}{\sum_{k=1}^K (\gamma_{jk} - 1 + \sum_{t=1}^T c_{jkt})} \tag{C.20}$$

$$\widehat{a}_{ij} = \frac{\eta_{ij} - 1 + \sum_{t=1}^T \xi_{ijt}}{\sum_{j=1}^J (\eta_{ij} - 1 + \sum_{t=1}^T \xi_{ijt})} \tag{C.21}$$

$$\widehat{\Pi}_i = \frac{\eta_i - 1 + \gamma_{i0}}{\sum_{n=1}^N (\eta_{ij} - 1 + \gamma_{i0})} \tag{C.22}$$

Los parámetros μ_{jk}^{m}, U_{jk}^{γ}, γ_{jk} y η_i hacen referencia a los valores a priori de la media, varianza y peso de la *k-ésima* Gaussiana en el estado *j-ésimo* de la unidad central y el peso inicial de la *i-ésima* unidad, respectivamente. Por otro lado, los parámetros c_{jkt}, S_{jk}^{γ} y ξ_{ijt} son la probabilidad de la *t-ésima* trama de la *j-ésima* Gaussiana en el estado *k-ésimo* , la varianza media de las tramas de los nuevos datos de adaptación y la probabilidad de transición del estado *i-ésimo* al estado *j-ésimo* en el instante de tiempo *t*. Por último, el parámetro τ_{jk}^{m} es el peso de la información a priori en la *j-ésima* Gaussiana en el estado *k-ésimo* en la unidad *m*.

Analizando las ecuaciones anteriores, se observa que cuanto menor sea el valor de τ, mayor importancia se otorgará a los nuevos datos de adaptación frente a los datos del locutor independiente.