UNIVERSIDAD POLITÉCNICA DE MADRID

E.T.S.I. DE TELECOMUNICACIÓN

TESIS DOCTORAL

ROBUSTEZ EN RECONOCIMIENTO FONÉTICO DE VOZ PARA APLICACIONES TELEFÓNICAS

Autor: José Ignacio Puertas Tera

Director: Ramón García Gómez

Madrid 2.000

Resumen

El reconocimiento de voz es un área de conocimiento de creciente importancia durante esta última década. La introducción de mejores algoritmos, de modelados cada vez más complejos, junto con la aparición de sistemas de cómputo cada vez más potentes y asequibles, hacen posible que los sistemas de interfaz hombre-máquina a través de la voz sean casi una realidad. Esta interfaz permite el acceso a gran cantidad de información a través de la voz, facilitando la introducción de multitud de servicios interactivos usando el teléfono (fijo o móvil) y la televisión como elementos de acceso a dicha información.

Uno de los aspectos más relevantes que han de abordar estos sistemas es el de la robustez, contemplada ésta desde diferentes puntos de vista. Los sistemas de reconocimiento diseñados específicamente para un locutor ya no tienen sentido, ya que una multitud de usuarios con rasgos fonéticos y dialectales muy variados pueden acceder a las aplicaciones de estos sistemas.

Aunque se está produciendo una digitalización de las redes de comunicaciones que facilita una menor distorsión de la señal y bajos niveles de ruido, el bucle de abonado telefónico sigue siendo mayoritariamente analógico, los micrófonos presentan características muy variables y la influencia de la acústica de la sala y de las voces interferentes pueden afectar de forma importante al correcto funcionamiento del reconocedor de voz. La robustez frente a todos estos aspectos resulta de crucial importancia para cualquier aplicación que utilice el reconocedor.

Por otro lado, debido al funcionamiento estadístico de los reconocedores de voz de tipo Markoviano, es necesario introducir mecanismos de detección de palabras fuera de vocabulario para hacer las aplicaciones más robustas frente a pronunciaciones incorrectas por parte del usuario o señales interferentes.

Esta tesis aplica todos los aspectos anteriormente comentados para diseñar reconocedores de voz fonéticos destinados a aplicaciones telefónicas, donde los principales objetivos son los de conseguir una alta flexibilidad con una estructura muy sencilla y, a la vez, robustez para que la variabilidad del locutor y la del canal de comunicaciones se pueda compensar y se mantengan los niveles de prestación que presentan en condiciones controladas de laboratorio.

El objetivo de funcionamiento del reconocedor de voz en condiciones realistas se consiguió con la integración del mismo dentro de una plataforma *ITO* (Integración de Telefonía y Ordenadores), que permite probar muchos aspectos que de otra manera no se hubieran podido analizar.

Finalmente, el presente trabajo deja abierta una serie de líneas de investigación para futuros desarrollos.

Summary

The speech recognition is a field of knowledge of increasing importance during this last decade. The introduction of better algorithms, more complex modellings and the appearance of more powerful and affordable systems of computation lets the human-machine-interface systems with the speech be almost a reality. This interface provides access to a great amount of information with the speech, making easy the introduction of many interactive services using the telephone (fixed or mobile) and the television support.

One of the most outstanding aspects that these systems have to face is the robustness from different points of view. User-specifically-designed recognising systems have no sense, because a mass of users with very diverse phonetic and dialectical features can utilise the applications from these systems.

Although there is a digitalisation process of the communication networks that provides a lower distortion of the signal and reduced noise levels, the telephone local loop is mainly analogical, the microphones have very variable features and the acoustic influence of the room and the speech interferences can affect importantly to the correct way of working of the speech recogniser. Robustness in all those aspects are of crucial importance for any application that uses the recogniser.

On the other hand, due to the statistical working of the Markovian speech recognisers, it is necessary to introduce mechanisms of out-of-vocabulary word

detection in order to make the applications more robust with incorrect speaker pronunciations or interferences.

This thesis applies all those previously mentioned aspects to design phonetic speech recognisers aimed to telephone applications where the main objectives are to obtain a high flexibility with a quite simple structure and, at the same time, robustness to make up for the speaker and channel-communications variabilities and to hold the laboratory-condition performances.

The objective of realistic-condition performances of the speech recogniser was reached with its integration in a *CTI* platform (*Computer-Telephony Integration*), which allows to test many aspects that otherwise had not been possible to analyse.

Finally, the current work opens doors to a series of research topics for future developments.

Prefacio

Quiero dedicar esta tesis a mis
padres Victoriano e Ignacia, a
mi hermana Rosa y a mi
hermano Víctor

Esta tesis es el resultado de mis últimos tres años de trabajo en el Grupo de Tratamiento de Señales en Tiempo Real del Departamento de Señales, Sistemas y Radiocomunicaciones de la E.T.S. de Ingenieros de Telecomunicación de la Universidad Politécnica de Madrid.

Deseo expresar mi gratitud a Ramón García por su confianza en mi persona y su apoyo en todo momento para el desarrollo de la presente tesis.

También deseo expresar mi gratitud a todos mis compañeros de Departamento en estos últimos tres años por el gran ambiente de trabajo y su inestimable ayuda y, en particular, a Ricardo López Barquilla.

José Ignacio Puertas Tera

Madrid, a 24 de Enero de 2.000

Índice

Capítulo

1

Introducción

En las postrimerías del siglo XX, y cercano ya el tercer milenio, la utilización cada vez mayor de la voz como interfaz de comunicación hombre-máquina permitirá aumentar la sinergia con los sistemas informáticos, aprovechando al máximo las prestaciones que ofrecen en rapidez y eficiencia.

Los avances que se producen en el reconocimiento de voz son día a día más significativos. Los reconocedores de voz actuales manejan cada vez vocabularios más grandes y con menores tasas de error gracias al uso de algoritmos más eficientes, a la aparición de equipos más potentes y baratos, y al aumento de la complejidad de estos sistemas, al emplearse modelados más sofisticados y refinados.

No obstante, cuando estos reconocedores de voz se utilizan en situaciones reales, sus prestaciones se degradan muy rápidamente. Es necesario solventar esta falta de robustez para que el reconocimiento de voz pueda llegar a generalizarse.

Esta carestía de robustez aparece cuando los sistemas de reconocimiento que se diseñan para funcionar en el laboratorio son utilizados en situaciones reales. En una aplicación que vaya a ser utilizada en cualquier circunstancia, es obligatorio que funcione con independencia del locutor y del canal de comunicaciones, y que se consideren todos aquellos aspectos que pueden hacer que el sistema no sea viable, como el de las palabras muy breves o palabras fuera de vocabulario.

Antecedentes

Durante los últimos tres años he trabajado en el Departamento de Señales, Sistemas y Radiocomunicaciones de la E.T.S. de Ingenieros de Telecomunicación de Madrid, investigando el diseño e implementación de reconocedores fonéticos de voz flexibles, robustos y eficientes, de complejidad no muy elevada y de altas prestaciones.

Para llevar a cabo esta tarea, me he basado en la experiencia acumulada por el grupo en el campo del diseño de reconocedores de voz de palabras aisladas, y en el de arquitecturas software y hardware de *Integración de Telefonía y Ordenadores* (*ITO*), que posibilitan que un programa de ordenador dialogue, a través de una línea telefónica, con un interlocutor humano.

La premisa de partida consistió en realizar el entrenamiento de unos modelos que fueran representativos desde el punto de vista del lenguaje, pero que, a la vez, su número no fuera demasiado elevado, lo que hubiera impedido una implementación práctica. Se optó por la introducción de modelos fonéticos que, con mayor o menor resolución, son capaces de abarcar cualquier vocabulario.

Uno de los puntos fundamentales que desde los inicios se tuvo en cuenta fue la utilización de unos modelos fonéticos que presentaran la máxima variabilidad posible en cuanto a voces diferentes, de manera que la independencia de locutor permitiera al reconocedor de voz ser práctico y no sólo una investigación de laboratorio. Asimismo, dada la variabilidad del canal telefónico hacia donde se enfocó el diseño, era fundamental la utilización de algoritmos simples y con una convergencia muy rápida para adaptarse a cada nueva comunicación.

Otro de los aspectos fundamentales era la estructuración de los diferentes módulos de procesado en bloques independientes, que pudieran repartirse entre diferentes dispositivos para balancear cargas y sacar el máximo provecho posible de los recursos disponibles. La utilización de una arquitectura de desarrollo formado por un *ordenador personal* (*PC*) y una *tarjeta de procesado digital* (*DSP*) específica fue clave para lograr este fin.

La presente memoria pretende exponer el estado actual de mi trabajo en un área de investigación que está en continua evolución, y en la que se empieza a producir una generalización de esta nueva interfaz hombre-máquina.

Objetivos

Los objetivos de esta tesis están centrados en el diseño de reconocedores fonéticos de voz robustos de vocabulario flexible, destinados a aplicaciones que funcionen a través del canal telefónico en sistemas de centralita automática.

Los dos aspectos fundamentales que se desarrollan son: la flexibilidad del reconocedor de voz para adaptarse a diferentes vocabularios y su robustez con respecto a diversos factores.

La flexibilidad de un reconocedor se consigue a través de la utilización de un conjunto de unidades convenientemente elegidas, cuya concatenación proporciona los modelos de palabras. El objetivo de esta tesis es la utilización de unidades fonéticas, sean éstas libres de contexto (monofonemas) o dependientes de contexto (bifonemas y trifonemas), y la posibilidad de introducción de modelos subfonéticos.

Otro aspecto considerado es la utilización de un algoritmo de decodificación sencillo y eficiente para ser usado en el reconocimiento de palabras conectadas.

Con el fin de aumentar la robustez del sistema de reconocimiento, se introducen una serie de técnicas que permiten mantener las prestaciones con respecto a una serie de factores:

- *Robustez frente a diferentes locutores*: En las aplicaciones telefónicas donde el sistema puede ser utilizado por cualquier usuario, la independencia de locutor resulta fundamental.
- *Robustez frente a diferentes canales de comunicación y ambientes acústicos:* Los canales de comunicación son muy variados debido a que se establecen a través de la red telefónica, lo que hace que éstos presenten diferentes niveles de ruido y distorsiones lineales. Las características acústicas del recinto y las voces interferentes también afectan al funcionamiento del reconocedor. El objetivo es utilizar algoritmos eficaces para compensar estos aspectos, y que permitan que el cómputo adicional no sea muy elevado para mantener la eficiencia del sistema.
- *Reconocimiento robusto de palabras breves:* Palabras como los dígitos (*uno*, *dos*, *tres*,...), *sí* y *no*, son muy importantes en cualquier sistema de reconocimiento, pero debido a su brevedad son las que más problemas presentan.
- *Robustez frente a palabras fuera de vocabulario*: En cualquier sistema de reconocimiento es necesaria la detección de palabras pronunciadas fuera del vocabulario para evitar un uso incorrecto por el usuario.
- *Evaluación de distancias entre modelos entrenados*: Un aspecto añadido es la utilización de mecanismos para evaluar distancias entre modelos, para fusionar los más similares entre sí, reduciendo su número, a la vez que se mantiene el nivel de discriminación.

Finalmente, se pretende llegar a incorporar el reconocedor de voz en un sistema *ITO* de procesado en tiempo real, para probar y verificar la flexibilidad del sistema y los mecanismos de robustez en situaciones reales. La eficiencia será un factor adicional para mejorar el funcionamiento en tiempo real.

Estructura de la Tesis

La memoria de esta Tesis Doctoral está compuesta por seis capítulos, el primero de los cuales se corresponde con esta introducción.

El capítulo 2 hace una exposición de los diferentes enfoques y tecnologías que existen en el reconocimiento de voz, así como una descripción de los modelos ocultos de Markov (*HMM*), que es la metodología básica que ha sido empleada para la realización de la presente tesis.

El capítulo 3 hace una descripción de una arquitectura de reconocimiento de voz de vocabulario flexible a partir del modelado de unidades fonéticas, destinado al funcionamiento en una aplicación de telefonista automática en tiempo real.

El capítulo 4 describe los diferentes métodos empleados en el reconocedor de voz para aumentar la robustez frente a diferentes locutores, diferentes canales de comunicación, palabras fuera de vocabulario y palabras cortas.

Los resultados experimentales, tanto los obtenidos en condiciones controladas de laboratorio como los conseguidos tras la implementación en tiempo real, se muestran en el capítulo 5.

Finalmente, en el capítulo 6 se exponen las conclusiones y las líneas de trabajo futuras que la presente tesis deja abiertas.

Capítulo

2

El Reconocimiento de Voz

Este capítulo pretende situar el reconocimiento de voz. Para ello, se contemplan las diferentes alternativas que históricamente han ido aparecido, los sistemas de reconocimiento de voz actuales y lo que todavía queda por mejorar en este campo.

El apartado 2.1 trata la distinción entre el reconocimiento de voz y la inteligencia humana, y compara el funcionamiento de los reconocedores de voz al estado del arte y los límites que imponen las capacidades humanas.

En el apartado 2.2 se describen los diferentes enfoques y metodologías que se han ido empleado a lo largo del tiempo para realizar la decodificación de una secuencia de palabras a partir de una señal de voz.

A continuación, en el apartado 2.3 se estudia el sistema de reconocimiento de voz más usado y que mejores resultados ha obtenido hasta el momento, los modelos ocultos de Markov (*HMM*). Se describe la metodología de entrenamiento-reconocimiento que utilizan, los diferentes tipos de modelado estadísticos que han aparecido y las diferentes alternativas que existen en la obtención de estos modelos en función de los requisitos que las aplicaciones necesitan.

Finalmente, en el apartado 2.4 se hace un análisis de las diferentes aplicaciones que utilizan reconocimiento de voz.

2.1 Límites del Reconocimiento. Entendimiento

El reconocimiento de voz forma parte del ámbito general del entendimiento y de la inteligencia, siendo su objetivo último que una máquina sea capaz de comprender lo que se está pronunciando. Por tanto, lo importante no es detectar la secuencia de fonemas, sílabas o palabras pronunciadas, sino extraer las ideas que se estructuran en palabras a partir de un conjunto de reglas sintácticas y semánticas, para que la máquina dé una respuesta en consecuencia y se pueda establecer un diálogo, teniendo en cuenta todas las ambigüedades y redundancias que el lenguaje natural presenta.

Todavía se está muy lejos de alcanzar este objetivo, aunque ya se están dando los primeros pasos, evolucionando desde reconocedores de palabras aisladas con vocabularios pequeños a reconocedores de grandes vocabularios con restricciones gramaticales, e incluso semánticas, estas últimas todavía muy incipientes. A este nivel de desarrollo, aunque los resultados que se obtienen en condiciones de laboratorio son muy buenos, hay que conseguir que éstos se mantengan cuando se apliquen en sistemas reales.

Comparando el funcionamiento de los actuales reconocedores de voz de vocabularios muy grandes (> 65000) en situaciones normales con las capacidades humanas, cometemos hasta diez veces menos errores que éstos [Lip96]. Esta diferencia es incluso superior cuando se trata de reconocer voz degradada. La introducción de sistemas para la compensación de los diferentes tipos de degradación que pueden aparecer en la voz es todavía una asignatura pendiente. Generalmente, se utilizan técnicas que intentan solventar esa pérdida de las propiedades utilizando muestras de voz degradada. Sin embargo, estos mecanismos resultan muy rudimentarios si se comparan con los robustos tipos de adaptación que los seres humanos presentan frente a una gran multitud de situaciones, por ejemplo, con señales de voz recortadas o eliminación de ciertas bandas de frecuencia. Además, los seres humanos presentan una adaptación muy rápida a las variabilidades que se producen de manera natural, ya sean causadas por nuevos locutores, variaciones de la velocidad del habla, reverberación, la acústica de la sala y las características del canal debido a la recepción de señales reflejadas y otros efectos acústicos. Finalmente, la capacidad de los seres humanos supera con creces a los reconocedores actuales en cuanto a la habilidad para distinguir nuevas palabras y sonidos ambientales no vocálicos de palabras correctas.

Se ha demostrado que es necesario mejorar considerablemente el análisis acústico-fonético de bajo nivel para obtener mayores beneficios en la utilización de reconocedores de voz con vocabularios muy grandes. Para conseguirlo es necesario tener en cuenta los tres puntos siguientes:

- El uso de modelados acústicos más refinados, para realizar una extracción de características más significativa. La introducción de mecanismos auditivos permite obtener unas mejoras importantes.

- La utilización de nuevas y mejores técnicas para aumentar la robustez y la capacidad de adaptación de los reconocedores de voz.
- Mecanismos para que los reconocedores sean capaces de manejar nuevas palabras y distinguirlas de las ya conocidas y que, a su vez, éstos puedan distinguir sonidos ambientales de lo que es voz.

2.2 Alternativas Actuales del Reconocimiento de Voz

En la actualidad, existen tres enfoques a la hora de plantear cualquier sistema de reconocimiento de voz [Del93]. Cada uno de estos enfoques presenta unas características diferentes, pudiéndose combinar varios de ellos para aprovechar las ventajas que cada uno ofrece por separado. Son los siguientes:

- *Enfoque acústico-fonético*: Engloba todos aquellos procesos destinados a realizar una decodificación de palabras a partir de las características diferenciadoras que la voz presenta y de un conjunto de reglas, dispuestas en forma de sistema experto, que existen en el habla.
- *Enfoque de patrones***:** Técnicas basadas en el reconocimiento de patrones, que decodifican lo pronunciado a partir de un conjunto de modelos que se captan de forma automática en una fase de entrenamiento, a diferencia del enfoque acústico-fonético que analiza la voz directamente para extraer las reglas que gobiernan el lenguaje. Para ello, se realiza una extracción de características con la mayor información posible sobre el habla, eliminando la información asociada con el locutor o el medio de comunicación. Estos métodos constan de dos fases bien diferenciadas: por un lado, la fase de entrenamiento, donde se generan los modelos que sirven de referencia, para lo cual se emplea un conjunto de bases de datos con voces grabadas que presenten la suficiente variabilidad; por otro lado, una fase de reconocimiento, donde se realiza una comparación entre las referencias obtenidas y las pronunciaciones, eligiéndose la secuencia de palabras cuya distancia a los modelos de referencia sea menor. Las técnicas más relevantes son:
    - *Dynamic Time Warping* (*DTW*): Técnicas de comparación de patrones de tipo determinista [Vin68].
    - *Modelos Ocultos de Markov* (*HMM*: *Hidden Markov Models*): Técnicas de comparación de patrones de tipo estocástico [Bak75] [Jel76].
- *Enfoque de la inteligencia artificial*: Se basa en las conocidas *redes neuronales*. Estas técnicas, al igual que en el caso del enfoque de patrones, obtienen las reglas que gobiernan el habla de una forma automática. Pero mientras en el enfoque de patrones los modelos se obtienen de forma

totalmente automática, en el reconocimiento basado en la comparación de patrones es necesario introducir cierto conocimiento.

2.2.1 Enfoque Acústico-Fonético.

Este enfoque se apoya en la fonética acústica, que postula que existe un número finito y diferenciado de unidades fonéticas en el lenguaje hablado, y que éstas se pueden caracterizar a través de un conjunto de propiedades que se manifiestan en la señal de voz, o en su espectro, a lo largo del tiempo. Los fonemas se estructuran en palabras, y éstas en frases, que son las que modelan las ideas. Esta metodología se basa en el diseño de sistemas expertos, que es la manera en la que supuestamente funciona el cerebro humano, imponiendo directamente las reglas que gobierna el lenguaje desde las características acústicas hasta los niveles superiores, a partir del análisis de la señal de voz, mientras que el resto de enfoques utilizan métodos automáticos para capturar dichas reglas.

Aunque las propiedades acústicas son altamente variables, dependiendo del locutor, y la concatenación de la secuencia de unidades fonéticas altera las propiedades que tendrían cada una de éstas por separado (*coarticulación*), se asume que las reglas que gobiernan la producción de estas unidades existen y se pueden aprender.

En un reconocedor basado en sistemas expertos, existe una fase que extrae las reglas de producción a partir del análisis de la señal de voz. En el momento del reconocimiento, se aplican estas reglas para obtener la secuencia de palabras reconocidas a partir de las características detectadas.

En cualquier reconocedor de voz basado en un enfoque acústico-fonético se distingue un conjunto de bloques de procesado:

- *Análisis Acústico de Voz*: Sobre la señal de voz, obtenida de las variaciones de presión sobre un micrófono, se aplica un conjunto de procesados iniciales similares a los que el sistema auditivo realiza. Este análisis es básicamente una transformación de la señal del dominio temporal al dominio frecuencial, por lo que suele ser típico la existencia de un banco de filtros perceptuales utilizando la transformada de Fourier. Otros tipos de análisis que se utilizan son el número de cruces por cero o la envolvente espectral, esta última obtenida a través del cálculo de los coeficientes de predicción lineal (*LPC*). Estas características obtenidas directamente de la señal de voz se denominan *características acústicas*.
- *Detección de Características Fonéticas*: A partir de las características acústicas obtenidas del análisis inicial de la señal de voz, se realiza la extracción de un conjunto de parámetros que se consideran discriminativos desde un punto de vista perceptual. Estos parámetros se denominan *características fonéticas*. Una característica fonética se puede definir como la propiedad mínima que presenta un fragmento de la señal de voz y que

discrimina a dos unidades fonéticas diferentes. Las características fonéticas más usuales son: la detección del grado de sonoridad, el cálculo de la frecuencia fundamental de las cuerdas vocales, el grado de fricción, los valores de los formantes de la voz (máximos del espectro de la señal que se corresponden con las frecuencias de resonancia de la cavidad bucal), etc.

- *Fase de segmentación y etiquetado*: Incluye la partición de la señal de voz en regiones donde las características fonéticas son similares y pueden asignarse a una, o posiblemente varias, categorías fonéticas. En esta fase se obtiene una secuencia ordenada de categorías fonéticas, que pueden llegar a solaparse, y que son las que se utilizan para realizar la decodificación de la secuencia de palabras.
- *Fase de discriminación*: Busca la decodificación de la secuencia de palabras pronunciadas a partir del conjunto de categorías fonéticas obtenidas en la fase de segmentación y etiquetado. Para ello, se utilizan las reglas sintácticas y semánticas que se obtienen del estudio de la señal de voz, que imponen un conjunto de restricciones que permiten discriminar entre la secuencia de palabras pronunciadas.

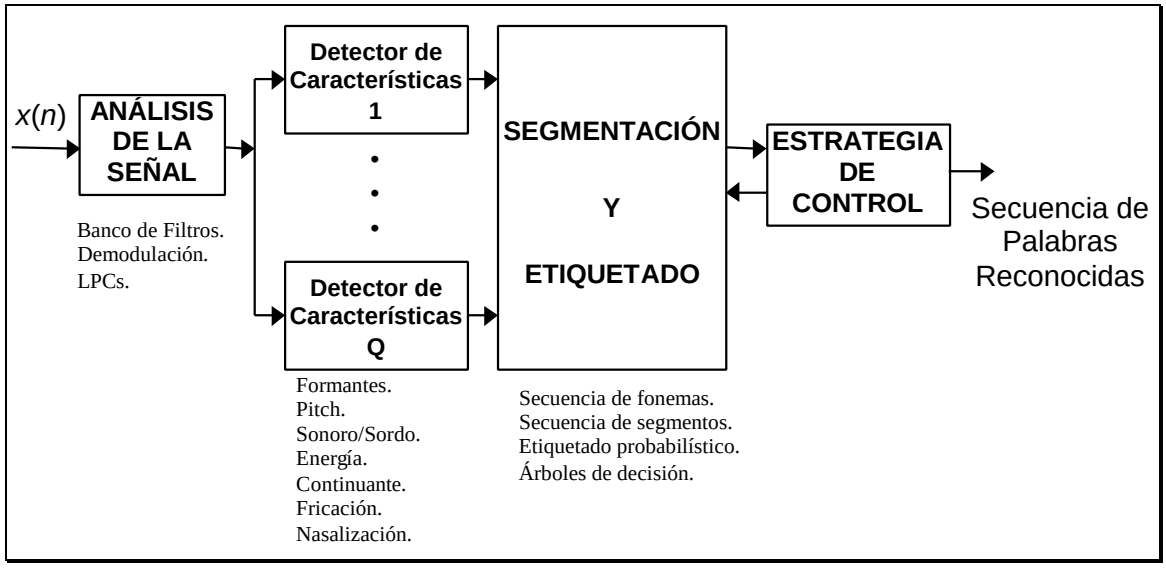

Figura 2-1. Bloques de un reconocedor de voz basado en un enfoque acústico-fonético.

En la Figura 2-1 se refleja la interrelación de los diferentes procesados antes mencionados.

La fase de discriminación se realiza a partir del bloque de estrategia de control. La estrategia de control puede funcionar de dos maneras diferentes:

- *Estrategia de control Bottom-Up*: Parte de las características obtenidas en los bloques de análisis acústico y fonético, para construir la secuencia de palabras reconocidas.

- *Estrategia de control Top-Down*: Esta estrategia empieza por escoger un conjunto de secuencias de palabras candidatas, las cuales van siendo descartadas en función de las características fonéticas encontradas. La secuencia de palabras reconocida se corresponde con la que tenga mayor posibilidad de haber sido generada.

Estas dos estrategias de control presentan el inconveniente de que una vez que se dan por buenas las categorías fonéticas detectadas, éstas no pueden volver a verificarse y si alguna fuera errónea el reconocedor no podría recuperarse. Por eso, suelen implementarse estrategias de control mixtas que combinan ambas.

Una estrategia de control de tipo mixto empieza por un control estratégico tipo *Bottom-Up*, que obtiene un conjunto de hipótesis a partir de las características fonéticas más fiables. Luego, usando el resto de características fonéticas se realiza un procesado *Top-Down* que se encarga de verificarlas. A continuación, se vuelve a generar otro nuevo conjunto de hipótesis que se vuelven a verificar. Este proceso cíclico acaba cuando se obtiene una secuencia de palabras bastante fiable.

2.2.2 Enfoque de Patrones.

Este enfoque, basado en el reconocimiento de patrones, es el que actualmente cuenta con un mayor desarrollo y el que ofrece unos mejores resultados. Como ya se mencionaba en [Cox88], todo clasificador de patrones tiene dos modos de funcionamiento:

- *Modo de entrenamiento*: Para cada una de las clases a entrenar se construye un modelo utilizando un conjunto de ejemplos que sirven de referencia.
- *Modo de reconocimiento*: El patrón a reconocer se compara, utilizando una métrica determinada, con todos los modelos de clase y se identifica con el más cercano.

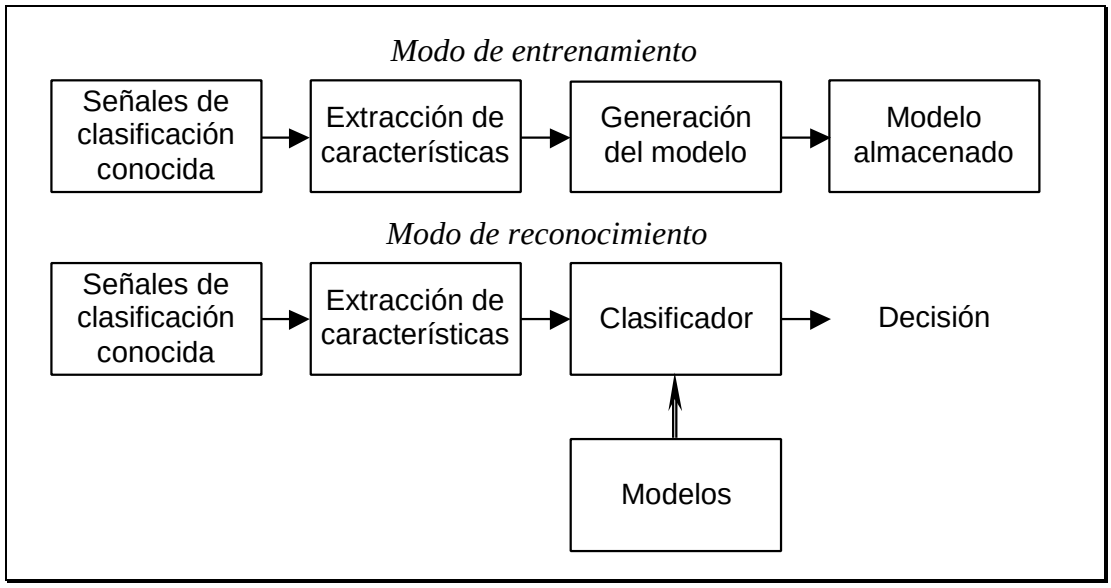

Figura 2-2. Sistema de clasificación de patrones en modo de entrenamiento y de reconocimiento.

En la Figura 2-2 se muestran los diagramas de bloques que un clasificador de patrones utiliza en los dos modos de funcionamiento: entrenamiento y reconocimiento. En ambos casos, se empieza con una extracción de características que puede llegar a resultar crítica en el correcto funcionamiento del sistema de reconocimiento porque:

- Es necesario extraer la información dentro de la señal que es más importante para la *discriminación de patrones* entre diferentes clases. Una buena extracción de características es aquella que resalta los parecidos dentro de cada clase y aumenta las diferencias entre diferentes clases.
- Se busca una *disminución de datos* de manera que la manipulación de los patrones sea computacionalmente factible. Por ejemplo, una señal de voz telefónica tiene un régimen binario de $64\ Kb/s$, que es demasiado grande para manejarse toda de una vez.

Las características que se extraen en el reconocimiento de voz suelen ser el espectro instantáneo de la señal de voz o la forma instantánea del tracto bucal. En reconocimiento de voz resulta muy importante la selección de las características, siendo la precisión de reconocimiento altamente dependiente del tipo y número de parámetros usados. Dado que las articulaciones que se utilizan en la producción de la voz cambian muy lentamente con el tiempo, para el caso del reconocimiento de voz es suficiente con extraer características en intervalos regulares de $10\ ms$ a $20\ ms$.

En el reconocimiento de voz basado en el enfoque de patrones existen dos metodologías diferentes. Son las siguientes:

- *Reconocimiento determinista*: La metodología más utilizada dentro de este ámbito es la *alineación dinámica de patrones*, en ingles denominada *Dynamic Time Warping* (*DTW*).

- *Reconocimiento estocástico*: Los modelos ocultos de Markov, más conocidos por su denominación en inglés como *Hidden Markov Models* (*HMM*), son los más representativos.

Los reconocedores de voz basados en *HMM* sobresalen sobre los basados en *DTW*, dada su mayor capacidad de modelado y su utilización en aplicaciones más complejas, siendo los que mejores resultados han obtenido hasta el momento.

2.2.2.1 Dynamic Time Warping (DTW).

La técnica denominada *Dynamic Time Warping* (*DTW*), o *alineamiento dinámico de patrones*, se basa en la alineación temporal de las características de una pronunciación respecto de otra que sirve de referencia, antes de calcular una puntuación global. Para ello, se utilizan técnicas de programación lineal.

La palabra inglesa *Warped* da la clave sobre este sistema de comparación de patrones. Esta palabra significa que la referencia se adapta estirándose o comprimiéndose en el tiempo, de manera que los vectores de características se alinean con el patrón de referencia antes de hacer la medida de distancia entre ambas pronunciaciones.

La principal ventaja de esta técnica es su sencillez, al usar algoritmos muy simples y con un grado de computación reducida, que permiten su utilización en aplicaciones sencillas con resultados muy buenos. La alineación dinámica de patrones parte de la necesidad de tener un *patrón* (*template*) de referencia con el que se debe alinear la pronunciación a reconocer. El principal problema de esta técnica es que es muy difícil de generalizar, siendo una metodología que requiere un procesado muy elevado y de difícil adaptación para tareas muy complejas y con vocabularios muy grandes.

Algoritmos de *programación dinámica* (*PD*) aplicados al reconocimiento de voz pueden verse en [Del93].

Al principio, antes de la generalización del reconocimiento basado en *HMM*, el *DTW* era la metodología más madura y con la que se obtenían los mejores resultados en el reconocimiento de palabras aisladas, e incluso palabras concatenadas. Para ello, incluso se fabricaron chips *VLSI* (*Very Large Scale Integration*) específicos para la realización de las tareas de programación dinámica. Sin embargo, con la aparición de los modelos de Markov, éstos tomaron el protagonismo, dada su mayor flexibilidad, robustez y su capacidad de abordar problemas más complejos.

2.2.2.2 Modelos Ocultos de Markov (HMM).

Los modelos ocultos de Markov (*HMM*: *Hidden Markov Models*) representan el otro conjunto de técnicas para el reconocimiento de voz. Están basados en la comparación de unos patrones de referencia denominados *modelos* con los vectores de características generados por la pronunciación a reconocer. Los *HMM* surgieron de la

necesidad de utilizar técnicas de modelado estadístico que fueran capaces de afrontar el problema de la variabilidad de la voz, que crece de manera importante cuando la complejidad y el tamaño del vocabulario aumentan. Ésta es la diferencia fundamental con los reconocedores de voz basados en *DTW*, que utilizan otras pronunciaciones deterministas como patrones de referencia.

Los modelos de Markov han sido la metodología elegida para el diseño e implementación de un reconocedor de voz fonético en aplicaciones telefónicas. En el siguiente apartado 2.3 se analizarán las diferentes posibilidades y tipologías que existen actualmente.

2.2.3 Enfoque de la Inteligencia Artificial. Redes Neuronales.

Este enfoque se basa en las denominadas *redes neuronales* (*ANN*: *Artificial Neural Networks*). Es el de aparición más reciente y está bastante desarrollado. Sin embargo, no existe todavía ningún sistema que funcione de modo exclusivo con este enfoque. En general, las redes neuronales se complementan con otras estrategias, por ejemplo con *HMM*, para formar los denominados *sistemas híbridos*. En [Bou96] se hace una descripción del estado del arte de sistemas que combinan las redes neuronales con los modelos de Markov. Otras veces, se utilizan para resolver problemas concretos, dada su sencillez y eficiencia, como el diseño de un cuantificador vectorial [Bea90].

El elemento fundamental de las técnicas basadas en la inteligencia artificial es la *neurona*. La neurona de la inteligencia artificial presenta un funcionamiento similar a las neuronas de cualquier sistema nervioso, aunque de una manera muy simplificada. Las neuronas se definen a partir de un conjunto de entradas y de salidas, que se conectan entre sí para crear una estructura más o menos extensa donde la información se va almacenando.

Las principales ventajas de este enfoque son: la sencillez con que esta red de neuronas interconectadas puede entrenarse a partir de un conjunto de datos de referencia de forma automática, proceso denominado *aprendizaje*, y la posibilidad de procesado en paralelo, por lo que no es necesario equipos de gran potencia y rapidez. Su principal inconveniente es la dificultad de controlar si este entrenamiento se está produciendo de forma adecuada o no, al ser un método no supervisado.

En [Del93] y [Bea90] se describen las nociones básicas sobre las redes neuronales, desde la definición de neurona, pasando por sus diferentes arquitecturas de interconexión (la que tiene mayor capacidad de modelado y obtiene unos mejores resultados es el denominado *perceptrón multicapa*), hasta sus aplicaciones, como el reconocimiento de voz o la construcción de cuantificadores vectoriales.

2.3 Reconocimiento de Voz con HMM

Los *HMM* son el enfoque de comparación de patrones de tipo estocástico más popular y exitoso en el reconocimiento de voz. Esto se debe a la existencia de algoritmos elegantes y eficientes tanto en entrenamiento como en reconocimiento.

Estos modelos se utilizan en dos niveles. Por un lado, para modelar palabras mediante la utilización de modelos acústicos. Por otro, para modelar el lenguaje, definiendo una red de reconocimiento en la que se imponen las restricciones gramaticales, a partir de las cuales se realiza la decodificación de la secuencia de palabras más probable utilizando la información del nivel acústico. En cualquier caso, la utilización de los *HMM* en el reconocimiento de voz se aplica teniendo en cuenta dos suposiciones:

- La voz se puede dividir en *segmentos* (*estados*) en los que la señal de voz puede considerarse prácticamente *estacionaria*, es decir, en la ventana de análisis la señal mantiene la estructura de principio a fin. Se asume que las transiciones entre segmentos contiguos son instantáneas.
- La probabilidad de observación de que un vector de características se genere sólo depende del segmento actual, y no de símbolos anteriores o posteriores. Esto recibe el nombre de *supuesto de independencia*.

2.3.1 Descripción de un Modelo de Markov (HMM).

Un modelo de Markov, tanto a nivel acústico como gramatical, está formado por un conjunto de *estados* (*nodos*) y de *transiciones* (*arcos*) definidos a partir de un conjunto de parámetros:

- N: Número de estados del modelo. Los estados se denotan como $s = \{s_1, s_2, ..., s_N\}$. $q_i\ (t)$ indica la probabilidad de que un vector en el tiempo t esté en el estado s_i.
- A: Matriz de transición de probabilidades de estados. Cada uno de los elementos se define como:

$$a_{ij} = p\left(q_j(t+1) \mid q_i(t)\right) \qquad [2.1]$$

Esta matriz no necesita estar completa, sólo en casos como el de la Figura 2-3 tendría definidas todas las posibilidades. En general, estos modelos almacenan la información temporal sobre la señal, por lo que presentan una estructura ordenada y las transiciones posibles están limitadas. En cualquier caso, las probabilidades de transición cumplen la propiedad de que la suma de valores de todos los arcos que salen de un estado valen uno. La ecuación [2.2] muestra dicha propiedad.

$$\sum_{j=1}^{N} a_{ij} = 1 \qquad [2.2]$$

- B: Matriz de probabilidades de generación de vectores o distribuciones. Cada uno de sus elementos se define como la probabilidad de que un determinado vector de características en un determinado tiempo haya sido generado por la distribución correspondiente:

$$b_j(y(t)) = p(y(t) \mid q_j(t)) \qquad [2.3]$$

Existen diferentes alternativas para modelar estas funciones densidad de probabilidad. En el apartado 2.3.2 se hace una descripción de los diferentes tipos de modelos de Markov asociados con estas distribuciones.

- p: Vector con las distribuciones de estado iniciales:

$$p_i = p(q_i(1)) \qquad [2.4]$$

Representa las condiciones iniciales de partida de los modelos de Markov.

Así, cualquier modelo de Markov se puede definir como:

$$l = (A, B, p) \qquad [2.5]$$

que se corresponde con los tres conjuntos de medidas de probabilidad que hay que especificar.

2.3.1.1 Forma de Mealy y Forma de Moore.

Como se describe en [Del93], existen dos posibilidades para definir los modelos de Markov dependiendo del lugar donde se localicen las funciones densidad de probabilidad:

- *Forma de Moore*: Los elementos de B se localizan en los nodos o estados, por eso, también se denominan como de *emisión de estado*. Es el tipo más común utilizado en el modelado acústico, aunque existen excepciones, como el caso de los reconocedores de *IBM* [Bah88], que utilizan la forma de Mealy. Las distribuciones se nombran como b_i, que indica la probabilidad de emisión del estado i-ésimo.
- *Forma de Mealy*: Los elementos de B se localizan en los arcos o transiciones, por lo que también se denominan como de *emisión de transición*. Es la forma característica que se utiliza en el modelado del lenguaje, cuando cada modelo de palabra se sitúa en un arco del modelo de red. Las distribuciones se nombran como b_{ij}, que indica la probabilidad de emisión del arco que va del estado i-ésimo al estado j-ésimo.

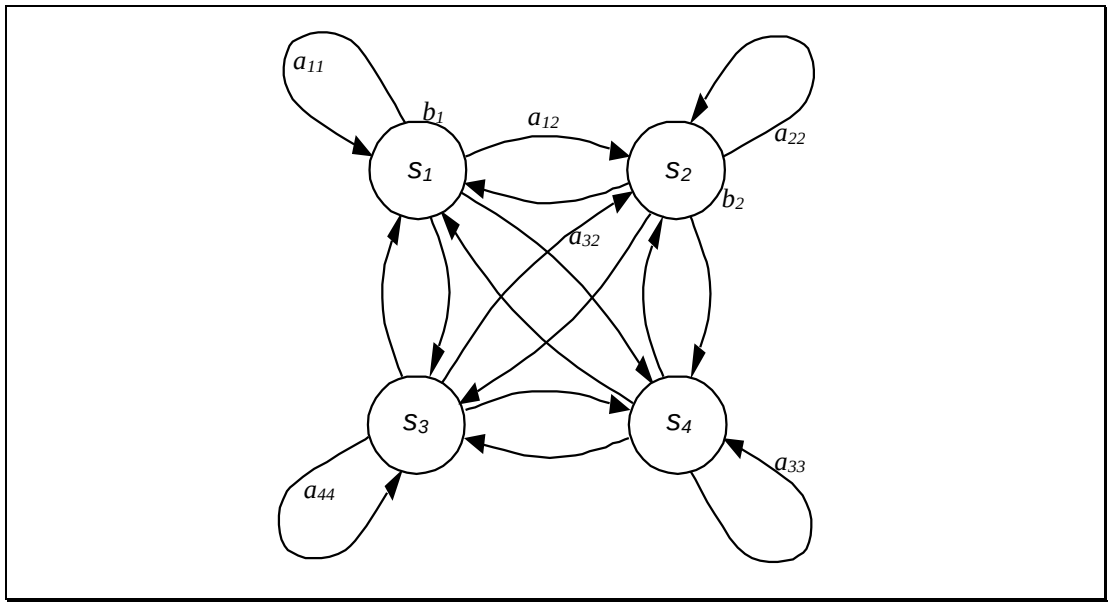

Figura 2-3. Ejemplo de un modelo de Markov formado por 4 estados que están totalmente interconectados. Cada uno de los arcos está caracterizado por una probabilidad de transición (a_{ij}: Probabilidad de ir del nodo *i* al nodo *j*) y, en este caso, es una forma de Moore al localizarse las distribuciones de probabilidad en los estados.

En la Figura 2-3 se puede ver un ejemplo de modelo de Markov formado por 4 estados totalmente interconectados. Se observa que en una forma de Moore las probabilidades de transición (elementos de la matriz A) se localizan en los arcos, mientras que las probabilidades de emisión (elementos de B) están en los nodos. En el reconocimiento de voz, donde existe una secuencia temporal de vectores de características, se utilizan modelos que tienen un inicio y un fin, se denominan modelos de *Bakis* o de *izquierda-derecha*. En la Figura 3-7 y en la Figura 3-8 se muestran dos estructuras de Markov que se usan para modelar unidades fonéticas. La primera estructura se denomina de *saltos simples* o de *primer orden*, mientras que la segunda de *saltos dobles* o de *segundo orden*.

2.3.2 Tipos de Modelos de Markov.

Un modelo de Markov se clasifica en función de la matriz B, o de distribuciones de probabilidad de salida. Tanto para las modelos de Markov con forma de Moore o de Mealy, dependiendo de cómo sean los elementos de esta matriz, se pueden clasificar los modelos de Markov en tres categorías:

- *Discretos*: También denominados *DHMM* (*Discrete HMM*). En este caso, las observaciones son vectores de símbolos de un alfabeto finito con M elementos diferentes, cada uno denominado *codeword*, que se agrupan en un denominado *codebook* $V = \{v_1, v_2, \cdots, v_M\}$. La distribución de probabilidad de un símbolo se define como $B = \{b_j(k)\}$ donde

$$b_j(k) = P[x = v_k \mid s_j], \qquad 1 \le k \le M \qquad [2.6]$$

es la distribución de símbolos en el estado j, con $j = 1, 2, ..., N$.

En este caso, basta con especificar N y M, las observaciones de símbolos V y los tres conjuntos de medidas de probabilidad A, B y p, que definen al modelo de Markov.

- *Continuos*: También denominados *CHMM* (*Continuous HMM*). Se asume que las distribuciones son densidades de probabilidad de espacios de observación continua. Se impone la restricción de que estas distribuciones presentan una determinada forma para obtener un número abordable de parámetros a estimar.

La forma más extendida de distribución es la de mezcla de un conjunto de densidades base g de una familia G con una forma paramétrica simple. Principalmente se utilizan densidades de base $g \in G$ de tipo gaussiano, para lo cual basta con definir la matriz de medias y la de covarianzas. Para la estimación de estos parámetros es necesario un gran número de vectores de entrenamiento, que aumentan de forma lineal con el número de modelos a entrenar. La función de distribución de probabilidad se puede escribir como:

$$b_j(x) = \sum_{k=1}^{M} C_{jk} \cdot \mathrm{N}\left(x, m_{jk}, \Sigma_{jk}\right), \qquad 1 \le j \le N \qquad [2.7]$$

donde x representa un vector de observación, $\mathrm{N}(.)$ una función normal, caracterizada por unos valores medios m_{jk} y una varianzas S_{jk}, M el número de mezclas por estado y C_{jk} los pesos asociados con cada una de las funciones, cumpliéndose que:

$$\sum_{j=1}^{N} C_{ij} = 1 \qquad [2.8]$$

- *Semicontinuos*: También denominados *SCHMM* (*Semicontinuous HMM*). Surgieron de la necesidad de entrenar un gran número de modelos con bases de datos limitadas [Hua90].

Los modelos semicontinuos, al igual que lo continuos, se modelan a partir de un conjunto de mezclas de funciones densidad de probabilidad gaussiana. La principal diferencia radica en que las funciones base son comunes a todos los modelos, como ocurre en el caso de los modelos discretos, donde existe un *codebook* común a todos ellos. Así, un modelo semicontinuo se define con los pesos asociados a cada una de las funciones base.

La forma de la función $b_j(x)$ es idéntica a la de la expresión [2.7], con la diferencia de que el conjunto de funciones normales es común a todos los modelos de estado de Markov. La expresión resulta ser:

$$b_j(x) = \sum_{k=1}^{L} C_{jk} \cdot \mathrm{N}\left(x, m_{jk}, \Sigma_{jk}\right), \qquad 1 \le j \le N \qquad [2.9]$$

donde L indica el número total de funciones normales en lugar del número de mezclas por modelo de estado.

2.3.3 Entrenamiento-Reconocimiento Usando HMM.

Como ya aparece en [Rab86], los modelos de Markov se caracterizan por tres problemas que hay que resolver para que se obtengan modelos útiles en aplicaciones reales:

- *Problema de evaluación*: Dada una secuencia de observaciones y un modelo, se busca cómo calcular la probabilidad de que la secuencia observada haya sido producida por dicho modelo.
- *Problema de estimación*: Dada una secuencia de observaciones y un modelo, se busca cómo elegir una secuencia de estados que sea óptima en algún sentido.
- *Problema de entrenamiento*: Dada una secuencia de observaciones de entrenamiento, se busca cómo obtener los parámetros del modelo de forma óptima.

Estos problemas se concretan en dos fases, tal como aparece en [Del93]. Son las siguientes:

- *Entrenamiento*: Dada una secuencia de observaciones de entrenamiento, cómo se entrena un modelo de Markov para que represente a dichas observaciones. Esto es, cómo se puede estimar la matriz de probabilidades de transición, A, las distribuciones de probabilidad, B, y las distribuciones de estado iniciales, p, a partir de una secuencia de observaciones.
- *Reconocimiento*: Dado un modelo de Markov ya entrenado, cómo encontrar la probabilidad de que una secuencia de observación haya sido generada por dicho modelo.

2.3.3.1 Algoritmos de Entrenamiento-Reconocimiento.

Tanto en entrenamiento como en reconocimiento, y para solucionar los problemas planteados en [Rab86], existen dos metodologías:

- *Reestimación de Baum-Welch basada en el algoritmo Forward-Backward (F-B)*: El *F-B* es un algoritmo eficiente que se usa para el cálculo de $f_{X|M}(X \mid l)$.

Para ello, el algoritmo *F-B* utiliza una variable de probabilidad hacia adelante denominada $a_t(i)$, como la probabilidad conjunta de la secuencia de observaciones $X = (x(0), x(1), \cdots, x(t))$ y el estado i en el tiempo t del modelo l. Se define y se puede expresar de manera recursiva como:

$$\begin{aligned} a_t(i) &= f_{X,S|l}(x(0), x(1), \cdots, x(t), s(t) = i \mid l) \\ &= \sum_{j=1}^{M} \left(a_{t-1}(j) \cdot a_{ji}\right) f_{X|S,l}(x(t) \mid s(t) = i \mid l) \end{aligned} \qquad [2.10]$$

Además, existe otra variable de probabilidad hacia atrás denominada $b_t(i)$ con unas expresiones muy similares a las de [2.10]:

$$\begin{aligned} b_t(i) &= f_{X,S|l}(s(t) = i, x(t+1), x(t+2), \cdots, x(T-1) \mid l) \\ &= \sum_{j=1}^{M} \left(a_{ij} \cdot b_{t+1}(j)\right) f_{X|S,l}(x(t+1) \mid s(t+1) = j \mid l) \end{aligned} \qquad [2.11]$$

El método de entrenamiento de reestimación *Baum-Welch* es una técnica de maximización de las probabilidades, también denominado *EM* (Estimate-Maximise), que de forma iterativa utiliza las probabilidades hacia adelante y hacia atrás que proporciona el algoritmo *F-B*.

- *Algoritmo de decodificación de Viterbi*: Es un algoritmo de tipo *MAP* (Máximo a posteriori). Proporciona la secuencia de estados s^{MAP} de un modelo l dada la secuencia de observaciones $X = (x(0), x(1), \cdots, x(T-1))$, obteniéndose:

$$\begin{aligned} s^{MAP} &= \arg\max_{S} f_{X,S|l}(X, s \mid l) \\ &= \arg\max_{S} \left(f_{X|S,l}(X \mid s, l) \cdot P_{S|M}(s \mid l)\right) \end{aligned} \qquad [2.12]$$

Este algoritmo, según se explica en [Cox88] permite una reestimación del modelo sin necesidad de utilizar las probabilidades hacia atrás, $b_t(i)$. Esto proporciona un gran ahorro computacional y un procedimiento de entrenamiento muy sencillo. También consigue:

- Obtener un valor de *score* (*puntuación*) de similitud entre una secuencia de observación y un *HMM*.
- Fragmentar una señal no estacionaria en un conjunto de segmentos cuasiestacionarios.

Ambos algoritmos pueden aplicarse con modelos discretos, continuos y semicontinuos. El más eficiente es el de Viterbi, pues siendo más sencillo, obtiene unos resultados muy similares a los que pueden conseguirse con la reestimación de *B-W*. Por ello, en la presente tesis, tanto en entrenamiento, como en reconocimiento, se emplea el algoritmo de Viterbi. En [Rab93] se muestra cómo entrenar modelos continuos basándose en la segmentación proporcionada por el algoritmo de Viterbi, similar al procedimiento de cálculo de cuantificación vectorial *k*-means. En el apartado 3.3.5 se hace una descripción detallada de éste método.

Independientemente de si el método de entrenamiento y reconocimiento se basa en la reestimación de *Baum-Welch* o en el algoritmo de *Viterbi*, es posible introducir una simplificación que permite disminuir de forma importante el número de posibilidades en la red de modelos de Markov. Consiste en *podar* (*prune*) todos aquellos caminos que presenten unas probabilidades muy bajas de corresponderse con una secuencia de estados correcta, consiguiéndose una mayor eficiencia computacional. A estas metodologías se las denomina *Beam Search* y se aplicaban ya en las técnicas de alineamiento dinámico de patrones, *DTW*. En 3.6.3 se muestra la aplicación de dicha técnica.

2.3.4 Clasificación de los Modelos de Markov.

Hasta el momento todo lo referido a modelos de Markov ha sido independiente de si éstos se utilizan para modelar características subfonéticas o frases enteras. A continuación, se establece una clasificación del tipo de modelados que pueden ser utilizados:

- *Modelos de palabras*.
- *Modelos de sílabas*.
- *Modelos de semi-sílabas* (*demisyllables).*
- *Modelos fonéticos*.
- *Modelos subfonéticos*.

2.3.4.1 Modelos de Palabras.

En este caso, los *HMM* modelan palabras completas. Este modelado es el que tradicionalmente ha sido utilizado en reconocedores de voz de palabras aisladas, como en el reconocimiento de dígitos.

Un reconocedor basado en este tipo de modelos es difícil de generalizar para cualquier vocabulario porque el número de modelos crece de forma lineal con el tamaño del vocabulario y, además, es necesario disponer de una base de datos con un número suficiente de repeticiones para cada nueva palabra que se quiera añadir. Por tanto, la flexibilidad que proporciona es nula.

2.3.4.2 Modelos de Sílabas.

Los modelos de sílabas proporcionan una cierta flexibilidad, ya que la concatenación de las mismas forma palabras, evitando la necesidad de obtener bases de datos nuevas si se desea aumentar el tamaño del vocabulario. Sin embargo, esto no implica que no se requieran grandes bases de datos que presenten las diferentes posibilidades de combinación entre consonantes y vocales para formar sílabas (CV, VC, CCV, CVC, CVV, etc.). Aunque parezca que no es un gran inconveniente, supone un

número tan elevado de modelos, que el entrenamiento de los mismos resulta prácticamente de la misma magnitud que el de modelos de palabras completas.

2.3.4.3 Modelos de Semi-Sílabas.

Los modelos de semi-sílabas están formados por un grupo de consonantes (que pueden no existir) y la parte inicial de una vocal, o bien, por la parte final de una vocal y un grupo de consonantes.

Al modelar las sílabas en dos partes, el número total de modelos a entrenar se reduce de forma importante.

2.3.4.4 Modelos de Unidades Fonéticas.

Los fonemas representan un nivel superior de fragmentación de palabras. La modificación de un fonema, que es la unidad mínima dentro de una palabra, puede cambiarle el sentido a la misma. Hay que distinguirlo de alófono, que es cada una de las pronunciaciones reales del modelo ideal que representa el fonema. El número de fonemas que existe en cualquier lengua no tonal está bastante limitado, aún cuando el número de alófonos puede resultar muy elevado. Así, por ejemplo, para el castellano existen unos 30 fonemas diferentes, pero se pueden distinguir hasta 5.000 alófonos, como se especifica en [Pro92], que es el documento de definición de la base de datos de frases del proyecto Albayzín.

En función del grado de resolución que se quiera que presenten las *unidades fonéticas*[1] se pueden obtener modelos más o menos específicos. La manera en que se hace la división en unidades fonéticas depende del contexto donde el alófono se localice:

- *Monofonemas*: Son unidades totalmente libres de contexto. Un monofonema tiene en cuenta todas las posibles realizaciones de un fonema independientemente de sus vecinos.
- *Bifonemas*: Son unidades que dependen sólo de uno de sus contextos, ya sea éste el derecho (*bifonema derecho*) o el izquierdo (*bifonema izquierdo*).
- *Trifonemas*: Son unidades que dependen de ambos contextos a la vez.
- *Trifonemas generalizados*: Dado que el número de unidades va aumentando al ir considerando más detenidamente la posición de los fonemas en su contexto, puede llegar a ser tan elevado que su entrenamiento no fuera posible. Surge el trifonema generalizado como un primer nivel de compartición, en el que varios trifonemas cercanos se agrupan para reducir el número de modelos y que el entrenamiento de éstos sea mejor.

[1] Se utiliza *unidad fonética* en lugar de fonema o alófono, porque es mucho más general. Una unidad fonética puede corresponderse con todos los alófonos que puede presentar un fonema, o modelar a un único alófono. En general, representa un subconjunto de alófonos de un fonema.

En el caso del castellano existen unos 30 monofonemas, unos 30*30 = 900 bifonemas derechos o izquierdos (un número menor, pues no todas las combinaciones son posibles) y unos 30*30*30 = 27.000 trifonemas (muchos menos, porque al igual que en el caso anterior, tampoco todas las combinaciones son posibles). En función de la aplicación, hay que analizar hasta qué nivel de resolución se desea trabajar, dependiendo del tamaño de las bases de datos accesibles.

Evidentemente, como en el caso del modelado de sílabas, si se aumenta la resolución de modelos, se requerirían bases de datos cada vez mayores, con un número suficiente de repeticiones para entrenar cada uno de ellos. Si, además, se tiene en cuenta que el número de estados de cada modelo varía entre uno y cinco estados, el problema del entrenamiento aumenta de forma significativa. Es aquí donde surge la necesidad de la reducción del número de parámetros. Si los trifonemas generalizados permiten cierta reducción, esto es a costa de que muchas veces los modelos generados pueden no resultar tan buenos al sumarse modelos de forma completa que no tienen por qué ser totalmente coincidentes. El paso siguiente, por tanto, es trabajar a nivel subfonético.

2.3.4.5 Modelos de Unidades Subfonéticas.

En los sistemas actuales de reconocimiento continuo de grandes vocabularios se realiza compartición de estados entre los diferentes *HMM*, con el fin de encontrar un buen compromiso entre la resolución de los modelos y la robustez.

Los trabajos más significativos sobre modelado subfonético son, por orden de aparición, los tres siguientes: el *fenón* de *IBM* [Bah88], el *senón* de *CMU* (Carnegie Mellon University)[Hwa92][Hwa93a] [Hwa93b] y el *genón* de *SRI* (Speech Technology and Research Laboratory, International) [Dig96].

En todos estos casos, la metodología es muy similar, y consiste en buscar un punto donde sea posible realizar una *atadura* (*tying*), normalmente al nivel de estado de *HMM*, de manera que se reduzca el número de parámetros a entrenar. Se pueden distinguir tres fases:

- *Estimación inicial*: Se diseñan unos modelos con un bajo nivel de entrenamiento, normalmente con *HMM* discretos, para obtener una primera estimación de unos modelos que pueden ser fonéticos.
- *Clustering* (*agrupamiento*): Se agrupan los estados entre los modelos que sean similares, utilizando una determinada distancia (*medida de distorsión*).
- *Reestimación*: Con las reglas de agrupación obtenidas, se realiza otro entrenamiento para conseguir unos modelos nuevos mucho más refinados y precisos.

En el agrupamiento se pueden imponer restricciones, como en el caso de los senones:

- *Restricción de dependencia de fonema*: En estas unidades no se permite agrupar estados que correspondan a unidades fonéticas diferentes, evitando que puedan llegar a confundirse.
- *Restricción de dependencia de estado*: Se impone la condición de no agrupar estados que no se correspondan con la misma posición dentro del modelo.

Estas restricciones proporcionan una disminución en el número de posibilidades de agrupamiento, facilitando la obtención de los parámetros. En [Hua95] se hace una descripción del cómputo eficiente de distancias a los senones dentro del sistema *WHISPER*^{TM} de *Microsoft*.

En [Luo99] se propone la utilización de un conjunto de clases compartidas por todos los modelos de estado. Cada uno de ellos se define como una combinación probabilística de dichas clases.

2.4 Aplicaciones del Reconocimiento de Voz

El reconocimiento de voz es una tecnología de importancia creciente. A la vez que los desarrolladores de algoritmos están trabajando para conseguir reconocedores más rápidos y mejores, los fabricantes de *DSPs* están evaluando arquitecturas que permitan afrontar de una manera más eficiente los problemas de la voz, y los fabricantes de productos como televisores, vídeos, ordenadores, máquinas de dictado dedicadas o sistemas de control de automóviles están investigando y sacando al mercado nuevos equipos de reconocimiento.

2.4.1 Factores del Reconocimiento de Voz.

Las diferentes aplicaciones que utilizan reconocedores de voz presentan unos factores de funcionamiento que hay que considerar para que éstos funcionen de manera adecuada.

Como se menciona en [Dav96], un reconocedor de voz se puede identificar a partir de un vector multidimensional. Cada una de las dimensiones refleja una determinado factor, cuya enumeración es:

- Independencia de locutor.
- Reconocimiento continuo o discreto.
- Tamaño del vocabulario.
- Portabilidad de los sistemas hardware.
- Micrófono remoto o cercano.

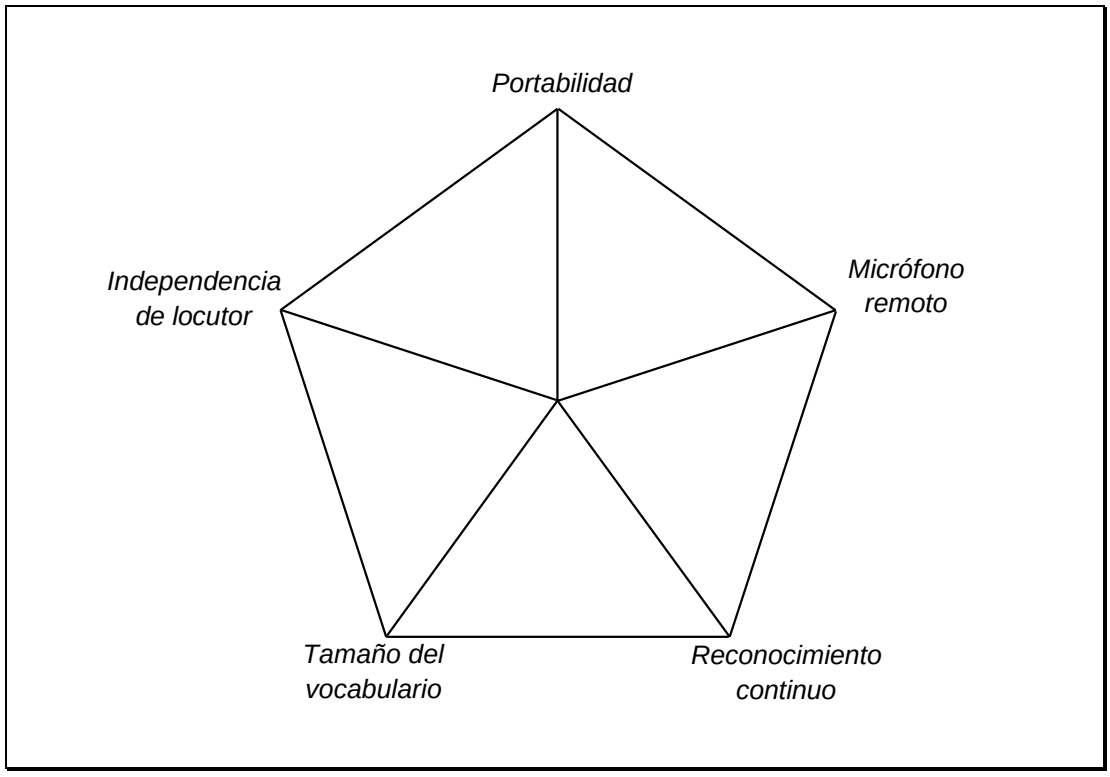

Figura 2-4. Los cinco factores que hay que tener en cuenta a la hora de diseñar un reconocedor de voz.

Como se ve en la Figura 2-4 todas estas propiedades se pueden distribuir en un gráfico en estrella, en el que cada uno de los factores se localiza en una de las puntas. En función de la distancia al centro se establece una medida de la cantidad del factor que es necesario considerar.

Existe otra característica adicional denominada *Cut Through*^{TM}, que es la habilidad de reconocer una pronunciación antes de que el sistema termine de hacer la pregunta o incluso antes de empezarla. Resulta muy útil cuando el usuario utiliza muchas veces el mismo sistema y ya conoce lo que se le pregunta, por lo que no es necesario esperar a que la máquina termine de reproducir para responder.

2.4.1.1 Independencia de Locutor.

La independencia de locutor es el factor determinante a la hora de clasificar los reconocedores de voz. Implica que cualquier persona puede utilizar el reconocedor de voz sin necesidad de realizar un entrenamiento previo para adaptarlo a las características de su voz. Para ello, es necesario utilizar bases de datos que incluyan un número suficientemente elevado de voces con la máxima variabilidad. La independencia de locutor es la única posibilidad cuando el sistema es usado por el público en general.

La dependencia de locutor requiere que el usuario entrene el sistema mediante la pronunciación de una serie de palabras. El tiempo necesario para esto varía desde unos minutos hasta varias horas, dependiendo de la lista de palabras a reconocer.

Estos sistemas son los que mejor funcionan, pues se adaptan a los diferentes acentos del locutor y a su velocidad de pronunciación.

Adicionalmente, algunos sistemas independientes del locutor se dice que son *adaptativos*, es decir, que no es necesario realizar un entrenamiento previo a la utilización del sistema, sino que éste se va adaptando según va siendo usado por el usuario. No es posible usar este tipo de reconocedores cuando el sistema es accedido por un número elevado de usuarios.

2.4.1.2 Reconocimiento Continuo o Discreto.

Este factor responde a la necesidad de que exista un silencio más o menos grande entre palabras para que el reconocedor de voz funcione correctamente. Se distinguen tres categorías:

- *De palabras aisladas*: El reconocedor selecciona entre un conjunto de palabras candidatas cuál es la más probable. Se requiere un silencio de al menos $250\ ms$.
- *De palabras concatenadas o conectadas*: El reconocedor es capaz de decodificar una secuencia de palabras entre un conjunto de posibilidades, al que se le pueden imponer restricciones (*gramáticas*). Se le impone la existencia de un silencio de al menos $50\ ms$.
- *Reconocimiento continuo*: Es similar a un reconocedor de palabras conectadas, pero no es necesario que existan pausas muy elevadas entre palabras, pudiendo funcionar hasta con silencios de $25\ ms$.

2.4.1.3 Tamaño del Vocabulario.

Otra clasificación que se suele hacer para dividir los reconocedores de voz es en función del número de palabras que son capaces de manejar:

- *Pequeños vocabularios*: Estos reconocedores manejan vocabularios desde decenas hasta centenares de palabras.
- *Vocabularios medios*: Éstos son reconocedores que manejan vocabularios desde cientos de palabras hasta unos cuantos miles.
- *Grandes vocabularios*: Estos reconocedores son capaces de manejar hasta varias decenas de miles de palabras.

Al igual que ocurre con la continuidad de las palabras, no existe una división categórica, sino que esta clasificación responde a los tipos de reconocimiento más usados.

2.4.1.4 Portabilidad de los Sistemas Hardware.

Algunas aplicaciones de reconocimiento relacionadas con cierta movilidad del usuario presentan limitaciones en cuanto al tamaño, potencia y memoria con que

cuentan. Sin embargo, en otros, como en los ordenadores personales actuales, la potencia de los procesadores y la existencia de suficiente memoria permite la utilización de algoritmos muy intensivos para conseguir mejores tasas de reconocimiento.

2.4.1.5 Micrófono Remoto o Cercano.

En aplicaciones de habla cercana, el locutor se encuentra próximo al micrófono o *headset* (auriculares con micrófono). Sin embargo, en las aplicaciones de habla remota existe una distancia suficientemente grande entre locutor y micrófono, y los problemas de eco y ruido provocan errores que hay que corregir. El caso de aplicaciones telefónicas también se considera como un tipo especial de habla remota, pues aunque el micrófono está cercano al locutor, existe una gran variabilidad de calidad del micrófono, y la comunicación entre el locutor y la máquina con el reconocedor de voz se realiza a través de la red telefónica, que presenta anchos de banda y niveles de ruido muy diferentes.

2.4.2 Principales Aplicaciones Actuales del Reconocimiento de Voz.

Las principales aplicaciones actuales del reconocimiento de voz se pueden dividir en tres grandes grupos, cada uno de los cuales presenta precios, funcionamiento y características diferentes:

- Aplicaciones telefónicas.
- Aplicaciones industriales.
- Aplicaciones de oficina.

2.4.2.1 Aplicaciones Telefónicas.

Entran dentro del ámbito de las tecnologías *ITO* (Integración de Telefonía y Ordenadores). Presentan unas propiedades muy particulares puesto que utilizan la red telefónica como canal de comunicaciones. Suelen ser aplicaciones de tipo conversacional, donde hay un diálogo en el que se guía al usuario a través de un conjunto de preguntas hasta conseguir el objetivo deseado. Suelen emplearse reconocedores de palabras aisladas o palabras conectadas con vocabularios no muy grandes.

El reconocimiento de voz es complicado debido a que al utilizar la red teléfonica el ancho de banda es reducido. Además, muchos de los micrófonos son de mala calidad y existen problemas de eco y ruido de fondo. Todo estas condiciones hacen que su funcionamiento sea mucho más difícil que el de las aplicaciones de oficina o industriales. El problema del reconocimiento de voz a través de teléfonos móviles es mayor.

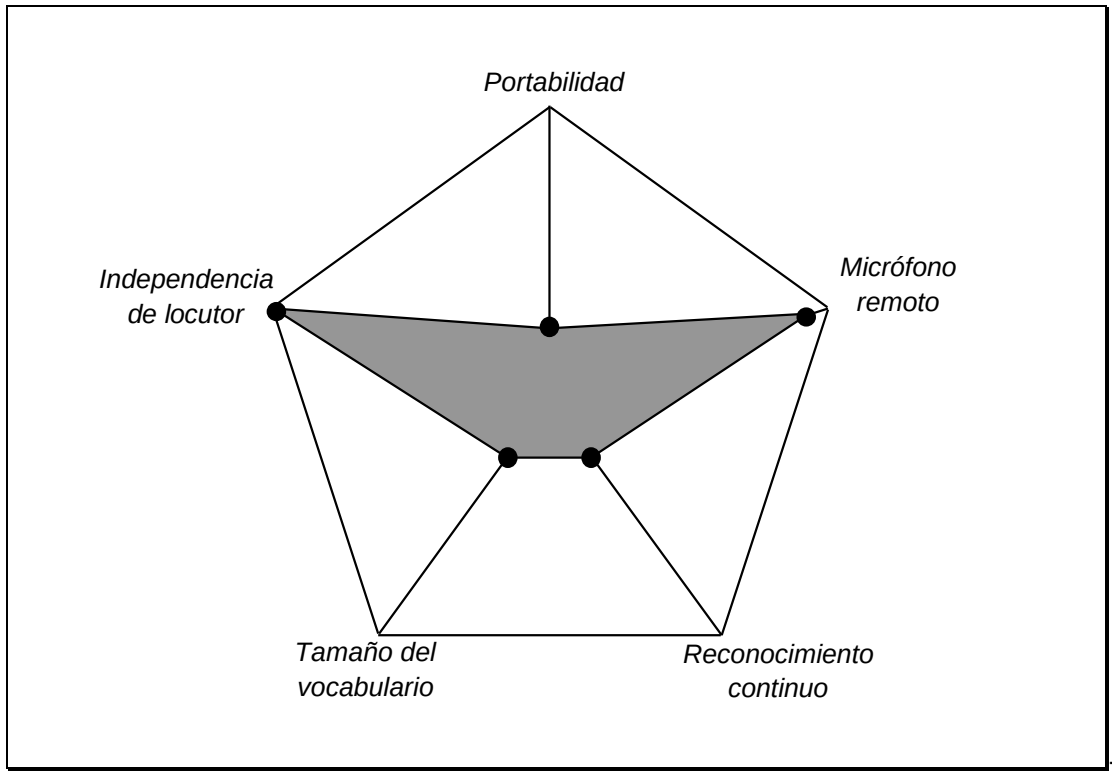

Figura 2-5. Valores de los factores en un reconocedor de voz pensado para aplicaciones telefónicas.

Como se puede observar en la Figura 2-5, los factores principales de un reconocedor de voz destinado a aplicaciones telefónicas son dos: por un lado, debe ser un sistema independiente de locutor; y por otro, hay que tener en cuenta que existe una distancia entre el locutor y la máquina de reconocimiento, con todos los problemas que se derivan de ello.

2.4.2.2 Aplicaciones Industriales.

Son las destinadas al manejo un aparato, una máquina o una herramienta, ya sea hardware o software, a partir de la invocación de órdenes a través de la voz.

A diferencia de las aplicaciones telefónicas, donde hay que establecer un canal de comunicaciones que varía en cada nueva utilización, en este caso, este canal está muy bien caracterizado. Aunque suele haber un mayor ancho de banda, suelen trabajar en situaciones en las que existen ambientes ruidosos.

Deben ser aplicaciones fáciles de usar, porque el operario suele estar ocupado realizando otras actividades. Por supuesto, deben ser aplicaciones portátiles, para lo cual se realiza la transmisión vía radio hacia un ordenador principal de procesado, requiriéndose terminales de voz portátiles con baterías de suficiente potencia. Además, pueden usarse reconocedores dependientes de locutor, puesto que requieren menor potencia de procesador, son más apropiados para aplicaciones de habla continua y son más fáciles de adaptar a un vocabulario específico en función de la tarea a la que van destinados.

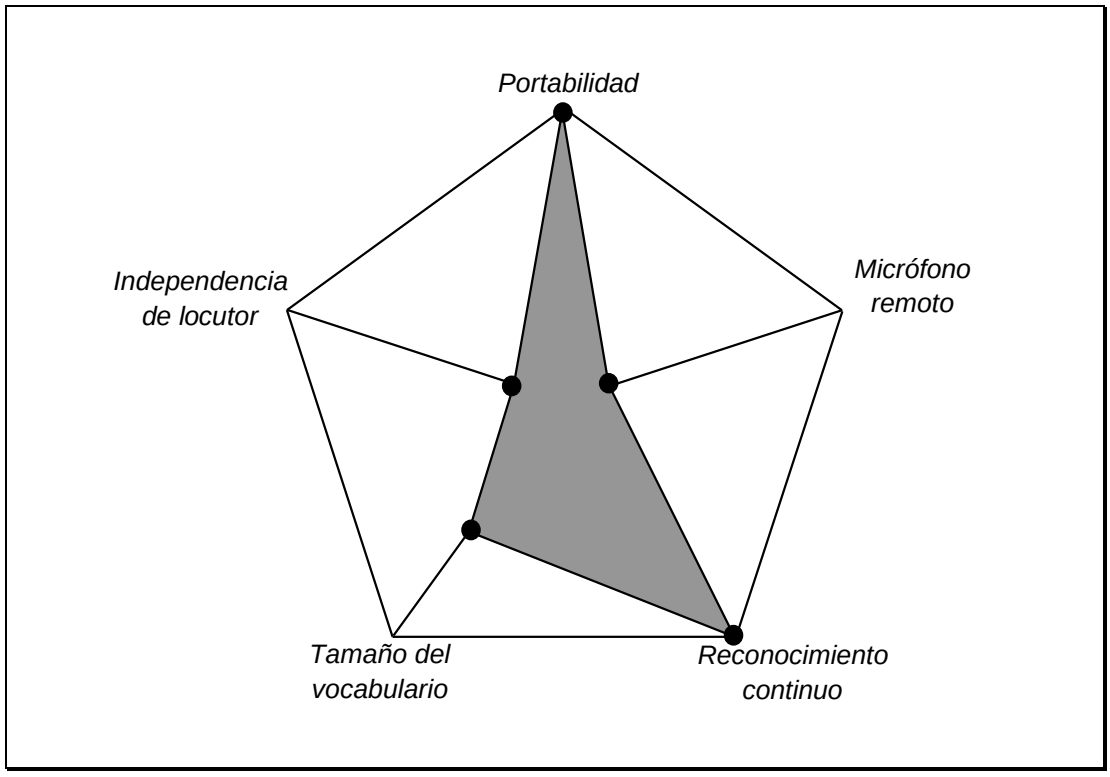

Figura 2-6. Factores de un reconocedor de voz destinado a aplicaciones industriales.

Como se observa en la Figura 2-6, los factores de portabilidad y de habla continua son los más importantes en una aplicación industrial que utilice reconocimiento de voz, seguidos de un vocabulario de tamaño intermedio adaptado a las necesidades de la tarea específica.

2.4.2.3 Aplicaciones de Oficina.

La mayoría de los sistemas de reconocimiento de voz están destinados a ofrecer una serie de mejoras adicionales a las características básicas de un ordenador personal de sobremesa. Por ello, las aplicaciones de oficina utilizan principalmente tecnología independiente de locutor (sin necesidad de entrenamiento previo), aunque suelen ser sistemas adaptativos (adaptación durante su utilización), y se suelen usar micrófonos de auriculares con casco o de mesa. Algunos sistemas requieren tarjetas especiales de procesado digital o tarjetas convencionales de procesado de sonido. Ultimamente, con el aumento de la potencia de los ordenadores, se tiende a emplear técnicas basadas sólo en sistemas software, ejecutándose los algoritmos únicamente en el procesador central de éste.

Los resultados de los reconocedores son diferentes en cuanto a velocidad de habla y tamaño del vocabulario. Normalmente, todas estas aplicaciones de oficina se venden como sistemas de dictado automático. Los reconocedores más extendidos y con mejores resultados son los de *IBM* y *Dragon Systems*.

Existen otras aplicaciones de voz destinadas a personas con discapacidades físicas para manejar el teclado o el ratón de un ordenador, o para el manejo de sillas de ruedas o de diferentes dispositivos.

Capítulo

3

Arquitectura de Reconocimiento de Voz Fonético

3.1 Introducción

En este capítulo se detallan las diferentes partes de un reconocedor de voz fonético basado en Modelos Ocultos de Markov (*HMM)* orientado hacia aplicaciones telefónicas y funcionamiento en tiempo real.

Para que un reconocedor de voz trabaje con tasas de error bajas y con suficiente robustez, para que no se degrade cuando las condiciones sean adversas, se necesita un óptimo diseño de cada uno de sus elementos. Así, en las páginas sucesivas, se analizarán los siete aspectos que tuvieron que ser considerados en este reconocedor de voz flexible para aplicaciones telefónicas:

- *Extracción de características*: Resulta fundamental en todo sistema basado en reconocimiento de patrones. Debe de estar adaptada al tipo de información que se quiera obtener, eliminando la característica de la fuente y del canal de comunicaciones.

- *Modelado de las unidades fonéticas*: Un reconocedor de vocabulario flexible necesita disponer de un conjunto de unidades que permita construir cualquier palabra o frase a partir de su concatenación. Como ya se explicó en el apartado 2.3.4.3, los fonemas representan un conjunto completo y reducido de unidades a partir de las cuales se puede generar cualquier palabra. También se mencionaba que estas unidades pueden modelarse con mayor o menor resolución en función del contexto a considerar. Es necesario especificar para un determinado idioma, en este caso para el castellano, cómo son estas unidades. Además, las diferentes alternativas de entrenamiento en función del acceso a las bases de datos será un factor determinante a la hora de establecer el conjunto de unidades a entrenar.
- *Modelado de palabras*: El siguiente paso, una vez establecida la extracción de características y el diseño de unas unidades fonéticas, es la generación de modelos de palabras. Se necesita un transcriptor fonético para conseguir la secuencia de unidades fonéticas de que consta un modelo de palabras. Otro aspecto a considerar es la introducción directa en el reconocedor de voz de modelos de palabras enteras, como son los dígitos, *sí* y *no*.
- *Cálculo eficiente de probabilidades*: La eficiencia es un aspecto que no puede dejar de considerarse, porque puede significar que un reconocedor sea viable o no. El cálculo de las probabilidades de los elementos de la matriz B es el más crítico. Si el número de modelos es elevado, la carga computacional puede llegar a ser excesiva. En este apartado, se analizan las simplificaciones aplicadas al cálculo de las verosimilitudes que permiten reducirlo de forma significativa, sin que por ello se produzca una degradación de las prestaciones del reconocedor.
- *Algoritmos de reconocimiento*: Representan el núcleo central de cualquier reconocedor de voz que pretenda ser flexible y eficiente. Si están bien definidos y bien utilizados, se obtendrán mejoras muy significativas tanto en la fase de entrenamiento como en la de reconocimiento. Se requieren algoritmos que permitan realizar una decodificación a dos niveles: el *nivel acústico*, decodificación de las características de la señal de voz dentro de una palabra, y el *nivel de lenguaje*, para poner fácilmente las restricciones gramaticales. El algoritmo utilizado permite realizar ambas fases de una manera combinada y trama a trama, con lo que se puede utilizar en un sistema en tiempo real. A este algoritmo se le ha introducido una serie de mejoras para conseguir una mayor flexibilidad.
- *Estructura de reconocimiento en aplicaciones telefónicas*: En este apartado se particulariza el reconocedor de voz para el caso de aplicaciones que funcionen como telefonista automática. Puesto que estas aplicaciones pueden ser usadas por cualquier usuario, y cada uno puede reaccionar de una manera distinta, es necesario introducir en la red de reconocimiento un

conjunto de elementos para que el reconocedor mantenga sus prestaciones. A las palabras que se añaden para conseguir este fin se las denomina *comodín*. Las palabras comodín son todas aquellas palabras o expresiones adicionales que se pueden pronunciar junto con la respuesta esperada (*por favor, gracias, don, doña, póngame con,* etc.) y que no aportan ninguna información.

- *Integración en un sistema ITO en tiempo real*: Finalmente, de nada serviría tener un reconocedor funcionando con cualquier vocabulario y adaptado al canal telefónico si éste no puede ser probado en situaciones reales. Por ello, se describe cómo el reconocedor de voz puede integrarse en un sistema *ITO*, formado por un *PC* y una tarjeta de procesado digital de señal conectada a una centralita automática. En todo sistema de reconocimiento funcionando en tiempo real surgen nuevos aspectos que hay que tener en cuenta, destacándose la detección de extremos, para determinar los bordes de principio y fin de palabra, que permitan realizar la búsqueda hacia atrás en el algoritmo de reconocimiento y detectar cuándo no se ha pronunciado nada o se ha empezado antes de tiempo.

En lo que queda de capítulo se seguirá el siguiente orden de exposición: en el apartado 3.2 se describe detalladamente la extracción de características implementada para el funcionamiento del reconocedor de voz a través de canal telefónico; en el apartado 3.3 se estudia el modelado de Markov de unidades fonéticas para el castellano, indicando los mecanismos de entrenamiento y generación de modelos empleados; en el apartado 3.4 se describe la obtención de modelos de palabras; en el apartado 3.5 se realiza un análisis del cálculo de verosimilitudes con el fin de simplificarlo y aumentar la eficiencia del sistema; en el apartado 3.6 se describen los algoritmos de reconocimiento utilizados tanto en entrenamiento como en reconocimiento con las diferentes alternativas propuestas; en el apartado 3.7 se introduce la información gramatical asociada con un reconocedor de voz que usado en aplicaciones telefónicas; y finalmente, en el apartado 3.8 se indica cómo se ha realizado la integración del reconocedor de voz en un sistema *ITO* que funciona en condiciones reales y cómo se ha implementado el detector de extremos.

3.2 Extracción de Características

3.2.1 Caracterización del Canal Telefónico.

Antes de plantearse qué tipo de extracción de características hay que utilizar en un sistema de reconocimiento de voz, es preciso analizar el canal de comunicaciones y su influencia sobre la señal de voz.

El primer aspecto a tener en cuenta es la variabilidad de canales de comunicación en las aplicaciones telefónicas. Mientras que en una aplicación de oficina las condiciones se mantienen prácticamente iguales en sus diferentes usos, en una aplicación telefónica, tanto el micrófono como el medio de transmisión varían en cada nueva comunicación establecida. La extracción de características debe tenerlo en cuenta para minimizar sus efectos.

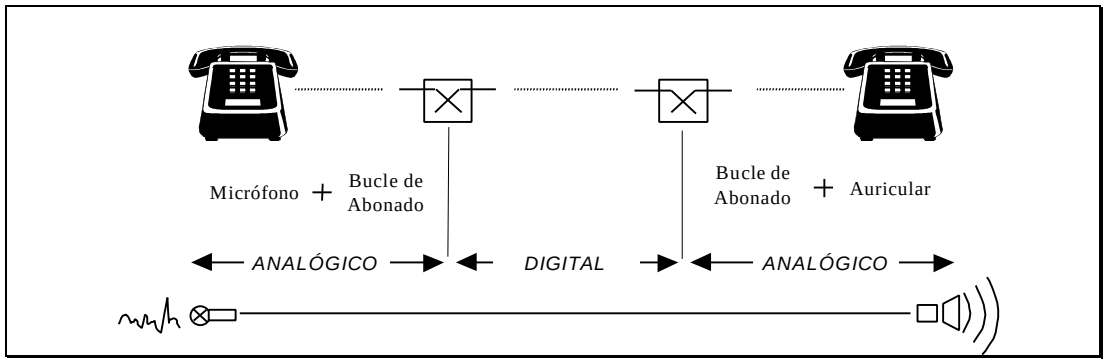

Figura 3-1. Esquema típico de una comunicación establecida entre el micrófono de un teléfono y el auricular de otro a través de la *red telefónica Conmutada* (*RTC*).

En la Figura 3-1 se observa una comunicación entre dos teléfonos a través de la *red telefónica conmutada* (*RTC*), que es la arquitectura más común dado el grado de digitalización actual de las redes telefónicas. Se distinguen dos partes, una *analógica* y otra *digital*. La parte *digital* puede existir o no; por ejemplo, en una llamada local no es necesario que exista una conversión a digital, pudiendo transmitirse enteramente en analógico. La parte *analógica* es la que capta, recibe y transmite la señal de voz, y consta de los siguientes elementos:

- *Micrófono*: Transductor que convierte las variaciones de presión de las ondas acústicas de la voz en ondas eléctricas.
- *Bucle de abonado*: Elemento que transporta las ondas eléctricas desde un teléfono hasta su central local o viceversa.
- *Auricular*: Elemento que convierte las ondas eléctricas en ondas de presión.

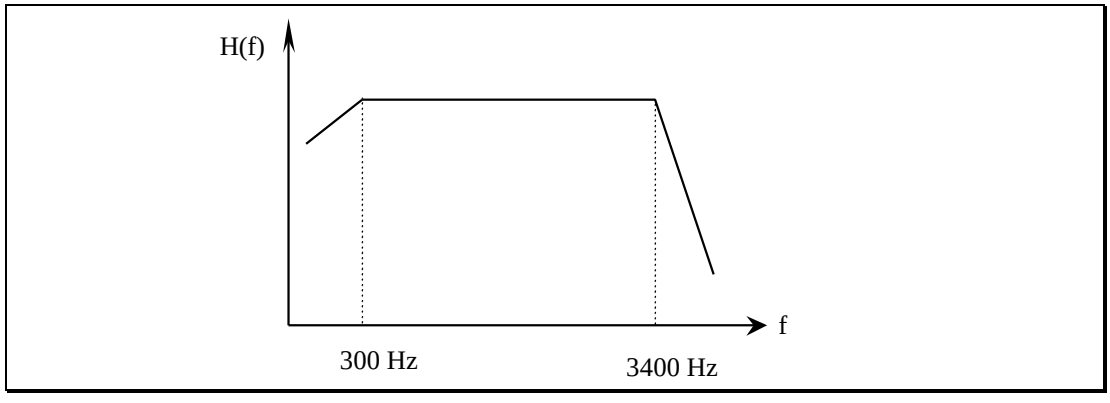

Figura 3-2. Función de transferencia típica de un canal telefónico. Se observa como el ancho de banda se extiende desde los 300 *Hz* a los 3400 *Hz* aproximadamente.

Todos los elementos que intervienen en la caracterización del canal telefónico desde la captación de la señal de voz hasta su recepción, más el conjunto de transformaciones, filtrados y procesados que se producen, generan una función de transferencia del sistema que va desde el micrófono de un teléfono hasta el auricular del otro, como la de la Figura 3-2.

La función de transferencia presenta un ancho de banda aproximado de $3.100\ Hz$, desde $300\ Hz$ hasta $3.400\ Hz$, donde la respuesta es aproximadamente plana, disminuyendo rápidamente hacia los bordes para evitar el efecto de solapamiento (*aliasing*). La utilización de unos márgenes de guarda tan grandes tiene su origen en la multiplexación de canales realizada de manera analógica. Estos márgenes permitían utilizar unos filtrados muy sencillos, comprobándose que esta banda era suficiente para que cualquier comunicación telefónica funcionara con un nivel de inteligibilidad muy elevada (mayor al $90\ \%$).

Esta función de transferencia no es siempre igual, varía en cada comunicación. Además, existen otros aspectos que se introducen y que no están reflejados en dicha función. Se distinguen:

- *Ruido*: Cualquier señal de tipo impredecible que se añada a la voz. Suele provenir de los elementos electrónicos y de las conversiones analógico-digital-analógico. Un nivel elevado de ruido puede llegar a enmascarar la señal de voz y hacerla irreconocible.
- *Distorsión lineal*: Los diferentes elementos que intervienen en la comunicación, incluyendo las condiciones acústicas de la sala, la respuesta del micrófono, los elementos de transmisión o el receptor, aportan múltiples funciones de transferencia.
- *Interferencias*: Cualquier señal procedente de otra línea que aparezca con un nivel de potencia elevado.

3.2.2 Esquema General del Extractor de Características.

Una vez analizadas las características del canal telefónico, se realiza una extracción de las mismas teniendo en cuenta la energía que se localiza en la banda que comprende entre los 300 y los $3.400\ Hz$, que es la menos variable entre diferentes canales. Esta extracción de características se basa en el cálculo de los coeficientes cepstrum [Dav80][Del93] y delta-cepstrum [Fur86].

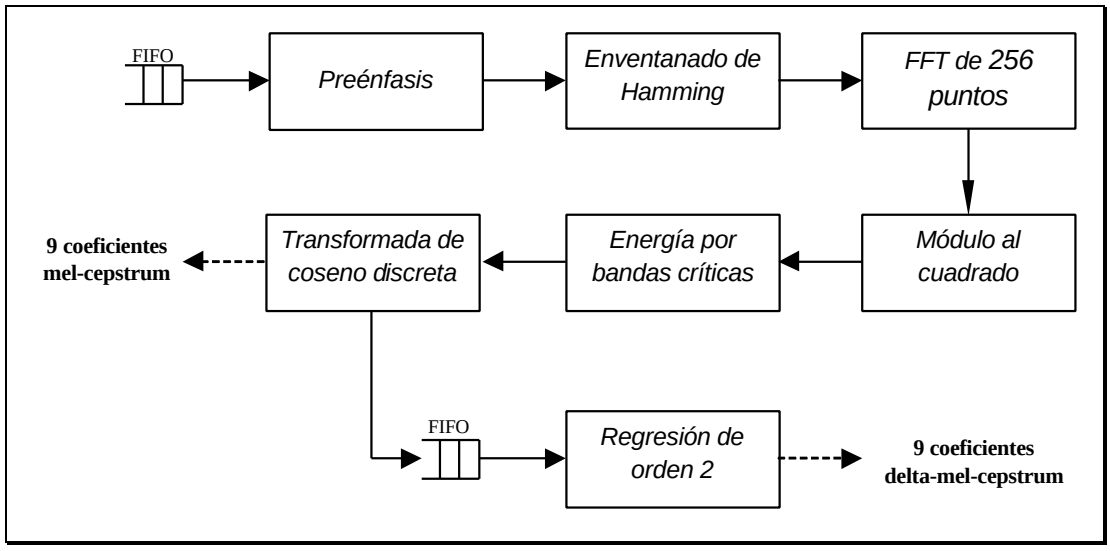

Figura 3-3. Diagrama de bloques de la extracción de características.

En la Figura 3-3 se muestra un esquema de los diferentes bloques de que consta la extracción de características.

El primer paso consiste en captar la señal de voz y muestrearla a una frecuencia de $8\ kHz$, con 16 bits por muestra. Estas muestras se van almacenando en una *FIFO* como enteros de punto flotante en bloques de 256 muestras ($32\ ms$), y se van extrayendo con un solapamiento de 128 muestras ($16\ ms$).

Antes de hacer ningún procesado, se realiza un filtrado paso alto con el fin de eliminar la componente continua, si la hubiera, y de resaltar las altas frecuencias. La función de transferencia tiene la siguiente expresión:

$$H(z) = 1 - a \cdot z^{-1} \qquad [3.1]$$

donde a tiene un valor de $0{,}95$, que es suficiente para enfatizar la banda de alta frecuencia y conseguir unos modelos mejores para los fonemas de tipo fricativo.

3.2.3 Enventanado de la Señal y Transformación Tiempo-Frecuencia.

La señal filtrada paso alto se multiplica por una ventana de Hamming de la misma longitud que la de análisis, en este caso 256 puntos. Los valores de la ventana se calculan como:

$$Hamming[k] = 0{,}54 - 0{,}46 \cdot \cos\left(\frac{2pk}{L-1}\right) \qquad [3.2]$$

donde $0 \le k < L$ y siendo L 256 muestras.

La forma de la ventana de Hamming puede verse en la Figura 3-4.

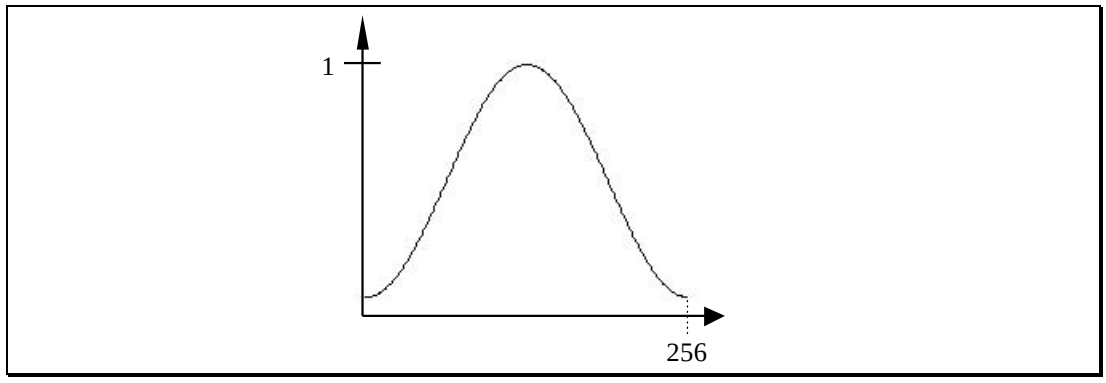

Figura 3-4. Forma de la ventana de Hamming que multiplica a la señal de voz.

A la 256 muestras que forman la trama enventanada se les aplica una transformada de Fourier rápida (*FFT*: *Fast Fourier Transform*) de 256 puntos. Puesto que la señal es real, basta con aplicar una *FFT* para número reales, que es el doble de rápida que una *FFT* para números complejos, y aprovecha el hecho de que en frecuencia las muestras son hermíticas, con lo que sólo es necesario calcular la mitad de puntos. Por ello, se utiliza una *FFT* basada en la transformación de Hartley. El resultado es un vector de 128 puntos, con su parte real e imaginaria, denominado $X[k]$, que ofrece una resolución en frecuencia de 31,25 *Hz*.

El siguiente punto consiste en obtener la energía en todas las frecuencias. Así, se calcula el módulo para cada frecuencia y se eleva al cuadrado.

3.2.4 Energía por Bandas Críticas en Escala Mel.

Se ha comprobado que la respuesta del oído no es igual para todas las bandas de frecuencia. Para reducir el número de valores, se agrupa la energía de la señal en bandas de energía, siendo su tamaño variable con la frecuencia. Para ello, se multiplican las muestras de energía al cuadrado por un banco de filtros triangulares utilizando la escala *mel*. La conversión de escala en *Hz* a escala *mel* tiene la siguiente expresión:

$$f_{mel} = 1.000 \cdot \log_2\left[1 + \frac{f_{Hz}}{1.000}\right] \qquad [3.3]$$

Tabla 3-I. Valores de los índices de la *FFT* de las frecuencias mínima y máxima que se corresponden con los filtros triangulares en escala mel.

Banda	Posiciones en la *FFT*	Frecuencia mínima	Frecuencia Máxima
0	5 – 11	156,25	343,75
1	9 – 14	281,25	437,50
2	12 – 18	375,00	565,50
3	15 – 22	468,75	687,50

4	**19 – 26**	**593,75**	**812,50**
5	**23 – 32**	**718,75**	**1.000,00**
6	**27 – 37**	**843,75**	**1.156,25**
7	**33 – 43**	**1.031,25**	**1.343,75**
8	**38 – 51**	**1.187,50**	**1.593,75**
9	**44 – 59**	**1.375,00**	**1.843,75**
10	**52 – 68**	**1.625,00**	**2.125,00**
11	**60 – 80**	**1.875,00**	**2.500,00**
12	**69 – 92**	**2.156,25**	**2.875,00**
13	**82 – 108**	**2.531,25**	**3.375,00**

En total se definen 14 bandas, las reflejadas en la Tabla 3-I. La primera banda está centrada aproximadamente en $250\ Hz$, que es un poco inferior a los 300 Hz teóricos del canal telefónico, y la última en $3.125\ Hz$. Se observa como la mayor parte de la energía que se extrae se localiza en la zona de los 250 a $3.000\ Hz$, captando cierta energía de baja frecuencia y muy poca de alta.

Una vez obtenidos los valores de la *FFT*, calculado el módulo y elevado al cuadrado, esto es, $|X\ [k]|^2$, se necesita definir los pesos por los que hay que multiplicar cada valor de la *FFT* para definir el filtro triangular elevado al cuadrado.

Tabla 3-II. Valores de los pesos asociados con las bandas de filtros triangulares.

- **Banda 0:** [0,00391; 0,14062; 0,47266; 1,00000; 0,47266; 0,14062; 0,00391]
- **Banda 1:** [0,09766; 0,39062; 0,87891; 0,56250; 0,19141; 0,01562;
- **Banda 2:** [0,06250; 0,31641; 0,76562; 0,71191; 0,34028; 0,10428; 0,00391]
- **Banda 3:** [0,02441; 0,17361; 0,45844; 0,87891; 0,66799; 0,33284; 0,11326; 0,00925]
- **Banda 4:** [0,03338; 0,17899; 0,44018; 0,81694; 0,75008; 0,41327; 0,17610; 0,03858]
- **Banda 5:** [0,01794; 0,12755; 0,33681; 0,64573; 0,95367; 0,61035; 0,34332; 0,15259; 0,03815]
- **Banda 6:** [0,00055; 0,04785; 0,17145; 0,37134; 0,64752; 1,00000; 0,66614; 0,39987; 0,20118; 0,07007; 0,00654]
- **Banda 7:** [0,03379; 0,13516; 0,30412; 0,54066; 0,84478; 0,83266; 0,57191; 0,36000; 0,19691; 0,08266; 0,01723]
- **Banda 8:** [0,00766; 0,05941; 0,16000; 0,30941; 0,50766; 0,75473; 0,95699; 0,70962; 0,49917; 0,32564; 0,18904; 0,08935; 0,02658; 0,00074]
- **Banda 9:** [0,00047; 0,02484; 0,08613; 0,18434; 0,31947; 0,49152; 0,70050; 0,94639; 0,81000; 0,60062; 0,42250; 0,27562; 0,16000; 0,07563; 0,02250; 0,00063]
- **Banda 10:**[0,01000; 0,05062; 0,12250; 0,22562; 0,36000; 0,52563; 0,72250; 0,95063; 0,84028; 0,66016; 0,50174; 0,36502; 0,25000; 0,15668; 0,08507; 0,03516; 0,00694]
- **Banda 11:**[0,00694; 0,03516; 0,08507; 0,15668; 0,25000; 0,36502; 0,50174; 0,66016; 0,84028; 0,96460; 0,79719; 0,64573; 0,51020; 0,39062; 0,28699; 0,19930; 0,12755; 0,07175; 0,03189; 0,00797]
- **Banda 12:**[0,00032; 0,01148; 0,03858; 0,08163; 0,14062; 0,21556; 0,30644; 0,41327; 0,53603; 0,67474; 0,82940; 1,00000; 0,84985; 0,71191; 0,58618; 0,47266; 0,37134; 0,28223;

```
                0,20532; 0,14062; 0,08813; 0,04785; 0,01978; 0,00391]
```

- **Banda 13:**
```
[0,00610; 0,02441; 0,05493; 0,09766; 0,15259; 0,21973;
 0,29907; 0,39062; 0,49438; 0,61035; 0,73853; 0,87891;
 0,97516; 0,85563; 0,74391; 0,64000; 0,54391; 0,45563;
 0,37516; 0,30250; 0,23766; 0,18063; 0,13141; 0,09000;
 0,05641; 0,03062; 0,01266; 0,00250]
```

En la Tabla 3-II se indican los pesos de los filtros triangulares obtenidos a partir de las bandas definidas en la Tabla 3-I. Teniendo en cuenta la discretización conseguida al calcular 128 valores en 4.000 *Hz* de banda, resulta un valor de 31,25 *Hz*.

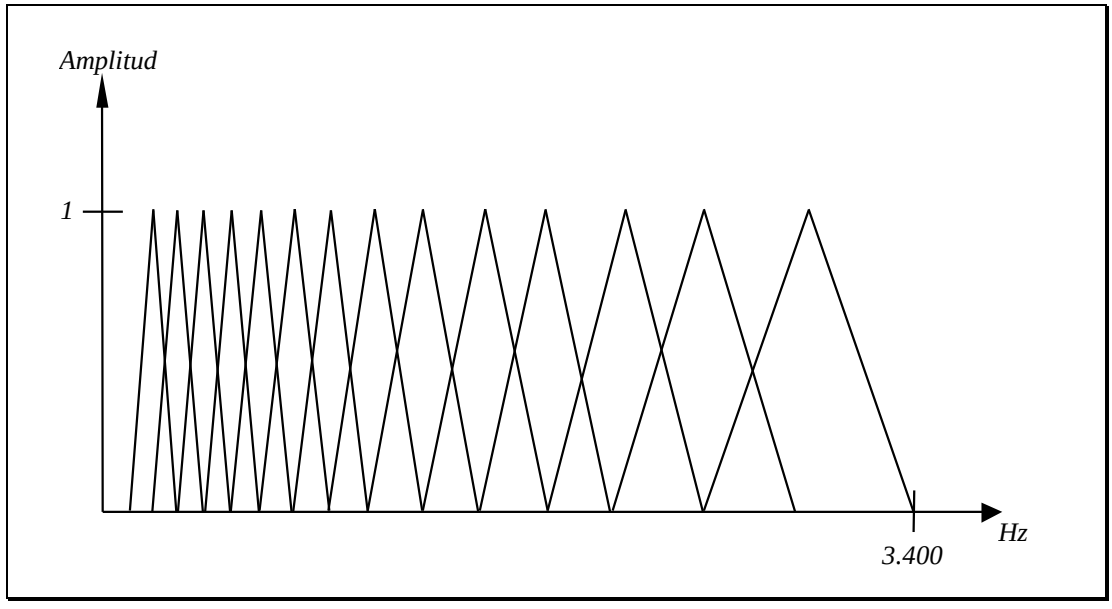

Figura 3-5. Disposición del banco de filtros triangulares en escala mel.

En la Figura 3-5 se ve como los valores anteriormente enumerados se corresponden con unos filtros triangulares. Estos filtros realmente no son continuos, sino que están muestreados cada 31,25 *Hz*.

Definiendo I_{bi} como el conjunto de índices de la Tabla 3-I y P_{bi} al conjunto de pesos de la enumeración anterior, se puede calcular la energía por bandas en escala logarítmica como:

$$E_{bi} = \log\left(\sum_{k \in I_{Bi}} X^2[k] \cdot P_{bi}[k - I_{bi,0}] \right), \quad 0 \le bi \le 13 \qquad [3.4]$$

3.2.5 Cálculo de los Coeficientes Mel-Frequency-Cepstrum.

Una vez calculadas las energías por banda, se obtienen los coeficientes *mel-frecuency-cepstrum*. Se pretende disminuir el número de componentes independientes a partir de la información en las 14 bandas, para lo cual, al conjunto de energías por bandas se aplica la *transformada coseno discreta* (*DCT*: *Discrete Cosine Transform*), de acuerdo con la expresión:

$$C(k) = \sum_{bi=0}^{13} E_{bi} \cdot \cos\left(\frac{p}{14} k(bi + 0{,}5)\right) \qquad 1 \le k \le NC \qquad [3.5]$$

donde, en este caso, NC vale 8. Así, se obtienen 8 componentes del vector, al que hay que añadir el *cepstrum 0 ó* logaritmo de la energía total, que según la ecuación de los cepstrum sería:

$$C(0) = \sum_{bi=0}^{13} E_{bi} = \sum_{bi=0}^{13} \left(\log_{10} \left(\sum_{k \in I_{Bi}} X^2[k] \cdot P_{bi}[k - I_{bi,0}] \right) \right) \qquad [3.6]$$

El problema de esta expresión es que puede ocurrir que en alguna banda el nivel de la señal sea muy bajo y al aplicar el logaritmo salga un valor muy negativo, que al sumarse a los demás sería el predominante. Para evitarlo, se procede a un cambio entre el logaritmo en base 10 y el sumatorio. Esta inversión no es equivalente pero ofrece mejores resultados que la anterior. Para aplicarlo es necesario un coeficiente corrector. La expresión queda:

$$C(0) = \log_{10} \left(Q \cdot \sum_{bi=0}^{13} \left(\sum_{k \in I_{Bi}} X^2[k] \cdot P_{bi}[k - I_{bi,0}] \right) \right) \qquad [3.7]$$

donde I_{bi} son los índices y P_{bi} los pesos de los filtros triangulares, y bi son las bandas, 14 en este caso. El factor Q vale $1{,}5258.10^{-5}$ para que los valores que se obtengan sean del mismo orden en [3.6] y [3.7]. Con este valor se tienen 9 coeficientes, que se irán almacenando en una *FIFO* de cinco vectores, consumiendo un elemento por trama.

3.2.6 Cálculo de los Coeficientes Delta-Mel-Frequency-Cepstrum.

Utilizando los cinco vectores de coeficientes cepstrum de la *FIFO* se calculan los coeficientes *delta-mel-frequency-cepstrum* de la siguiente forma:

$$\Delta C(k, t-2) = -2 \cdot C(t-4) - C(t-3) + C(t-1) + 2 \cdot C(t) \qquad [3.8]$$

donde k va de 0 a 8, puesto que también se calcula la regresión de la energía.

Como se observa en la expresión [3.8], es necesario calcular los coeficientes *cepstrum* de las siguientes dos ventanas de análisis antes de poder calcular los *delta-cepstrum*. Este retardo de dos tramas se corresponde con 256 muestras ó 32 *ms*, que hay que tener en cuenta cuando se utiliza la segmentación manual de una frase durante el entrenamiento de los modelos de Markov, para evitar desfases no deseados que pueden llevar a la obtención de modelos mal entrenados.

Una vez calculados ambos conjuntos de parámetros, se unen todos para formar los coeficientes de la trama actual $S(t)$, que consta de 18 componentes y que se forman de la siguiente manera:

$$S(t) = [\Delta C(0,t), \cdots, \Delta C(NC,t), C(0,t), \cdots, C(NC,t)] \qquad [3.9]$$

El orden de las componentes es el mostrado en [3.9], pero podría ser otro. $S(t)$ es el vector que contiene las características de la voz y puede utilizarse directamente tanto en entrenamiento como en reconocimiento.

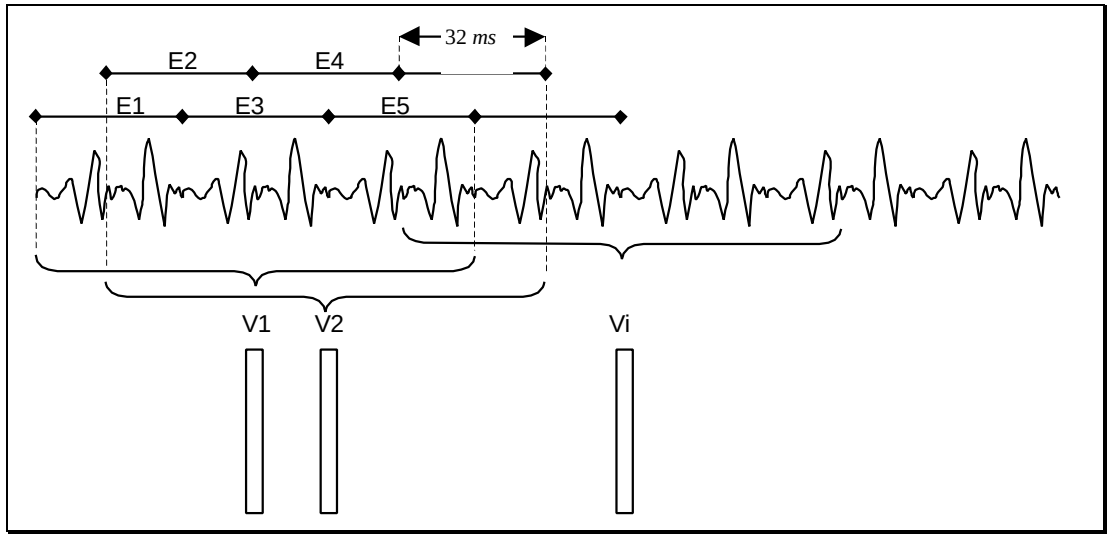

Figura 3-6. Esquema de la generación de los vectores de características.

En la Figura 3-6 se muestra la manera en la que se van generando los vectores de características. Se observa que pasados $32 * 3$ *ms* se puede generar el primer vector de características y cada 16 *ms* va apareciendo el siguiente.

3.2.7 Ponderación del Vector de Características.

A la hora de calcular las verosimilitudes de un vector de características a partir de la función densidad de probabilidad, tanto para modelos continuos como semicontinuos, siempre aparece la distancia euclídea entre el vector de características y el vector de medias ponderado por las varianzas. En cualquier caso, la distancia se puede expresar como:

$$d(S(t-2), M) = \sum_{i=1}^{LV} \left(\frac{S(i,t-2) - m(i)}{s_i} \right)^2 \qquad [3.10]$$

donde LV es el número de componentes del vector de características.

Como se observa en [3.10], las varianzas que se utilizan no dependen de los modelos usados, sino que son valores fijos. Esto es así, porque se ha comprobado que no produce una degradación significativa de la tasa de error y sí se consigue una reducción del número total de parámetros a entrenar. Esta mayor eficiencia es tanto en memoria como en computación.

Además del hecho de que la estimación de las varianzas se realiza una única vez al principio del entrenamiento, dado que siempre es la misma, se puede introducir

como un factor de ponderación del vector de características. Este vector de pesos, denominado por el factor de pesos V, se introduce como un paso adicional en el cálculo de los vectores de características. El vector de características resultante, si $S(t - 2)$ es el obtenido en el apartado anterior, es:

$$Z(i,t-2) = S(i,t-2) \cdot V(i) \qquad [3.11]$$

con $i = 1, ..., LV$.

3.2.7.1 Cálculo de la Ponderación de los Vectores de Características.

El vector V de pesos se obtiene, antes de empezar a entrenar, a partir de todos los vectores de características que aparezcan en la base de datos de entrenamiento. Su cálculo se resume en la siguiente expresión:

$$V(k) = \frac{1}{s_k} = \frac{1}{\sqrt{\left(\frac{1}{N}\sum_{i=1}^{N}(S_i(k) - m_k)^2\right)}} \qquad [3.12]$$

donde las componentes de la media se calculan como:

$$m_k = \frac{1}{N}\sum_{i=1}^{N} S_i(k) \qquad [3.13]$$

con N el número de vectores de características de las bases de datos de entrenamiento.

Por tanto, la nueva distancia, una vez ponderados todos los vectores de características, queda como una distancia euclídea sin normalizar entre los vectores de la extracción de características ponderados y los vectores de medias generados a partir de vectores también ponderados:

$$d(S(t-2), M') = \sum_{k=0}^{17}(Z(k,t-2) - M'(k))^2 \qquad [3.14]$$

$M'(k)$ son los vectores de medias de los modelos usando los vectores de características con varianza normalizada a uno. El modo de calcular los vectores de medias se explica en el apartado dedicado al entrenamiento de modelos de Markov.

3.3 Modelado de Unidades Fonéticas

3.3.1 Unidades Fonéticas para el Castellano Usando el Código SAMPA.

Tal como se menciona en el apartado 2.3.4.4, las unidades fonéticas permiten modelar palabras con gran flexibilidad. Un modelo de palabra se forma a partir de la

concatenación de dichas unidades fonéticas, obteniéndose un reconocedor de voz que permite construir cualquier vocabulario que se desee.

Para un idioma en particular es necesario establecer el conjunto de unidades que hay que modelar para generar cualquier modelo de palabra. Existen diferentes códigos para nombrar dichas unidades. Los más extendidos son: el código *IPA* y el código *SAMPA*. Ambos son internacionales e identifican los fonemas de un idioma y sus similitudes con los de otros. El código elegido es el *SAMPA*, que tiene la ventaja sobre el código *IPA* de que no necesita caracteres especiales para identificar las unidades fonéticas. Para establecer de manera clara el conjunto de fonemas, se empieza por una enumeración de los mismos, con una breve descripción y unos ejemplos ilustrativos.

Tabla 3-III. Definición y contexto de los fonemas del español en código *SAMPA*.

Fonema	*DEFINICIÓN Y CONTEXTO*
p	Oclusiva bilabial sorda.
b	Oclusiva bilabial sonora; en posición inicial absoluta o contacto con nasal precedente.
t	Oclusiva dental sorda.
d	Oclusiva dental sonora; en posición inicial absoluta o en contacto con [n] precedente.
k	Oclusiva velar sorda.
g	Oclusiva velar sonora; en posición inicial absoluta o en contacto con nasal sucesora.
m	Nasal bilabial.
n	Nasal alveolar.
J	Nasal palatal.
N	Nasal velar; en contacto con velar precedente.
tS	Africada palatal sorda.
B	Aproximante bilabial sonora; en distribución complementaria con [b].
f	Fricativa labiodental sorda.
T	Fricativa interdental sorda.
D	Aproximante bilabial sonora; en distribución complementaria con [d].
s	Fricativa alveolar sorda.
z	Fricativa alveolar sonora; en contacto con consonante sonora sucesora.
Z	Fricativa palatal sonora; en inicial de sílaba no precedida de [n] o [l] y en inicial absoluta.
x	Fricativa velar sorda.
G	Aproximante velar sonora; en distribución complementaria con [g].
l	Lateral alveolar.
L	Lateral palatal.
rr	Vibrante alveolar (simple).
r	Vibrante alveolar (múltiple).
i	Anterior cerrada.
j	Semivocal-semiconsonante anterior; aparece en los diptongos.
e	Anterior media.
a	Central abierta.
o	Posterior media.
u	Posterior cerrada.
W	Semivocal-semiconsonante posterior; aparece en diptongos.

Tabla 3-IV. Ejemplos de los fonemas del español en código *SAMPA*.

Fonema	*Ejemplos*	*Fonema*	*Ejemplos*
p	profesor, grupo, Pamplona	**z**	desde, nos dijo, más grande
b	vimos, con blanca, Lumbreras	**Z**	yerno, ya, yo, hierba
t	tampoco, mitad, pantano	**x**	ejes, Julio, Gema, lógico
d	después, abundante, Andrés	**G**	García, algún, agua, grasienta
k	causa, aquel, escrito, basket	**l**	hablo, las, Pamplona, militar
g	lenguaje, ganaban	**L**	apellido, llevar, allí, aquellos
m	nombre, militar, un viento	**rr**	retando, tertulia, organización
n	nuestra, dentro, dinero, nunca	**r**	dinero, era, creo, pared
J	niño, España, Cataluña, araña	**i**	mirábamos, admiten, teníamos
N	inglés, lengua, ningún, nunca	**j**	muy, grasienta, diez, tertulia
tS	chorizo, ocho, noche, hecho	**e**	entra, paredes, que, esos
B	Vizcaya, cubierto, vida	**a**	final, sangre, ensangrentada
f	funeraria, profesor, fiesta,	**o**	solo, hizo, gloriosos, total
T	riqueza, ciudad, asociado, hizo	**u**	actitud, unas, segundo, mundo
D	quedan, mitad, el día	**w**	bueno, cuajar, cuartel, nuevo
s	caso, bastante, gustaba		

En la Tabla 3-III se muestra la lista de los fonemas del castellano en código *SAMPA*, y en la

Tabla 3-IV se reflejan unos ejemplos. En total, para el castellano, existen 31 fonemas. A éstos hay que añadir los referidos a los silencios en diferentes posiciones.

Tabla 3-V. Unidades fonéticas de silencio que se añaden a los de fonemas, en función de su posición.

	Tipo de Silencio
<	**Inicial de frase**
&	**Intermedio**
>	**Final de frase**

En la Tabla 3-V se muestran las tres unidades fonéticas adicionales, que se incorporan a las de la Tabla 3-III, para modelar el silencio en función de su posición dentro de la frase. De esta manera, se establece que existen 34 unidades fonéticas que es necesario modelar.

3.3.2 Tipos de Unidades Fonéticas.

Una vez visto qué fonemas existen para el castellano, es posible ver qué tipos de unidades fonéticas se pueden entrenar en función del contexto. Se distinguen tres diferentes alternativas: *monofonemas*, *bifonemas* y *trifonemas*.

A la hora de identificar una unidad fonética se utiliza la siguiente estructura:

<*fon_actual*>[<*fon_anterior*>,<*fon_posterior*>]

fon_anterior indica la unidad fonética que se localiza a la izquierda y *fon_posterior* la que se localiza a la derecha. En cualquiera de los casos, si uno de los

contextos no se considera se utiliza el símbolo "*". De esta manera, los tipos de unidades fonéticas que existen son:

- *Monofonema:* $f_2[*,*]$
- *Bifonema izquierdo:* $f_2[f_1,*]$
- *Bifonema derecho:* $f_2[*,f_3]$
- *Trifonema:* $f_2[f_1,f_3]$

donde f_1 indica el fonema anterior, f_2 indica el fonema actual y f_3 indica el fonema posterior.

3.3.3 Estructura de Markov de las Unidades Fonéticas.

A continuación, se establece la forma de los modelos de Markov que se utiliza para modelar las unidades fonéticas. Como se menciona en 2.3.1 todos los modelos que estén relacionados con la variable temporal deben tener un estado de inicio y uno de fin. En general, la estructura de red más usada y simple es la denominada de izquierda-derecha o de Bakis. Esta estructura no tiene la posibilidad de que a partir de un estado salgan transiciones hacia estados anteriores, sino que siempre hay que avanzar, y no presenta bucles cerrados. Existen otras alternativas como el sistema *Sphinx* de *CMU* [Hua93], donde el modelo permite modelar un mayor número de posibilidades, pero complica un poco más la estructura de cálculo.

Todos las unidades fonéticas, ya sean libres de contexto (como los monofonemas) o con contexto (bifonemas derechos e izquierdos y trifonemas) se modelan utilizando esta estructura, pero se consideran dos alternativas distintas:

- *Modelos de Bakis de primer orden* (o *de salto simple*): Sólo existen dos tipos de transiciones de estado: de un estado al mismo (del estado *i* hacia el estado *i*), o que salga de un estado y conecte con el inmediatamente a su derecha (del estado *i* al estado *i* +1). En la Figura 3-7 se muestra un ejemplo de un modelo formado por tres estados.

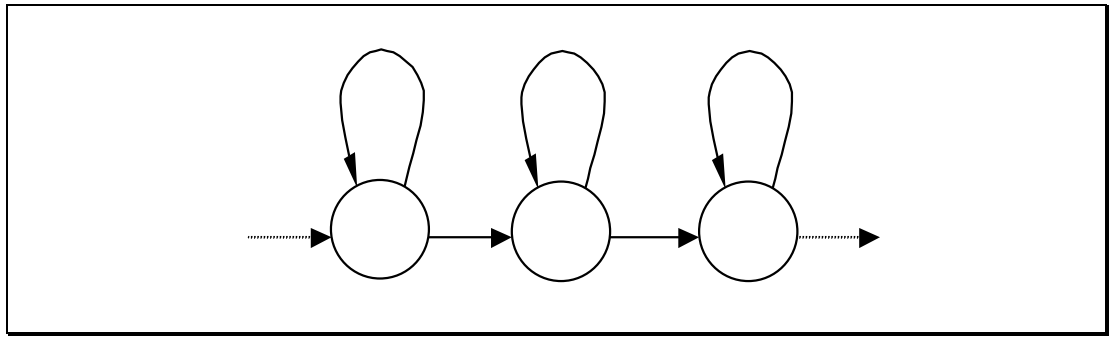

Figura 3-7. Modelo de Bakis de primer orden con tres estados.

- *Modelos de Bakis de segundo orden* (o *salto doble*): Adicionalmente a las posibilidades del modelo de salto simple, se permite que un estado esté conectado con otro dos posiciones más a la derecha (del estado i al estado $i + 2$). En la Figura 3-8 se muestra un ejemplo de un modelo formado por cuatro estados.

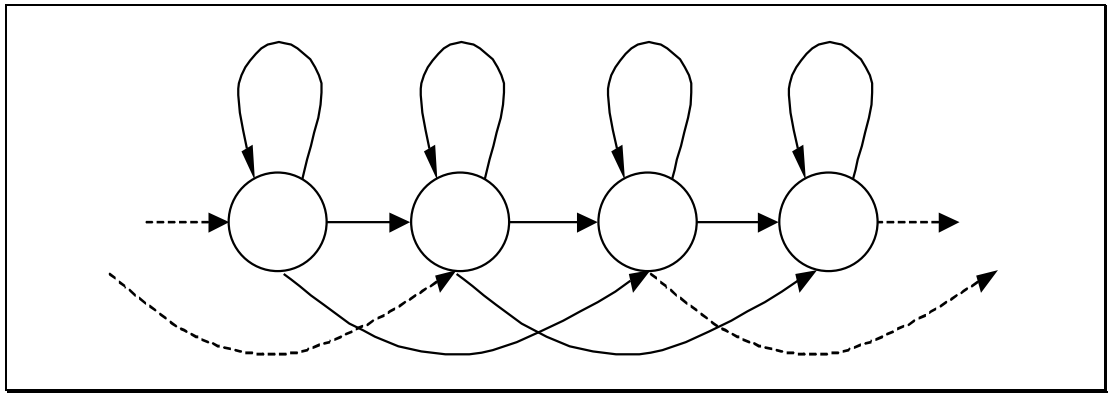

Figura 3-8. Modelo de Bakis de segundo orden con cuatro estados.

A continuación se establece el número de estados que se utiliza para modelar cada unidad fonética. Hay dos posibilidades: realizar una asignación uniforme, correspondiendo a todos los modelos el mismo número de estados; o modelar cada uno con un número diferente dependiendo de las peculiaridades del mismo. Dado que la extracción de características utiliza una ventana de análisis de $32\ ms$ con un desplazamiento de $16\ ms$, puede ocurrir que una unidad fonética esté formada por una única trama, por lo cual, se puede asignar un número de estados diferente a cada modelo en función de la duración media de cada una de ellas.

Considerando la segunda posibilidad, se asignan uno, dos o tres estados para una misma unidad fonética, independientemente de si se modela con contexto o sin él. En todos los casos, se está hablando de emisión en los nodos, es decir, de una *máquina de Moore*.

Las ventajas de esta estrategia son principalmente dos. Por un lado, el número medio de tramas por estado tiene una variabilidad menor; al reconocer palabras, el número de tramas que se asigna a cada estado es más homogéneo. Por otro lado, al tener, en casos concretos con fonemas muy cortos, un número de estados más bajo, hay más vectores de características de entrenamiento y, por tanto, estarán mejor entrenados.

Tabla 3-VI. Número de estados que se asigna a cada modelo fonético.

Fonema	*Nº estados*	*Fonema*	*Nº estados*	*Fonema*	*Nº estados*
<	**1**	**N**	**3**	**L**	**3**
&	**1**	**tS**	**3**	**rr**	**3**
>	**1**	**B**	**2**	**r**	**2**
P	**3**	**f**	**3**	**i**	**3**
b	**1**	**T**	**3**	**j**	**2**
t	**3**	**D**	**2**	**e**	**3**
d	**1**	**s**	**3**	**a**	**3**
k	**3**	**z**	**2**	**o**	**3**
g	**1**	**Z**	**3**	**u**	**3**
m	**3**	**x**	**3**	**w**	**2**
n	**3**	**G**	**2**		
J	**3**	**l**	**2**		

En la Tabla 3-VI se muestra el resultado obtenido en lo que se refiere al número de estados que para cada unidad fonética se emplea en su modelado.

Las vocales tienen tres estados y las semivocales *j* y *w* sólo tienen dos. En el caso de las oclusivas, mientras las sordas (*p*, *t* y *k*) tienen tres estados, las sonoras (*b*, *d* y *g*) tienen sólo uno, esto es debido a que todas ellas modelan el silencio que las acompaña, pero en las sonoras su duración es mucho menor que en las sordas. El resto de unidades fonéticas varían entre dos y tres, nunca uno, pues tienen una duración bastante larga.

3.3.4 Frases de Entrenamiento. Base de Datos Albayzín.

3.3.4.1 Base de Datos Albayzín.

A la hora de entrenar unos modelos fonéticos, es necesario utilizar bases de datos que presenten unas características adaptadas al tipo de reconocimiento que se realice. En [All97] se puede comprobar el efecto de utilizar modelos fonéticos entrenados a partir de frases pronunciadas de forma natural en el reconocimiento de palabras aisladas. No sólo la forma de hablar tiene que ser similar tanto en entrenamiento como en reconocimiento, sino que las diferentes posibilidades deben estar recogidas en la base de datos.

En el presente reconocedor de voz se ha utilizado la base de datos del proyecto Albayzín. Esta base de datos consta de dos corpus. El primero está formado por 200 frases leídas por locutores masculinos y femeninos. Fue grabada a través de micrófono a una frecuencia de muestreo de 16 *kHz*, existen 24 repeticiones por frase, con un total de 4.800 frases. La cuarta parte de esta base, es decir, 1.200 frases, se encuentra manualmente segmentada usando el código *SAMPA*, del resto únicamente se tiene su transcripción fonética. El segundo está formado por 500 frases repetidas en grupos de 50 frases por locutor, con 4 repeticiones por frase, en total 2.000 frases adicionales. No obstante, este segundo corpus no puede utilizarse para entrenar unidades fonéticas sin

utilizar la segmentación automática, puesto que no tiene marcas de comienzo y de fin de fonemas.

3.3.4.2 Adaptación de la Base de Datos Albayzín.

Como se menciona en el apartado anterior, la base de datos fue grabada a través de micrófono y con una frecuencia de muestreo de $16\ kHz$. Sin embargo, el reconocedor está pensado para funcionar a través de canal telefónico, con las peculiaridades que fueron mencionadas en el apartado 3.2 sobre la extracción de características. Es necesario realizar una conversión para que se adapte a las condiciones de funcionamiento que este presenta.

Primeramente, es preciso realizar un diezmado de la base de datos de $16\ kHz$ a $8\ kHz$, y filtrar la señal a través de un canal similar al telefónico para que ésta presente unas características similares de atenuación y ruido.

El siguiente paso, consiste en realizar la extracción de características de la base de datos, añadiéndole las marcas de principio y fin de los vectores de características de los fonemas en la parte que está segmentada manualmente. Hay que establecer un criterio para poder asignar los vectores de características a uno u otro fonema en función de la marca temporal asociada. Además, es preciso tener en cuenta que el extractor de características introduce un retardo de dos tramas, puesto que los coeficientes se calculan utilizando una regresión de orden dos sobre la señal. En la Figura 3-6 puede verse este proceso de generación de vectores de características.

Para convertir las marcas temporales en los índices de tramas de características generadas se establece el siguiente criterio: primeramente, se convierten las marcas temporales en número de muestras al multiplicar por la frecuencia de muestreo. A continuación, se divide entre el número de muestras por desplazamiento para poder numerar las marcas. Finalmente, teniendo en cuenta que cada marca indica tanto el final de un fonema como el principio del siguiente, se resta al valor obtenido un número de tramas relacionado con la forma de calcular los vectores, uno en el caso de ser el indicador de trama final y dos si es del principio. En las siguientes expresiones se muestra el criterio seguido:

- Si es *marca final de fonema* (*MF*):

$$N\_VC = \frac{MARCA\ \ FINAL}{LONGITUD\ \ DE\ \ TRAMA} - 2$$

- Si es *marca inicial de fonema* (*MI*):

$$N\_VC = \frac{MARCA\ \ INICIAL}{LONGITUD\ \ DE\ \ TRAMA} - 1$$

Adicionalmente, y debido a la manera en que está obtenida la segmentación y la transcripción fonética, hay que considerar los siguientes puntos:

- Las frases siempre comienzan con silencio inicial (**<**) y acaban con silencio final (**>**). El resto de silencios son intermedios (***&***).
- Siempre existe el silencio inicial de las frases, que es suficientemente grande como para que tenga por lo menos una trama, por lo que la primera trama obtenida es la del silencio inicial (**<**). Ocurre exactamente igual con el silencio final (**>**).
- Dada la forma en que se asignan los vectores de características, es posible, para ciertos fonemas con una duración inferior a $32\ ms$, que el número del vector final sea menor que el inicial. En estos casos no se considera dicho vector. Sólo se aceptan fonemas con una duración de al menos un vector.

Esta segmentación, que asigna los vectores de características a una determinada unidad fonética, es sólo para el entrenamiento de éstos fuera del contexto de la frase completa, es decir, utilizando la segmentación manual que existe sobre las frases de entrenamiento.

Para realizar la segmentación automática se requieren unos modelos iniciales obtenidos a partir de esta segmentación manual, pues si no se hiciera así y se empezara, por ejemplo, con una segmentación uniforme, no se garantizaría que los modelos entrenados correspondieran a las unidades fonéticas deseadas, y tardaría mucho en converger, llegando a mínimos locales.

3.3.5 Procedimiento de Entrenamiento Segmental de k-Medias de Modelos Continuos de Markov (CHMM).

3.3.5.1 Esquema General de Entrenamiento.

Independientemente de utilizarse la segmentación manual proporcionada por la base de datos, o la segmentación automática, el esquema de entrenamiento de los modelos de Markov sigue exactamente la misma estrategia. Únicamente variará la manera de aplicar el algoritmo de entrenamiento.

Se podría considerar segmentar y etiquetar cada trozo de voz manualmente, pero resultaría muy laborioso e incluso podría generar errores debido a las inconsistencias en la determinación de los bordes entre unidades fonéticas. El procedimiento de entrenamiento implementado es similar al ya descrito en [Jua90] o [Rab93] para palabras aisladas, y que se basa en el algoritmo de las *k*-medias.

El entrenamiento de modelos continuos consta de los siguientes pasos:

1-) Segmentación uniforme de los vectores de entrenamiento para la obtención de unos modelos iniciales.

2-) Segmentación óptima o automática de las tramas en estados utilizando el algoritmo de Viterbi para asignar tramas a estados. Los parámetros del

modelo de cada estado se obtienen por procedimientos de agrupamiento de cuantificación vectorial típico, como el algoritmo *k*-medias.

3-) Evaluación de la convergencia a partir de las probabilidades obtenidas en el paso 2. Si las probabilidades mejoran y el cambio ha sido superior a un determinado umbral, se continúa en el paso 2 para mejorar los modelos. Si mejoran, pero el cambio no supera dicho umbral, se dan por buenos los modelos y se pasa al punto 4. Si las probabilidades empeoran porque los modelos anteriormente entrenados son mejores, se pasa al punto 4 con ellos.

4-) Se guardan los modelos generados.

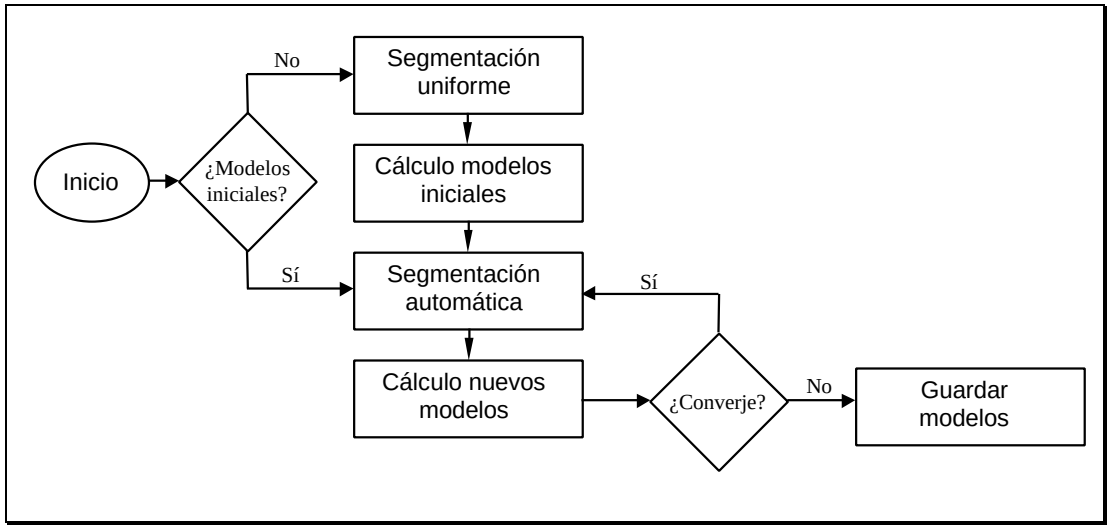

Figura 3-9. Algoritmo de entrenamiento de modelos de Markov continuos (*CHMM*).

En la Figura 3-9 puede verse el esquema utilizado para realizar el entrenamiento, y se observa que el entrenamiento se puede comenzar de dos maneras: partiendo de unos modelos iniciales o empezando por una segmentación inicial uniforme.

Como ya ha sido mencionado a la hora de hablar de los *HMM*, un modelo de Markov se define a partir de un conjunto de parámetros, que utilizando la forma Moore resultan ser:

A: Matriz de transición de estado.

B: Matriz de probabilidades de observación, que proporciona las probabilidades de observación respecto de todos los vectores de características y para todos los tiempos.

$\pi(1)$: Vector de probabilidades de estado inicial.

Todos los estados tienen idénticas probabilidades de estado iniciales, $\pi(1)$, es decir, al empezar a entrenar no existe ningún estado más importante que el resto. Este

término es una constante y, por tanto, no es necesario considerarlo para el cómputo del algoritmo de reconocimiento.

Otra simplificación que se hace es el cálculo inicial de la matriz de transición de estados A. Se ha probado que las probabilidades de transición no afectan de manera significativa al funcionamiento del reconocimiento. Se considera que todos los valores de las probabilidades de transición son independientes del estado, es decir, sólo dependen de la posición relativa de los mismos y, además, estos valores se calculan una vez, considerándose constantes para el resto del entrenamiento. Esto ofrece las siguientes ventajas:

- Simplifica el cálculo del resto de parámetros de los modelos de Markov. El cálculo de los modelos de Markov se restringe al cálculo de la matriz B de probabilidades de observación.
- Simplifica el algoritmo de reconocimiento, aumentándose la eficiencia, tanto en espacio de memoria necesario para almacenar los parámetros como en cómputo.

Las ventajas que se obtienen con esta imposición son claramente superiores a las desventajas que se pudieran generar. La principal desventaja sería la de un incremento en la tasa de error, al no haberse entrenado con unos parámetros calculados de forma óptima. Sin embargo, hay que considerar que eliminar ciertos parámetros, que a la hora de la verdad no aportan demasiado, puede suponer un incremento notable en ahorro de memoria y computación y, por tanto, un incremento en el número de pruebas que se pueden realizar.

Cuando se está hablando del diseño de un reconocedor que funcione en aplicaciones reales en equipos de mercado, hay que considerar que existen otros factores que pueden llegar a ser más importantes, como por ejemplo, el uso de vocabularios más grandes, medidas para aumentar la robustez en diferentes aspectos o, incluso, que un mismo equipo sea capaz de funcionar con varios reconocedores a la vez.

Tabla 3-VII. Probabilidades de transición para una red de Bakis de primer orden o salto simple.

	Valor
p(i, i)	***0,20***
p(i, i + 1)	***0,80***

En la Tabla 3-VII se muestran los valores de probabilidad de transición para un modelo de Bakis de primer orden.

Tabla 3-VIII. Probabilidades de transición para una red de Bakis de segundo orden o salto doble.

	Valor
p(i, i)	***0,15***
p(i, i + 1)	***0,75***
p(i, i + 2)	***0,10***

En la Tabla 3-VIII aparecen las probabilidades de transición consideradas para un modelo de Bakis de segundo orden o de saltos dobles.

3.3.5.2 Tipos de Segmentaciones.

Teniendo en cuenta las diferentes alternativas con relación a utilizar o no la segmentación fonética de la base de datos, se pueden distinguir los siguientes tipos de asignaciones de tramas a estados:

- **Segmentación uniforme:** Este tipo es el que se utiliza cuando no se tiene información a priori sobre cómo realizar la primera segmentación. Sería necesaria, por ejemplo, en el caso del entrenamiento de las unidades fonéticas utilizando la segmentación fonética de las frases. En caso de no saber cómo realizarla lo mejor es repartir las tramas entre los estados disponibles para así poder generar unos modelos iniciales.
- **Segmentación manual:** No es una segmentación como tal, sino que consiste en utilizar directamente en el entrenamiento de los modelos la información temporal asociada con la segmentación en fonemas.
- **Segmentación automática a través del algoritmo de Viterbi**. Una vez que se dispone de unos modelos iniciales, ya se puede realizar una asignación automática de las tramas a estados. Este algoritmo puede utilizarse tanto para fragmentar modelos fonéticos como modelos de palabras aisladas.

3.3.5.3 Generación del Modelo de Markov de cada Estado.

Independientemente de haberse realizado una segmentación uniforme o automática, se genera un conjunto de vectores de características para generar la función de distribución correspondiente.

Un elemento b_i, en el caso de una función multigaussiana, se define a partir del conjunto de los siguientes parámetros:

- Los vectores de medias.
- Los pesos de ponderación de las gaussianas.
- La matriz de covarianzas.

Como ya se mencionó a la hora de hablar de la ponderación del vector de características en el apartado 3.2.7, se supone independencia de cada una de las componentes, por lo cual la matriz de covarianzas es diagonal. Adicionalmente, se impuso la condición de que una vez calculada dicha matriz no es necesario reestimarla.

Como también ha sido mencionado, la varianza de cada componente es la varianza que tiene esa componente entre todos los vectores de características de entrenamiento. Al realizarse la normalización se obtiene un conjunto de vectores cuya varianza es la unidad, y el cálculo de las log-verosimilitudes pueden utilizar directamente la distancia euclídea.

Para el cálculo de las medias de las gaussianas y el valor de los pesos asociados con cada una de ellas se aplica una cuantificación vectorial, para lo cual se divide el conjunto de vectores de características en un conjunto de grupos (*cluster*), cada uno de los cuales define una de las gaussianas. El vector medio de cada cluster (*centroide*) será el vector de medias asociado con cada gaussiana. Los pesos de ponderación de cada una de estas gaussianas se corresponden con el porcentaje de vectores que cada cluster contiene. Para realizar la cuantificación vectorial se utiliza el algoritmo *LBG* o de *Lloyd Generalizado por "División de Centroides"*.

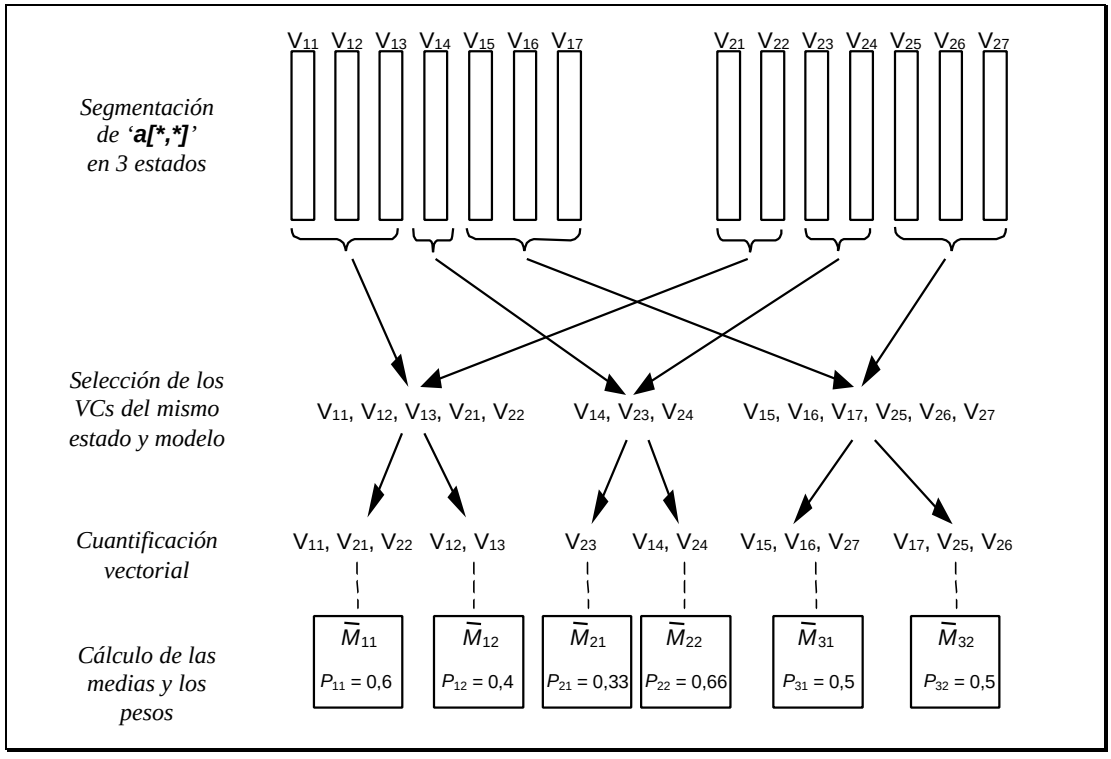

Figura 3-10. Esquema del cálculo de las medias y los pesos de las gaussianas asociadas a cada estado.

En la Figura 3-10 se muestra de forma gráfica cómo se obtendrían los parámetros asociados con un modelo de Markov a partir de la segmentación de las frases en estados. En este caso, se corresponde con la unidad fonética ***a[*,*]*** que consta de tres estados. V_{ij} identifica al vector de características j-ésimo del fonema i-ésimo. Por otro lado, M_{kl} identifica a la media l-ésima del estado k-ésimo, P_{kl} es el peso asociado a la gaussiana l-ésima del estado k-ésimo.

3.3.5.4 Algoritmo LBG o de Lloyd Generalizado por "División de Centroides".

Este algoritmo permite obtener un número de gaussianas que es múltiplo de 2^I, donde I es el número de veces que se itera el algoritmo. Partiendo de un *centroide*, éste se divide en dos y se busca la asignación óptima. A su vez se divide en otros dos, obteniéndose cuatro, iterándose el algoritmo sucesivamente hasta llegar al número de *centroides* deseado.

Una forma más formal de expresar el algoritmo es:

- **Inicialización:** Encontrar el centroide de la población entera de vectores. Éste se denomina el vector de código inicial, que es único.
- **Recursión**: En total hay *I* iteraciones y se obtienen 2^I *centroides*. Denominando a las iteraciones *i* = 1, 2,... , *I*, para cada iteración *i* se tiene:

  1- Dividir cualquier centroide o vector de códigos existente $\bar{x}$, en dos nuevos centroides, esto es, $\bar{x}(1-e)$ y $\bar{x}(1+e)$, donde e es un número pequeño. El resultado es un conjunto nuevo de centroides $\bar{x}_k^i$, con *k* = 1, 2, ... , 2^i.

  2- **Recursión**: Se hace otro conjunto de iteraciones hasta que se obtenga un conjunto de centroides con distorsión mínima.

  2.1- Para cada vector de características, *x*, del conjunto de entrenamiento, se cuantifica *Q*(*x*) en el código $\bar{x}_{k^*}^i$, con

  $$k^* = \arg\min_k d\left(x, \bar{x}_k^i\right) \qquad [3.15]$$

  La distancia d(,), es la medida de distorsión que, en este caso, es la distancia euclídea.

  2.2- Calcular la distorsión total actual como resultado de la cuantificación,

  $$D = \sum d[x, Q(x)] \qquad [3.16]$$

  donde el sumatorio se realiza sobre todos los vectores *x* del conjunto de entrenamiento y *Q*(*x*) indica el centroide al que *x* es asignado en la actual iteración.

  2.3- Para cada *k*, calcular el centroide de todos los vectores *x* tales que $\bar{x}_k^i = Q(x)$, actualizándolos para la iteración actual. Este nuevo conjunto de centroides forma el nuevo *codebook*. Si el valor de *D* calculado en 2.2 es suficientemente pequeño ir a 3, si no volver a 2.1.

  3- Si i < I, volver al paso 1.

Los *centroides* así obtenidos corresponden a las medias de cada gaussiana y, una vez ejecutado este algoritmo, se obtiene el peso asociado con cada una de ellas calculando el porcentaje de vectores de entrenamiento *x* utilizado en la estimación de la media de cada gaussiana.

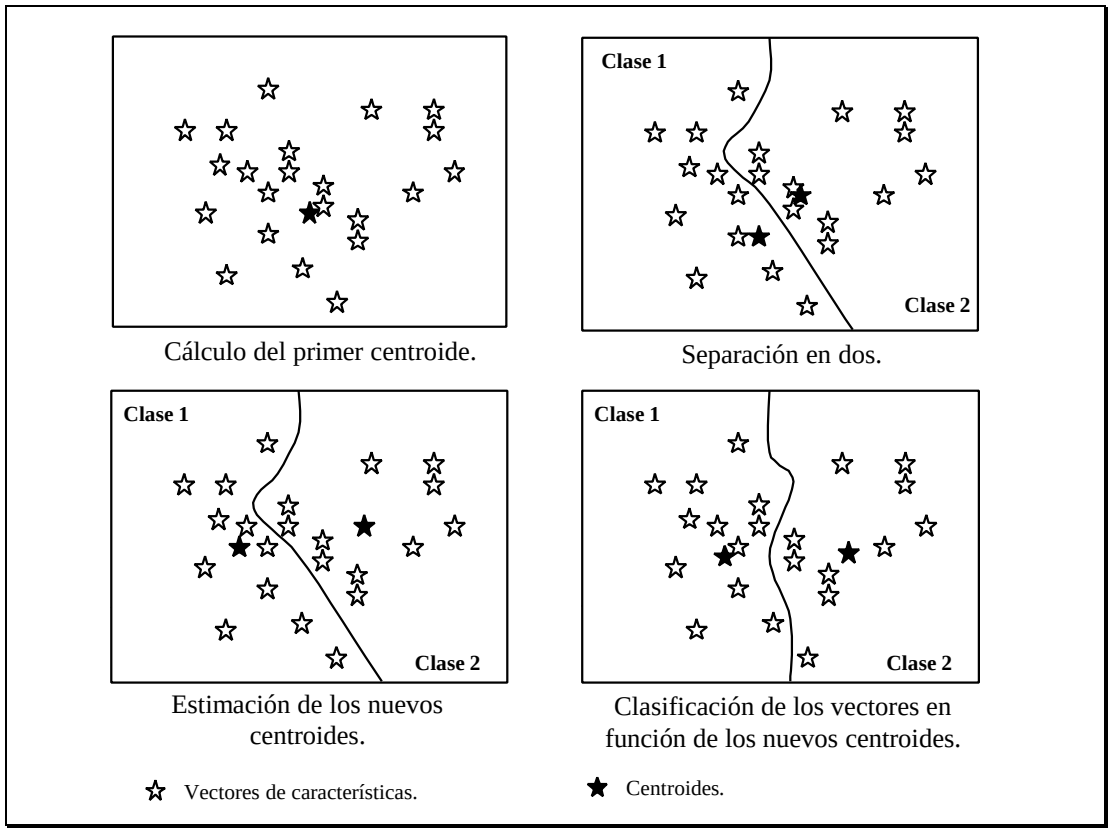

Figura 3-11. Algoritmo *LBG* para la estimación de los centroides que serán las medias de las gaussianas de los modelos de estado.

En la Figura 3-11 se muestra gráficamente la aplicación del algoritmo *LBG* de cuantificación vectorial para la estimación de las medias de las gaussianas de los modelos de estado continuos. Las medias se corresponden con los centroides calculados.

3.3.6 Conversión a Modelos Semicontinuos de Markov (SCHMM).

Cuando se maneja un conjunto reducido de modelos de unidades fonéticas, como en el caso de los monofonemas, el número de parámetros que se entrena no es muy elevado y el sistema es muy eficiente. Por el contrario, cuando se desea aumentar el número de modelos con unidades fonéticas dependientes del contexto, éste empieza a aumentar en gran medida proporcionalmente a la cantidad de nuevos modelos que se consideran.

Una manera de controlar el aumento de memoria y de carga computacional debido al aumento del número de modelos, para evitar que disminuyan las prestaciones en cuanto al tamaño de vocabulario que el reconocedor es capaz de manejar cuando se utilicen monofonemas, consiste en utilizar modelos semicontinuos de Markov. Como ya se indica en [Hua90], la compartición de las gaussianas entre todos los modelos posibilita reducir el número de parámetros a la vez que éstos, a los que les corresponde un mayor número de vectores de entrenamiento, estarán mejor caracterizados.

En [Hua90] se hace una extensión del algoritmo de entrenamiento basado en Baum-Welch, que inicialmente fue diseñado para modelos discretos y continuos, en modelos semicontinuos. Tal como se establece en 2.3.3, en la presente tesis se ha optado en todo momento por la utilización del algoritmo de Viterbi tanto en entrenamiento como en reconocimiento, dada su mayor sencillez, sin que por ello los modelos obtenidos se degraden de forma elevada.

Para la obtención de unos modelos semicontinuos de Markov se propone realizar una conversión de los *CHMM* en los *SCHMM*. En todos los casos, primeramente se buscan los *centroides* que representan las medias de las funciones densidad de probabilidad gaussianas que comparten todos los modelos. El paso siguiente consiste en obtener los pesos que a cada *CHMM* le corresponde en función de la distancia a las mismas. Se pueden distinguir dos variantes en función de cómo se realiza la obtención del conjunto de *centroides* común a los modelos:

- Método basado en la cuantificación vectorial.
- Método basado en el agrupamiento de gaussianas de los *CHMM*.

Una vez obtenido este conjunto de *centroides*, en función de la distancia de los *CHMM* a los mismos, se establece el conjunto de pesos para cada modelo. El cálculo de estos pesos es igual, independientemente del método con el que se hayan obtenido las medias de las gaussianas.

3.3.6.1 Método Basado en la Cuantificación Vectorial.

A partir del conjunto de vectores de características de la base de datos de entrenamiento se realiza una cuantificación vectorial, similar a la que existe en los modelos discretos de Markov, pero con la diferencia de que aquí representa la media de unas funciones densidad de probabilidad gaussianas.

El principal inconveniente que a priori presenta esta técnica es que el cuantificador vectorial proporciona un conjunto de centroides cuya distribución depende en gran medida del número de apariciones de vectores que de cada unidad fonética existe en la base de datos de entrenamiento. Por ejemplo, para el caso de la base de datos Albayzín y el entrenamiento de monofonemas, los silencios representan sólo tres modelos de un estado, frente al total de 82 estados que existen para todos los modelos. Sin embargo, el $20\ \%$ de los vectores de características de entrenamiento representan al silencio y, por tanto, habrá un número muy significativo de centroides que estén asociados con el mismo, superior a lo que debería corresponderle. Este mismo fenómeno ocurre en el caso de las vocales, que son el otro conjunto que mayor número de centroides generaría.

Sin embargo, en el resto de unidades fonéticas, el número de centroides sería muy reducido y aparecerían grandes problemas de discriminación.

3.3.6.2 Método Basado en la Reducción de Gaussianas de los CHMM.

Si la obtención del conjunto de centroides de la cuantificación vectorial de los datos de entrenamiento no es muy adecuada, se puede utilizar directamente las medias de los modelos continuos entrenados.

La utilización de modelos de unidades fonéticas dependientes de contexto aumenta el número de parámetros a entrenar y, por tanto, a cada parámetro le corresponde un menor número de vectores de características. Puesto que estos modelos así entrenados no son muy fiables, se puede realizar una disminución de su número agrupando los más cercanos dos a dos hasta conseguir el número deseado de centroides.

3.3.6.3 Obtención de los Pesos de los Modelos Semicontinuos.

Obtenido el conjunto de centroides que representa al conjunto de gaussianas comunes, se realiza el cálculo de los pesos asociados:

- Para cada centroide obtenido se calculan las probabilidades asociadas de las medias de los modelos semicontinuos ponderando por el peso de las mezclas de los continuos. Se tendrá un valor para cada uno de los centroides.
- Selección de los centroides cuya distancia sea la mayor para quedarse con un número reducido de ellas.
- Normalización de los pesos para que se cumpla que su suma sea igual a la unidad.

Por ejemplo, en el caso de los monofonemas con unos CHMM de 2 pesos cada uno existe un total de 164 gaussianas. Se puede reducir a 128 con 4 pesos por modelo teniendo una reducción del 22 % de memoria.

3.4 Modelado de Palabras

3.4.1 Introducción.

Los fonemas son las unidades mínimas con significado cuya concatenación forma palabras, que son los elementos de que consta el lenguaje. En el apartado 3.3 se ha descrito cómo es posible entrenar modelos de unidades fonéticas a partir de una base de datos formada por frases de las que se tiene su transcripción fonética.

Adicionalmente, hay que considerar otro tipo de modelado de palabras. Existen palabras, como los dígitos, que suelen ser muy breves y muy similares entre sí, por lo que resulta difícil obtener tasas de error bajas utilizando su transcripción fonética. Por ello, también es posible introducir en el reconocedor modelos de palabras completas.

En este apartado se describe cómo obtener modelos de palabras utilizando dos metodologías diferentes:

- Modelado de palabras a partir de la concatenación de unidades fonéticas.
- Entrenamiento de palabras aisladas.

Ambas alternativas no son categóricas, pues es incluso posible introducir en una misma red de reconocimiento modelos de los dos tipos.

3.4.2 Modelado Fonético de Palabras.

El castellano presenta la ventaja de que dada una palabra, cualquier persona es capaz de pronunciarla aunque no la haya visto ni oído nunca, ya que existe un conjunto directo de reglas que indican de forma prácticamente unívoca la secuencia de fonemas que hay que pronunciar. Esto simplifica el diseño de un transcriptor fonético, pues el conjunto de reglas está muy bien acotado y el número de excepciones es muy bajo. En otros idiomas, como el inglés, esto supone un gran problema, porque el número de excepciones es casi equivalente al de palabras.

Para este caso concreto, el del castellano, y utilizando el código *SAMPA* se tiene la lista de fonemas enumerados en la Tabla 3-III. Únicamente es necesario que el transcriptor fonético analice la secuencia de palabras que aparece en las frases del vocabulario y proporcione, para cada una de ellas, la secuencia de unidades fonéticas en trifonemas que habría que pronunciar.

A continuación, se establece de una manera más formal cómo formar la secuencia de modelos de palabras de una frase a partir de la secuencia de unidades fonéticas de cada una de las palabras, considerando que entre dos de ellas existe un silencio (reconocimiento de palabras concatenadas, como se puede ver en el apartado 2.4.1.2, sobre la diferencia entre reconocimiento continuo y discreto).

Un aspecto fundamental a la hora de realizar la transcripción fonética es el de los efectos contextuales entre palabras, incluso aunque exista un silencio entre dos de ellas. La razón es que en castellano ciertos fonemas no se pronuncian igual si se encuentran en posición inicial absoluta o en medio de palabra, existiendo o no silencio entre ambos fonemas. En la Tabla 3-III se comprueba que existen ciertas unidades fonéticas, como las oclusivas sonoras *b*, *d* y *g*, que pronunciadas fuera de la posición inicial se convierten en las fricativas sonoras *B*, *D* y *G*.

Para denominar una frase se utilizan las letras ***fr***, para una palabra ***p*** y para una unidad fonética ***fn***. Como ya se ha mencionado, las tres unidades para modelar el silencio en las diferentes posiciones son:

- **<**: Silencio inicial de frase.
- ***&***: Silencio entre palabras dentro de frase.
- **>**: Silencio final de frase.

De este modo, cualquier frase se puede expresar como la concatenación de un conjunto de *N* palabras separadas por silencios:

$$\textbf{\textit{fr1}}: < p_1 \ \& \ p_2 \ \& \ p_3 \ \& \ ... \ \& \ p_N >$$

A su vez, cada palabra puede expresarse como una secuencia de *M* fonemas.

$$p_i: \ fon_1 \ fon_2 \ fon_3 \ ... \ fon_M$$

Asimismo, cada palabra puede definirse como una secuencia de unidades fonéticas. El caso más general es el de los trifonemas. La palabra anterior como secuencia de trifonemas tendría la siguiente expresión:

$$p_i: \ fon_1 [\&, fon_2] + fon_2 [fon_1, fon_3] + fon_3 [fon_2, fon_4] + ... + fon_M [fon_{M-1}, \&]$$

Si la palabra fuese la inicial de la frase o la final habría que sustituir el contexto correspondiente por el símbolo del silencio inicial o final.

Los modelos de Markov de cada unidad fonética están formados por una red de Bakis con un número de estados que dependen de la longitud media de dichas unidades fonéticas. Los modelos de palabras basados en características fonéticas se obtendrían mediante la concatenación de dichos modelos fonéticos.

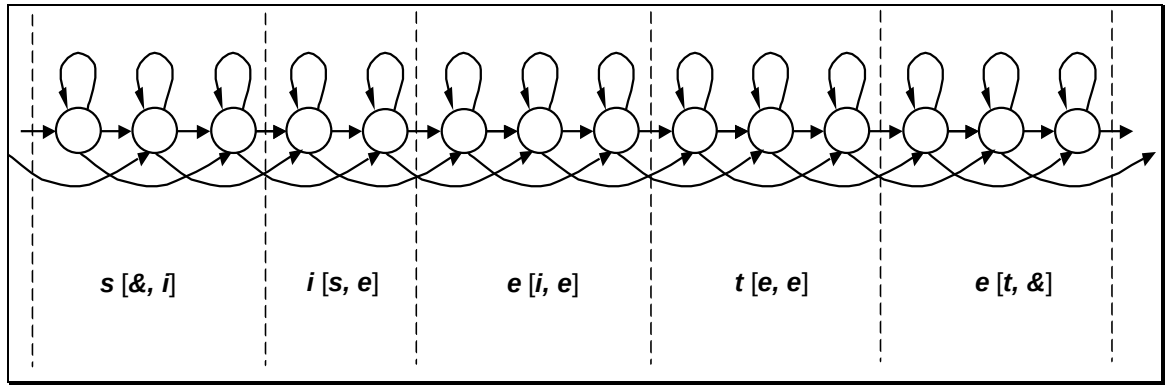

Figura 3-12. Ejemplo en la formación de un modelo para la palabra *siete* a partir de los modelos fonéticos de los trifonemas correspondientes, utilizando estructuras de Bakis de segundo orden.

En la Figura 3-12 se muestra un ejemplo de palabra generada por concatenación de modelos fonéticos, de manera que el último estado de una unidad fonética se yuxtapone al primer estado de la siguiente. Esta estructura uniforme no permite discriminar si se trata de una secuencia de modelos fonéticos o una palabra modelada entera, si se eliminaran las etiquetas fonéticas.

Se observa que la información temporal asociada con las duraciones de fonemas está incluida implícitamente en los modelos fonéticos entrenados.

3.4.3 Modelado de Palabras Aisladas.

En ciertos casos, puede resultar interesante la utilización de modelos de palabras enteras para conseguir aumentar el grado de reconocimiento, que debido a la similitud entre palabras breves pronunciadas aisladamente ocasionaría un aumento de la tasa de error.

Un caso particular es el reconocimiento de las palabras *sí* y *no*. Reconocer estas palabras utilizando modelos de Markov no es una tarea sencilla. En muchas ocasiones el fonema *s* tiene muy poca energía, debido a las características del canal telefónico, que presenta una frecuencia de corte situada en los $3.400\ Hz$. Por eso, realmente consiste en discriminar entre *i* y *no*, que resulta ser aún más difícil, debido a la similitud que presenta el fonema *i* y el *n* en el espectro.

El reconocimiento fonético de Markov se basa en la existencia de fonemas con un grado de discriminación elevado, pero más importante es la imposición de las restricciones que aparecen al forzar la secuencia de fonemas. Por ello, en el caso de las palabras *sí* y *no*, la tasa de error que presentan está relacionada con el reconocimiento de cada unidad fonética entre sí.

El capítulo 4 está dedicado al análisis de técnicas para aumentar la robustez del reconocedor de voz, y se mencionan otras posibilidades para mejorar la discriminación mediante unidades fonéticas utilizando transcripciones alternativas de estas palabras.

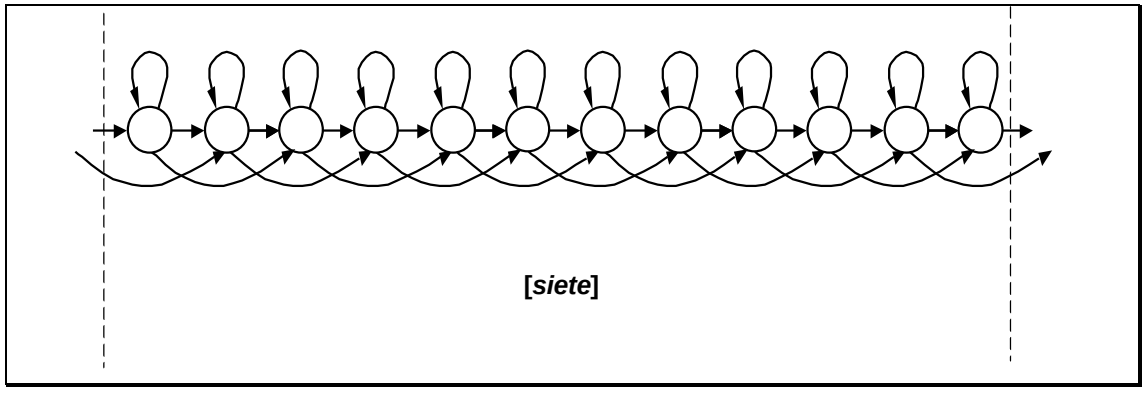

Figura 3-13. Ejemplo de un modelo de palabra *siete* entrenada de forma completa, utilizando una estructura de Bakis de segundo orden con doce estados.

El número de estados necesarios para modelar una palabra de forma aislada hay que adaptarlo en función de la duración media de esa palabra y de la necesidad de aumentar la diferencia entre diversas palabras. Si se utilizan más estados, las palabras se discriminan mejor. Es el mismo efecto que en el caso de palabras obtenidas fonéticamente, a mayor duración de palabra más fácil resulta su reconocimiento.

En la Figura 3-13 se muestra como ejemplo el modelado de la palabra *siete*, al igual que la de la Figura 3-12, pero modelada de forma completa. En este caso, el modelo está formado por doce estados, frente a los catorce de que consta la misma palabra si se modela fonéticamente.

3.5 Cálculo Eficiente de Probabilidades

3.5.1 Introducción.

Como se menciona en el apartado 2.3.3, el algoritmo de Viterbi es usado tanto en reconocimiento como en entrenamiento, es decir, a través de este algoritmo se entrenan los modelos por medio de métodos de agrupamiento segmental y también se utiliza para seleccionar la secuencia de palabras cuya probabilidad sea la mejor.

El cálculo de la probabilidad de cada secuencia de palabras se obtiene a partir del producto de las probabilidades que aparecen a lo largo de la cadena de Markov. La utilización directa de estos valores en unidades naturales es imposible, debido a que éstos suelen ser muy pequeños y al multiplicarse entre sí el resultado tiende a cero, apareciendo grandes errores de precisión y de saturación. La manera de solventarlo es mediante la utilización de las denominas *log-verosimilitudes* (*log-likelihood*), esto es, el logaritmo de las probabilidades al que se le cambia el signo para ser positivas.

La función logaritmo permite reducir de manera importante el rango de variación de cualquier variable. Pero tiene una propiedad incluso más interesante, el logaritmo del producto de términos se convierte en el sumatorio del logaritmo de cada uno de los términos. Ambos aspectos permiten manejar el problema de la precisión en el cálculo de las probabilidades gracias a los rangos dinámicos utilizados.

Existe otro inconveniente adicional. En todos los casos, ya sean los modelos continuos o semicontinuos, a la hora de calcular las probabilidades hay que enfrentarse al problema de obtener la probabilidad de la suma de gaussianas multidimensionales. Si las probabilidades se aplicaran directamente en unidades naturales sería necesario, para cada una de las gaussianas, aplicar una función exponencial. Si se calculan en log-verosimilitudes, además, hay que aplicar la función logaritmo al sumatorio resultante.

El uso de estas funciones, que habría que aplicarlas muchas veces, resulta muy costoso computacionalmente. Para evitar la utilización de estas funciones, se propone el denominado *modelo híbrido de selección de gaussianas* (*HGS*).

3.5.2 Modelo Híbrido de Selección de Gaussianas (HGS).

En el cálculo de las log-verosimilitudes en una cadena de Markov de modelos continuos o semicontinuos, siempre aparece un término que requiere calcular un logaritmo de suma de exponenciales, donde cada una de ellas estaría relacionada con una gaussiana. La expresión tiene la siguiente forma:

$$\log\left(P\left(x \mid q_i\right)\right) = \log\left(\sum_{j=1}^{NMezclas} c_j \cdot \exp\left(d\left(x, m_{ij}\right)\right)\right) \qquad [3.17]$$

La selección de gaussianas (*GS*) es un tema que aparece en la literatura ya con [Boc93] y últimamente en [Gal99], y junto con [Dem96], plantean el tema de la reducción del cálculo de gaussianas y la obtención de modelos semicontinuos con un número reducido de pesos.

En [Dem96] se indica que no es necesario utilizar todas las gaussianas que componen un modelo continuo de Markov, sólo aquellas que más importancia tengan a la hora de calcular las probabilidades. Se mencionan dos alternativas a la hora de elegir qué gaussianas son las que se deben usar:

- *Dependiente de los datos*: De entre el conjunto de gaussianas de los modelos se seleccionan aquellas cuyas distancias al vector de características sean menores.
- *Dependiente de los modelos*: Para cada modelo de estado se utilizan únicamente las que tengan los mejores pesos.

Aunque el enfoque no es exactamente el mismo que en las referencias indicadas, la utilización de un número reducido de gaussianas puede permitir simplificar el cálculo de la expresión [3.17]. Por tener cierta similitud, se propone denominarla como *modelo híbrido de selección de gaussianas* (*HGS*). Este método consiste en considerar para el cálculo de la probabilidad sólo una única gaussiana, aquella que al aplicarla proporcione el peso del modelo con valor mayor.

Si se introduce la simplificación de una única exponencial en la expresión [3.17], la función logaritmo y la función exponencial son inversas y se cancelan, reduciéndose el cálculo de las log-verosimilitudes al de distancias entre las medias de las gaussianas y el vector de características ponderados por la varianza.

En los siguientes subapartados se muestra cómo quedarían las expresiones simplificadas para el caso de *CHMM* (apartado 3.5.2.1), y de *SCHMM* con varios *codebooks* (apartado 3.5.2.2).

3.5.2.1 HGS Aplicado a Modelos Continuos de Markov.

Para el caso de modelos continuos, la expresión general del cálculo de la probabilidad de generación, escrita en forma natural, queda como:

$$P(S \mid q_j) = \sum_{m=1}^{NM} c_{mj} \prod_{i=1}^{LV} \left[\frac{1}{\sqrt{2ps_i}} \exp\left(-\frac{1}{2} \left(\frac{s_i - m_{imj}}{s_i} \right)^2 \right) \right] \qquad [3.18]$$

Ésta representa la probabilidad de observación del vector de características S en un estado j; NM el número de mezclas de gaussianas, en este caso, se utilizan dos; LV el número de componentes, en este caso, dieciocho; s_i^2 las varianzas y m_{imj} las medias. Esta expresión puede simplificarse teniendo en cuenta que:

- Los pesos dependen de la mezcla y del estado, pero no de la componente del vector.
- Las varianzas dependen únicamente de la componente del vector.

Con estas nuevas consideraciones la expresión se puede reescribir como:

$$P(S \mid q_j) = \left(\prod_{i=1}^{LV} \frac{1}{\sqrt{2p}\, s_i} \right) \cdot \sum_{m=1}^{NM} \left[c_{mj} \cdot \exp\left(-\frac{1}{2} \sum_{i=1}^{LV} \left(\frac{s_i - m_{imj}}{s_i} \right)^2 \right) \right] \qquad [3.19]$$

En la expresión [3.19] se identifica la distancia del vector de características (término mencionado en el apartado de extracción de características) con un vector de medias, considerando el nuevo vector de medias normalizado como:

$$h_{imj} = \frac{m_{imj}}{s_i} \qquad [3.20]$$

y sustituyendo, se obtiene:

$$P(S \mid q_j) = \left(\prod_{i=1}^{LV} \frac{1}{\sqrt{2p}\, s_i} \right) \cdot \sum_{m=1}^{NM} \left[c_{mj} \cdot \exp\left(-\frac{1}{2} d(S, h_{mj}) \right) \right] \qquad [3.21]$$

Como se ha mencionado anteriormente, en lugar de calcular el producto de este tipo de probabilidades, se calcula su logaritmo natural cambiándole el signo:

$$-\log\left(P(S \mid q_j)\right) = -\log\left(\prod_{i=1}^{LV} \frac{1}{\sqrt{2p}\, s_i} \right) - \log\left(\sum_{m=1}^{NM} \left[c_{mj} \cdot \exp\left(-d(S, h_{mj}) \right) \right] \right) \qquad [3.22]$$

Puede verse que existe un término que es constante y otro que es la suma de exponenciales ponderadas según la expresión [3.17]. Aplicando el selector de gaussianas híbrido, se puede sustituir la suma de exponenciales por la que junto al peso sea mejor, sin que esto deba afectar de forma importante al error del reconocedor. De hecho, en las pruebas realizadas se ha demostrado que al calcular la log-verosimilitud sólo se modifica el tercer dígito. Teniendo en cuenta dicha reducción se puede reescribir, eliminando el correspondiente logaritmo de una exponencial, como la siguiente expresión más simple:

$$\log\left(P(S \mid q_j)\right) = K + \max_{m \in NM} \left(\log c_{mj} - \frac{1}{2} d(S, h_{mj}) \right) \cong \max_{m \in NM} \left(\log c_{mj} - \frac{1}{2} d(S, h_{mj}) \right) \qquad [3.23]$$

Para calcular el valor de las probabilidades basta con calcular el vector de características normalizado por el vector de pesos de ponderación. Luego se calcula la distancia de este vector a todos los vectores de medias normalizadas de la gaussianas

y, finalmente, se le suma el peso correspondiente de la mezcla asociada, eligiéndose el valor máximo en este caso.

3.5.2.2 HGS Aplicado a Modelos Semicontinuos de Markov.

El HGS también puede aplicarse a modelos semicontinuos que incluso estén formados por varios codebooks. En la siguiente expresión se indica cómo es la probabilidad en este caso:

$$P(S \mid q_j) = \prod_{k=1}^{NCB}\left[\sum_{m=1}^{NM_k} c_{mjk} \prod_{i=1}^{LV_k}\left[\frac{1}{\sqrt{2ps_i}}\exp\left(-\frac{1}{2}\left(\frac{s_i - m_{imjk}}{s_i}\right)^2\right)\right]\right] \qquad [3.24]$$

En esta expresión, que es muy similar a la del caso continuo, lo único nuevo que aparece es que se tiene un conjunto de codebooks que agrupan a ciertas componentes del vector, siendo NCB el número de c*odebooks*, NM_k el número de mezclas por codebook y LV_k el número de componentes por codebook.

Haciendo el mismo desarrollo que anteriormente, con las mismas simplificaciones, se llega a una expresión muy parecida a la [3.23]:

$$\log\left(P(S \mid q_j)\right) = K + \sum_{k=1}^{NCB}\left[\max_{m \in NM}\left(\log c_{mjk} - \frac{1}{2}d\left(S, h_{mjk}\right)\right)\right] \cong \sum_{k=1}^{NCB}\left[\max_{m \in NM}\left(\log c_{mjk} - \frac{1}{2}d\left(S, h_{mjk}\right)\right)\right] \quad [3.25]$$

En esta expresión, usando unos modelos semicontinuos con un único codebook, se tendría la misma expresión que para modelos continuos, pero con la ventaja de que los vectores de medias son comunes a todos los modelos, por lo que el número de distancias a calcular puede ser muy reducido.

3.5.3 Estructura Computacional de las Probabilidades.

Aunque el objetivo fundamental de la presente tesis es el diseño de reconocedores de voz robustos, hay que tener también en cuenta los aspectos de eficiencia. En este apartado se muestra cómo, analizando detenidamente el cálculo de probabilidades, es posible conseguir unos procesados individuales que permiten simplificar el funcionamiento del reconocedor, ahorrando memoria y, sobre todo, carga computacional.

Realizando una reestructuración del cómputo de las verosimilitudes se puede conseguir que el reconocedor de voz funcione independientemente de que los modelos que se entrenen sean continuos o semicontinuos, fonéticos o subfonéticos, o incluso modelos de palabras completas. El resultado es una división funcional en los siguientes tres bloques:

- ***Distancia a las medias de las gaussianas***: En este bloque de procesado se calculan las distancias entre el vector de características y los vectores de medias de todas las gaussianas de los modelos activos. La diferencia entre

usar unos modelos continuos o semicontinuos reside en el hecho de que en el primer caso cada gaussiana es utilizada por un único modelo, mientras que si son semicontinuos se comparten entre varios.

- ***Distancia a los modelos de estados activos***: A partir de las distancias a las medias de las gaussianas, y teniendo en cuenta los conjuntos de pesos asociados con cada modelo de estado, se elige como valor de verosimilitud aquel que teniendo en cuenta las dos componentes a la vez proporcione un valor mayor. En el apartado 3.5.2.1 se explica para el caso de la utilización de modelos continuos y en el 3.5.2.2 para modelos semicontinuos.
- ***Distancia a los modelos de palabras***: El tercer bloque es el del cálculo de las distancias a los modelos de estado de las palabras para ser usados por el algoritmo de reconocimiento. Cada modelo de estado de palabra se corresponde con alguno de los modelos de estado, ya sean fonéticos, subfonéticos u otros, cuya distancia ya se ha calculado en el nivel anterior. Por eso, este bloque no se corresponde con el cálculo de distancias a los modelos propiamente, sino a la lectura de las mismas en el lugar donde éstas se han calculado.

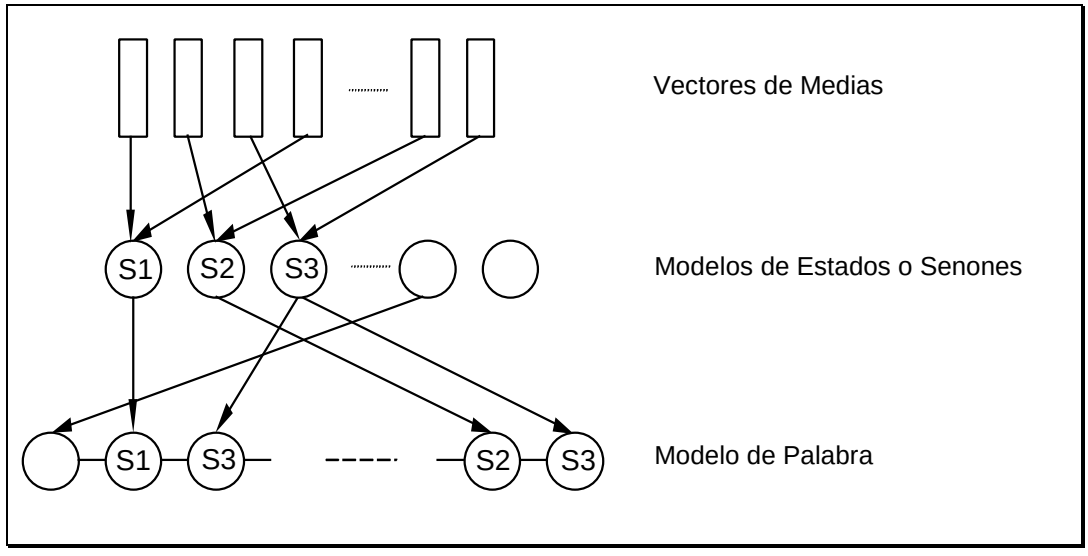

Figura 3-14. Estructura de cálculo de las verosimilitudes de los estados de modelo de una palabra.

La Figura 3-14 muestra la estructura de cálculo de la distancia de los vectores de características a los modelos de estados. El hecho de que los modelos sean continuos o semicontinuos sólo significa que existe un menor número de gaussianas, puesto que éstas se utilizan entre varios modelos. En este caso, será más eficiente para dos vocabularios con el mismo número de modelos de estado y el mismo número de mezclas por modelos, pues si se trata de modelos semicontinuos habrá que calcular un menor número de distancias.

La diferencia entre utilizar modelos subfonéticos o no es la misma que en el caso de ser semicontinuos o continuos. Si no existe compartición, el número de distancias a

modelos de estado es mayor. Si por el contrario, estos estados se pueden compartir entre varios modelos, como en el caso de los senones, se disminuye la memoria y el tiempo necesario para procesar.

Con esta estructura de cálculo es posible utilizar en el reconocedor modelos de palabras completas, de sílabas, fonéticos ó subfonéticos, pero además es posible hacerlo con varios tipos a la vez. Por ejemplo, se puede tener un vocabulario formado por nombres de ciudades y los dígitos, utilizando modelos fonéticos en el primer caso y, en el segundo, modelos de palabras aisladas.

Con la estructura planteada según la Figura 3-14, la distancia a un modelo de estado que sea compartido por dos palabras se realiza una única vez y, por tanto, la carga computacional no depende del número de palabras, sino del número de modelos de estado diferentes que cada aplicación utilice. Puede ocurrir que dos aplicaciones, una que utilice un vocabulario con el doble de palabras que la otra, presenten la misma carga computacional en el cálculo de las distancias a los modelos de palabras si ambos vocabularios representan un número de modelos de estado similar.

La reestructuración del cómputo de las verosimilitudes asociadas con el algoritmo de reconocimiento proporciona una serie de ventajas que se pueden resumir de la siguiente manera:

- Permite concebir el reconocedor de voz como la llamada a una secuencia de bloques independientes especializados en un procesado en particular. Hay que distinguir: el cálculo de la extracción de características, las distancias a las medias de las gaussianas, las distancias a los modelos de estado y el algoritmo de reconocimiento. Esta división en bloques separados permite distribuir la carga computacional entre elementos de procesado separados. Por ejemplo, entre un *PC* y una tarjeta de procesado digital se puede balancear la carga computacional en función de las características que cada uno de estos equipos presente.
- Con la forma de realizar los cálculos, diferenciando distancias a gaussianas, a modelos de estado y a palabras, al reconocedor le es indiferente si los modelos con los que trabaja son continuos o semicontinuos, fonéticos o subfonéticos, o incluso de palabras aisladas. Esta información está implícita en los ficheros con los modelos. Únicamente el programa que genera los ficheros de configuración con la información de los modelos es el que debe discriminar.
- La carga computacional de las distancias a los modelos de palabras no depende del tamaño del vocabulario, sino de la variabilidad de unidades fonéticas o subfonéticas que éste presente, puesto que es el número de modelos de estado diferentes que existe en el vocabulario lo que determina dicha carga.

3.6 Algoritmos de Reconocimiento

3.6.1 Introducción.

En este apartado se aborda el diseño del algoritmo de reconocimiento, que es el encargado de la asignación de tramas a estados y que decodifica la secuencia de palabras con mayor probabilidad.

Existen varias técnicas y estructuras de red de reconocimiento pensadas para la decodificación de una palabra o secuencia de palabras en una red de reconocimiento de dos niveles. Los primeros trabajos en este campo aparecen con Vinstyuk en 1971 [Vin71], que proponía la utilización de la programación dinámica para la extracción de la mejor secuencia de palabras. En [Whi81] puede verse formalmente la interrelación entre la programación dinámica y el algoritmo de Viterbi.

En el campo de algoritmos de reconocimiento de palabras concatenadas se distinguen tres alternativas diferentes:

- *Programación dinámica en dos niveles* (*the two-level dynamic programming*): La obtención de la secuencia de palabras se realiza en dos fases. La primera a nivel acústico, extrayendo información sobre alineación de las palabras del vocabulario. La segunda, a partir de la información acústica, genera la secuencia óptima de palabras. Fue inicialmente propuesta por Sakoe [Sak79].
- El algoritmo *Level-Building* de Mayers y Rabiner [Mye81]: Es un algoritmo similar al anterior, pero que se realiza de forma iterativa con la trama.
- El algoritmo *One-Stage Aproach* o *de una etapa* (también denominado *One-Pass* o *FSLB-Frame-Synchronous Level Building*): La decodificación de los dos niveles se realiza a la vez, simplificando la cantidad de información que hay que almacenar y el número de alternativas posibles [Lee88].

Los tres algoritmos, para una misma red de reconocimiento y unos mismos modelos, obtendrían la misma secuencia de palabras, puesto que son equivalentes. Sin embargo, la cantidad de cómputo necesario no es igual. En este sentido, los algoritmos de una etapa son los más eficientes.

Para tener un reconocedor de voz que pueda ser implementado y usado en aplicaciones que funcionen en tiempo real, es necesario disponer de un algoritmo de decodificación síncrono, al que se le impone cualquier estructura de red y cualesquiera restricciones gramaticales, para que pueda realizar la decodificación tanto a nivel de dentro de palabra como a nivel de frase en una única fase y, además, pueda funcionar trama a trama.

Como ya se mencionó a la hora de hablar del entrenamiento de los modelos en el apartado 3.3, el algoritmo puede ser usado tanto en entrenamiento como en reconocimiento. Las principales diferencias son:

- En caso de ser usado en entrenamiento, únicamente se crea una red con la secuencia de modelos esperados, mientras que en reconocimiento se proporciona un conjunto de alternativas para seleccionar la más probable.
- En entrenamiento no interesa saber qué palabra ha sido reconocida, puesto que se impone, sino la segmentación óptima en estados.
- En reconocimiento es suficiente conocer la palabra o secuencia de palabras reconocidas y sus duraciones (para poder realizar la búsqueda hacia atrás o *backtracking*). En entrenamiento, además, hay que conocer cómo han sido asignadas las diferentes tramas entre los diferentes estados y así poder reestimar los nuevos modelos.

3.6.2 Algoritmo Síncrono de Búsqueda a través de la Red.

El algoritmo de reconocimiento de voz utilizado es el descrito en [Lee88] y ampliado más tarde en [Lee89]. Fue propuesto por Lee y Rabiner, es del tipo *One-Stage Approach* y puede ser implementado síncronamente con la extracción de características. Es un algoritmo para reconocimiento de palabras conectadas muy flexible, al que incluso se le puede incorporar costes de transición de palabras y costes por duraciones de palabra. Recoge todo lo mejor de diferentes técnicas que habían sido propuestas hasta la fecha.

Utiliza estructuras de Markov de primer orden. Sin embargo, su extrapolación a saltos dobles puede derivarse de una manera bastante directa sólo teniendo en cuenta pequeños detalles sobre la forma en que las palabras confluyen y emergen en los nodos. Este algoritmo tiene la ventaja de que con él es fácil realizar una extracción del segundo o posteriores mejores caminos dentro de la red, con un incremento no excesivamente elevado en el cómputo.

Este apartado se desglosa de la siguiente manera: en el subapartado 3.6.2.1 se analizan los dos niveles de la red de reconocimiento; en el subapartado 3.6.2.2 se describe el conjunto de módulos de procesado de que consta el algoritmo de reconocimiento síncrono de Viterbi; en el 3.6.2.3 se muestra el desarrollo del algoritmo de búsqueda a través de la red de reconocimiento; y en los siguientes cinco se describe:

- El *algoritmo de decodificación de Viterbi modificado* (*DP1_1*), para el caso de una estructura de Bakis de primer orden, en el subapartado 3.6.2.4.
- En el 3.6.2.5 la extensión del *DP1_1* a un *algoritmo de decodificación de Viterbi modificado de segundo orden* (*DP1_2*).

- El *algoritmo de combinación de caminos en un nodo gramatical* (*DP2*), que utiliza la información a nivel acústico para realizar la decodificación a nivel gramatical, en el subapartado 0.
- El bloque de *postprocesado* (*PP*), donde se puede introducir la información sobre probabilidades y duraciones de palabra, se muestra en el subapartado 3.6.2.7.
- Finalmente, en el 3.6.2.8, la *búsqueda hacia atrás* o *backtracking* (*BT*), que realiza la decodificación de la secuencia de palabras reconocidas.

3.6.2.1 Estructura de la Red de Reconocimiento de Palabras Conectadas.

Se ha demostrado que para reconocer palabras conectadas es suficiente con descomponer la red de reconocimiento en dos niveles denominados de dentro de palabra (*acústica*) y de frase (*gramática*), cuyas propiedades son completamente diferentes:

- *Nivel acústico*: El nivel de dentro de palabra se corresponde con el modelo de palabra como en los descritos en el apartado 3.4. Los nodos del nivel dentro de palabra se corresponden con los estados de Markov y los arcos con las transiciones de estados.
- *Nivel de frase*: Se corresponde con el nivel gramatical, en el que los nodos se asocian con los límites de las palabras y los arcos con los modelos de las mismas. Esta estructura puede ser muy simple o muy complicada dependiendo de la aplicación y de la información gramatical utilizada.

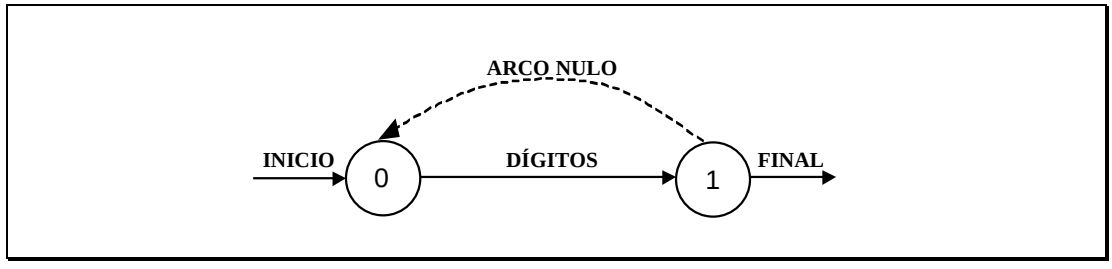

Figura 3-15. Ejemplo de red gramatical para el reconocimiento de dígitos.

En la Figura 3-15 se muestra un ejemplo de red gramatical para el reconocimiento de un número indeterminado de dígitos. El arco que va desde el nodo 0 al nodo 1, que está etiquetado como DIGITOS, en realidad es un conjunto de arcos en paralelo. En este caso, podrían ser los modelos de palabras desde el *cero* hasta el *nueve*. Se observa que existe un punto de entrada (nodo INICIO) y uno de salida (nodo FINAL). La decodificación de la secuencia de palabras se realiza desde los nodos finales hacia los iniciales.

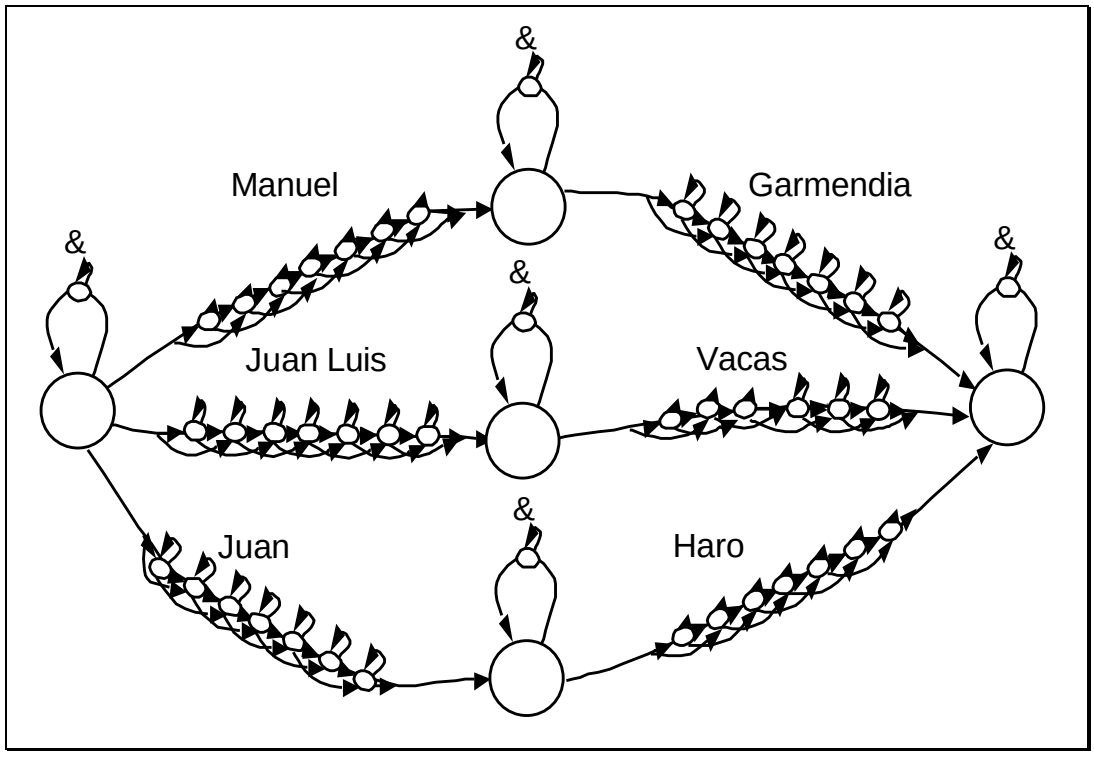

Figura 3-16. Ejemplo de red de reconocimiento donde se observan los dos niveles de que consta la red de reconocimiento.

En la Figura 3-16 se muestra más detenidamente la red de reconocimiento con los dos niveles anteriormente mencionados. El algoritmo utilizado permite combinar y realizar la decodificación de palabras directamente en un paso, evitando tener que esperar hasta el final de la primera fase para realizar la segunda decodificación.

3.6.2.2 Módulos del Algoritmo de Reconocimiento.

En el algoritmo de reconocimiento de palabras conectadas se pueden distinguir seis módulos, cada uno de ellos está perfectamente caracterizado, pudiéndose identificar como tareas independientes interconectadas. Son los siguientes:

- **Extracción de características** (*FE*: *Feature Extraction*): Es el encargado de realizar la extracción de los vectores de características. Para el caso de la presente tesis es la descrita en el apartado 3.2.
- **Probabilidad local** (*LL*: *Local Likelihood*): Es el encargado de la evaluación de la probabilidad de observación de los vectores de características en todos los estados internos de los modelos de palabra. El entrenamiento de modelos continuos y semicontinuos de Markov se define en 3.3, cuyo cálculo de las probabilidades simplificadas se describe en el apartado 3.5.
- **Decodificación dentro de palabra** (*DP1*): Es el algoritmo de reconocimiento encargado de realizar una *decodificación de Viterbi modificada* capaz de incluir penalizaciones por duraciones de palabra. Tal como se ha mencionado anteriormente existen dos variantes, la primera para redes de

primer orden (*DP1_1*), que es la que aparece en [Lee89], la segunda es una extensión de la anterior para redes de segundo orden (*DP1_2*).

- **Decodificación a nivel gramatical o de frase** (*DP2*): Realiza la fusión de la información procedente de los arcos que contienen los diferentes modelos de palabra que llegan a un nodo gramatical, teniendo en cuenta las restricciones sintácticas que se pueden imponer a la red de reconocimiento.
- **Postprocesado** (*PP*: *PostProcessing*): Es el módulo encargado de combinar las penalizaciones por duraciones con las probabilidades de los caminos.
- **Búsqueda hacia atrás** (*BK*: *BackTracking*): Finalmente, este módulo se encarga de la decodificación del conjunto de palabras candidatas óptimas. Este algoritmo tiene en cuenta que pueden existir varios nodos finales, eligiendo el que proporcione la mejor secuencia de palabras.

3.6.2.3 Descripción del Algoritmo de Búsqueda Síncrono.

El conjunto de bloques descritos anteriormente se muestra en la Figura 3-17, donde se observa la disposición de los diferentes módulos del algoritmo y la manera en que se sincroniza el mismo respecto a las tramas:

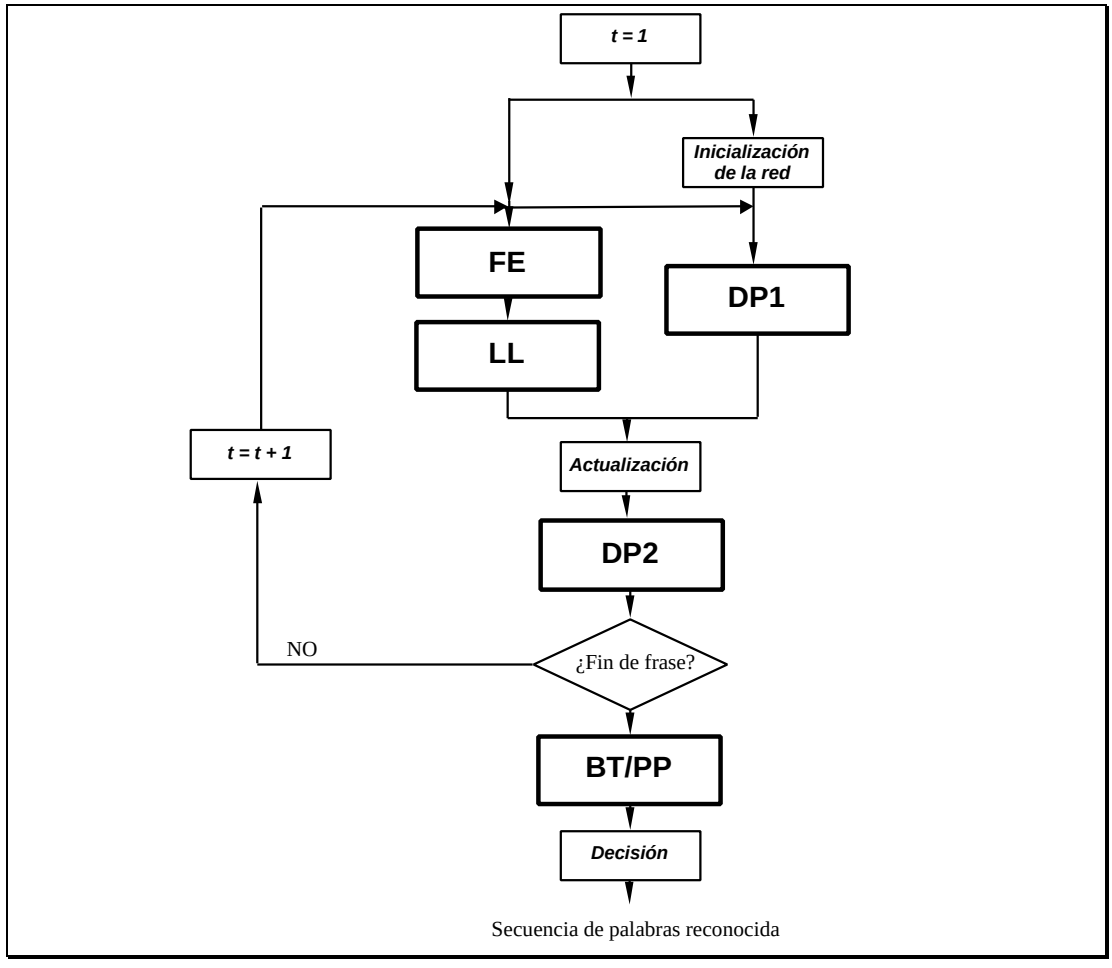

Figura 3-17. Diagrama de bloques del algoritmo síncrono de búsqueda a través de la red.

En la Figura 3-17 aparece cómo se pueden ejecutar en paralelo dos procesados a la vez. Por una parte, la extracción de las características (*FE*) y el cálculo de las probabilidades locales (*PP*), por otra, el algoritmo de Viterbi modificado (*DP1*). El proceso de actualización es el encargado de cambiar las probabilidades acumuladas de los caminos óptimos a todos los nodos internos a partir de las probabilidades locales. La decisión de finalización y extracción de las palabras reconocidas en una aplicación en tiempo real se realiza, en este caso, a través de un detector de extremos que estima que ya ha aparecido el silencio final.

El algoritmo busca la mejor opción que llega a cada nodo de la red, en cualquier tiempo t, y construyendo el camino óptimo de duración t a ese nodo a partir de todos los mejores caminos de duración $(t - 1)$. El principio de la *programación dinámica* (DP: *Dynamic Programing*) establece que el mejor camino a cualquier nodo i, en el tiempo t, se puede determinar a partir de los mejores caminos a todos los nodos j, en el tiempo $(t - 1)$, junto con las mejores transiciones desde el nodo j hasta el nodo i.

Por ello, será necesario almacenar las probabilidades acumuladas a todos los nodos de la red, la información del camino (los nodos de la gramática de origen y destino para cada arco de la gramática en el camino), y las duraciones de estado y palabra para calcular las probabilidades de duración y realizar la búsqueda hacia atrás.

El algoritmo puede describirse de la siguiente manera:

- **Paso 1:** Inicializar toda la información de búsqueda hacia atrás en todos los nodos.
- **Paso 2**: Ejecutar la construcción del camino óptimo trama por trama.

  para cada trama de la señal de voz (mientras no se llegue al final de la frase):
  - Realizar la extracción de características (*FE*).
  - Realizar el cálculo de las probabilidades locales para todos los estados activos en los modelos de palabras (*LL*).

  para cada nodo de la gramática activo en la red (bucle de nodos gramaticales):

  para cada nodo gramatical anterior activo:

  para cada arco de la gramática activa entre ellos (bucle de modelo de palabra):
  - Calcular el algoritmo de Viterbi modificado (*DP1*) dentro de una palabra para cada nodo interno activo (bucle de estados internos).
  - Actualizar las probabilidades acumuladas de los caminos y la información de los caminos.

  fin de cada arco de gramática activa.

  fin de cada nodo gramatical anterior activo.
  - Ejecutar el algoritmo de decodificación a nivel gramatical (*DP2*).
  - Actualizar las probabilidades acumuladas de los caminos y la información de los mismos.
  - Actualizar los buffers acumulados de búsqueda hacia atrás.

fin de cada nodo gramatical activo en la red.

fin de cada trama de la señal de voz.

- **Paso 3**: Ejecutar el postprocesado (*PP*) y la búsqueda hacia atrás (*BK*).

  para cada nodo terminal activo en la red:

  - Ejecutar el postprocesado de duraciones y puntuaciones (*PP*).
  - Ejecutar la búsqueda hacia atrás (*BK*) para identificar la secuencia de palabras reconocidas.

  fin de cada nodo terminal activo.

- **Paso 4**: Determinar la secuencia de palabras reconocidas.

3.6.2.4 Algoritmo de Decodificación de Viterbi de Primer Orden (DP1_1).

El algoritmo consiste en calcular los siguientes tres tipos de probabilidades acumuladas, a la vez que se guarda información sobre las duraciones por estado y de la palabra:

- *inbest* (LG, $t-1$): Representa la probabilidad del mejor camino que llega al nodo de la gramática izquierda (LG: Left Grammar) en el tiempo ($t-1$). Si a este valor se le añade una penalización por transición de palabra, que puede ser determinista o estocástica (como el logaritmo de la probabilidad del modelo del lenguaje), se denomina *like* (0, $t-1$). Se puede escribir como:

$$like\,(0, t-1) = inbest\,(LG, t-1) + penalización\ por\ transición\ de\ palabra$$

- *like* (i, t): Representa la probabilidad acumulada en el estado i-ésimo en el tiempo t. Se calcula a partir de los valores en el tiempo anterior ($t-1$) del mismo estado o del anterior, y a partir de las probabilidades locales. Si existen N estados dentro del modelo de palabra, se debe calcular desde el estado primero hasta el N-ésimo. Su cálculo se puede escribir como:

$$\begin{aligned} like\,(i,t) = max\,[&like\,(i,t-1) + a\,(i,i),\\ &like\,(i-1,t-1) + a\,(i-1,i) + penalización\ por\ duración\ de\ estado] +\\ &+ p\log local\,(i,t) \end{aligned}$$

  donde *ploglocal*(i, t) representa el logaritmo de la probabilidad local del vector de características en el tiempo t, del estado i del modelo de palabra, $a(j, i)$ es el logaritmo de la probabilidad de hacer una transición desde el estado j al estado i, y la *penalización de la duración de estado* es el logaritmo de la probabilidad de haber permanecido en el estado ($i-1$) durante d tramas.

- *outbest* (RG, t): Representa la probabilidad del mejor camino de salida que alcanza el nodo de la gramática derecha (RG: Right Grammar) en el tiempo t. Se calcula como:

$$outbest(RG,t) = like\,(N,t) + penalizaciones\ por\ duraciones\ de\ estado\ y\ palabra$$

El mejor camino desde este arco de palabra se incorpora a los procedentes de otros arcos que llegan al nodo gramatical, y entre todos ellos se elige el mejor camino para ser usado como entrada de los arcos que partan de ese nodo gramatical, y si es un nodo terminal, el camino de la secuencia de palabras en el tiempo *t*.

En resumen, este algoritmo parte de las probabilidades acumuladas en el tiempo $(t-1)$ en todos los estados de una palabra, *like*$(i, t-1)$, y la del nodo de inicio de palabra *inbest*$(LG, t-1)$, y obtiene las probabilidades acumuladas en el tiempo t de todos los estados, *like*(i, t), y del nodo de salida de palabra, *outbest*(RG, t), utilizando adicionalmente las probabilidades locales calculadas en el tiempo t, *ploglocal*(i, t), las probabilidades de transición entre estados, $a(i, j)$, y las penalizaciones por duraciones de estados y palabra.

En la Figura 3-18 puede verse este mismo algoritmo presentado de forma gráfica. Se observa que por el nodo gramatical de la izquierda van entrando las probabilidades procedentes de las palabras que llegan a él. Por otro lado, esta palabra va obteniendo información sobre la probabilidad de recorrerla, a la que se le añaden las penalizaciones por las duraciones del último estado y de la palabra en su conjunto. Sin embargo, todas estas penalizaciones no son obligatorias y pueden utilizarse como si fueran constantes, independientemente de la palabra y de las propias duraciones.

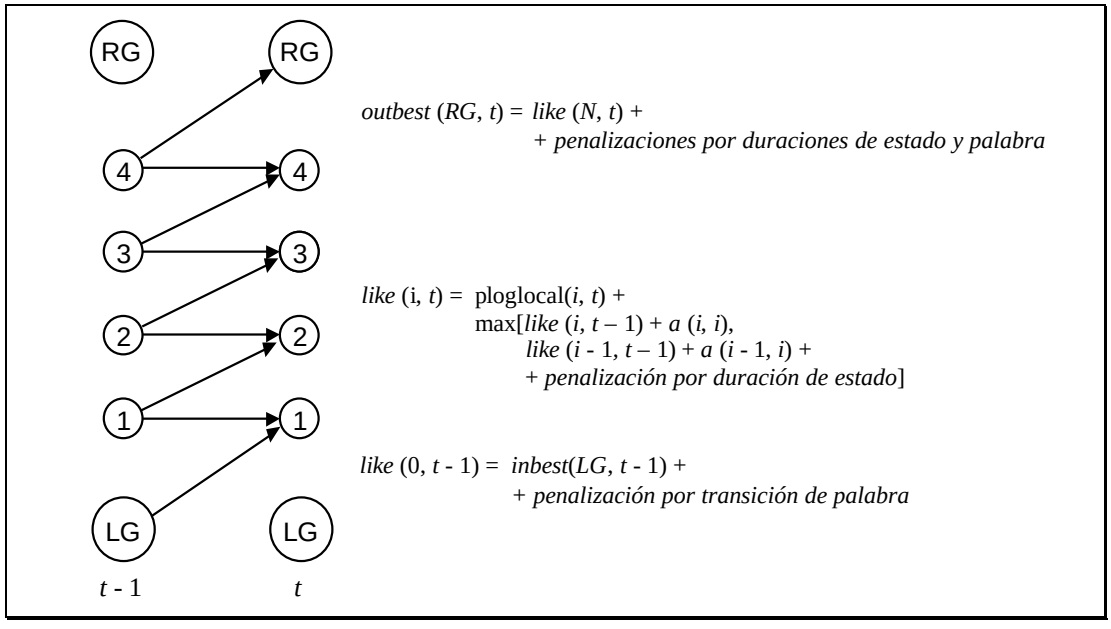

Figura 3-18. Representación esquemática del algoritmo de decodificación de Viterbi modificado para un modelo de palabra con una estructura de Bakis de primer orden.

A continuación, se describe el algoritmo de Viterbi con todos los detalles necesarios para su implementación y obtención de la información útil, como pueden ser las duraciones de estado y palabra. La duración de los estados es necesaria para poder entrenar los modelos, al proporcionar la segmentación de las frases, y la duración de la

palabra permite realizar la búsqueda hacia atrás. Previamente se definen una serie de variables usadas por el algoritmo.

- *like* (*i*): Probabilidad acumulada del mejor camino que llega al nodo *i*.
- *elapse* (*i*): Duración del mejor camino al nodo *i* procedente del nodo gramatical de la izquierda (*LG*).
- *dwell* (*i*, 1:*i*): Tiempos de permanencia de todos los estados hasta el nodo *i* para el mejor camino que llega al nodo *i*.
- *a* (*j*, *i*): Logaritmo de la probabilidad de transición desde el nodo *j* hasta el nodo *i*.
- *inbest* (*LG*): Probabilidad del mejor camino que entra al nodo *LG* procedente de nodos gramaticales previos.
- *outbest* (*RG*): Probabilidad del mejor nodo que alcanza el nodo gramatical derecho (*RG*) en nodos posteriores.
- *ploglocal* (*i*): Probabilidad de observación actual para el nodo *i*.
- *scratch* (*i*): Array auxiliar necesario para almacenar los valores *like* (*i*) de la trama anterior.

El *Algoritmo de decodificación de Viterbi modificado de primer orden* (*DP1_1*) es:

- **Paso 1:** Inicializar:
  - *like* (0) = *inbest* (*LG*) + *penalización por transición de palabra*;
  - *elapse* (0) = 0;
  - *dwell* (0, 0) = 0;
  - *a* (0, 1) = 0;
- **Paso 2**: Calcular el mejor camino que llega al nodo *i*, con $1 \leq i \leq N$, en el tiempo *t*:
  - *scratch* (*i*) = *like* (*i*) + *a* (*i*, *i*);
  - *scratch* (*i* – 1) = *like* (*i* – 1) + *a* (*i* – 1, *i*) + penalización por duración de estado;

  si (*scratch* (*i*) ≥ *scratch* (*i* – 1)) *entonces*
  - *elapse* (*i*) = *elapse* (*i*) + 1;
  - *dwell* (*i*, *i*) = *dwell* (*i*, *i*) + 1;

  si no
  - *scratch*(*i*) = *scratch* (*i* – 1);
  - *elapse* (*i*) = *elapse* (*i* – 1) + 1;
  - *dwell* (*i*, *i*) = 1;
  - *dwell* (*i*, 1:*i* – 1) = *dwell* (*i* – 1, 1:*i* – 1);

  fin de la sentencia condicional.
- **Paso 3**: Actualizar las probabilidades de estado acumuladas y generar las puntuaciones de los caminos de salida en el tiempo *t*:

- *like* (1:*N*) = *scratch* (1:*N*) + *ploglocal* (1:*N*);
- *outbest* (*RG*) = *like* (*N*) + *penalizaciones por duraciones de estado y palabra*;

3.6.2.5 Algoritmo de Decodificación de Viterbi de Segundo Orden (DP1_2).

La introducción de saltos dobles hace que el algoritmo tenga que ser modificado. Sin embargo, esto no supone un gran cambio respecto al algoritmo de saltos simples en lo que se refiere a la forma en que el mismo se desarrolla, pero sí en que hay que tener en cuenta otros casos, y en que hay cuatro fórmulas para calcular las probabilidades acumuladas frente a las tres anteriores. Éstas son las siguientes:

- *inbest* (*LG*, *t* – 1): En este caso sigue significando lo mismo, y no hay ninguna modificación. Por tanto,

$$like\,(0,t-1)=inbest\,(LG,t-1)+\,penalización\ por\ transición\ de\ palabra$$

- *like* (*i*, *t*): Esta vez hay que tener en cuenta que se permite saltar al estado *i* desde el estado *i* – 2. Sin embargo, no ocurre cuando *i* es igual a 1, que utiliza las expresiones de saltos simples. La fórmula para el primer estado se escribe como:

$$\begin{aligned} like\,(1,t) = max[&like\,(1,t-1)+a\,(1,1),\\ &like\,(0,t-1)+a\,(0,1)+\,penalización\ por\ duración\ de\ estado]+\\ &+p\log local\,(1,t)\end{aligned}$$

  Para el caso general habría que utilizar la fórmula:

$$\begin{aligned} like\,(i,t) = max[&like\,(i,t-1)+a\,(i,i),\\ &like\,(i-1,t-1)+a\,(i-1,i)+\,penalización\ por\ duración\ de\ estado,\\ &like\,(i-2,t-1)+a\,(i-2,i)+\,penalización\ por\ duración\ de\ estado]+\\ &+p\log local\,(i,t)\end{aligned}$$

- *outbest* (*RG*, *t*): Hay que tener en cuenta que existen dos estados del arco que pueden llegar al nodo gramatical izquierdo y habrá que seleccionar el mejor:

$$\begin{aligned} outbest(RG,t) = max[&like\,(N,t)+\,penalizaciones\ por\ duraciones\ de\ estado\ y\ palabra,\\ &like\,(N-1,t)+\,penalizaciones\ por\ duraciones\ de\ estado\ y\ palabra]\end{aligned}$$

En la Figura 3-19 se muestra el algoritmo de Viterbi para saltos dobles. Ahora el algoritmo se complica ligeramente al poder saltar directamente dos estados del modelo de palabra, y a la hora de obtener la probabilidad acumulada de salida existen dos posibilidades de llegar.

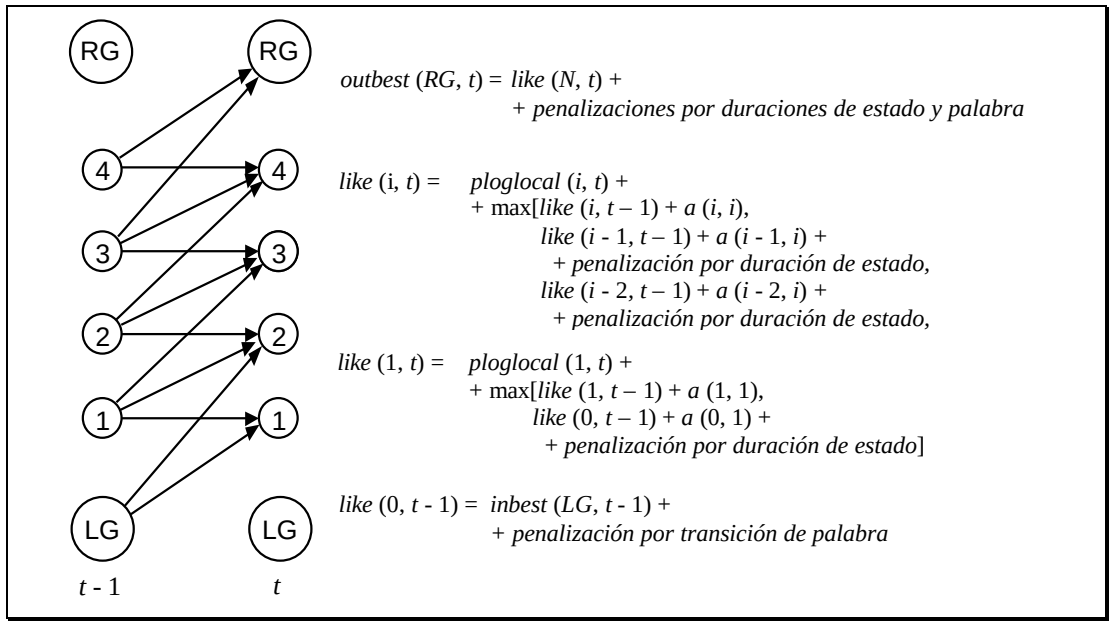

Figura 3-19. Representación esquemática del algoritmo de decodificación de Viterbi modificado para un modelo de palabra con una estructura de Bakis de segundo orden.

El *Algoritmo de Decodificación de Viterbi Modificado de Segundo Orden* (*DP1_2*) es muy similar al *DP1_1*, únicamente hay que tener cuidado con en cálculo de las probabilidades acumuladas para el primer estado del modelo de palabra, que es igual que en el *DP1_1*. El algoritmo queda de la siguiente manera:

- **Paso 1:** Inicializar:
  - *like* (0) = *inbest* (*LG*) + *penalización por transición de palabra*;
  - *elapse* (0) = 0;
  - *dwell* (0, 0) = 0;
  - *a* (0, 1) = 0;
  - *a* (0, 2) = 0;
- **Paso 2**: Calcular el mejor camino que llega al nodo *i*, con $1 \leq i \leq N$, en el tiempo *t*. Para el caso de *i* = 1, la descripción es igual que en el **Paso 2** del algoritmo *DP1_1*. Para el resto de los casos, $2 \leq i \leq N$, las instrucciones son las siguientes:
  - *scratch* (*i*) = *like* (*i*) + *a* (*i*, *i*);
  - *scratch* (*i* – 1) = *like* (*i* - 1) + *a* (*i* – 1, *i*) + penalización por duración de estado;
  - *scratch* (*i* – 2) = *like* (*i* - 1) + *a* (*i* – 2, *i*) + penalización por duración de estado;

  si ((*scratch* (*i*) ≥ *scratch* (*i* – 1)) & (*scratch* (*i*) ≥ *scratch* (*i* – 2))) *entonces*
  - *elapse* (*i*) = *elapse* (*i*) + 1;
  - *dwell* (*i*, *i*) = *dwell* (*i*, *i*) + 1;

  si (*scratch* (*i*) ≥ *scratch* (*i* – 1)) *entonces*
  - *scratch*(*i*) = *scratch* (*i* – 1);
  - *elapse* (*i*) = *elapse* (*i* – 1) + 1;

- *dwell* (*i*, *i*) = 1;
- *dwell* (*i*, 1:*i* – 1) = *dwell* (*i* – 1, 1:*i* – 1);

si no

- *scratch*(*i*) = *scratch* (*i* – 2);
- *elapse* (*i*) = *elapse* (*i* – 2) + 1;
- *dwell* (*i*, *i*) = 1;
- *dwell* (*i*, *i* – 1) = 0;
- *dwell* (*i*, 1:*i* – 2) = *dwell* (*i* – 1, 1:*i* – 2);

fin de la sentencia condicional.

- **Paso 3**: Actualizar las probabilidades de estado acumuladas y generar las puntuaciones de los caminos de salida en el tiempo *t*:
  - *like* (1:*N*) = *scratch* (1:*N*) + *ploglocal* (1:*N*);

  si (*like* (*N*) ≥ *like* (*N* – 1)*) entonces*
  - *outbest* (*RG*) = *like* (*N*) + *penalizaciones por duraciones de estado y palabra*;

  si no
  - *outbest* (*RG*) = *like* (*N* – 1) + *penalizaciones por duraciones de estado y palabra;*

3.6.2.6 Algoritmo de Combinación de Caminos en un Nodo Gramatical (DP2).

Este algoritmo es el encargado de recoger la información de todos los arcos gramaticales que, procedentes de otros nodos, llegan a éste. El algoritmo debe seleccionar entre todos estos caminos cuál es el mejor y guardar toda la información necesaria para realizar la búsqueda hacia atrás una vez que se haya detectado el fin de frase. Asimismo, usando los tiempos de duración de palabra y de permanencia en cada estado, es el encargado de calcular las penalizaciones por duraciones que pueden ser usadas por el módulo de postprocesado (*PP*).

Los elementos necesarios para realizar la implementación del algoritmo son:

- *P* (*g*): Conjunto de todos los arcos que llegan al nodo $g = \{k|RG(k) = g\}$.
- *glike* (*g*, *t*): Probabilidad acumulada del mejor camino que llega al nodo *g* en el tiempo *t*.
- *word* (*g*, *t*): Identificación del mejor arco de modelo de palabra del mejor camino que llega al nodo *g* en el tiempo *t*.
- *bp* (*g*, *t*): Duración del mejor arco de modelo de palabra del mejor camino que llega al nodo *g* en el tiempo *t*.

- *bpnode* (*g*, *t*): Identificación del nodo gramatical de inicio del mejor arco que llega al nodo *g* en el tiempo *t*.
- *outbest* (*g*, *k*): Probabilidad del mejor camino que alcanza el nodo *g* procedente del arco *k*.

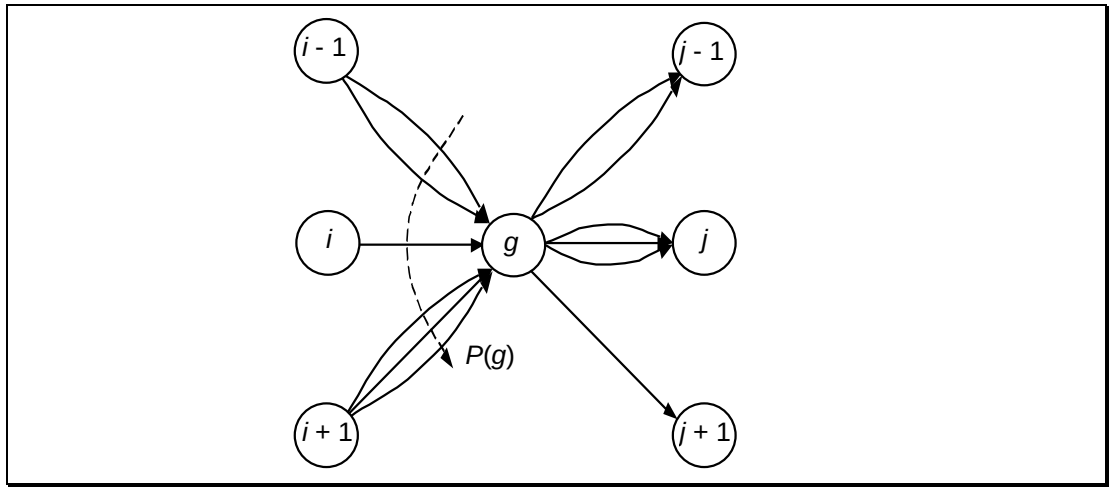

Figura 3-20. Esquema de funcionamiento del algoritmo de fusión de caminos en un nodo gramatical.

En la Figura 3-20 se muestra un esquema de cómo funciona el algoritmo. Éste captura información de arcos conectados a nodos anteriores y la envía a arcos que tienen otros nodos destinos. El algoritmo consta de tres pasos y su descripción es la siguiente:

- **Paso 1:** Realizar la fusión de todos los caminos que llegan al nodo gramatical *g* en el tiempo *t*:
  - $glike(g,t) = \max_{k \in P(g)} [outbest(g,k)]$;
  - *inbest* (*g*) = *glike* (*g*, *t*);
- **Paso 2**: Salvar la información de búsqueda hacia atrás del mejor camino que llega al nodo gramatical *g* en el tiempo *t*:
  - $word(g,t) = \arg\max_{k \in P(g)} [outbest(g,k)]$;
  - *bp* (*g*, *t*) = *elapse*[*word* (*g*, *t*)];
  - *bpnode* (*g*, *t*) = *LG*[*word* (*g*, *t*)];
- **Paso 3**: Usar los tiempos de permanencia y la duración y calcular las penalizaciones por duración para ser usadas por el módulo de postprocesado (*PP*).

3.6.2.7 Postprocesado (PP).

Una vez realizada la búsqueda hacia adelante, habrá una serie de candidatos posibles, uno por cada nodo terminal. El *postprocesado* permite poder añadir más información antes de elegir la secuencia de palabras más probable. Suele ser información referida a penalizaciones por probabilidades de duración de estados. Los algoritmos de decodificación dentro de palabra, *DP1_1* y *DP1_2*, pueden incorporar dicha información, tal como se mostró en su descripción, así este módulo puede simplificarse y requerir un cómputo muy bajo.

3.6.2.8 Búsqueda Hacia Atrás ó Backtracking (BT).

La búsqueda hacia atrás o *backtracking* permite recorrer desde un nodo terminal cualquiera, y a partir de la información que se ha ido almacenando en los nodos, la secuencia de palabras, saltando de nodo en nodo y restando al tiempo acumulado lo que cada una de ellas dura hasta llegar al tiempo cero o estado inicial.

El algoritmo puede describirse de la siguiente manera:

- **Inicializar**: $i = g$, $j = t$, $k = 0$.

mientras ($j <> 0$)

- $k = k + 1$;
- *iword* (k) = *word* (i, j);
- *idur* (k) = *bp* (i, j);
- i = *bpnode* (i, j);
- $j = j$ - *idur* (i, j);

fin del bucle.

Una vez aplicado el algoritmo se obtiene un total de k palabras concatenadas, que están almacenadas en *iword* y cuyas duraciones están en *idur*. Puede ocurrir que no sea necesario conocer qué palabras son, pero sí sus duraciones, por tanto, es posible modificar la información que se puede almacenar. Por ejemplo, sería interesante almacenar la información asociada con las permanencias en los estados para el entrenamiento de modelos. Esta información servirá para realizar la segmentación de las frases.

3.6.3 Poda de Caminos. Beam Search.

La aplicación directa del algoritmo anteriormente descrito es muy poco eficiente cuando el tamaño del vocabulario es de varias decenas de palabras. Es necesario introducir algún mecanismo de búsqueda del mejor camino (*beam search*) para reducir

el cómputo necesario. En [Dell93] se hace un análisis más detallado de esta técnica que se utilizó por primera vez en el *DTW*.

Todos los caminos cuyo coste caiga por debajo de un *factor* $d(t)$ al que se le suma la mejor hipótesis en la trama anterior no avanzan. Si se tiene (t, i_t^*) como el conjunto de todos los puntos activos en la red de reconocimiento en el tiempo t, el camino a cualquier (t, i_t) sólo será candidato para crecer en el tiempo $t+1$ si se cumple:

$$D_{min}(t,i_t) \leq D_{min}(t,i_t^*) + d(t) \qquad [3.26]$$

3.7 Estructura de Reconocimiento en Aplicaciones Telefónicas

3.7.1 Introducción.

El diseño del nivel gramatical es fundamental para el correcto funcionamiento de los reconocedores de voz. La red con la información gramatical debe ser lo suficientemente flexible para reconocer de manera robusta cualquier pronunciación, y a la vez, lo suficientemente sencilla para que la aplicación que la use resulte ser muy simple.

En este apartado se describen diferentes redes de reconocimiento que pueden aparecer en aplicaciones telefónicas.

3.7.2 Telefonista Automática.

Aunque en un reconocedor de voz el nivel gramatical podría ser mucho más complicado, de manera que éste funcionara como uno de habla continua, en la mayoría de los casos es suficiente que sea de palabras conectadas, por ejemplo, para detectar una secuencia de números o el nombre de alguna persona, lugar o cosa de entre un conjunto de posibilidades.

Una aplicación de telefonista automática se basa en el establecimiento de un diálogo entre un usuario y el sistema. Este diálogo se establece de manera que el sistema guíe en todo momento al usuario. El diseño del diálogo entra dentro del campo de la comunicación y el lenguaje, y se encuentra fuera de los objetivos de la presente tesis. Sin embargo, se pueden mencionar las características que todo sistema conversacional debería poseer:

- El número de preguntas a realizar por el sistema debe ser lo más reducido posible.
- El sistema de diálogo debe ser lo suficientemente robusto como para permitir que durante el transcurso del mismo se corrijan posibles errores de reconocimiento.

- La conversación debe ser lo más natural posible para que el locutor se sienta cómodo.

A continuación, se enumeran los tipos de reconocimiento más normales que utilizan las aplicaciones telefónicas actuales:

- ***Sí y no*****:** La discriminación entre estas dos palabras es típica en cualquier aplicación telefónica. Se suele utilizar para la confirmación de una respuesta anterior o para guiar al usuario a través del diálogo. Debido a su brevedad, los errores de reconocimiento suelen ser importantes. Para mejorar la robustez de reconocimiento de este tipo de palabras el apartado 4.4 describe diferentes técnicas.
- ***Dígitos*****:** Una de las primeras aplicaciones del reconocimiento de voz fue la de los dígitos. Al igual que en el caso de la discriminación entre *sí* y *no*, presentan problemas por la brevedad de las mismas. Tradicionalmente, se han utilizados modelos entrenados de forma completa para el reconocimiento de estas palabras.
- ***Secuencia de dígitos*****:** Del reconocimiento de dígitos se pasa al de secuencias de dígitos en aplicaciones como la pronunciación de un número de teléfono o el del Documento Nacional de Identidad. Este tipo de reconocimiento resulta algo más complejo que los anteriores por la dificultad añadida de que el número de palabras que se puede esperar es variable.
- ***Nombres de personas, lugares o cosas*****:** La selección entre un conjunto de nombres de personas, lugares o cosas resulta mucho más natural que la pronunciación de dígitos. Una aplicación que utiliza vocabularios con listas como las anteriores resulta más sencilla que otra en la que haya que esperar a que el sistema enumere un conjunto de posibilidades y el usuario tenga que pronunciar el número asociado con la opción elegida. En este caso, es fundamental que el reconocedor sea flexible en la utilización de diferentes vocabularios.

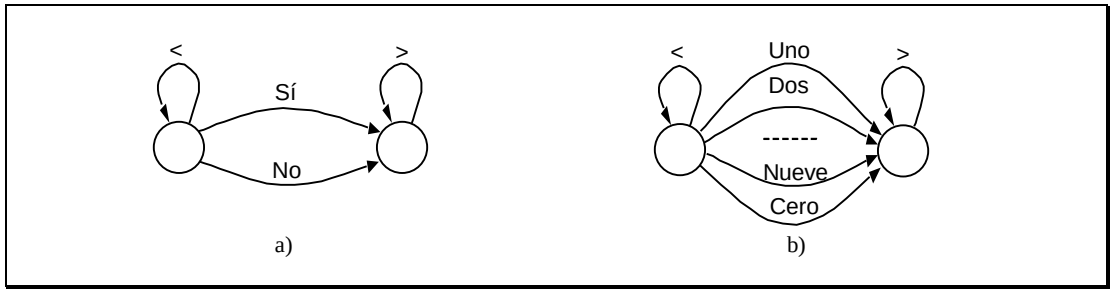

Figura 3-21. Ejemplos de redes de reconocimiento para el caso de a) reconocimiento entre las palabras *sí* y *no*, y b) reconocimiento de dígitos aislados.

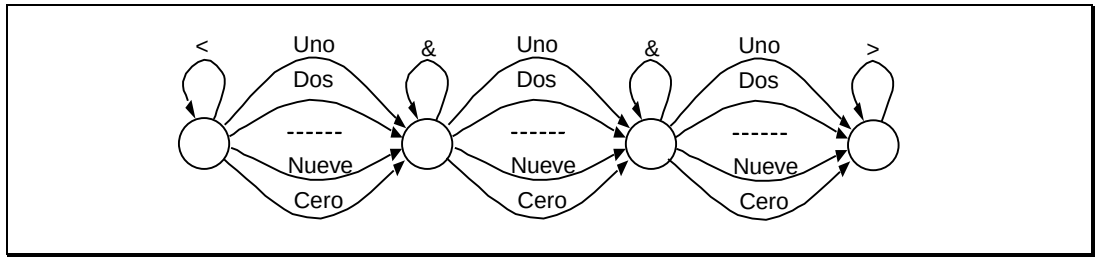

Figura 3-22. Ejemplo de red para el reconocimiento de secuencias de tres dígitos. Se observa que en cada uno de los nodos gramaticales se localizan arcos asociados con modelos de silencio. La utilización de los tres últimos nodos gramaticales como finales permite el reconocimiento también de uno o dos dígitos.

La apariencia de una red gramatical es similar a la de un modelo de Markov a nivel acústico, pero existe una diferencia respecto a éste, los modelos de palabras se localizan en los arcos, o lo que es lo mismo, son de emisión en la transición también denominada forma de Mealy, frente a los de emisión en los estados o forma de Moore, como ya se indicaba en el apartado 2.3.1.1.

En todas estas estructuras hay al menos un nodo origen o inicial, y un nodo terminal o final. Otro característica de estas redes es que todos los nodos poseen arcos con modelos de silencio de antes o después de frase (< y >) o de entre palabras (*&*).

En la Figura 3-21 se muestran dos de los casos anteriormente mencionados: en a) el reconocimiento entre *sí* y *no*, y en b) el reconocimiento de dígitos. En la Figura 3-22 se muestra una red gramatical para el reconocimiento de una secuencia de hasta tres dígitos, pueden utilizarse tres de los últimos nodos como finales para también detectar secuencias de sólo uno o dos dígitos.

3.7.2.1 Red de Reconocimiento de Nombres.

En el reconocimiento de dígitos o palabras como *sí* y *no*, el usuario suele responder directamente con la opción elegida. En el caso de que tenga que pronunciar entre diferentes nombres de personas, lugares o cosas el usuario tiende a responder añadiendo palabras antes y después, como por ejemplo, *póngame con*, *por favor*, *don*, *doña*, etc. Para que el reconocedor de voz funcione adecuadamente con estas palabras adicionales es necesario introducir estos modelos, denominados *comodín*, en la red y pueden utilizarse tanto antes como después de las palabras a reconocer.

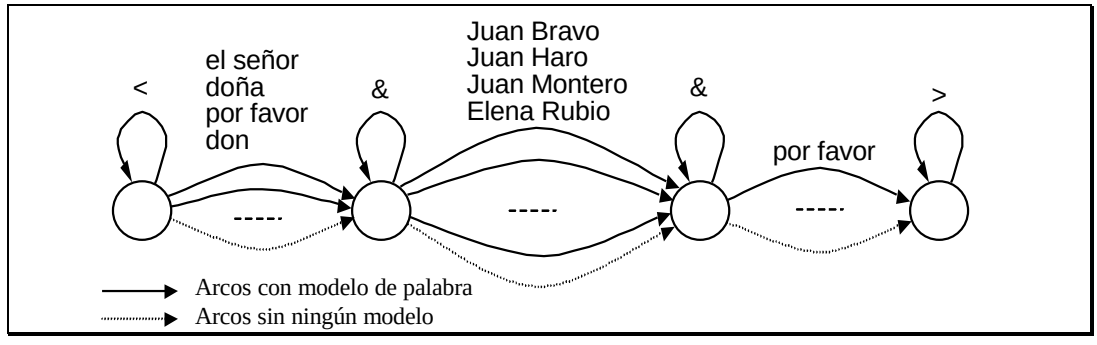

Figura 3-23. Estructura gramatical para el reconocimiento de nombres y apellidos a través del canal telefónico.

En la Figura 3-23 se muestra la estructura de una red de reconocimiento de nombres y apellidos, que por ser genérica puede ser utilizada con cualquier otro vocabulario.

El uso de un *arco nulo*, que sirve para conectar nodos de la red directamente sin ningún modelo de palabra, permite al reconocedor saltarse directamente los modelos comodín.

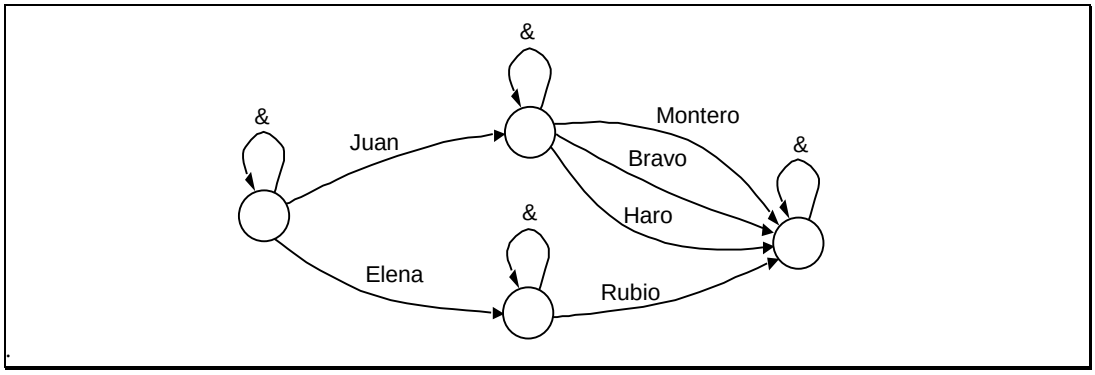

Figura 3-24. Detalle de la estructura de red de los nombres y apellidos. Al realizar esta división es posible compartir arcos de modelos de nombres aumentando la eficiencia, al tener que recorrer menos arcos durante el algoritmo de reconocimiento.

Hay que tener en cuenta dos aspectos que se muestran en la Figura 3-24:

- La introducción de un modelo de silencio entre nombre y apellido permite modelar la pausa que pudiera existir en la pronunciación del mismo. Para ello, basta con dividir el modelo de nombre y apellido en dos partes, asociar cada uno con un arco gramatical diferente e introducir entre los dos nuevos modelos otro nodo gramatical que tenga un arco de silencio intermedio.
- La segunda es una simplificación que se obtiene a partir de la anterior. Efectivamente, si se tienen dos arcos asociados con el mismo modelo de nombre, pueden agruparse en uno sólo y ser utilizados por varios modelos de apellidos a la vez.

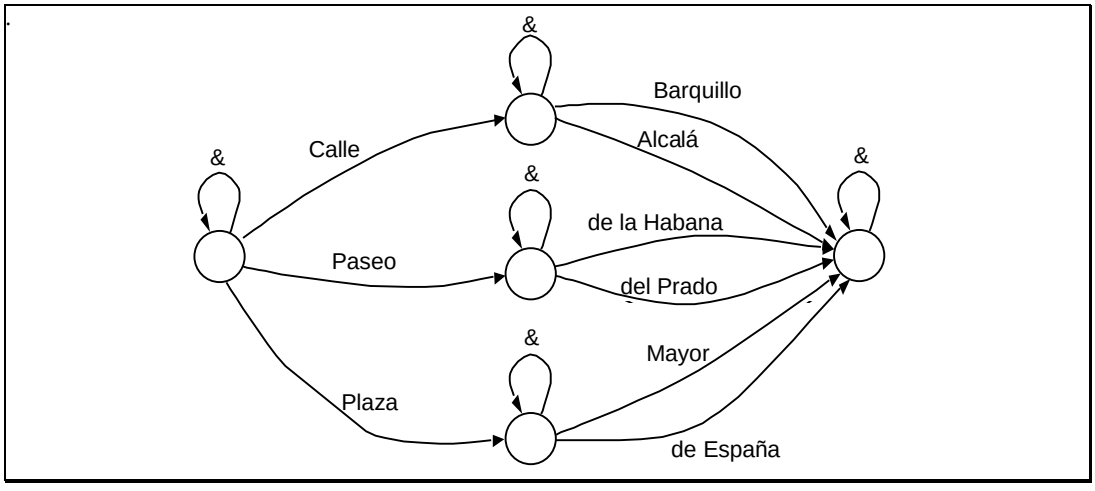

Figura 3-25. Estructura de red para el reconocimiento de direcciones.

Habría que especificar el conjunto de reglas necesarias para dividir cualquier nombre y apellido en dos partes a partir de su texto. Estas reglas pueden enumerarse de la siguiente manera:

- Si un nombre consta únicamente de una palabra (porque falte el apellido, por ejemplo), se conserva dicho modelo de forma completa y no se introduce ningún nodo intermedio con silencio.
- Si está formado por dos palabras, se divide por el único silencio o pausa que existe.
- Si existen tres o más palabras, se divide por el silencio cuyo centro de gravedad esté situado más hacia el centro tendiendo a la izquierda. Esto es debido a que suele pronunciarse un mayor número de palabras sin silencio antes de hacer ninguna pausa. Se tiene en cuenta que todos aquellos conjuntos de palabras que empiecen por *de*, *del* ó *de la* se agrupan con la palabra que acompañan, considerándose como una palabra. En la Figura 3-25 hay un ejemplo para el caso del reconocimiento de direcciones.

El objetivo de la división en nombre y apellido no es sólo modelar la posible pausa de respiración en una secuencia de palabras larga, sino también posibilitar una reducción del tamaño de la red de reconocimiento, con una disminución considerable de la memoria y del cómputo necesario que, para el caso de la base de datos de prueba, es de un 20 %. Esta base de datos fonéticamente balanceada consta de 100 nombres y apellidos españoles.

3.8 Integración del Reconocedor en un Sistema PC-DSP para su Funcionamiento en Tiempo Real

3.8.1 Introducción.

Hasta ahora, se han descrito los elementos necesarios para diseñar un reconocedor fonético de voz y su particularización en aplicaciones telefónicas. Sin embargo, a la hora de diseñar el reconocedor y de probarlo, no es suficiente que funcione como un programa que lee los datos almacenados en un fichero y extrayendo una respuesta. Hay que considerar otros aspectos importantes cuando un reconocedor funciona en condiciones reales.

Un aspecto fundamental e indispensable en cualquier aplicación real es el *detector de extremos*, que indica cuándo se ha empezado a hablar y cuándo se ha dejado de pronunciar, para que el algoritmo de reconocimiento se detenga y realice la búsqueda hacia atrás o *backtracking*.

A la hora de la integración en un sistema formado por una tarjeta de procesado digital de la señal, que realiza la interfaz con la red telefónica, y un ordenador personal, es necesario analizar modularmente las diferentes partes en que se puede dividir el reconocedor. A partir de estos módulos es posible realizar una distribución de las tareas en cada uno de los dispositivos y la cantidad de información a transmitir entre ambos.

3.8.2 División Funcional de un Reconocedor de Voz Fonético.

Como ya se ha mencionado, el reconocedor de voz consta de una serie de procesados desde que se capta la señal hasta que el algoritmo de reconocimiento síncrono procesa una nueva trama. Un detector de extremos o el fin de silencio marca el momento en el que hay que parar de analizar y detectar la secuencia de palabras reconocidas.

Los módulos del reconocedor de voz, similares a los del apartado 3.5.3, son:

- ***Extracción de características***: Recoge las muestras de la señal de voz enventanada y genera los vectores de características.
- ***Distancia a los vectores de medias***: Tanto si son modelos continuos como modelos semicontinuos, hay que calcular la distancia entre los vectores de características y las medias de los modelos.
- ***Distancia a los modelos fonéticos o subfonéticos***: Las distancias a las medias se combinan con los pesos asociados a los modelos para dar las probabilidades de pertenencia. Estos valores calculados son los utilizados por el algoritmo de reconocimiento.
- ***Algoritmo de búsqueda síncrono en la red de reconocimiento***: Utilizando las probabilidades calculadas anteriormente, se actualiza la red de

reconocimiento, en la que se mantiene la referencia a las distancias de los modelos de estado. El algoritmo de reconocimiento accede a la referencia y la utiliza. De esta manera, aunque el modelo de estado sea usado en múltiples lugares, su distancia se calcula una única vez.

- ***Detector de extremos***: Este módulo es el encargado de detectar cuándo es necesario interrumpir el análisis de la señal de voz y extraer la secuencia de palabras reconocidas. Tan sólo se necesita que el extractor de características le proporcione el nivel de energía de la trama actual para tomar la decisión.

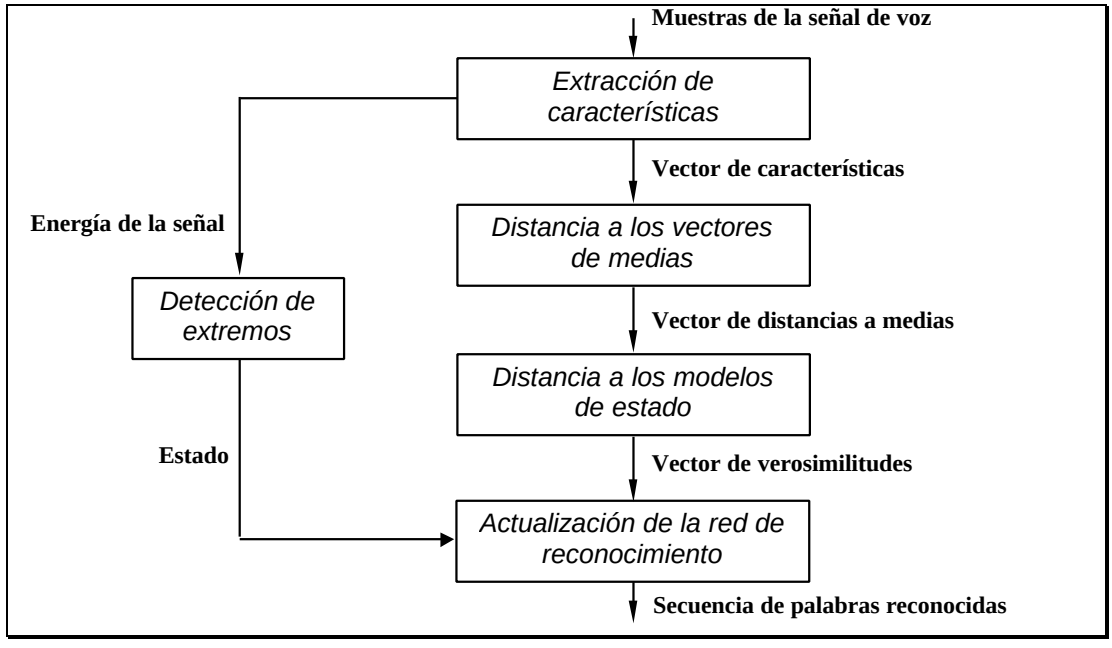

Figura 3-26. Estructura modular del reconocedor de voz. Se distinguen cinco bloques que pueden integrarse sobre el sistema *PC-DSP* en cualquiera de sus interfaces.

En la Figura 3-26 se muestra la distribución de los cinco módulos que se pueden identificar en el reconocedor de voz y la forma de conectarse entre sí. Se distinguen varios bloques de procesados, algunos de los cuales pueden ejecutarse en paralelo y otros sólo secuencialmente. En cualquier caso, estos módulos pueden repartirse entre diferentes dispositivos de procesado y, en función de la velocidad de transferencia entre ambos, es posible realizar la asignación de cargas computacionales que optimice el sistema en el que se encuentre para conseguir el máximo número de reconocedores con vocabularios grandes.

Para realizar esta división hay que tener en cuenta que el volumen de información a transmitir entre algunos módulos de procesado es proporcional al número de modelos de estado. Esto ocurre en el caso de la salida del cálculo de distancias a medias y el cálculo de las verosimilitudes. El uso de modelos semicontinuos y de modelos subfonéticos permite aumentar el nivel de compartición entre modelos, disminuyendo estas transferencias de datos. Por el contrario, el módulo de extracción de características siempre presenta un mismo nivel de procesado.

3.8.3 Incorporación del Reconocedor Fonético en un Sistema ITO.

Una vez definidas las partes en que se puede dividir el reconocedor de voz, hay que analizar el sistema en el que se integra para aprovechar las características que presenta. En un sistema *ITO* se pueden distinguir dos partes: una tarjeta de procesado de señal *DSP* y un *PC*.

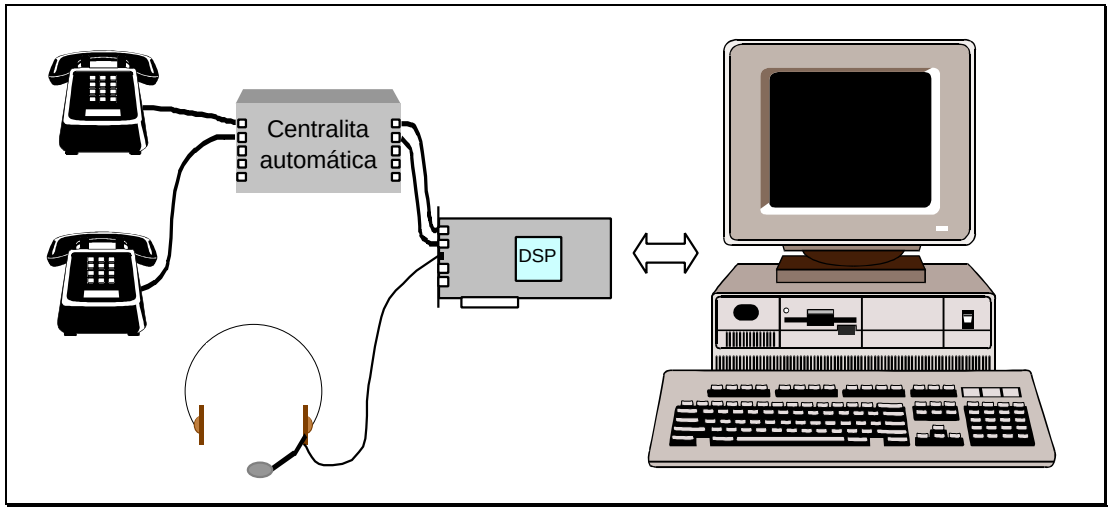

Figura 3-27. Descripción de los diferentes elementos que intervienen en el reconocedor en tiempo real integrado dentro de un sistema *ITO*.

En la Figura 3-27 pueden verse los diferentes elementos que intervienen en el reconocedor en tiempo real y su interconexión. La tarjeta de procesado de señal posee una interfaz a la red telefónica, para un total de cuatro líneas y una entrada headset (auriculares + micrófono) para pruebas sin canal telefónico. La conexión a la línea telefónica puede realizarse a una centralita automática a través de la cual se pueden simular comunicaciones telefónicas y llamadas ficticias usando dos líneas a la vez, una de salida y otra de entrada. A través del headset se puede utilizar el reconocedor sin necesidad de realizar una llamada telefónica.

El *PC* puede ser *Pentium*^{TM} o superior, contra más rápido sea es posible conseguir sistemas con un mayor número de reconocedores funcionando a la vez, o con vocabularios mayores. La tarjeta se conecta a través de un bus *ISA*, que es el encargado de la comunicación entre la tarjeta con *DSP* y el *PC*.

Se distinguen tres modos de utilización:

- ***Headset***: Se usan los auriculares conectados a la tarjeta de DSP. Permite realizar pruebas sin red telefónica.
- ***Teléfono***: Para hacer pruebas a través de una centralita mediante llamadas telefónicas.
- ***Simulación de llamada***: Se utilizan dos líneas de la tarjeta a la vez. Una línea realiza una llamada a través de la centralita hacia la otra. Por un canal

se reproducen ficheros de voz y, por otro, el reconocimiento se lleva a cabo en tiempo real.

Todo lo anterior se describe desde el punto de vista de los elementos hardware de que se compone y sus interconexiones. Desde un punto de vista funcional interesa saber cómo integrar el reconocedor dentro del sistema *ITO*.

El funcionamiento es el siguiente: cuando el sistema detecta que debe arrancar el reconocedor avisa a la tarjeta y ésta empieza a capturar muestras. Estas muestras pueden enviarse directamente al *PC* o elaborarse parcial o totalmente dentro de la tarjeta de *DSP*. Si sólo se realiza una parte o incluso no se procesa nada, los datos se pasan al *PC* a través del bus de datos. En el *PC* se inicia un *proceso* (*thread*) que va recibiendo los datos de la tarjeta y los va procesando de forma síncrona. Cuando el detector de extremos encuentra el fin de palabra, se realiza la búsqueda hacia atrás, el *Thread* finaliza el análisis y extrae la secuencia de palabras reconocidas.

Como paso previo al inicio de funcionamiento del reconocedor, hay que preparar el vocabulario, que consiste en generar el conjunto de ficheros de configuración con los modelos usados y la estructura de la red a partir de un fichero de texto con las palabras del vocabulario. Cada vez que se pretenda usar el reconocedor, hay que cargar dichos ficheros en memoria e inicializar la información de la red. El reconocedor puede funcionar todas las veces que se desee para, finalmente, liberar los recursos cuando se cambie de vocabulario o se cierre la aplicación.

Independientemente de cómo se realice la división del software de reconocimiento entre la tarjeta con *DSP* y el *PC*, el funcionamiento del reconocedor es idéntico. La única diferencia es la carga computacional que se asigne a cada elemento y la cantidad de datos que se transfieren. Entre todas las configuraciones destacan:

- ***Todo en el PC***: La tarjeta captura las muestras y las envía al *PC*, que realiza todo el procesado. En este caso, la tarjeta es un elemento de interfaz y no aporta nada más.
- ***Distancia del vector de características a los vectores de medias en el DSP***: La tarjeta captura las muestras, extrae los vectores de características y calcula las distancias a los vectores de medias de las gaussianas. Los vectores de medias, junto con la información de la energía de cada trama, son enviados al *PC*, que se encarga de realizar el resto del procesado. Cuando el *PC* detecta el fin de la pronunciación, avisa a la tarjeta para que ésta deje de transmitir datos. Esta configuración tiene la ventaja de que los procesados, como son la *FFT* y productos escalares, se realizan de manera muy eficiente en el *DSP*, reduciéndose el cómputo que debe hacer el *PC*.

3.8.4 Algoritmo de Detección de Extremos.

La detección de extremos es algo que resulta fundamental a la hora de tener un sistema funcionando en tiempo real. Cuando se está evaluando un reconocedor de voz

a partir de una base de datos, la segmentación de la voz puede ir incorporada en los ficheros y, por tanto, no hay necesidad de utilizar un detector de extremos. Sin embargo, en un sistema en tiempo real, no sólo es necesario conocer cuándo se ha terminado de pronunciar una frase, sino también si ésta ha empezado antes de tiempo o incluso si no se ha pronunciado. Un mal diseño del detector de extremos puede hacer que las características del reconocedor de voz se degraden, perdiendo todas las ventajas que proporcionan otras técnicas de robustez.

Como se menciona en [Rab93], el diseño de un detector de extremos para voz limpia es muy fácil. Sin embargo, pueden aparecer problemas que complicarían el funcionamiento de los detectores, destacándose:

- *Los problemas causados por el locutor y su manera de pronunciar*: Muchas veces, debido a las características del locutor, éste puede generar unos ruidos cortos causados por los labios o la boca, que pueden ser confundidos con sonidos oclusivos. Por otro lado, la respiración fuerte puede ser confundida con fonemas fricativos.
- *Condiciones ambientales*: Lo ideal es una habitación sin ruidos u otras señales diferentes a las pronunciadas por el locutor. En la realidad aparecen ruidos de fondo (ruidos de máquinas), ruidos no estacionarios (bocinas, puertas que se cierran), interferencias (TV, radio, conversaciones) y situaciones hostiles (por ejemplo, si el locutor que está hablando se encuentra en una situación estresante como volar).
- *Canal de comunicaciones*: En condiciones ideales no debería introducir ninguna modificación a la señal, sin embargo, suele aparecer cacofonía, distorsión de intermodulación o interferencia tonal.

Existen tres familias amplias de detectores de extremos dependiendo de cómo se realice junto con la comparación de patrones:

- ***Extrínsecos o explícitos***: La detección de voz se puede realizar independientemente de las técnicas de comparación de patrones de fases posteriores en el reconocimiento de voz.

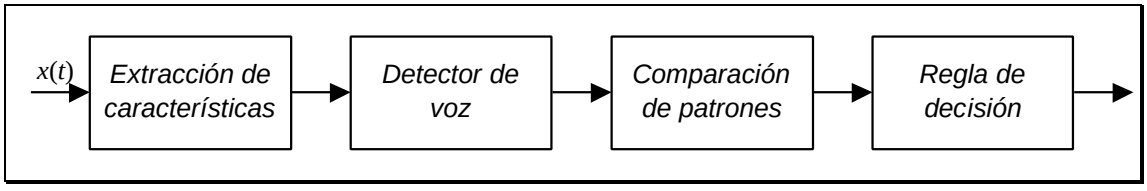

Figura 3-28. Diagrama de bloques de un detector de voz explícito.

En un detector de voz explícito primero se realiza una extracción de características, que es usada para la detección de extremos. La comparación de patrones se realiza a continuación, incluso con una extracción de características

diferente. En la Figura 3-28 se muestra el diagrama de bloques. Este tipo de detectores fallan cuando existe ruido o interferencias no estacionarias.

- ***Intrínsecos o implícitos***: La determinación de los puntos inicial y final de la frase a reconocer se determinan únicamente por las fases de reconocimiento y decisión del sistema. No existe un mecanismo externo como en el caso de los métodos explícitos.

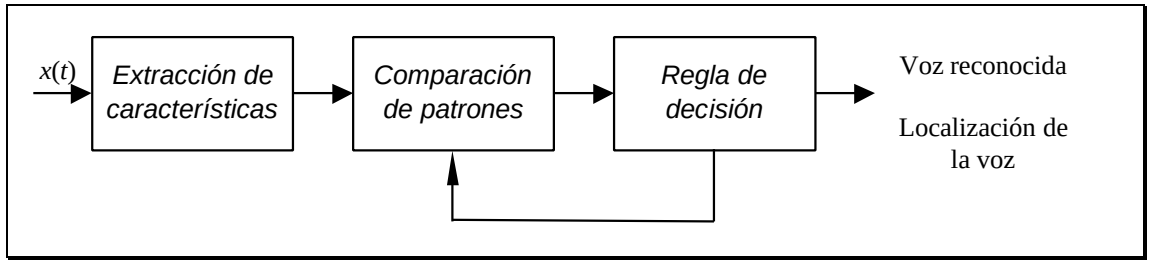

Figura 3-29. Diagrama de bloques de un detector de voz implícito.

Los bloques de un detector que trabaja de forma intrínseca al proceso de comparación de patrones aparecen en la Figura 3-29. El detector es más complejo, pero se obtienen unos resultados mejores.

- ***Híbridos***: Se utiliza información obtenida de medidas realizadas a partir de la señal de voz, como en los métodos explícitos, y a la vez se incorpora información en el algoritmo de reconocimiento, como en los métodos implícitos. Permite reducir la complejidad considerando sólo una parte del conjunto de estimaciones que un detector de extremos implícito realiza.

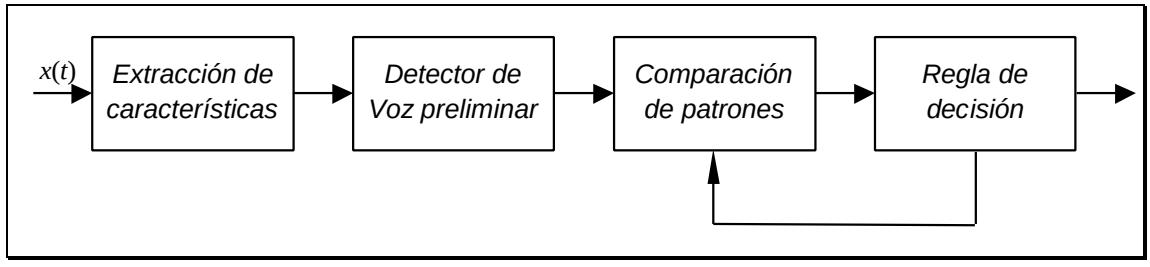

Figura 3-30. Diagrama de bloques de un detector de voz híbrido.

La carga computacional de este detector es muy similar al del explícito, pero presenta un funcionamiento más parecido al implícito. La Figura 3-30 muestra su diagrama de bloques.

En el caso de la red de reconocimiento que utiliza el algoritmo de búsqueda síncrona, es fácil añadir información implícita sobre la detección de extremos, basta usar un modelo de silencio como el descrito en el apartado 3.7. Con este mecanismo, se asegura que tanto el silencio inicial como el final se adapten en el reconocedor, eliminándose a la hora de extraer la mejor secuencia de palabras.

En [Lam81] o [Wil84] se observa cómo una estrategia de detección de extremos híbrida obtiene unos resultados muy similares a un método explícito, pero con una complejidad mucho más baja.

En el caso de este reconocedor de voz se necesita que la detección de extremos explícita se encargue de:

- Parar cuando se detecte el silencio final: Una vez alcanzado el silencio final, no es necesario que transcurra un tiempo para realizar la búsqueda hacia atrás. Fijando un valor máximo de silencio final, el sistema será capaz de parar el reconocedor dando una respuesta más rápida.
- Detectar que no se ha pronunciado nada: Es algo necesario, si no fuera así, el reconocedor no sabría si realmente hay o no palabras. Esto no es del todo cierto, es posible introducir un modelo de silencio intermedio en paralelo con los modelos de palabra. En tal caso, si el sistema descubre que es el mejor modelo, puede detectar que no ha habido voz. Sin embargo, esto no resulta muy fiable cuando el nivel de ruido es muy alto y, por tanto, una estrategia explícita es recomendable.
- Detectar el falso comienzo: Finalmente, es necesario saber si se empezó a hablar antes de que el reconocedor empezará a procesar. En tal caso, hay que pararlo, pues lo que éste obtenga puede que no tenga sentido.

3.8.4.1 Algoritmo de Detección de Extremos Basado en la Energía a Largo y a Corto Plazo.

Este algoritmo está basado en la diferencia entre la energía a corto plazo y la de largo plazo. Este valor debe cumplir una serie de criterios que están parametrizados. El detector pasa por una serie de estados, hasta que se llega a la detección de una palabra, silencio o falso comienzo. Los parámetros que hay que definir son:

- ***Umbral de silencio*** ($UMBSIL$): Este valor ha sido fijado experimentalmente en $0{,}05$. Cuando la diferencia de energía está por debajo del mismo, se considera que es ruido; por encima, se detecta el comienzo de una palabra.
- ***Umbral de final*** ($UMBEND$): Este valor está fijado en $-0{,}001$, por encima del cual, una vez detectado el comienzo de una palabra, se considera que la señal sigue correspondiendo a la misma. Cuando la diferencia de energía cae por debajo de este umbral, se detecta el final de palabra.
- ***Duración mínima de una palabra*** ($DURPAL$): Es el tiempo que debe permanecer la diferencia de energía por encima de $UMBSIL$, al menos durante $100\ ms$, que equivale a $6{,}25$ tramas.
- ***Duración mínima del silencio posterior*** ($DURSIL$): Es el tiempo en que la diferencia de energías debe mantenerse por debajo del umbral $UMBEND$. Debe ser como mínimo $100\ ms$, que equivale a $6{,}25$ tramas.

Es evidente que estos valores deben estar ajustados según lo que se desee reconocer. No es lo mismo reconocer nombres y apellidos, donde hay que colocar *DURPAL* del orden de 250 *ms* y *DURSIL* de al menos 500 *ms* (para que los silencios intermedios no sean interpretados como silencios finales, a costa de dar una respuesta más lenta), que el caso de discriminar entre *sí* y *no*, donde es necesario disminuir *DURPAL* por debajo de los 170 *ms* y *DURSIL* a 100 *ms* (para ser detectados y dar una respuesta más rápida).

Otro aspecto que se puede añadir al detector de extremos es que si el número de tramas de silencio previo es inferior a 225 *ms*, hay que indicar que la palabra empezó a pronunciarse antes de arrancarse el reconocedor.

En el cálculo de la energía de largo plazo (*LTE*: Long-Time-Energy) y la de corto plazo (*STE*: Short-Time-Energy) se utiliza el coeficiente cepstrum cero ($C(0)$), que se corresponde con la energía de la señal. *LTE* y *STE* se obtienen de un filtrado sobre la energía de la señal en cada trama. Las expresiones son las siguientes:

$$LTE[t] = 0.95 \cdot LTE[t-1] + 0.05 \cdot C[0,t] \qquad LTE[0] = 2.0$$

$$STE[t] = 0.90 \cdot STE[t-1] + 0.10 \cdot C[0,t] \qquad STE[0] = 1.0$$

$$SNR[t] = LTE[t] - STE[t]$$

El algoritmo de detección de extremos se puede definir como una máquina de cinco estados denotados por *Q0*, *Q1*, *Q2*, *Q3* y *Q4*. El estado *Q0* se corresponde con el estado inicial de partida o silencio previo. El estado *Q1* se corresponde a la detección de palabra, permaneciendo el tiempo que dura la pronunciación. El estado *Q2* se alcanza cuando se detecta un silencio entre segmentos vocales o el silencio final. La transición al estado *Q3* marca el fin del proceso de detección. El estado *Q4* se corresponde con un error que impide el reconocimiento.

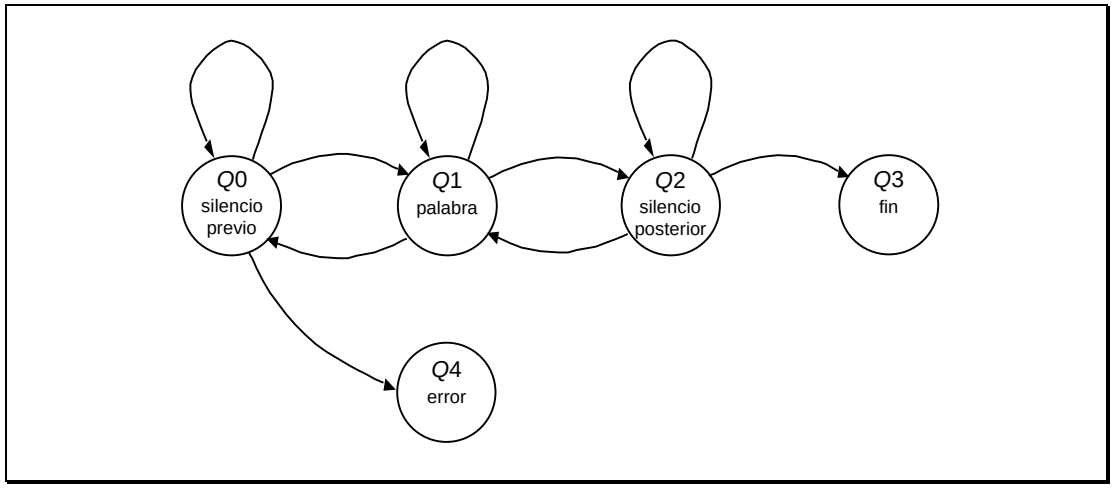

Figura 3-31. Máquina de estados asociada con la detección de extremos. Sólo tiene dos posibilidades de salir: cuando se ha detectado un silencio posterior de una duración dada, o cuando se ha detectado un error.

Los datos de entrada a la máquina de estados son: la relación señal a ruido (*SNR*) y el número de trama en curso *T*. Las operaciones se denotan como *Qi*-*Qj*, que corresponden al paso del estado *i* al estado *j*. En la Figura 3-31 se muestran las posibilidades de transición entre los diferentes estados. A continuación, se definen las condiciones y las operaciones que se realizan:

- ***Q*0-*Q*0**: si (*SNR* <= *UMBSIL*).
- ***Q*0-*Q*1**: si (*SNR* > *UMBSIL* && *T* >= 14).
  {*T1* = 0}, *T1* es un contador de tramas asociado al estado *Q1*.
- ***Q*0-*Q*4**: si (*SNR* > *UMBSIL* && *T* < 14).
- ***Q*1-*Q*0**: si (*SNR* < *UMBEND* && ++*T1* < *DURPAL*).
- ***Q*1-*Q*1**: si (*SNR* >= *UMBEND*).
  {*T1* ++}.
- ***Q*1-*Q*2**: si (*SNR* < *UMBEND* && ++*T1* < *DURPAL*).
  {*T2* = 0}, *T1* es un contador de tramas asociado al estado *Q2*.
- ***Q*2-*Q*1**: si (*SNR* >= *UMBEND*).
  {*T1* = *T1* + *T2*}.
- ***Q*2-*Q*2**: si (*SNR* > *UMBEND* && *T2* < *DURSIL*).
  {*T2* ++}.
- ***Q*2-*Q*3**: si (*SNR* < *UMBEND* && T2 >= *DURSIL*).

Al aplicar el algoritmo hay que tener en cuenta que con la extracción de características existe un retraso de dos tramas respecto al instante actual, con lo que el comienzo de palabra (transición *Q*0-*Q*1) se retrasa implícitamente dos tramas, igual que con el final de palabra (transición *Q2*-*Q3*).

Otro aspecto importante a considerar es la existencia de un tiempo máximo (*time-out*) para el proceso de reconocimiento. El reconocedor se detiene por sí mismo si se ha sobrepasado el tiempo, y analiza lo que haya hasta ese momento. Si no hay nada, se considerará que el usuario no ha pronunciado nada.

Capítulo

4

Robustez en el Reconocimiento de Voz

4.1 Introducción

En el capítulo 3 se describe con profundidad un reconocedor fonético de voz para aplicaciones telefónicas en un sistema en tiempo real. Sin embargo, si lo que se quiere es un sistema que mantenga la robustez cuando las condiciones para las que fue diseñado difieren de las condiciones reales en las que va a funcionar, no es suficiente con lo anterior, es necesario añadir una serie de procesados adicionales que no supongan una carga computacional excesiva para no hacer disminuir las prestaciones conseguidas.

Para aumentar la robustez es necesario considerar las desviaciones o alteraciones respecto de las condiciones que fueron consideradas. El diseño del reconocedor, desde la extracción de características hasta el entrenamiento de los

modelos, ha sido pensado para disminuir la posible desadaptación respecto de las condiciones en situaciones reales. Sin embargo, esto no elimina toda la variabilidad y es necesario pensar en estrategias particularizadas con el fin de mejorar las características del reconocedor de voz.

Algunas de las variabilidades posibles en el sistema de reconocimiento ya se han tenido en cuenta en el reconocedor de voz. Por ejemplo, a la hora de diseñar la red de reconocimiento se han introducido palabras comodín para permitir al reconocedor adaptarse a las diferentes formas en que los usuarios pueden responder.

A continuación, se enumera un conjunto de aspectos bien definidos, aunque no totalmente independientes entre sí, que son analizados en este capítulo. Para cada uno de ellos se propone al menos una metodología para mantener las propiedades del reconocedor cuando las características acaecidas son diferentes a las esperadas. Son los siguientes:

- ***Independencia de locutor***: Éste es sin duda uno de los aspectos fundamentales. Hay que tener en cuenta que las voces y la forma de hablar de las diferentes personas no son iguales, puesto que presentan diferentes frecuencias fundamentales, velocidades de locución, volumen de emisión, etc. Como ya se comentó en el apartado 2.4.2.1 sobre las aplicaciones telefónicas, surge la necesidad de tener reconocedores de voz que funcionen para cualquier persona. Esta información hay que considerarla tanto en las bases de datos utilizadas como en la generación de los modelos de Markov y las redes de reconocimiento utilizadas.
- ***Independencia del canal de comunicaciones***: Si en una aplicación telefónica los usuarios son diferentes para cada nueva utilización, el canal de comunicación también lo será. Además, en este caso, resulta ser extremadamente variable dentro de las peculiaridades propias del canal telefónico, existiendo tantos canales como posibles caminos dentro de la red telefónica. En el apartado 4.2.4.1, una vez caracterizado el canal telefónico y enumeradas las diferentes posibilidades propuestas, se analizan dos alternativas como son el filtrado *RASTA* y la técnica *CMN*.
- ***Palabras breves***: En el reconocimiento de voz no todas las palabras presentan la misma dificultad. Resulta mucho más sencillo discriminar entre palabras que tengan una longitud mayor, puesto que el número de características que deben poseer es mayor. Por el contrario, las palabras breves y muy similares entre sí son de gran dificultad de reconocimiento. Sin embargo, en todo reconocedor en aplicaciones telefónicas siempre surge la necesidad de que el usuario responda con palabras como *sí* y *no*, o con un número. Además, los algoritmos de detección de extremos se diseñan para el reconocimiento de palabras con unas duraciones determinadas y, muchas veces, ante consonantes con muy poca energía (como las fricativas) no se

activan, no detectándose nada. Las palabras breves necesitan un tratamiento especial para que la calidad del sistema de reconocimiento sea superior y presente una mayor robustez y fiabilidad.

- ***Palabras fuera de vocabulario***: El reconocedor de voz descrito en el capítulo anterior está pensado para reconocer entre un conjunto de palabras o secuencia de ellas, en función de la red de reconocimiento, cuál es la más probable. Los modelos de Markov presentan el problema de ser un método estadístico, por tanto, siempre dan un resultado aunque lo pronunciado no se corresponda con ninguna de las palabras posibles. Introduciendo en el reconocedor algún mecanismo que proporcione una puntuación sobre el nivel de calidad, es posible utilizarlo para rechazar o validar la palabra o secuencia de palabras reconocidas. El grado de calidad exigido puede imponerse en función de las necesidades de robustez. En una aplicación donde sea crítico saber que lo reconocido está bien, es preferible ser muy estricto, rechazar la palabra y sugerir al usuario que la vuelva a pronunciar.
- ***Distancias entre modelos***: Hay otro aspecto que, si bien no se encuentra ligado directamente con la robustez del sistema de reconocimiento, es necesario considerar. Muchas veces se entrenan unos modelos de las bases de datos de palabras y no se sabe si aportan o no información útil al sistema de reconocimiento o, por el contrario, introducen mayor incertidumbre. Por eso, a veces resulta más conveniente agrupar un conjunto de modelos similares con un bajo grado de fiabilidad y, así, aunque se obtiene un numero menor de modelos, éstos están mejor entrenados. Esto puede ocurrir en el caso de las oclusivas donde, dada su brevedad, la variabilidad de los vectores de características puede ser muy elevada, pudiendo interesar entrenar todos los fonemas oclusivos juntos en un único modelo, introduciendo la información del fonema en el siguiente modelo.

El orden de exposición es el siguiente: en el apartado 4.2 se describen las metodologías para conseguir independencia de locutor; el apartado 4.3 se destina a la independencia del canal de comunicaciones; en el apartado 4.4 se menciona una técnica para aumentar la robustez de palabras breves basada en el uso de diferentes transcripciones fonéticas para la misma palabra; en el apartado 4.5, se hace un análisis de métodos para valorar la calidad de las palabras reconocidas y poder usarse para rechazar las mal reconocidas o fuera de vocabulario; y en el apartado 4.6 se describe la técnica denominada proyección de Sammon [Sam69], que se utiliza para medir distancias entre los modelos entrenados.

4.2 Robustez Frente a Diferentes Locutores

4.2.1 Introducción.

La independencia de locutor es un aspecto determinante para el correcto funcionamiento de un reconocedor de voz. Lo ideal sería que un reconocedor de voz estuviera completamente adaptado al locutor que lo usa, de manera que la incertidumbre fuera muy pequeña y la tasa de reconocimiento muy elevada. La implementación de reconocedores dependientes de locutor tiene sentido cuando se está hablando de aplicaciones que se vayan a usar por un único locutor y se requiera un grado de precisión elevado.

Como se puede ver en el apartado 2.4, destinado a la caracterización de las propiedades que debería tener un reconocedor de voz para diferentes aplicaciones, en función de la misma se exige o no independencia de locutor. En aplicaciones telefónicas, donde cada nueva llamada supone un nuevo usuario, no es factible tener reconocedores dependientes o independientes de locutor adaptativos (puesto que no hay tiempo suficiente para que el usuario entrene el reconocedor, ni que éste se adapte al mismo).

En aplicaciones telefónicas hay que intentar caracterizar las fuentes de variabilidad de las diferentes voces en los modelos de Markov o en la estructura de la red de reconocimiento, para que el funcionamiento del reconocedor no se degrade cuando los usuarios sean muy diferentes.

4.2.2 Rasgos Característicos de la Voz.

Para empezar, a continuación se enumeran las diferentes propiedades que se pueden encontrar entre distintos locutores y que pueden dar como resultado que el reconocedor de voz no funcione igual para todos ellos:

- ***Pitch***: Es una de las propiedades que más caracteriza los diferentes tipos de voces. El rango de valores de la frecuencia fundamental es muy grande, principalmente si se comparan las voces de mujeres, hombres y niños. Así, en las mujeres y niños, al tener unas cuerdas vocales de menor longitud que la de los hombres, presentan una frecuencia mayor.

  El pitch es la manera en la que al generar los sonidos sonoros se producen diferentes respuestas espectrales. Las voces con un pitch de frecuencia fundamental muy grande tienen el problema de que pierden resolución para la obtención de la envolvente espectral, aumentan el rango de variación y resultan mucho más complicados de reconocer.

  En resumen, para voces con pitch de frecuencia fundamental baja no hay ningún problema, porque con la extracción de características propuesta se

obtiene fácilmente la envolvente espectral. Para el caso contrario, pueden aparecer inconvenientes.

- ***Forma y tamaño de la cavidad bucal***: Al igual que en el caso anterior, la forma de la cavidad bucal, que es la que filtra la señal generada por las cuerdas vocales en los sonidos sonoros, no es igual en todas las personas.
- ***Velocidad de elocución***: Cada persona tiene una velocidad de pronunciación diferente. En general, cambiar la velocidad del habla supone que los silencios entre palabras y dentro de ellas son más o menos largos y los sonidos sostenidos (como las vocales, las nasales y las fricativas) duran más o menos, mientras que el resto de consonantes (como las oclusivas) siempre duran aproximadamente lo mismo, lo único que puede ocurrir es que no se pronuncien. La velocidad de pronunciación de los fonemas es un rasgo característico de cada persona y varía incluso en función del estado físico o emocional del momento.
- ***Diferentes características fonéticas***: Dentro de un idioma existen varios dialectos o registros, de manera que ciertos fonemas pueden ser pronunciados de forma diferente según su región de procedencia. Además, la variación del pitch con el tiempo es altamente dependiente del dialecto

Para cada una de las anteriores características es posible aplicar distintos mecanismos de compensación de la variabilidad. Se pueden distinguir dos alternativas:

- ***Bases de datos de entrenamiento***: Consiste en introducir la variabilidad de los locutores en las bases de datos de entrenamiento para el caso de voces pronunciadas con diferentes pitch y diferentes cavidades bucales.
- ***Estructura de las unidades fonéticas***: La adaptación a diferentes velocidades de locución y con diferentes características fonéticas se consigue introduciendo dentro de la unidad fonética correspondiente las alternativas de pronunciación, permitiendo saltos entre modelos o incluso modelos en paralelo.

4.2.3 Incorporación de Variabilidad de Locutor en las Bases de Datos de Entrenamiento.

Como se ha mencionado anteriormente, hay una serie de aspectos de la voz (como el pitch, información sobre el tamaño y forma de la cavidad bucal) que hay que introducir en la base de datos de entrenamiento como único modo de tener en cuenta esta fuente de información en los modelos de Markov, ya que es muy difícil extraerla y compensarla de otra manera utilizando un esquema de análisis sencillo y, aunque en la extracción de características se utiliza un banco de filtros con una anchura suficiente para que el pitch y sus armónicos no alteren la envolvente espectral, la manera en que

el pitch en sí altera la posición de los formantes[2] de la voz no está muy clara, pero es relevante.

Surge la posibilidad de tener dos tipos de modelos de Markov, unos para los de pitch bajo y otro alto, con dos redes de reconocimiento independientes. Sin embargo, además de incrementar al doble la potencia de cálculo y el tamaño de la memoria, habría que decidir cuál de los dos tipos es el más adecuado. Por eso, es preferible entrenar unos únicos modelos válidos para cualquier persona.

Mediante la utilización de la base de datos del proyecto Albayzín, se tiene suficiente representatividad de la variabilidad, y gracias a la utilización de la segmentación automática de las frases es posible utilizarla de forma completa.

4.2.3.1 Variabilidad de la Base de Datos Albayzín.

A continuación, se describe la base de datos utilizada para el entrenamiento de los modelos. En [Pro92] se realiza un análisis más detenido. La base de datos Albayzín consta de dos corpus diferenciados. El primero está formado por 200 frases, mientras que el segundo por 500.

- ***Corpus de aprendizaje***: Está formado por un total de 4.800 frases, ya que cada una de ellas se pronuncia 24 veces. La mitad de las frases son pronunciadas por mujeres (2.400 frases) y la otra mitad por hombres (las otras 2.400 frases). 4 locutores pronunciaron las 200 frases y otros 160 un subcorpus de 25 frases cada uno. En total hay 164 locutores, la mitad mujeres y la otra mitad hombres.

  Las frases fueron diseñadas de manera que se cumpliera que:

  - La frecuencia relativa de aparición de los sonidos sigue la distribución del castellano. Para garantizar la capacidad de entrenamiento de los sonidos menos frecuentes se impuso un mínimo de 40 apariciones en las 200 frases (lo que implica un mínimo de 960 apariciones en el total del corpus).
  - Los contextos considerados relevantes estuvieran presentes al menos 4 veces. Como criterio general, se consideraron relevantes para un sonido aquellos contextos que ocurren al menos el 10 % de las apariciones del sonido; adicionalmente, se consideraron relevantes contextos que pueden significar modificaciones fonéticas importantes.
  - Las proporciones de cada sonido en sílaba tónica y átona siguieran las del castellano.

[2] Los formantes se definen como los máximos del espectro de la señal y coinciden con las frecuencias de resonancia de la cavidad bucal. La posición de estos máximos, en el caso de una vocal, sirve para discriminar entre ellas. Sin embargo, su posición es alterada en función de la frecuencia fundamental del pitch.

- ***Corpus de prueba***: Está formado por un total de 2.000 frases, ya que cada una de ellas se pronuncia 4 veces. La mitad de las frases son pronunciadas por mujeres (1.000 frases), la otra mitad por hombres (las otras 1.000 frases). Hay 40 locutores que pronunciaron cada uno un subcorpus de 50 frases. En total hay 80 locutores.

  En el diseño de las 500 frases se garantizó un mínimo de 50 realizaciones para cada sonido.

Tabla 4-I. Número de frases de locutores para cada uno de los corpus de aprendizaje y de prueba en función del género de la base de datos del proyecto Albayzín.

	Aprendizaje	Prueba	Totales
Mujeres	82	40	122
Hombres	82	40	122
Total locutores	164	80	244

Tabla 4-II. Número de frases de entrenamiento para cada uno de los corpus de aprendizaje y de prueba en función de las edades de la base de datos del proyecto Albayzín. En cada una de las categorías el 50 % corresponde a mujeres y el otro 50 % a hombres.

	Aprendizaje	Prueba	Totales
De 18 a 31 años	2.400	1.000	3.400
De 31 a 40 años	1.200	500	1.700
De 41 a 55 años	1.200	500	1.700
Todas las edades	4.800	2.000	6.800

Como conclusión, en la Tabla 4-I se muestra el resumen del número de locutores por corpus y por género. Mientras que en la Tabla 4-II se muestra la disposición por edades del número de frases de cada uno de los corpus de que consta la base de datos del proyecto Albayzín. La mitad de cada una de las categorías corresponde a mujeres y la otra mitad a hombres.

Se comprueba que la base de datos tiene un número suficiente de locutores y la distribución en cuanto a edades y género es suficientemente amplia como para que sea representativa.

4.2.4 Incorporación de Variabilidad de Locutor en la Estructura de Red de Reconocimiento.

Los otros dos aspectos mencionados en el apartado 4.2.2 sobre rasgos característicos de la voz (las diferentes velocidades de pronunciación y las diferentes características dialectales) pueden ser introducidos directamente en la estructura de la red de reconocimiento.

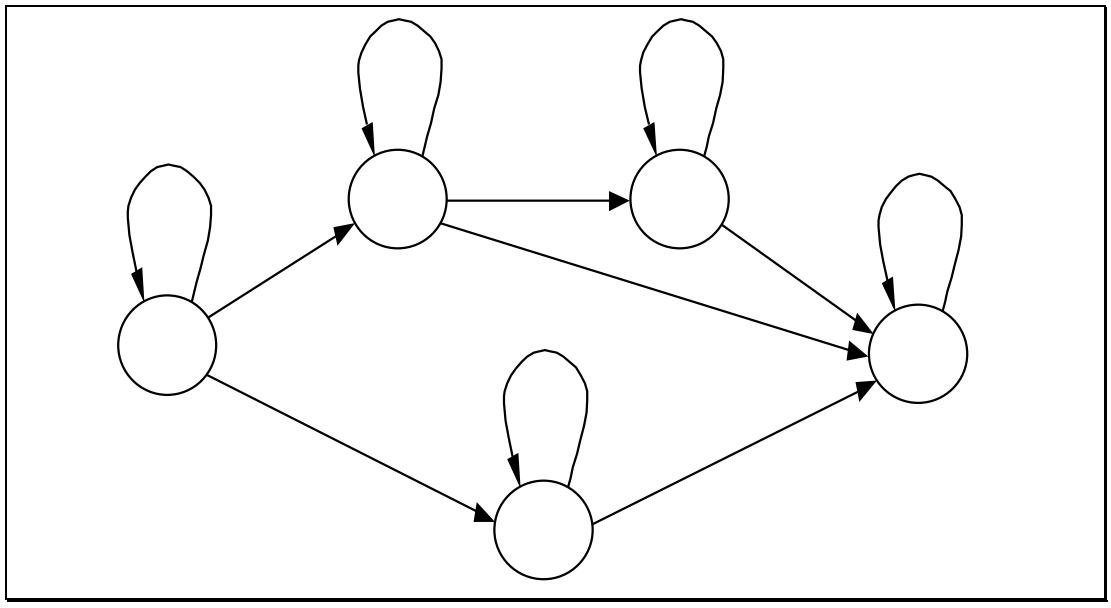

Figura 4-1. Esquema de un modelo fonético de Markov con alternativas subfonéticas del proyecto *Sphinx II* de *CMU*.

Para las diferentes características fonéticas o alternativas de pronunciación, ya se propuso en el reconocedor *Sphinx II* de *CMU* [Hua93] una estructura como la de la Figura 4-1. Esta red fonética tiene la ventaja de ser muy flexible, pues permite introducir una gran variabilidad de características para una misma unidad fonética. Tiene el inconveniente de que complica el entrenamiento de las mismas y, además, se requiere una base de datos de mayor tamaño para que cada modelo de estado esté mejor entrenado.

Puesto que uno de los problemas más complicados y que es necesario solventar es el de las diferentes velocidades de pronunciación, además de la omisión de características fonéticas por parte del usuario, se analizan alternativas a la red de reconocimiento para permitir adaptaciones a estos tipos de variabilidades.

Una alternativa sencilla, cuyo incremento computacional y de memoria no es muy elevado, es la utilización de segmentación automática de frases en lugar de la segmentación manual. Además, se analizan las prestaciones que proporciona una red de reconocimiento de saltos simples o de saltos dobles.

Al estar la base de datos de entrenamiento formada por frases leidas de forma natural, la asignación de número de estados por modelo fonético tiene en cuenta las duraciones medias de los fonemas que aparecen en ella. Si el locutor pronunciase más rápido, los saltos dobles pueden evitar estados intermedios no representados en la pronunciación, ya sea porque ésta dure menos o porque se haya omitido alguna característica.

En este apartado se analizan las diferencias existentes a la hora de segmentar automáticamente una frase a partir de una red de reconocimiento de saltos simples y de saltos dobles, explicando las ventajas que cada una proporciona por separado y comparándolas con la segmentación manual.

4.2.4.1 Entrenamiento con la Segmentación Manual de las Frases.

La cuarta parte de la base de datos Albayzín está segmentada, tal como se mencionó a la hora de hablar del entrenamiento de los modelos fonéticos en el apartado 3.3.4.2, pudiéndose utilizar esta segmentación para el entrenamiento de las unidades fonéticas.

Esta segmentación está dada en número de muestras que hay que convertir en número de tramas, teniendo en cuenta si es inicio o final de fonema para realizar la asignación de las tramas generadas a cada una de las unidades fonéticas.

La utilización de la segmentación manual de las frases de entrenamiento en unidades fonéticas proporciona las siguientes ventajas:

- La secuencia de unidades fonéticas se corresponde exactamente con la que aparece en la frase pronunciada, por lo ***que no se producen errores de transcripción*** por asignación de tramas a modelos fonéticos que no hayan sido pronunciados.
- Se impide que, por una velocidad elevada de pronunciación, tramas vecinas sean asignadas forzosamente a modelos fonéticos erróneos, puesto que si algún fonema tuviera una duración muy pequeña (inferior a una trama) o simplemente no hubiera sido pronunciado, no existiría a todos los efectos.

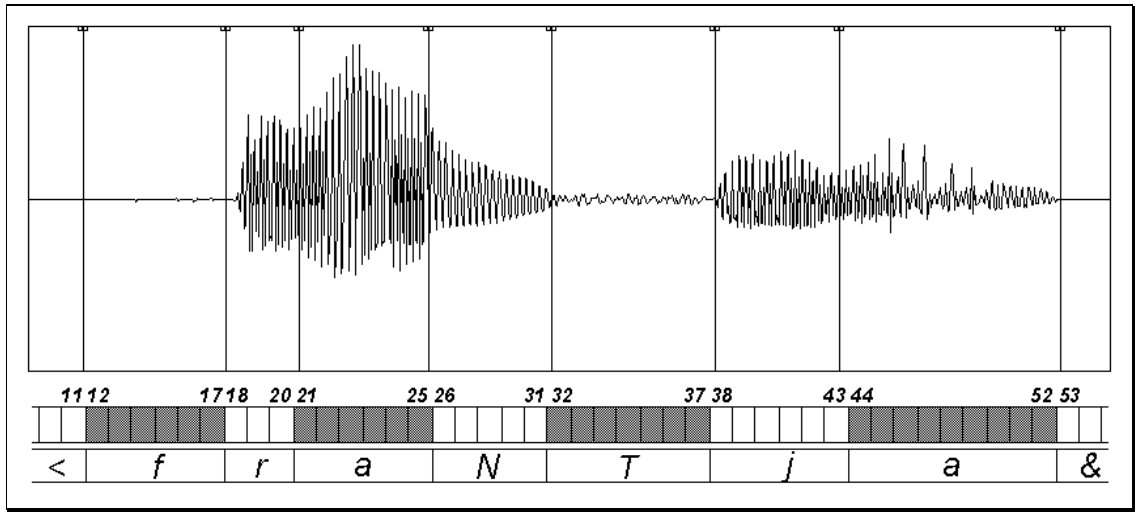

Figura 4-2. Segmentación manual de la palabra *"Francia"* de la frase *"Francia, Suiza y Hungría ya hicieron causa común."* y la correspondiente asignación a tramas de cada unidad fonética en función de las fórmulas de conversión de tiempo a tramas.

En la Figura 4-2 se muestra la relación entre la segmentación manual hecha por un experto fonetista sobre la señal de voz y su reasignación en tramas según las fórmulas del apartado 3.3.4.2. Puesto que los vectores de características se calculan cada $16\ ms$, se puede comprobar que el error máximo con respecto a la segmentación sobre la señal de voz es de $8\ ms$. En este caso, las duraciones son bastante grandes, por lo que todas las unidades fonéticas tienen un número suficiente de estados para poder realizar la segmentación utilizando el algoritmo de Viterbi.

Un dato importante es que cada vector de características obtiene información procedente de un trozo de señal de 96 *ms* (32 + 32 + 32 *ms*), que se corresponde con el cálculo de los cepstrum de la señal de voz (ventana de 32 *ms*) y una regresión de segundo orden que utiliza los vectores calculados dos tramas antes (los 32 *ms* previos) y dos tramas después (los 32 *ms* posteriores), con lo que el grado de integración de señal es muy elevado (es como tener tramas con una ventana de 96 *ms*).

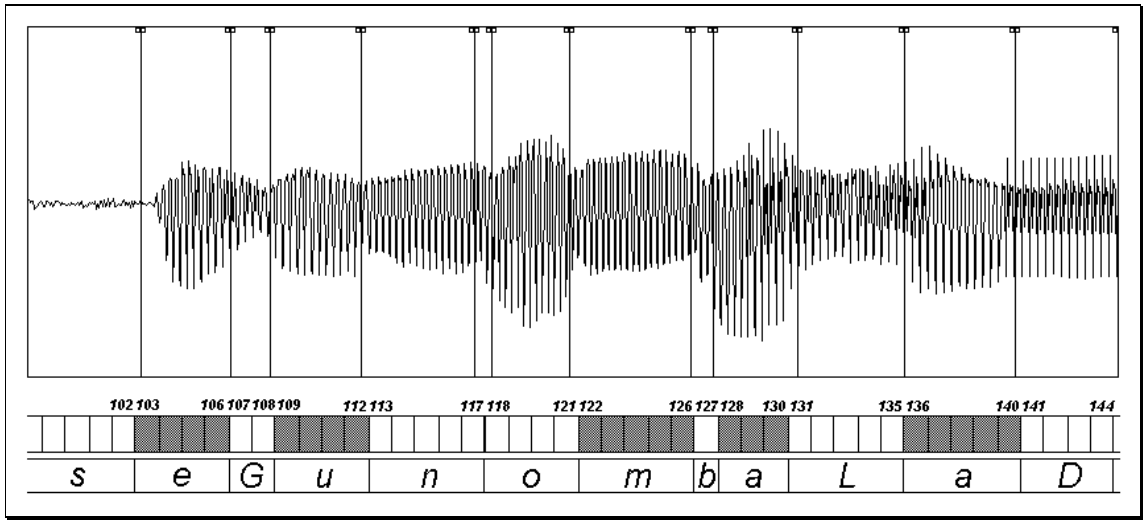

Figura 4-3. Segmentación de *"...segundo en Vallad..."* de la frase *"El primero en Guipúzcoa y el segundo en Valladolid."* y la asignación de tramas a cada unidad fonética en función de las fórmulas de conversión de tiempo a tramas. El fonema *"d"* entre el *"n"* y el *"o"* no tiene ninguna trama.

Por otro lado, en la Figura 4-3 se muestra la segmentación de *"...segundo en Vallad..."* dentro de la frase *"El primero en Guipúzcoa y el segundo en Valladolid"*. En este caso, se muestra el fenómeno de la no aparición de trama alguna del fonema *d* entre los fonemas *n* y *o*, debido a que su duración es muy pequeña. De hecho, a todas las unidades fonéticas cuya duración sea inferior a 16 *ms* nunca se les asigna trama, y en las que tienen entre 16 y 32 *ms* depende de donde se localicen las segmentaciones (respecto del principio de la pronunciación, pueden tener una o ninguna trama, como el caso del fonema *b*).

También destaca en la Figura 4-3 la no existencia de silencios entre las palabras *segundo*, *en* y *Valladolid*. Para la segmentación automática es importante porque puede que otros locutores sí lo pronuncien y, por tanto, es necesario considerarlo. Además, se observa que el fonema *e* ha sido omitido y únicamente el fonema *m* se pronuncia en la palabra *"en"*. Al no existir silencio ni dicho fonema *e*, hay dos unidades fonéticas seguidas omitidas, pero que hay que considerarlas porque en otros casos puede que sí aparezcan.

Cada una de estas tramas correspondiente a una unidad fonética hay que asignarla a cada uno de sus estados, cuyo número se diseñó en función de la duración media que aparece en la base de datos.

Utilizando una red fonética de saltos simples para realizar la asignación de tramas a estados del modelo fonético, el algoritmo de Viterbi no es capaz de decodificar la secuencia de tramas de salida si el número de tramas es inferior al número de estados, debido a que no se avanza lo suficiente como para alcanzar el último estado del modelo fonético. Aunque la duraciones medias han tenido en cuenta este fenómeno asignando menos estados a los que tienen menor duración, muchas veces ocurre que el número de tramas es inferior al de estados. Para aprovechar estos vectores en el entrenamiento e incorporar el conocimiento sobre este tipo de fonemas de duración muy breve, se utiliza un método para asignar estos vectores a los modelos de unidades fonéticas en función del número de estados y de tramas:

- ***Número de Estados = 3***. El criterio utilizado es:
  - Si el número de tramas es ***mayor que tres*** se utiliza el algoritmo de Viterbi para realizar la segmentación en estados, ya sea la red de saltos dobles o de saltos simples.
  - Si el número de tramas es ***igual a tres***, en el caso de la utilización de saltos dobles se aplica Viterbi, si se utilizan saltos simples se asigna una trama a cada estado.
  - Si el número de tramas es ***igual a dos*** se asigna la primera trama al primer estado y la segunda trama al tercer estado. De esta manera, estas tramas que se encuentran en la frontera con los fonemas anteriores y posteriores se utilizan para caracterizar el inicio y fin del modelo.
  - Si el número de tramas es ***igual a uno*** se asigna al estado central. Este estado también caracterizará a las unidades fonéticas de menor duración.
- ***Número de Estados = 2***. El criterio utilizado es:
  - Si el número de tramas es ***mayor que dos*** se utiliza el algoritmo de Viterbi para realizar la segmentación en estados, sea la red de saltos dobles o simples.
  - Si el número de tramas es ***igual a dos*** se asigna una trama para cada estado.
  - Si el número de tramas es ***igual a uno*** se mira cuál de los dos estados modela mejor esa trama.
- ***Número de Estados = 1***. El criterio utilizado es:
  - Sea cual sea el número de tramas, todas ellas son asignadas al único estado que existe, por lo que no es necesario aplicar Viterbi ni ningún criterio como en los casos anteriores.

En los casos donde no exista ninguna trama, porque su duración sea inferior a los 16 *ms* (nunca le corresponde ninguna trama) o entre 16 y 32 *ms* (tal como se ha mencionado, a veces puede no corresponder trama alguna), no hay ningún problema, simplemente se pasa a segmentar la siguiente unidad fonética.

4.2.4.2 Segmentación Automática de las Frases con Saltos Simples para el Entrenamiento.

En el apartado anterior, al tener la frase ya segmentada en fonemas, la única asignación de tramas a estados se produce dentro de una unidad fonética. Sin embargo, esto es muy costoso en tiempo y recursos. Si se desea añadir una nueva base de datos con nuevos locutores se requiere realizar la segmentación de la frase en fonemas para incorporar dichos datos a los modelos.

Una solución consiste en realizar durante el entrenamiento de los modelos la segmentación en unidades fonéticas. En lugar de aplicar Viterbi a un modelo de dos o tres estados con las tramas indicadas según la segmentación en fonemas, se crea uno de palabra a partir de la concatenación de los estados de las unidades fonéticas según la transcripción fonética de la frase.

La primera aproximación al entrenamiento automático es la utilización de una red de reconocimiento con saltos simples, imponiendo que exista al menos una trama para cada modelo de estado.

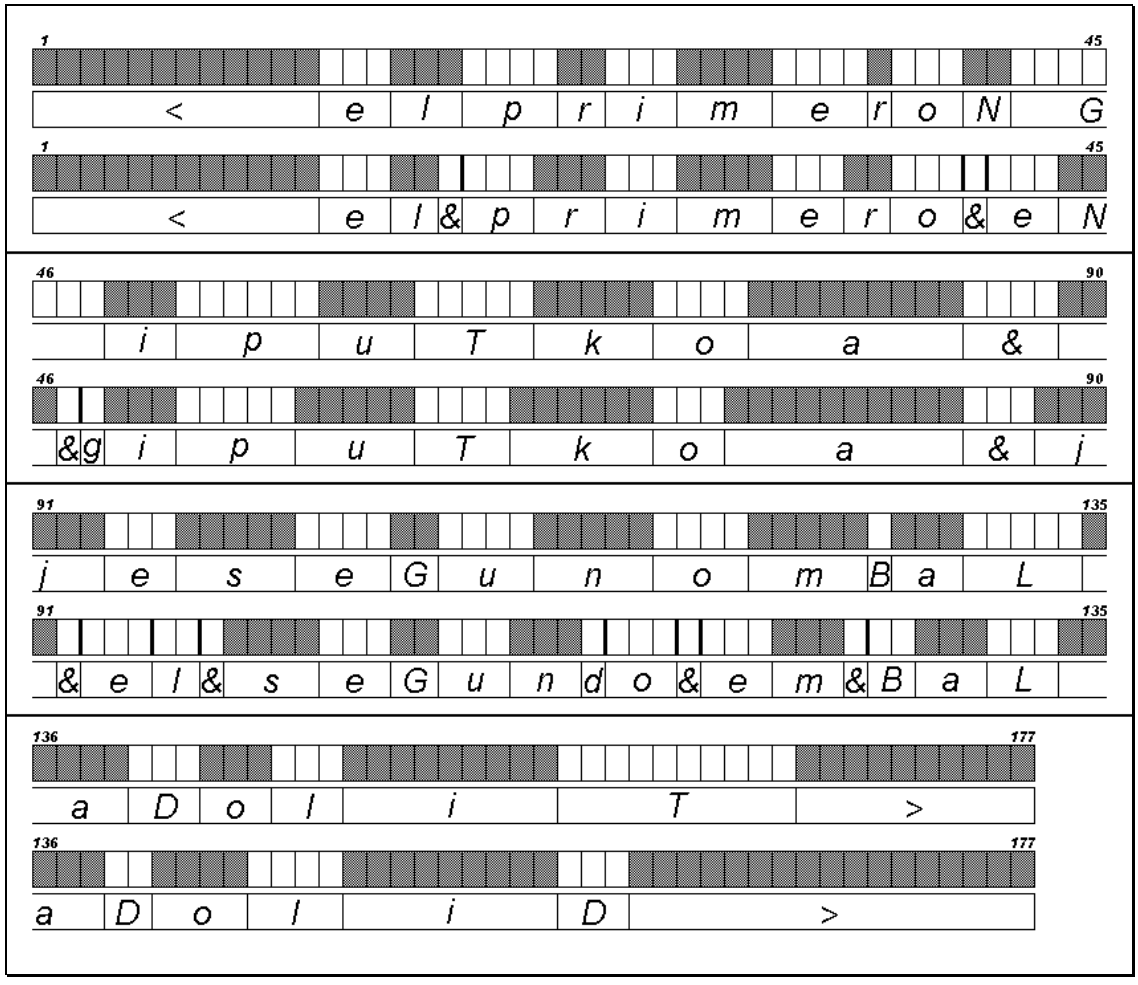

Figura 4-4. Comparación de la segmentación de la frase "*El primero en Guipúzcoa y el segundo en Valladolid*" realizada manualmente por un fonetista con la realizada por el segmentador automático de saltos simples.

En la Figura 4-4 se muestra la relación que existe entre la asignación de tramas a estados de una frase a partir de la segmentación manual hecha por un fonetista experto, y la segmentación en estados realizada de forma automática utilizando una red de reconocimiento de saltos simples a través del algoritmo de Viterbi. Comparando ambas se comprueba que:

- Para todos aquellos fonemas pronunciados lentamente, su segmentación funciona bien, por ejemplo en el caso de la palabra *primero*. Sin embargo, analizando la asignación en tramas dentro de cada estado, puesto que todos los estados requieren como mínimo una trama aunque no le corresponda, la asignación no es la óptima (algunas tramas son asignadas a estados que no les corresponde).
- Si la velocidad de pronunciación es rápida o se omiten fonemas, el resultado es crítico, ya que se obliga a asignar una trama por estado y eso supone un offset de asignación que se propaga durante varios fonemas. Por ejemplo, en *"...segundo en Gui..."*, donde a los fonemas *d*, *o*, *&* y *e* se les asignan varias tramas incorrectamente.
- Los fallos de transcripción al incorporar dicho offset afectan muy significativamente a la asignación de las tramas, como en el caso de la sustitución del fonema *g* por el fonema *G* en *Guipúzcoa*, donde se asignan tramas a los fonemas *N*, *&* y *g* sin corresponderles. Si se suma el número de tramas que provoca el fallo de transcripción y la velocidad de pronunciación se obtiene un conjunto de quince tramas seguidas mal asignadas.

La segmentación automática de las frases para el entrenamiento de los modelos fonéticos aporta una serie de ventajas sobre la segmentación manual:

- Al no tener que analizar previamente la frase y realizar una segmentación en fonemas, se puede incorporar cualquier base de datos para el entrenamiento de los mismos, pues ***la única información que se necesita es la transcripción fonética***, que es algo inmediato si se sabe lo que se ha pronunciado y, en cualquier caso, es menos costoso.
- Al incorporar un mayor número de bases de datos en entrenamiento, el número de apariciones de cada unidad fonética es mayor y los modelos tendrán ***más representatividad*** proveniente de diferentes locutores.
- Al entrenarse todas las unidades fonéticas al mismo tiempo, un cambio en una frontera entre dos unidades afecta a los dos modelos. La convergencia tiende a ***minimizar el error de forma global***, considerando todos los modelos a la vez en lugar de tener en cuenta sólo los de dentro de cada unidad fonética.

Sin embargo, el entrenamiento con segmentación automática utilizando redes de primer orden presenta los siguientes inconvenientes:

- Al imponer la existencia de al menos una trama por estado, si la frase ha sido pronunciada muy deprisa por el locutor puede ocurrir que se produzca un desplazamiento de todas las tramas hacia los modelos de los fonemas posteriores o anteriores. ***El entrenamiento*** en cada nueva iteración proporciona unos modelos cada vez peor entrenados y el mismo ***diverge***.
- Si el locutor pronuncia deprisa, puede que alguna característica de algún fonema o incluso algún fonema de duración muy breve (como una oclusiva) hayan sido omitidos. Utilizando esta segmentación, los modelos no estarían entrenados de forma óptima y se produciría una disminución del grado de discriminación y, por tanto, un aumento de la tasa de error.

4.2.4.3 Segmentación Automática de las Frases con Saltos Dobles para el Entrenamiento.

Una alternativa a la utilización del algoritmo de Viterbi con una red de reconocimiento de saltos simples consiste en utilizar saltos dobles. Los saltos dobles dan mayor flexibilidad puesto que es posible evitar estados intermedios.

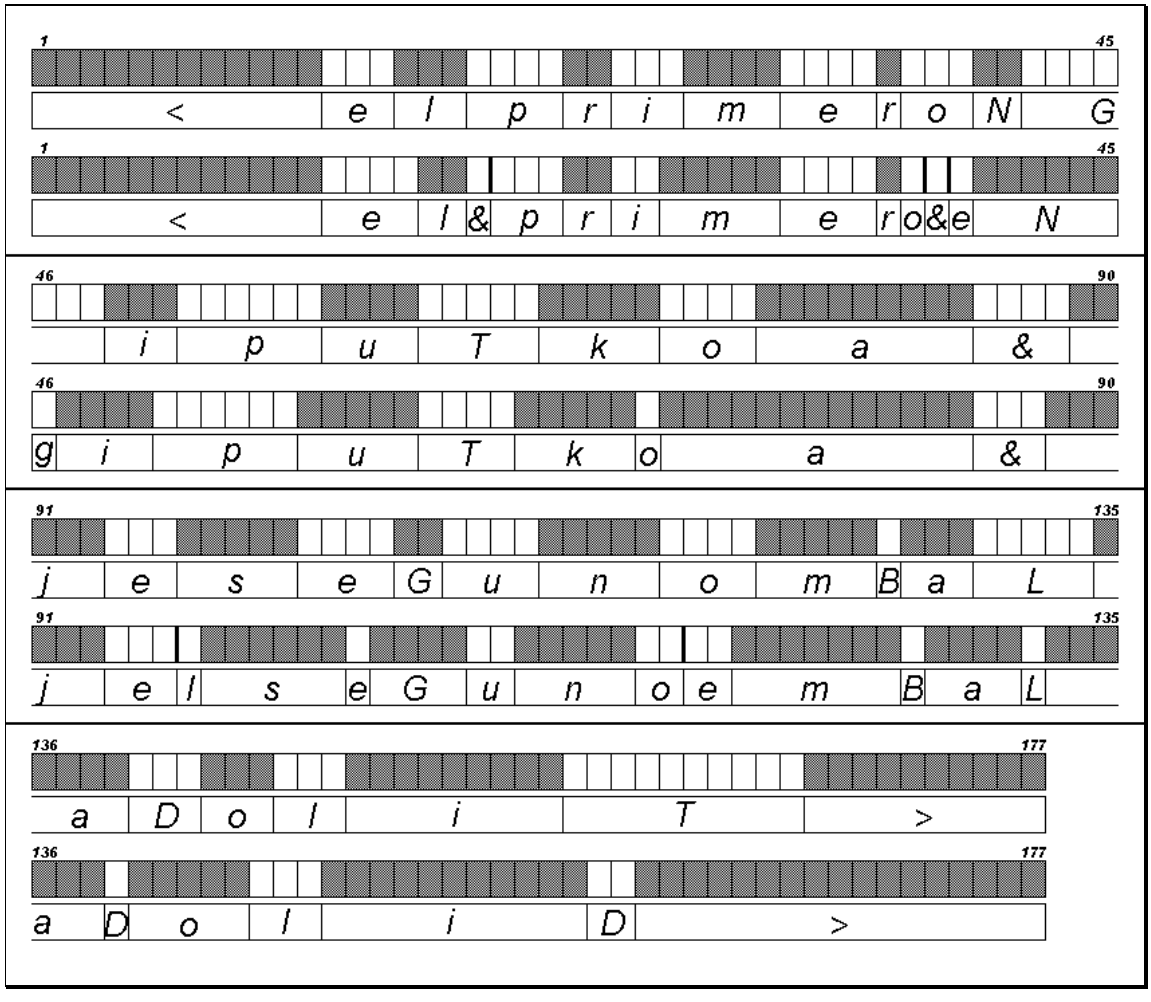

Figura 4-5. Comparación de la segmentación de la frase "*El primero en Guipúzcoa y el segundo en Valladolid*" realizada manualmente por un fonetista con la realizada por el segmentador automático de saltos dobles.

En la Figura 4-5 se compara la segmentación automática de una red de reconocimiento de saltos dobles con la correspondiente a la segmentación manual realizada por un experto fonetista. La frase sigue siendo la misma de la de la Figura 4-4 y, por tanto, entre dos palabras siempre se introduce un silencio intermedio *&*. En general, la segmentación automática realizada es muy similar a la manual, con una serie de diferencias que se enumeran a continuación:

- En el fonema *p* de *primero*, la primera trama ha sido asignada a un silencio intermedio, porque está formado por tres estados y el primero de ellos modela tramas de silencio previo a la oclusión.
- Aparece un fonema *e* en la palabra *en* cuando en la segmentación real no existe. Esto es debido a que se han impuesto dos fonemas que no existen, uno es el silencio intermedio *&*, el otro la *e*. Entre los dos tienen cuatro estados. Los saltos dobles obligan a la existencia de dos tramas (frente a las cuatro de los saltos simples).
- Se puede comprobar el fallo que supone un error de transcripción. En la frase real, el fonema inicial de la palabra *Guipúzcoa* es el fricativo sonoro *G*, mientras que en la transcripción se ha utilizado el oclusivo sonoro *g*. Ambos son tan diferentes que las tramas correspondientes a *G* se han asignado al fonema anterior, el *N*. Además, existen dos fonemas seguidos que son *&* y *g* con dos estados en total. Los saltos dobles obligan a la aparición de una trama (frente a las dos de saltos simples), que el segmentador asigna a *g* que es más similar al *&*.
- Se puede ver la influencia que la coarticulación produce sobre los fonemas. Tanto el fonema *o* de *Guipúzcoa*, como los fonemas *L* y *D* de *primera en Valladolid*, están altamente influenciados por las vocales vecinas (en el primer caso, por la *a*, en el segundo por *a* y *o*). Esta similitud hace que la segmentación haga desaparecer parte de las tramas que según el fonetista experto no le corresponden.
- En *"...el seg..."* ocurre lo mismo que en el caso de dos fonemas seguidos sin aparecer, uno es el *l*, el otro la *&*. Puesto que ocupan tres estados, el segmentador les ha asignado una trama (si hubiera sido de saltos simples, se le asignarían tres tramas).
- Finalmente, el último fonema de *Valladolid* se impuso en la transcripción fonética como el fonema *T*, aunque lo que se pronunció fue el fonema *D*. El resultado es que parte de las tramas que le corresponden a dicho fonema *T* son asignadas al fonema más parecido, en este caso, el silencio final <, frente al *D* que es mucho más energético.

La segmentación automática de saltos dobles permite que:

- Cuando un conjunto de ***modelos no existe***, el error que se comete sea menor, al asignar un número de tramas que no supera la mitad del número total de estados, frente al de saltos simples que le asigna un número de tramas igual al de estados.
- Los ***fallos de transcripción*** acomoden la asignación de manera que se adapten en cierta medida a los fonemas más similares. La incertidumbre que se introduce no es tan elevada como en el caso de los saltos simples.
- El fenómeno de la coarticulación asigne tramas a modelos aunque no se correspondan exactamente con los esperados o asignados por un fonetista experto.

En resumen, las ventajas que la segmentación de saltos dobles tiene sobre la de saltos simples son:

- ***Permite la utilización de todas las frases de entrenamiento***: Se ha comprobado que muchas de las frases están pronunciadas a tal velocidad que el número de tramas es incluso menor al número de estados. Con una red de saltos simples estas frases no pueden ser utilizadas, porque no se ha recorrido la red de reconocimiento en su totalidad para poder sacar la segmentación, y los errores de offset y de fallos de transcripción serían muy críticos.
- ***El entrenamiento es convergente***: Al no existir un offset motivado por la asignación de tramas a estados, aún cuando no corresponda realmente a este modelo, los modelos de una iteración a la siguiente van mejorando. En el de saltos simples, el error llega a ser muy adverso.
- ***Los modelos están mejor entrenados***: Por la misma razón que la del párrafo anterior, las tramas son asignadas al mejor modelo, siendo el algoritmo de Viterbi capaz de dejar algún estado intermedio sin ninguna asignación. De este modo, los modelos reflejan mejor las características fonéticas de los locutores, incluyendo los efectos de coarticulación, y se produce un mayor grado de discriminación. Los fallos de transcripción no son tan importantes.

4.2.5 Influencia de los Saltos Dobles en Reconocimiento.

De la misma manera que en entrenamiento los saltos dobles posibilitan que el algoritmo de Viterbi se adapte a las propiedades fonéticas de la secuencia de vectores de características, en reconocimiento se produce el mismo fenómeno, ya que prácticamente no existe diferencia entre entrenamiento y reconocimiento.

La única diferencia entre ambos radica en que en entrenamiento se fija a priori la palabra o frase a reconocer, puesto que se conoce de antemano, y lo que se desea es segmentarla de forma óptima en función de los modelos. Sin embargo, en

reconocimiento lo que se pretende es que entre el conjunto de palabras candidatas se extraiga la secuencia que haya sido generada con mayor probabilidad.

Todas las ventajas obtenidas en el entrenamiento, gracias a que con la utilización de los saltos dobles se realiza una asignación más óptima de las tramas a estados cuando el locutor habla más deprisa o cuando omite alguna característica fonética, son válidas también en reconocimiento. Aunque en todos los modelos de la red de reconocimiento se obtiene una mejor probabilidad, ya que con saltos dobles se aumenta los grados de libertad del algoritmo de Viterbi, para la palabra correcta la mejora es incluso superior al resto, por lo que el grado de discriminación del reconocedor aumenta.

Igual que ocurre en entrenamiento, en reconocimiento se pueden introducir silencios intermedios entre palabras, puesto que el algoritmo de Viterbi es capaz de saltarse estos estados cuando no aparece ninguno, por lo que la robustez del reconocedor respecto a diferentes velocidades de locución y de pausas entre palabras se puede incorporar fácilmente en la red de reconocimiento, sin que ello provoque problemas de asignación de tramas a estados que no deberían existir.

4.3 Robustez Frente a la Variabilidad del Canal de Comunicaciones

Este apartado está dedicado a la influencia del canal de comunicaciones sobre el reconocimiento de voz. El desarrollo seguido es el siguiente: primeramente, se analiza la influencia del canal de comunicaciones sobre el funcionamiento del sistema de reconocimiento y, en particular, en aplicaciones telefónicas; a continuación, tras realizar una caracterización del canal de comunicaciones, se enumeran las alternativas que existen para disminuir su influencia en el entrenamiento y en el reconocimiento; finalmente, se estudian dos técnicas sencillas y de adaptación muy rápida que resultan interesantes en aplicaciones telefónicas, como son el filtrado *RASTA* y la técnica de *restado cepstral* o *CMN*.

4.3.1 Influencia del Canal de Comunicaciones en el Reconocimiento.

En un sistema de reconocimiento de voz se pueden plantear tres escenarios posibles en función de las características del canal de comunicaciones en entrenamiento y en reconocimiento:

- Voces limpias en entrenamiento y en reconocimiento.
- Condiciones idénticas del canal en entrenamiento y en reconocimiento.
- Condiciones diferentes del canal en entrenamiento y en reconocimiento.

La situación ideal de un sistema de reconocimiento de voz es *el entrenamiento y el reconocimiento a partir de voces limpias*. Por voces limpias se entiende todas aquellas pronunciaciones realizadas en ambientes donde el nivel de ruido es bajo o prácticamente nulo, donde no existen interferencias por canales adyacentes o voces de fondo, los micrófonos utilizados son todos iguales y presentan una respuesta plana y las características acústicas de la sala de locución no distorsionan la señal. En definitiva, que la función de transferencia del canal de comunicaciones sea plana, siendo la señal a la salida del canal idéntica a la de entrada. Cuando se dan estas circunstancias se dice que el *canal de comunicaciones es transparente y no influye en el funcionamiento del sistema de reconocimiento*.

Otra circunstancia en la que un sistema de reconocimiento empieza a verse influenciado por el canal de comunicaciones es cuando aquel presenta unos niveles de ruido, distorsiones lineales e interferencias y, además, *similares en entrenamiento y en reconocimiento*. Si esto ocurre, los modelos entrenados habrán captado las características del canal de comunicaciones, pero puesto que en reconocimiento se dan las mismas, se dice que existe un *acoplo entre entrenamiento y reconocimiento*. Por tanto, *la influencia sobre el reconocimiento será debida solamente a los niveles de ruido, distorsiones e interferencias que existan*.

Sin embargo, la situación más real es que exista un cierto nivel de ruido, aparezcan distorsiones debidas al micrófono y a la acústica de la sala, se produzcan interferencias por voces de fondo y éstas sean *diferentes en entrenamiento y en*

reconocimiento. En este caso, se dice que *el sistema de reconocimiento está desacoplado*. Al llevar a cabo el reconocimiento en unas circunstancias diferentes a las de entrenamiento, la degradación del sistema es elevada y aumenta en mayor medida cuanto más distintas sean las condiciones.

De los tres escenarios presentados, el más normal para una aplicación telefónica es el tercero, ya que la conexión que se establece a través de la *RTC* es diferente en cada nueva comunicación, con niveles de ruido, distorsión e interferencias muy variables, tal como ya se comentó en el apartado 3.2.1 dedicado a la caracterización del canal telefónico. La variabilidad de micrófono es también muy elevada, al presentar cada uno unas características diferentes de distorsión. Finalmente, las condiciones de la sala son tan cambiantes como los lugares donde se localizan los teléfonos y, por tanto, las características acústicas de la sala y el nivel de interferencias por voces y música de fondo modifican la señal captada por el reconocedor.

Resulta fundamental, principalmente en aplicaciones telefónicas, realizar una compensación del canal de comunicaciones y que se produzca la adaptación entre el entrenamiento y el reconocimiento de manera que se disminuya su influencia sobre el funcionamiento del sistema y, además, sea un procedimiento simple y de adaptación rápida, ya que el tiempo disponible no es muy grande.

4.3.2 Metodologías de Adaptación del Canal.

A la hora de aplicar una determinada técnica o metodología para solventar o disminuir los efectos que el canal pudiera introducir en el reconocimiento de voz, es preciso analizar primeramente cómo se puede caracterizar y qué ecuaciones son las que hay que considerar.

4.3.2.1 Tipos de Alteraciones.

Todo canal telefónico y, en general, todo canal de comunicaciones, puede ser caracterizado por un modelo que represente todas sus características. Sobre este modelo se pueden identificar las siguientes perturbaciones o efectos:

- ***Distorsión lineal***: El canal de comunicación presenta una respuesta en amplitud diferente para cada una de las frecuencias. En la extracción de características ya se tuvo en cuenta esta posible variabilidad, llegándose a la conclusión de que lo mejor era no considerar aquellas bandas de frecuencia de mayor variabilidad y que pudieran introducir mayor incertidumbre en el reconocimiento. A la distorsión lineal también se la denomina ***ruido convolucional***, puesto que, como posteriormente se verá, si se le aplica el logaritmo del espectro de la respuesta del filtro, la distorsión se convierte en un factor aditivo, como si fuera un ruido. En los *cepstrum*, al trabajar con una expresión relacionada con el logaritmo del espectro de la señal, este factor también es aditivo.

- ***Ruido aditivo***: Señal aleatoria estacionaria que se supone independiente de la señal principal y que, por tanto, se suma en potencia a ésta.
- ***Señales interferentes***: Señales no estacionarias que se introducen en el canal de comunicaciones y modifican la señal de voz que se pretende reconocer. No son propias del canal de comunicaciones, ya que pueden aparecer también en la fuente y en el destino. Resultan interesantes los trabajos en [Raj97] para analizar la influencia de la música de fondo en el funcionamiento del reconocimiento de voz.
- ***Acústica de la sala***: Un micrófono que se localice en una sala donde se pronuncia una señal de voz capta la señal original y el conjunto de todas sus reflexiones (*ecos* y *reverberaciones*), cuya distribución depende de las dimensiones de la sala, de los objetos y materiales que existan y su distribución. Es un tema que no se suele tener en cuenta en el funcionamiento de los reconocedores. En [Rec92] se analizan alternativas para la caracterización de salas.

Los cuatro problemas anteriores son importantes y pueden afectar al funcionamiento de un sistema de reconocimiento de voz cuando no se tienen en cuenta. Los problemas que conviene estudiar son la disminución de los efectos del ruido y de la distorsión lineal, que, históricamente, han sido los primeros en intentar solucionarse.

Sobre las señales interferentes existen muchas alternativas basadas en la captación de la señal con múltiples micrófonos para extraer la señal de las interferencias, pero resultan métodos muy complicados y, en general, poco efectivos. No resultan prácticos cuando se intentan utilizar en aplicaciones telefónicas.

La acústica de la sala puede caracterizarse como una distorsión lineal cuando las dimensiones de la misma son pequeñas o las atenuaciones son elevadas. Cuando aparecen problemas de eco resulta muy difícil su compensación.

4.3.2.2 Modelo de Canal de Comunicaciones.

A continuación, se plantean diferentes alternativas para modelar la influencia del ruido aditivo y de las distorsiones lineales o ruido convolucional.

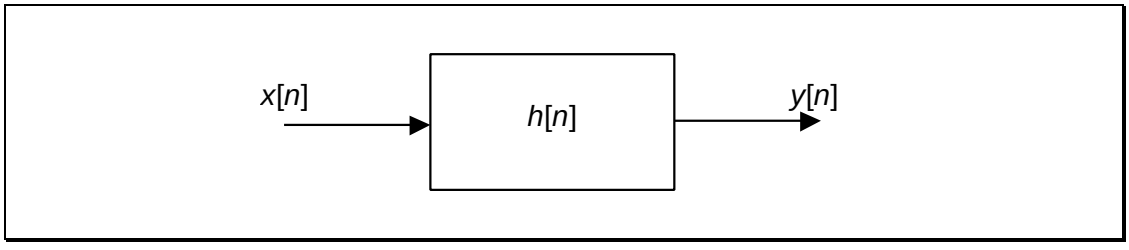

Figura 4-6. Caracterización de un canal de comunicación con distorsión lineal.

La Figura 4-6 representa el esquema de canal de comunicaciones más simple, caracterizado únicamente como un filtro lineal invariante que hace distorsionar linealmente a la señal. La expresión sería:

$$y[n] = x[n] * h[n] \qquad |Y(w)| = |X(w)| \cdot |H(w)| \qquad [4.1]$$

donde $x[n]$ representa la señal de voz temporal, $y[n]$ la señal una vez que ha pasado a través del canal y $h[n]$ la respuesta al impulso del canal. Los valores en mayúsculas representan la transformada de Fourier de los mismos. En este caso, bastaría con aplicar un filtro de ecualización que presentara la función de transferencia inversa.

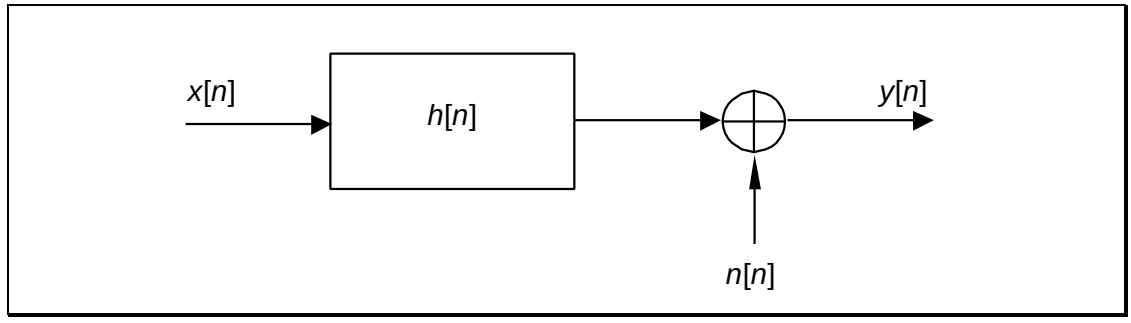

Figura 4-7. Caracterización de un canal de comunicaciones que presenta distorsión lineal y ruido aditivo que se añade a la salida.

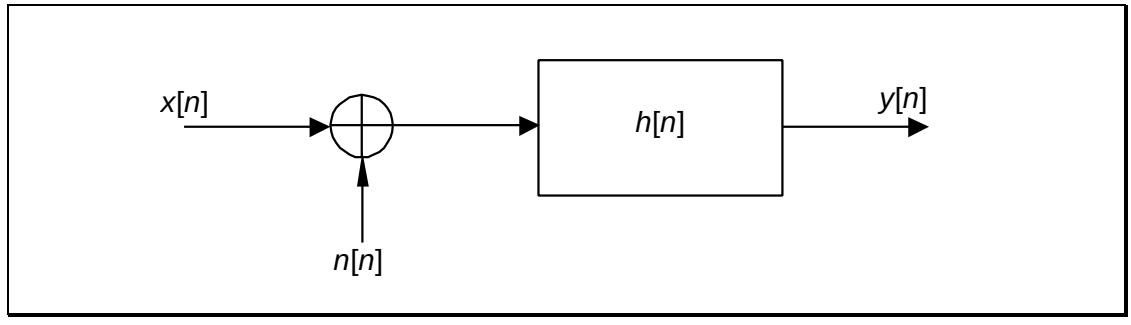

Figura 4-8. Caracterización de un canal de comunicaciones que añade ruido aditivo previo a la distorsión lineal.

Si al esquema de canal de la Figura 4-6 se le incorpora el ruido aditivo a la salida del filtro, se obtendría el esquema de la Figura 4-7. Si se coloca primero el ruido y luego el filtrado quedaría el de la Figura 4-8. Ambos esquemas son equivalentes, al tratarse de un sistema lineal invariante. Sin embargo, el valor del ruido gaussiano y aditivo ($n[n]$ y su transformada de Fourier $X(w)$) no es el mismo en ambos casos. La expresión para la Figura 4-7 sería:

$$y[n] = x[n] * h[n] + n[n] \qquad |Y(w)| = |X(w)| \cdot |H(w)| + |N(w)| \qquad [4.2]$$

Mientras que para la Figura 4-8 resultaría:

$$y[n] = (x[n] + n[n]) * h[n] \qquad |Y(w)| = |X(w)| \cdot |H(w)| + |N(w)| \cdot |H(w)| \qquad [4.3]$$

El ruido de la expresión [4.2] es el ruido de la expresión [4.3] que ha atravesado el filtro lineal junto con la señal. Si se denomina al $log(|X(w)|)$ como X_t, al $\log(|Y(w)|)$ como Y_t, al $\log(|H(w)|)$ como H_t y al $\log(|N(w)|)$ como N_t, en las expresiones [4.1], [4.2] y [4.3] y se les aplica la función logaritmo, se obtienen las siguientes ecuaciones:

$$Y_t = X_t + H_t \qquad [4.4]$$

$$Y_t = \log(\exp(X_t + H_t) + \exp(N_t)) \qquad [4.5]$$

$$Y_t = H_t + \log(\exp(X_t) + \exp(N_t)) \qquad [4.6]$$

La ecuación [4.4] es el resultado de aplicar el logaritmo y sustituir el nombre de las variables a la expresión [4.1], la [4.5] en el caso de la [4.2] y la [4.6] en el de la [4.3].

Entre las expresiones [4.5] y [4.6] interesa más la segunda, puesto que el término de distorsión lineal (ruido convolucional) es aditivo sobre la señal. Si esta última expresión se elabora un poco más se obtiene:

$$Y_t = H_t + X_t + \log(1 + \exp(N_t - X_t)) \qquad [4.7]$$

La ecuación [4.7] resulta muy interesante, ya que teniendo unos parámetros de la señal relacionados con el logaritmo del espectro, los efectos de ruido aditivo y ruido convolucional se introducen aditivamente a la señal y se podrían eliminar mediante su cancelación si fueran conocidos.

Ahora bien, las características de distorsión lineal del canal son constantes o varían muy lentamente, es decir, es información de bajas frecuencias y, por el contrario, el ruido aditivo suele aparecer en alta frecuencia. Si se tiene en cuenta que la mayor parte de la información se localiza en una zona central del espectro, sería posible eliminar ambos términos mediante la utilización de un filtrado paso banda, siempre y cuando el nivel de ruido no sea muy elevado. Esto tiene sentido cuando la relación señal a ruido sea suficientemente elevada.

4.3.2.3 Alternativas de Compensación de Canal.

En [Pon97] y en [Ste96] se presentan unos estudios sobre las diferentes metodologías que existen a la hora de abordar el problema de conseguir que el canal de comunicaciones afecte lo menos posible al reconocimiento de voz, corrigiendo la desadaptación que pudiera existir entre el entrenamiento y el reconocimiento. Una enumeración de la misma es la siguiente:

- ***Entrenamiento en las mismas condiciones***: Consiste en entrenar los modelos con bases de datos que presenten las condiciones que existen en reconocimiento. Si bien cuando se utilizan estos modelos en las mismas condiciones en la que fueron entrenados se obtienen unos resultados muy buenos, en unas condiciones algo diferentes la degradación empieza a ser

elevada, empeorando los resultados que hubieran sido obtenidos utilizándose modelos entrenados en condiciones de voces limpias. Otra desventaja es la necesidad de conocer por adelantado las condiciones del canal donde el sistema va a funcionar, algo que en la mayoría de los casos no es posible.

- ***Características robustas***: Estas metodologías consisten en adaptar la extracción de características a las condiciones con las que el reconocedor trabaja. Sin embargo, existen otros procesados sobre dichas características que permiten que sean aún mejores. Entre estos nuevos parámetros, obtenidos de la modificación de otros ya calculados destaca, por encima de todos, el filtrado *RASTA* [Her94], que parte del conocimiento del tipo de distorsiones que presenta un determinado canal de comunicaciones para minorizarlas. Es una de las metodologías más sencillas e interesantes para aplicaciones donde se puede conocer qué tipo de distorsiones pueden aparecer, pero no se sabe de manera exacta cómo serán. Existen otras alternativas como en [Ver96] donde se propone una modificación de los cepstrum para aumentar la energía en altas frecuencias sin aumentar el ruido.

El filtrado *RASTA* se introduce dentro de esta categoría porque inicialmente en su concepción estaba inmerso en el cálculo de la extracción de características. Sin embargo, para otros parámetros se podría comportar como adaptación de canal.

- ***Adaptación de los modelos***: Estas técnicas se utilizan cuando no se puede reentrenar los modelos para todas las condiciones posibles, resultando más interesante entrenar unos únicos modelos con voz limpia y combinarlos con un modelo que represente las condiciones esperadas del canal, introduciendo el mismo dentro del algoritmo de reconocimiento. En [Mor96] se muestra una metodología de combinación de modelos de voz limpia y de canal en paralelo (*PMC*).
- ***Adaptación del canal****:* Estas técnicas buscan estimar una corrección o modificación de las tramas o de la frase antes de realizar el entrenamiento o el reconocimiento.

En muchos sistemas, el cálculo de las probabilidades se basa en la diferencia entre los vectores de características y unos vectores de medias. Por eso, muchas veces los sistemas que se consideran como adaptación del canal pueden llegar a englobarse como adaptación de modelos en los que únicamente se modifican las medias. Esta correspondencia es total cuando la adaptación del canal consiste en añadir una desviación a los vectores de características generados.

El filtrado *RASTA*, como posteriormente se verá, es una modificación del canal, pero no de forma constante, sino que depende de los vectores anteriores, por eso no se engloba exactamente dentro de estas metodologías, pero tiene una similitud muy elevada con ellas.

En los métodos donde el ruido convolucional y el ruido aditivo es independiente de la señal, lo mejor es realizar una estimación de dichos valores. Existen métodos que tienen la peculiaridad de ser independientes del contenido de la voz, como son los de normalización de la media cepstral o *CMN* [Liu93], pensados para la eliminación del ruido convolucional. Existen otros conjuntos de técnicas que ya sí tienen en cuenta la señal que se está produciendo. En la mayoría de estos métodos es necesario trabajar con BD estéreo[3]. En [Ace90] y [Mor96] se analizan diferentes alternativas como *SDCN* (*SNR-Dependent Cepstral Normalization*), o normalización cepstral dependiente del nivel de señal a ruido de la señal; *CDCN* (*Codeword-Dependent Cepstral Normalization*), donde se hace una normalización específica para cada media de unos modelos continuos o semicontinuos; *PDCN* (*Phone-Dependent Cepstral Normalization*), donde la normalización es dependiente del fonema al que pertenezcan dichos modelos y unos derivados de éstos. En [Ace90], [Ace90b], [Ace91], [Liu93] y [Liu94] pueden verse resultados comparativos entre estos métodos y los independientes como son *RASTA* y *CMN*.

En [Rah94], y más detalladamente en [Rah96], se describe una técnica denominada *SBR* (*Signal Bias Removal*). Esta técnica está basada en la obtención de la desviación a partir de la estimación de máxima probabilidad (*MLE*: *Maximun Likelihood Estimation*). Esta técnica puede ser aplicada tanto a modelos discretos, semicontinuos y continuos. En ciertas circunstancias, cuando la desviación se calcula por frase, es equivalente al *CMN*.

En [Mor95] se presenta una técnica denominada *RATZ*, que se basa en asumir que los parámetros de la voz se pueden modelar mediante una distribución gaussiana multivariable con mezclas, y que los efectos del ambiente en la voz limpia se pueden modelar por compensaciones aditivas en los vectores de medias y en las matrices de varianzas. La técnica busca obtener los parámetros de voz limpia y voz ruidosa y así compensar esta última. Por tanto, para esta técnica también se requieren bases de datos estéreo.

- ***Realce de la voz*** **(***Speech Enhancement***)**: Estas técnicas están íntimamente relacionadas con la obtención de una calidad de voz elevada desde un punto de vista subjetivo, como lo agradable que resulta, u objetivo, como medidas de distorsión o valores de relación señal a ruido. Su fin no está orientado a la

[3] Esta BD estéreo suele obtenerse a partir de una BD de voz limpia, grabada en unas condiciones con ausencia de cualquier tipo de ruido o de distorsión. Luego se le aplica dicho ruido o distorsión de forma simulada para tener parametrizado y controlado el grado de suciedad que se le ha añadido.

mejora del reconocimiento en sí, pero una vez aplicadas pueden llegar a contribuir a una mejora del mismo.

El método de substracción espectral formaría parte de este conjunto de técnicas que buscan aumentar la calidad de la voz. Se han propuesto diferentes alternativas como se describe en [Vas96]. Algunas consiguen, en ciertos casos, muy buenos resultados. En [Bec98] se describe cómo mejorar la substracción espectral con un modelo para el ruido aditivo.

- ***Sistemas con múltiples micrófonos***: Es una técnica que intenta obtener la señal con información y eliminar el ruido, utilizando para ello varios micrófonos. El principal inconveniente radica en que a veces es imposible tener varios micrófonos (como en el caso de una aplicación telefónica) y, además, requieren conocer las dimensiones de la sala y la distribución de los micrófonos dentro de la misma para poder calcular las distancias entre la fuente y cada uno de ellos, y entre éstos en sí, resultando muy poco práctico en la mayoría de los casos.

4.3.3 Filtrado RASTA.

El procesado *RASTA* surgió tras el estudio del modelo auditivo humano, a partir del cual se comprobó que el oído presenta la máxima sensibilidad para captar información lingüística alrededor de los $4\ Hz$, y que ésta disminuye cuando la frecuencia es mayor o menor. Además, se añade el resultado obtenido por diversos experimentos que mostraban que la percepción de sonidos similares a los de la voz dependía del sonido que acaecía previamente.

Partiendo del hecho de que el lenguaje y la información que se quiere transmitir tienden a ser enviados dentro de esta banda de frecuencias de mayor sensibilidad, surge la posibilidad de eliminar las señales que se localizan lejos de esta banda al no aportar información relevante, disminuyendo el grado de variabilidad de los parámetros. Como ya se vio en el apartado anterior, normalmente la zona de bajas frecuencias se asocia con el ruido convolucional del canal, mientras que las altas frecuencias se asocian principalmente con el ruido aditivo. Es entonces cuando surge la técnica *RASTA* (*RelAtive SpecTrAl*: diferencia espectral), que es un filtrado paso banda propuesto por H. Hermansky [Her92].

Esta técnica inicialmente estaba integrada dentro de una extracción de características denominada *RASTA-PLP*, que puede estudiarse más con detenimiento en [Her92] [Her93] y [Her94]. Esta extracción busca la obtención de unos parámetros denominados *RASTA-PLP* o coeficientes cepstrum, pero que no coinciden con los mel-frequency-cepstrum descritos en el apartado de extracción de características. Para el cálculo de estos parámetros propuestos, tras la transformada de Fourier de la señal enventanada y una integración por bandas críticas, se aplican una compresión logarítmica y un filtrado paso banda. A continuación, se realiza una descompresión

usando la función antilogarítmica (o exponencial) para volver a escala lineal, y luego se procede al cálculo de los coeficientes *RASTA-PLP*, usando para ello la recursión de Durbin. La denominación de *PLP* procede del hecho de aplicar al espectro comprimido una curva de ponderación similar a la ley de potencia del sistema auditivo antes de hacer la descompresión.

Para aplicar el filtrado paso banda en la extracción de estos parámetros se han considerado diferentes alternativas sobre distintos tipos de compresión, desde la ya comentada compresión logarítmica, hasta sistemas con compresión lineal para valores pequeños y logarítmicos para grandes, también denominada *J-RASTA*. En estos sistemas, es necesario realizar el cálculo del umbral óptimo entre ambas zonas para conseguir una disminución de la tasa de error, donde J se corresponde con el parámetro de ajuste. La expresión de compresión tiene la siguiente forma:

$$Y = \log(1 + J \cdot X) \qquad [4.8]$$

Para la extracción de características propuesta, la aplicación directa de cualquiera de las técnicas de filtrado *RASTA* anteriormente descritas resultaría complicada. Sin embargo, como se menciona en [Her94], el procesado *RASTA* puede realizarse sobre trayectorias temporales de cualquier otro conjunto de parámetros, aunque teniendo en cuenta que los efectos pueden llegar a ser algo diferentes. Puesto que los coeficientes mel-frequency-cepstrum tienen relación con el desarrollo en serie del logaritmo del espectro, el filtrado *RASTA* se puede aplicar directamente a éstos, excepto al *cepstrum cero* o energía total, que se deja tal como está. El cálculo de los parámetros delta-mel-frequency-cepstrum no sufre ninguna modificación respecto a lo descrito en el apartado 3.2.6, si bien, ahora se utilizan los cepstrum filtrados.

4.3.3.1 Ajuste del Filtro RASTA.

El filtro usado en el cálculo de los coeficiente *RASTA_PLP* que aparece en [Her94] tiene la siguiente expresión:

$$H(z) = 0.1z^4 * \frac{2 + z^{-1} - z^{-3} - 2z^{-4}}{1 - Kz^{-1}} \qquad [4.9]$$

En la ecuación [4.9] pueden distinguirse dos términos diferentes:

- En el denominador se encuentra una constante K que se asocia con la frecuencia de corte inferior, cuyo valor indica el grado de integración de la información de las tramas anteriores. Cuando el valor de esta constante es $0{,}98$ le corresponde un grado de integración de unos $500\ ms$, que es el valor teórico para el cual se filtraría de manera óptima. Sin embargo, en [Her94] se demostró experimentalmente que con un valor de $0{,}94$, que se corresponde con $160\ ms$, se obtenían los mejores resultados.

- En el numerador se observa una regresión de orden dos similar a la de los coeficientes delta-cepstrum.

En el cálculo de los coeficientes delta-mel-frequency-cepstrum se realiza una regresión de hasta orden dos, por tanto, el filtro no necesita una expresión como la del numerador, pues ya se está considerando. En [Liu93], donde se habla del filtrado *RASTA* como un método de filtrado cepstral, se menciona un filtro paso alto más sencillo que el de la ecuación [4.9], que resulta ser de orden uno. Su función de transferencia tiene la siguiente expresión:

$$H(z) = \frac{1}{1 - Kz^{-1}} \qquad [4.10]$$

donde la constante K indica el grado de filtrado de los parámetros. Es un filtro paso alto en el que dependiendo del valor de la constante K le corresponde una frecuencia de corte inferior mayor o menor.

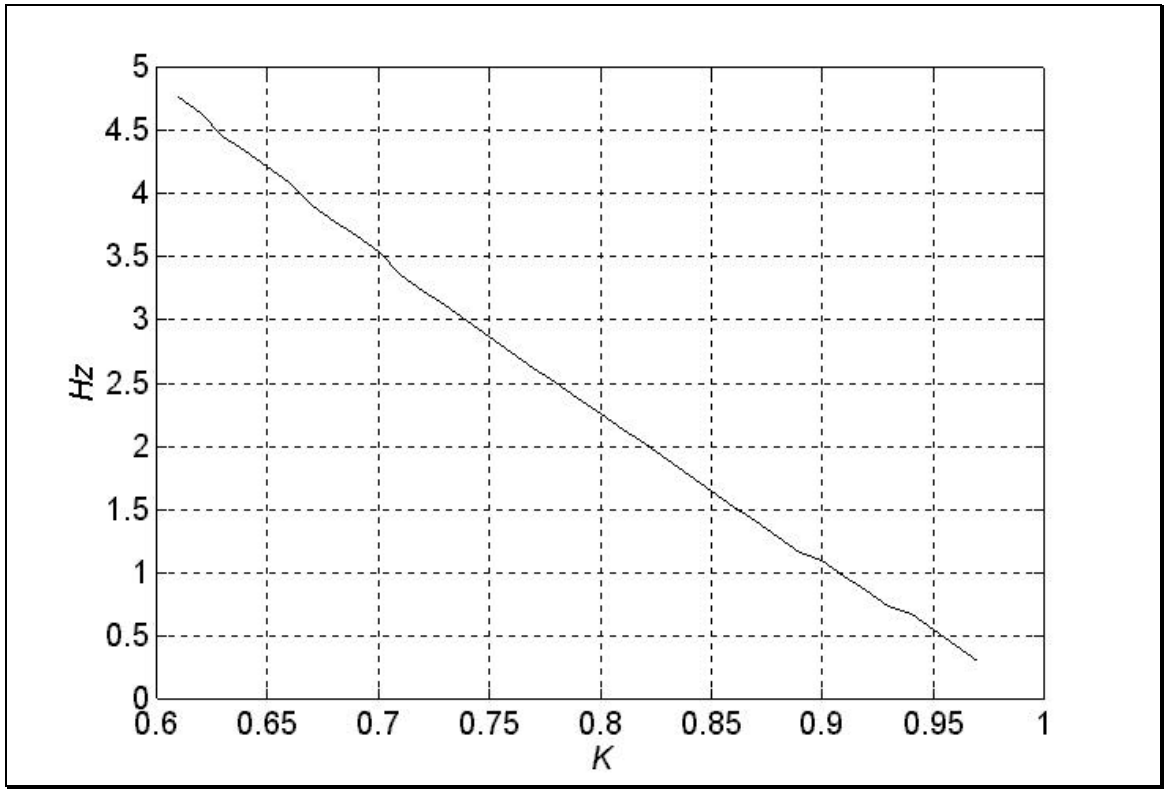

Figura 4-9. Diagrama que muestra la frecuencia de corte del filtro paso alto con la función de transferencia en función del valor de la constante *K*.

En la Figura 4-9 se puede ver la representación de la frecuencia de corte del filtro paso alto en función de la constante K para la ecuación [4.10]. Se considera que la frecuencia de muestreo es de $62{,}5\ Hz$, que se corresponde con el número de tramas generadas en un segundo (una trama cada $16\ ms$) con la extracción de características utilizada, teniendo en cuenta que las tramas se obtienen cada $16\ ms$. En esta gráfica puede verse que la frecuencia de corte es inversamente proporcional a la constante K.

Puesto que la información se localiza en la banda alrededor de los $4\ Hz$, se debe usar, al menos, un valor de K igual a $0{,}7$, aunque será la experiencia la que indicará cuál es el valor óptimo.

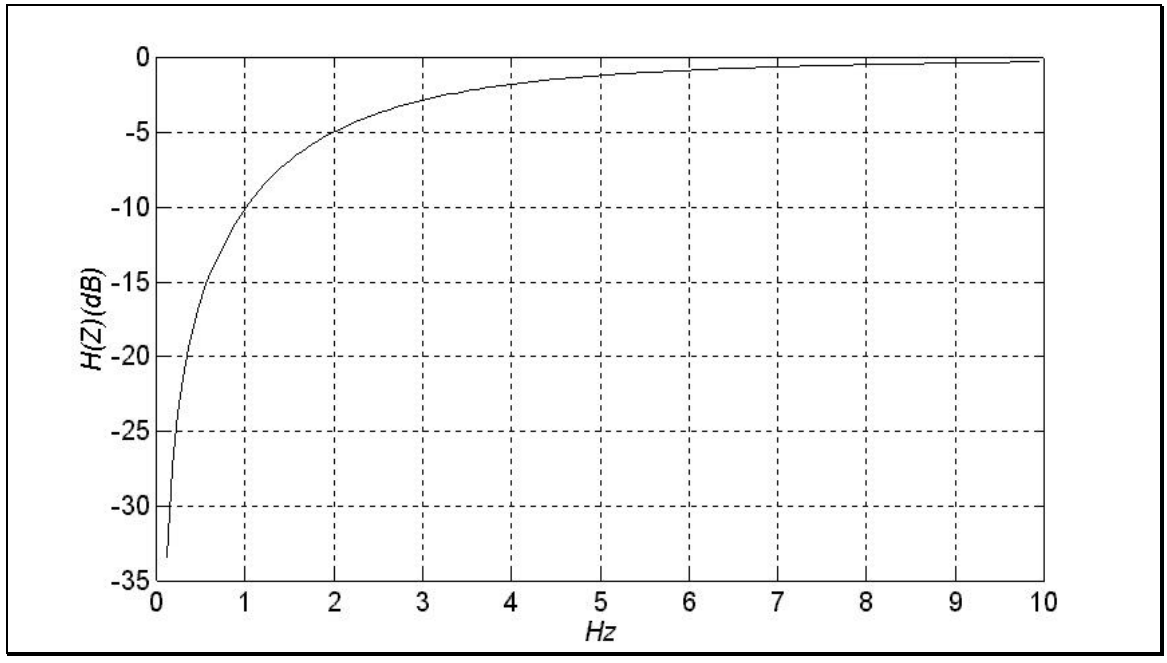

Figura 4-10. Función de transferencia normalizada del filtrado *RASTA* para una *K* igual a 0,75. Se observa que la frecuencia de corte está ligeramente por encima de los 2,8 *Hz* y, por tanto, no elimina la zona de los 4 *Hz* que corresponde a la máxima sensibilidad de la información lingüística.

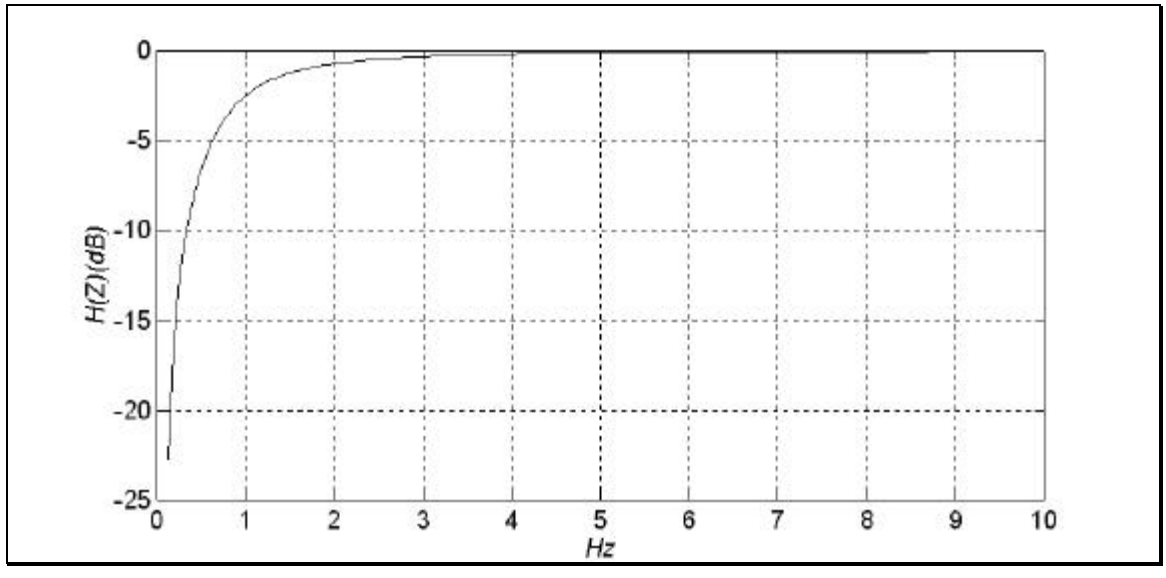

Figura 4-11. Función de transferencia normalizada del filtrado *RASTA* para una *K* igual al 0,92. Se observa que la frecuencia de corte está ligeramente por debajo de 1 *Hz*.

En la Figura 4-10 se observa la función de transferencia del filtro para una K de $0{,}75$ y en la Figura 4-12 para una K de $0{,}92$. En el primer caso, le corresponde una frecuencia de corte de $2{,}8\ Hz$, donde cae la curva $3\ dB$, mientras que en el segundo caso es de $0{,}8\ Hz$. En ambas situaciones se observa cómo este filtro tiene un cero a la

frecuencia de $0\ Hz$, es decir, elimina la componente continua. Este aspecto es interesante para compararlo con el funcionamiento de la otra técnica aplicada *Cepstral Mean Normalization* (*CMN*).

Existe un factor multiplicativo del filtro que no se ha incluido en la expresión del filtrado *RASTA* de la ecuación [4.10]. Como se observa en la Figura 4-10 y en la Figura 4-11, las funciones de transferencia están normalizadas. Esto es así porque para representarlas se ha eliminado un factor de ganancia, de manera que el máximo sea siempre $0\ dB$. Realmente, en función de la constante K, la función de transferencia presenta una ganancia mayor a la unidad en altas frecuencias, sin embargo, esto no es ningún problema, ya que al calcular el vector de ponderación este incremento no considerado es compensado, quedando los vectores correctamente normalizados.

La ventaja de utilizar un procesado como este filtrado es que su funcionamiento es en tiempo real, pues sólo requiere información pasada para la estimación actual y, además, usa muy poca memoria, sólo los valores cepstrum anteriores filtrados y no filtrados.

Se puede pasar de la función de transferencia a la ecuación en diferencias para obtener de esta manera la implementación del filtro. La ecuación resulta ser:

$$y(t) = K \cdot y(t-1) + x(t) - x(t-1) \qquad [4.11]$$

4.3.3.2 Variantes del Filtrado RASTA.

Además de aplicar el filtrado tal como se acaba de describir, se han propuesto alternativas para eliminar ciertos efectos de la aplicación del mismo.

El principal inconveniente surgió cuando en los experimentos realizados para obtener las tasas de error [Her94] se comprobó que ciertos ficheros de voz se habían analizados incluyéndose los de la cabecera. Puesto que el filtrado *RASTA* hace que la información de una trama se propague hasta un determinado tiempo después, los modelos fonéticos obtenidos no estaban bien y causaron una disminución de la tasa de reconocimiento.

Para evitar este problema en [Vet97] se describe una metodología *RASTA* alternativa que intenta eliminar la dependencia del contexto izquierdo, proponiendo el uso de un filtrado de amplitud constante para compensar las diferencias de fase que a diferentes frecuencias provoca el *RASTA*, por ser un filtro *IIR*. El principal inconveniente de este esquema es que no puede ser implementado en tiempo real, ya que es necesario esperar hasta la última trama para poder realizar un filtrado de las tramas obtenidas ordenadas de manera inversa, no pudiendo aplicarse el algoritmo de reconocimiento hasta ese momento, por lo que no resulta práctico.

Otro de los aspectos que se han reconsiderado es el de ajustar el filtrado en función del canal. En lugar de aplicar siempre el mismo filtro, se pretende adaptar la

señal de voz para después filtrar los parámetros de una forma óptima y conseguir normalizar el canal.

En [Ave96] se describen dos metodologías para el diseño de estos filtros. Ambas se basan en comparar la voz limpia con la misma voz una vez introducida distorsiones lineales y ruido aditivo. Como en el caso de otras técnicas de adaptación mencionadas, requieren el uso de bases de datos estéreo. Para una aplicación telefónica, no resulta práctico usar técnicas que requieren una información que no se puede aplicar porque no se tiene la señal original limpia.

4.3.4 Cepstral Mean Normalization (CMN).

Como se describe en al apartado anterior sobre el procesado *RASTA* y se muestra en la Figura 4-10 y en la Figura 4-11, este procesado elimina toda la señal que se encuentra por debajo de la frecuencia de corte determinada por la constante *K*, incluida la componente continua. Existe un método denominado *Cepstral Mean Normalization* (*CMN*), también conocido como *Cepstral Mean Susbstration* (*CMS*), que realiza una normalización de los cepstrum eliminando únicamente la componente continua. Para ello, a partir de los cepstrum calculados de la señal de voz completa, se calcula el vector de cepstrum medio y se resta a todos los vectores de la pronunciación. En [Liu93] se realiza una descripción de esta técnica, cuya expresión general tiene la forma:

$$\hat{X}_t = X_t - \frac{1}{N}\sum_{i=1}^{N} X(i) \qquad [4.12]$$

donde X_t es el vector con los coeficientes cepstrum en el tiempo t antes de realizar la normalización y $\hat{X}_t$ es el mismo vector una vez realizada dicha normalización. N indica el número de la trama de la pronunciación.

El resultado de obtener el vector de medias para realizar la sustracción a partir de las frases de entrenamiento, es equivalente a calcular un vector de compensación proveniente del cálculo de los vectores de medias de unos determinados conjuntos de parámetros físicos (o una presunta identidad fonética), promediados por el porcentaje de tramas utilizadas para estimar cada uno de ellos. Es decir, que según cuál sea la distribución estadística de las unidades fonéticas, este vector de compensación global puede variar. Ésta es una de las razones por la que resulta interesante utilizar en entrenamiento una base de datos fonéticamente balanceada que represente la distribución aproximada de sonidos en una situación real, para conseguir un resultado lo más óptimo posible.

Este método puede ser aplicado de diferentes maneras dependiendo de cómo se realice el cálculo del vector de medias. El vector de medias se puede obtener de las bases de entrenamiento y reconocimiento completas, de una única frase o de manera que pueda funcionar en tiempo real de forma síncrona con el algoritmo de

reconocimiento, calculándolo de forma incremental y recursiva. En todos los casos existe la posibilidad de considerar o no las tramas de ruido para el cálculo de este vector de medias.

Se puede comprobar que este algoritmo es equivalente al procesado *RASTA*, al comportarse como un filtrado paso alto que elimina bajas frecuencias, aunque sólo sea la componente continua. Sin embargo, ambos algoritmos compensan los efectos de filtrados lineales desconocidos, ya que los filtros lineales se comportan como vectores de compensación estáticos en el dominio de los cepstrum, que es la diferencia que existe entre los cepstrums obtenidos en el entrenamiento y en el reconocimiento.

No obstante, como se consideró a la hora de describir el modelo de canal, estos sistemas no compensan el efecto del ruido, puesto que éste es una función de la relación *SNR* en el dominio de los cepstrum. Como se mencionó en el apartado 4.3.2.3 sobre las diferentes alternativas de adaptación del canal, existen otras metodologías capaces de compensar a la vez el ruido aditivo y la distorsión lineal. Sin embargo, necesitan mucho más procesado y no pueden utilizarse en situaciones donde las variaciones del canal son tan elevadas como en aplicaciones telefónicas.

Existe una diferencia entre el procesado *CMN* y el *RASTA*, y es que el primero es un filtrado no lineal, puesto que utiliza un sumatorio de valores de toda una pronunciación. Tampoco es casual, puesto que para aplicarlo de forma ideal hay que esperar a tener la frase completa, calcular el valor medio y luego restárselo a los vectores de características de toda la frase.

Una primera clasificación de tipos de normalización *CMN* es la siguiente:

- ***CMN Puro***: La estimación del vector de medias se realiza a partir de una única pronunciación. Además, normalmente se tiene un detector qué determina qué tramas son de voz y cuáles de silencio dentro de una frase. Las tramas de voz se utilizan para realizar la estimación de los valores medios de los cepstrum para realizar la sustracción.
- ***GCMN (Global CMN)***: Es una variante del anterior. Consiste en obtener la media del conjunto total de frases, no una media para cada una de las frases de entrenamiento o de reconocimiento. A la hora de aplicar esta técnica, sólo es necesario calcular la diferencia entre el vector de medias estimado de la base de datos de entrenamiento y el de medias de reconocimiento.

Independientemente de cuál de los dos tipos sea utilizado, ninguno puede ser usado en tiempo real, excepto para la obtención de la tasa de error que presenta el reconocedor de voz, pues en reconocimiento habría que esperar a tener la frase completa para empezar a analizar, y esto no resulta nada práctico, pues se pierde la capacidad de procesado en tiempo real. Por ello, hay que considerar estrategias de incorporación de esta técnica dentro de la arquitectura del reconocedor de voz.

En este sentido han aparecido estrategias como en [Vii98], donde se propone una metodología para calcular los parámetros de normalización para cada pronunciación de manera recursiva denominado ***CMN Recursivo***.

4.3.4.1 CMN Recursivo (CMNR).

El algoritmo *CMN Recursivo* parte del hecho de que analizar una porción inicial de la señal de voz lo bastante grande puede ser suficiente para obtener una estimación inicial del valor de medias similar a la que se obtiene si se aplicara el algoritmo *CMN* con una estimación de la frase completa. De esta manera, este algoritmo puede ser aplicado en tiempo real con un retardo no muy elevado. El algoritmo consta de dos fases que son las siguientes:

- ***Inicialización***: Se realiza el cálculo de las N primeras tramas correspondientes al tamaño de una ventana de la que se quiere extraer una estimación inicial. De estas primeras tramas se obtiene una media y una varianza inicial. Esos valores se aplican a la primera trama. Las expresiones de cálculo para esta trama son:

$$m_t(i) = \frac{1}{N}\sum_{t=1}^{N} o_t(i) \qquad [4.13]$$

$$s_t(i) = \sqrt{\frac{1}{N}\sum_{t=1}^{N}\left[o_t{}^2(i)\right] - \left[m_t(i)\right]^2} = \sqrt{\overline{s_t{}^2(i)} - \left[m_t(i)\right]^2} \qquad [4.14]$$

  con $o_t(i)$ la componente i-ésima del vector de características en el tiempo t, $m_t(i)$ la componente i-ésima del vector de medias, y $s_t(i)$ la componente i-ésima del vector de varianzas.

- ***Actualización***: Una vez que se conocen los valores iniciales, se procede a la actualización del vector de características de la siguiente manera:

$$\hat{x}_{t-N}(i) = \frac{x_{t-N}(i) - m_t(i)}{s_t(i)} \qquad [4.15]$$

  Por cada nuevo vector de características generado, se desplaza la ventana y se actualizan los valores de medias y varianzas de la siguiente manera:

$$m_t(i) = l \cdot m_{t-1}(i) + (1-l)\cdot o_t(i) \qquad [4.16]$$

$$\overline{s_t^2(i)} = l \cdot \overline{s_{t-1}^2(i)} + (1-l)\cdot o_t^2(i) \qquad [4.17]$$

  donde l se corresponde con el tamaño del paso de actualización.

En el caso de la extracción de características del reconocedor propuesto, las varianzas se calculan una única vez y se aplican a todos los vectores para normalizarlos, por lo que las ecuaciones [4.14] y [4.17] no son necesarias. La ecuación [4.15] quedaría reducida a aplicar la sustracción cepstral de la media estimada sin necesidad de dividir por la varianza.

El principal problema es el retardo en el cálculo, que puede ser en ocasiones demasiado elevado para poder utilizarse en una aplicación en tiempo real, en la que se desea que el usuario obtenga una respuesta inmediata. Los resultados experimentales permitirán evaluar los valores óptimos.

4.4 Robustez Frente a Palabras Breves

En este apartado se describe una alternativa para aprovechar la flexibilidad de un reconocedor de voz fonético para generar cualquier modelo de palabra con un número reducido de modelos de estados, pero disminuyendo la tasa de error que se obtiene cuando se utiliza este tipo de modelos en lugar de modelar palabras enteras debido a que las unidades fonéticas no tienen un grado de precisión tan elevado.

4.4.1 Introducción.

Para el entrenamiento de cualquier conjunto de palabras y, en particular, de palabras breves se pueden distinguir dos alternativas de modelado:

- ***Modelos de palabras aisladas***: Utilizan una base de datos con las palabras a reconocer para entrenar unos modelos de Markov específicos.
- ***Modelos silábicos, fonéticos o subfonéticos***: A partir de un conjunto de modelos silábicos, fonéticos o subfonéticos entrenados a partir de bases de datos con frases pronunciadas por una gran cantidad de locutores y de la transcripción fonética de las palabras a reconocer, se generan los modelos de palabras mediante la concatenación de estas unidades silábicas, fonéticas o subfonéticas.

El reconocimiento de palabras breves como son *sí*, *no* o los dígitos fueron las primeras aplicaciones que se utilizaron con los modelos de Markov en el reconocimiento de voz. Estos reconocedores de voz tenían una estructura gramatical fija, y sólo podían reconocer una palabra de entre un conjunto muy limitado de ellas entrenadas a partir de bases de datos de repeticiones de la misma. Este tipo de reconocedores permitían disponer de vocabularios de hasta algunas decenas de palabras.

Las tasas de error de estos reconocedores de voz de palabras aisladas han ido mejorando gradualmente gracias al diseño de mejores algoritmos de reconocimiento, de mejores entrenamientos de los modelos de Markov y de mejores extracciones de características. Éste es el caso, por ejemplo, de [Fur86] donde Furui propuso la incorporación de los parámetros *delta-cepstrum* que tienen en cuenta la información de las diferencias entre un conjunto de tramas a la vez, consiguiéndose una mayor robustez en el reconocimiento de palabras aisladas.

A la vez que se iba llegando al límite del reconocimiento de palabras aisladas con tasas de error cada vez más bajas, poco a poco y gracias a mejores algoritmos de entrenamiento, a la disposición de bases de datos más grandes y a la aparición de sistemas de cómputo más potente, los reconocedores de voz basados en el modelado de unidades menores a una palabra como sílabas, fonemas o características subfonéticas fueron mejorando e imponiéndose. En el apartado 2.3.4 se hace una exposición más detallada de estos tipos de modelados.

El problema surge cuando se intenta utilizar los modelos fonéticos o subfonéticos para el reconocimiento de palabras muy breves, ya que las tasas de error que se obtienen son bastante mayores que las que tenían los predecesores sistemas de palabras aisladas. Lo que ocurre en estos casos es que las unidades que se utilizan para reconocer estas palabras tan breves están entrenadas a partir de un conjunto muy elevado de vectores de características (provenientes de multitud de contextos fonéticos diferentes), precisamente para tener una mayor representatividad y variabilidad. Los modelos así obtenidos no tienen una precisión tan elevada como para que las diferencias entre estas palabras breves y muy similares entre sí sean suficientemente elevadas para poder discriminarlas correctamente.

La solución adoptada en la mayoría de los casos consiste en mantener dentro de la estructura de los reconocedores fonéticos de voz modelos de palabras aisladas para poder mantener la tasa de reconocimiento. En [Hwa93a], por ejemplo, en el reconocedor fonético del proyecto *Sphinx* de *CMU* se introducen modelos de las letras del abecedario, que dada su gran brevedad son las más difíciles de reconocer.

La utilización de modelos de palabras aisladas dentro de una estructura de un reconocedor fonético tiene los siguientes inconvenientes:

- Para cada una de las palabras breves que se quieren introducir surge la necesidad de tener una base de datos específica. Es el mismo inconveniente que aparece con la utilización de modelos de palabras aisladas para reconocer cualquier palabra, pero más restringido.
- Mientras que con la utilización de modelos fonéticos o subfonéticos se dispone de un número reducido de modelos compartidos por las diferentes palabras a reconocer, los entrenados para palabras breves son específicos y únicamente válidos para ser utilizados por ellos mismos.

Comparando el número de estados de unos modelos de palabras aisladas compuestos por trece palabras (*sí*, *no*, *información* y los dígitos desde el *cero* hasta el *nueve*) con los estados de un conjunto de modelos fonéticos libres de contexto, se llega a la conclusión de que la carga computacional asociada con el cálculo de distancias a estados es muy similar. En el primer caso de reconocimiento con modelos de palabras aisladas, existen unos 104 estados (13 palabras * 8 estados de media para cada uno de los modelos) y para el caso del reconocedor de voz fonético descrito en la presente tesis, utilizando monofonemas, se dispone de 82 estados.

A simple vista, podría parecer que la diferencia en el número de modelos de estados no es muy significativa y, por tanto, es preferible utilizar los modelos de palabras aisladas, que mejoran el grado de reconocimiento. Sin embargo, las cosas cambian cuando se considera el factor tamaño del vocabulario, ya que mientras que con los 82 modelos de monofonemas es posible tener un reconocedor formado por cualquier número de palabras, los modelos aislados únicamente se pueden utilizar para las palabras entrenadas.

Para obtener mejores tasas de reconocimiento de palabras breves y mantener las ventajas que el reconocedor de voz fonético proporciona, se propone la utilización de un método que asocie cada una de las palabras a reconocer con varias transcripciones fonéticas diferentes.

4.4.2 Método de las Transcripciones Fonéticas Alternativas.

En general, dada una palabra, existe una transcripción fonética ideal para el castellano, que es la secuencia de fonemas que de forma consecutiva hay que pronunciar para emitirla.

Tabla 4-III. Transcripción fonética de los dígitos, *sí, no* e *información.*

Palabra	Transcripción fonética
sí	*s i*
no	*n o*
información	*i N f o r m a T j o N*
ayuda	*a u D a*
cancelar	*k a N T e l a rr*
terminar	*t e rr m i n a rr*
cero	*T e r o*
uno	*u n o*
dos	*d o s | D o s*
tres	*t r e s*
cuatro	*k w a t r o*
cinco	*T i N k o*
seis	*s e j s*
siete	*s j e t e*
ocho	*o tS o*
nueve	*n w e B e*

En la Tabla 4-III se muestra la transcripción fonética habitual de las trece palabras consideradas. En el caso de la palabra *dos*, existen dos alternativas de transcripción fonética dependiendo de si se pronuncia una palabra antes con fonema nasal al final o no, aunque la primera es la más habitual por pronunciarse aisladamente la mayoría de las veces.

Si a estas transcripciones asociadas con una palabra se les asocia otras transcripciones fonéticas que como tal puede que no tengan ningún significado, se podría obtener un grado de discriminación mayor.

Tabla 4-IV. Transcripciones fonéticas alternativas de los dígitos y otras palabras breves.

Palabra	Transcripciones fonéticas alternativas
sí	*i i* | *s i i*
no	*o o* | *n o o* | *n n o*
ayuda	*a j u d a* | *a i u a* | *a i u d a*
cero	*s e r o*
dos	*d o o* | *o o s*
tres	*t r e e* | *t e e s* | *t r e e s*
cuatro	*k u a t r o*
cinco	*s i N k o*
seis	*s e i s* | *T e i s*
ocho	*o s o*
nueve	*o o e e*

En la Tabla 4-IV se muestran diferentes alternativas de transcripción fonética para las mismas palabras de la Tabla 4-III. Los cambios introducidos consisten en duplicar alguna vocal, sustituir una semivocal por la vocal correspondiente, eliminar algún fonema de consonante o cambiar alguna consonante por otra con la que se suele confundir.

En cualquier caso, con las palabras que estén activas en el vocabulario en ese momento, no se puede dar la misma transcripción fonética a dos palabras distintas porque no habría distinción posible. Las transcripciones fonéticas alternativas deben confiar sobre todo en que las vocales, nasales y fricativas son los fonemas que más duran y que son pronunciados con más probabilidad.

4.5 Robustez Frente a Palabras Fuera de Vocabulario

Los reconocedores de Markov tienen el problema de que son de tipo estadístico, es decir, entre un conjunto de alternativas eligen siempre la que tiene mayor probabilidad de haber sido generada, siendo posible que sin haber pronunciado ninguna de las palabras candidatas, o incluso con cualquier señal interferente con suficiente energía, el reconocedor se active y dé un resultado erróneo.

Hay que incorporar dentro del reconocedor mecanismos para luchar contra lo que se denomina *palabras fuera de vocabulario* (*OOV*: *Out-Of-Vocabulary*), que se da cuando lo pronunciado no se corresponde con lo esperado por el reconocedor. Se distinguen tres situaciones a las que se tiene que enfrentar un reconocedor y que sería interesante detectar:

- ***Señales espúreas***: Por un lado, es importante evitar que el reconocedor se active con cualquier señal indeseada captada por el micrófono, como puede ser la respiración, música o cualquier ruido de fondo. Esto puede ocurrir si dicha señal posee suficiente energía y una duración mínima determinada. El reconocedor debe detectarlo y corregirlo. Un ejemplo típico consiste en soplar al micrófono, donde la señal generada es de gran amplitud y hace saltar al detector de extremos y, si además es de longitud suficiente, generaría una palabra reconocida de forma errónea.
- ***Palabras fuera de vocabulario***: Este segundo aspecto es el que, en general, se entiende como detección de palabras no pertenecientes a la red de reconocimiento. Es necesario considerarlo cuando exista la opción de pronunciar un determinado nombre para que el reconocedor sea capaz de identificarlo como no perteneciente a la lista y rechazarlo.
- ***Confusión de palabras***: No se corresponde exactamente con palabras fuera de vocabulario, sin embargo, la metodología y el procesado que se requiere para la detección de señales espúreas o palabras fuera de vocabulario sirve igualmente para este objetivo. Debido a que el reconocedor no es perfecto y tiene una tasa de error, no se puede estar seguro de que lo reconocido se corresponda con lo pronunciado y un modelo distinto al pronunciado puede dar una probabilidad mayor, pero estableciendo una puntuación de calidad, podría ser rechazada.

Se han propuesto diversas metodologías para medir el grado de calidad de lo reconocido y poder realizar un rechazo de palabras. En [Riv96] se propone la utilización de una medida a partir de la distancia a los fonemas. Se proponen dos alternativas, una denominada ACM_1 (Acoustic Confidencial Measure), basada en el valor medio del logaritmo de las probabilidades de pertenencia de cada una de las tramas a cada modelo fonético; la otra se denomina ACM_2, que es similar a la anterior, pero las

probabilidades se normalizan por la duración de los fonemas y, así, todos ellos valen lo mismo independientemente de que duren más o menos.

Sin embargo, cualquier medida absoluta realizada sobre probabilidades en una red de Markov no puede dar una gran confidencialidad, debido a la gran variabilidad de los valores que se pueden obtener. Es fundamental obtener una medida de distancia entre dos valores absolutos, de manera que la diferencia entre ambos valores pueda ser más significativa. Por eso, se propone usar un modelo denominado de *Basura* que se encargue de ofrecer una medida de referencia para validar la probabilidad obtenida por el reconocedor.

4.5.1 Modelos de Basura.

Existen varias alternativas a la hora de definir un modelo de basura. Se pueden clasificar de la siguiente manera:

- ***Explícito***: Al igual que se tiene un modelo de silencio entrenado a partir de las tramas donde únicamente se localiza el ruido de fondo, es posible hacer un modelo específico. Podría modelarse a partir de todo tipo de ruidos, interferencias o música.
- ***Implícito***: No existe un modelo específico, sino que se va calculando a partir de las distancias a los modelos actuales, ya sean de palabra o fonéticos.

La ventaja del modelo de basura explícito es que permite modelar todo aquello que sea extraño y diferente a voz. Además, funciona como otro modelo cualquiera, por lo que se puede introducir fácilmente en la red de reconocimiento. El inconveniente que presenta consiste en que es difícil utilizar un modelo que permita discriminar palabras fuera de vocabulario.

La ventaja del modelo implícito de basura es que es más robusto y es capaz de discriminar también palabras fuera de vocabulario, al proporcionar una puntuación sobre la calidad de las palabras reconocidas. El principal inconveniente es que obliga a introducir un procesado adicional que, si no se realiza con cuidado, puede suponer una carga computacional elevada.

4.5.2 Implementación de un Modelo de Basura Implícito.

La opción elegida para aumentar la robustez frente a palabras fuera de vocabulario y, en general, para discriminar palabras erróneas, es la implementación de un modelo de basura implícito.

Basándose en el hecho de que el reconocedor de voz funciona a partir del modelado de unidades fonéticas, se propone que la distancia al modelo de basura se calcule utilizando las distancias a los modelos de estados de las unidades fonéticas. Para la implementación de dicho modelo de basura se fijaron dos criterios que éste tenía que cumplir:

- La distancia obtenida debería ser independiente de los modelos activos de la red de reconocimiento.
- La carga computacional asociada tenía que ser lo más baja posible para no disminuir las prestaciones de eficiencia del reconocedor.

Para conseguir que la medida fuera independiente del vocabulario activo, se utiliza la distancia a los modelos de estado de las unidades independientes de contexto, es decir, monofonemas. Además de conseguir que dicha medida, dada una pronunciación, proporcione el mismo valor con independencia del vocabulario, se obtienen las siguientes ventajas:

- Por un lado, ya se trabaje a nivel de monofonema, bifonema o trifonema, los monofonemas siempre se utilizan, porque en las frases del vocabulario muchas unidades fonéticas no están entrenadas para todos los contextos posibles y, en estos casos, los monofonemas los sustituyen.
- Para un vocabulario medio (alrededor de cien palabras), aparecen la mayoría de los modelos fonéticos, por lo que las distancias a dichos modelos ya están calculadas y sólo es necesario procesarlas.
- Para el castellano, con las 34 unidades fonéticas libres de contexto y con la asignación de uno, dos o tres estados por modelo fonético, se obtiene un total de 82 distancias a modelos de estado. Este número es muy reducido si lo comparamos con el número de modelos fonéticos utilizando unidades dependientes de contexto.

Cuando se trabaja con vocabularios grandes, el coste computacional asociado con el cálculo de la distancia al modelo de basura es muy pequeño, puesto que parte de la información necesaria ya está calculada y porque el resto de procesados (como el recorrer la red de reconocimiento) presenta una carga computacional mayor. Por el contrario, cuando el número de palabras es muy reducido, una gran mayoría de los modelos no existe en el vocabulario pero, sin embargo, hay que realizar su cómputo para evaluar la distancia al modelo de basura.

4.5.2.1 Cálculo de la Distancia al Modelo de Basura.

El modelo de basura se define como el promedio de las mejores distancias a los modelos de estado de los monofonemas. Pretende establecer un valor de referencia para realizar comparaciones a los modelos de distancia de la frase reconocida.

Para la realización del cálculo de la distancia al modelo de basura es necesario considerar tres procesados diferentes:

- ***Cálculo de log-verosimilitudes***: Distancias del vector de características a cada uno de los modelos de estado.
- ***Selección de las K mejores distancias***: A partir de la ordenación de las anteriores distancias se seleccionan las mejores. Para ello, se puede utilizar

cualquier algoritmo eficiente de ordenación, por ejemplo, el algoritmo de la burbuja.

- ***Cálculo del valor medio***: Las distancias seleccionadas se promedian para obtener una medida normalizada.

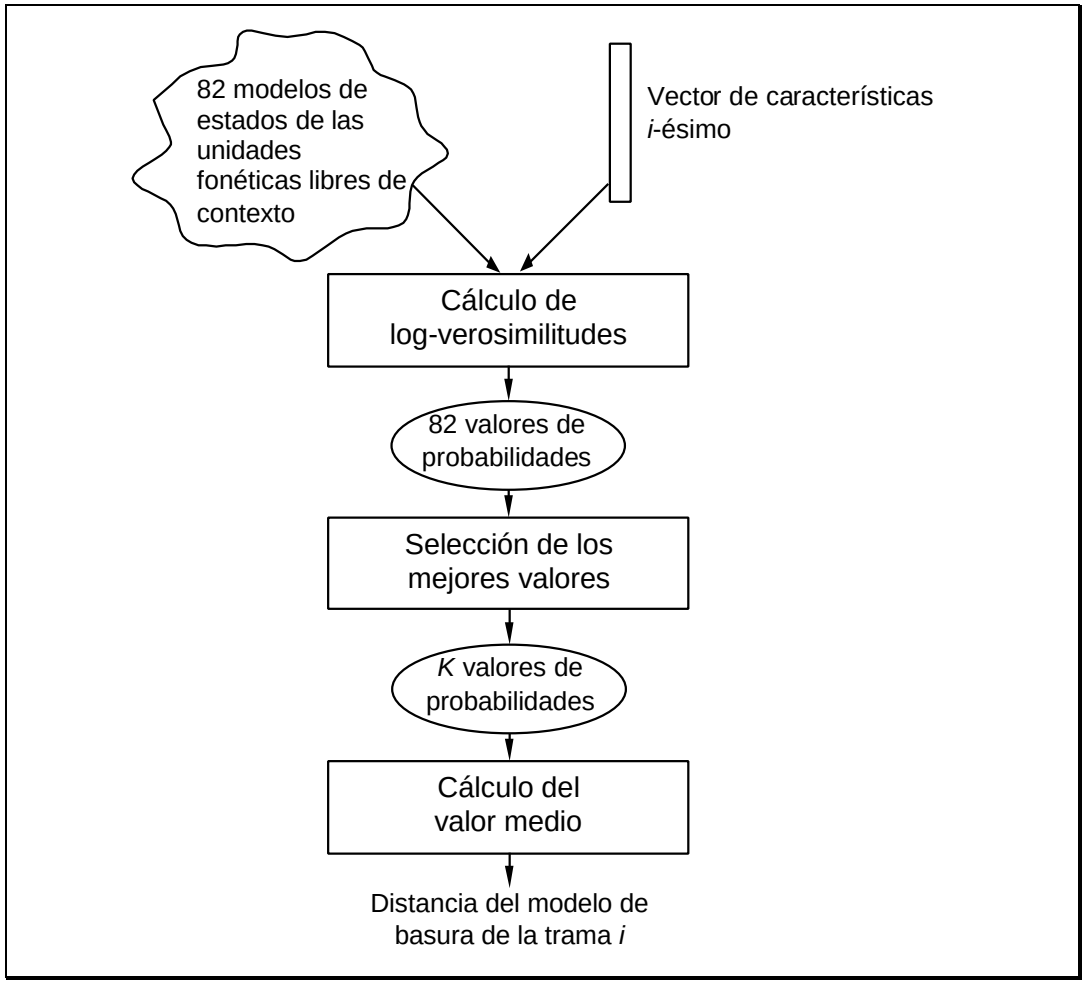

Figura 4-12. Diagrama de cálculo de la distancia al modelo de basura para la trama *i*.

En la Figura 4-12 se muestra el diagrama de cálculo para la obtención de la distancia al modelo de basura por trama. Estas distancias así obtenidas se almacenan de forma acumulativa para después ser utilizadas en la extracción del valor de la puntuación.

Dados los bloques de procesados y el diagrama de la Figura 3-14, la expresión matemática del ***cálculo de la distancia al modelo de basura*** puede escribirse de la siguiente manera:

$$DistBasura(i) = \frac{1}{K} \sum_{j \in \left\{ \substack{K\ mayores \\ (\log P(j))} \right\}} \log P(j) \qquad [4.18]$$

donde ($\log P_j(i)$) representa el logaritmo de la probabilidad de que la trama i pertenezca al modelo j, y K representa el parámetro de diseño, que es el número de distancias que se utiliza para promediar. Para almacenar de manera acumulativa estos valores así calculados se utiliza la expresión:

$$CosteAcumBasura(i) = CosteAcumBasura(i-1) + DistBasura(i) \qquad [4.19]$$

4.5.2.2 Diferencia entre el Modelo de Basura y el Modelo de Palabra.

Tal como se menciona anteriormente, la distancia al modelo de basura implícito se utiliza para compararla con la de la probabilidad acumulada de las palabras reconocidas. Lo que se quiere conseguir es que cualquier palabra fuera de vocabulario o señal interferente tenga una diferencia de probabilidad lo suficientemente baja como para que se pueda discriminar.

Los valores de probabilidades calculados son log-verosimilitudes, es decir, todos son negativos ya que representan el logaritmo de números menores a la unidad. Esto hay que tenerlo en cuenta a la hora de hacer comparaciones relativas para obtener el resultado adecuado.

Una ver realizada la búsqueda hacia atrás y obtenidas las probabilidades acumuladas, representadas por $CosteAcum(k)$ (que indica la suma de todas las probabilidades desde el instante inicial a la trama k), se puede calcular el ***coste de la palabra reconocida*** ($CostePalabra$). El valor de ese coste se obtiene a partir de la probabilidad acumulada en la trama final de la palabra f, $CosteAcum(f)$, al que se resta el valor de coste acumulado hasta la trama anterior al inicio de la palabra i, representando f el índice de la última trama perteneciente a la palabra, y siendo $i+1$ el índice de la primera trama perteneciente a dicha palabra. Al coste de la palabra reconocida hay que eliminarle todos los costes de transición entre estados de la red para que el resultado sólo tenga en cuenta las distancias a los estados del modelo de palabra. En resumen, el coste de la palabra se puede escribir como:

$$\begin{aligned} CostePalabra = \quad & CosteAcum(f) - CosteAcum(i) - \\ & - Trans_0 \cdot NumSaltos_0 - \\ & - Trans_1 \cdot NumSaltos_1 - \\ & - Trans_2 \cdot NumSaltos_2 \end{aligned} \qquad [4.20]$$

donde $CostAcum(f)$ indica el coste acumulado en la red hasta la trama f correspondiente a la última palabra, $CostAcum(i)$ la probabilidad acumulada hasta antes del inicio de la palabra, $Trans_0$ indica el logaritmo de la probabilidad de transición de un estado al mismo, $Trans_1$ el de un estado al siguiente y $Trans_2$ el de un estado hasta dos posteriores. $NumSaltos_0$, $NumSaltos_1$ y $NumSaltos_2$ indican el número de cada uno de los saltos dentro de la palabra.

Por otro lado, está el ***coste de la basura***, *CosteBasura*, que similarmente al del *CostePalabra*, se puede expresar como el coste acumulado del modelo de basura hasta la trama f menos el coste acumulado del modelo de basura hasta la trama i. En este caso, no hay que eliminar ningún coste de transición porque no existe. La expresión queda como:

$$CosteBasura = CosteAcumBasura(f) - CosteAcumBasura(i) \qquad [4.21]$$

siendo *CosteAcumBasura*(f) el coste acumulado hasta la trama f, y *CosteAcumBasura*(i) el coste acumulado hasta la trama i.

A partir de los costes de la basura y de la palabra se calcula una diferencia. Para realizar la comparación es necesario que el coste de basura esté multiplicado por un factor de ponderación Q menor que la unidad. Al ser dicho coste de basura negativo, porque es la suma de log-verosomilitudes, se obtiene un valor mayor de manera absoluta. La expresión de la ***diferencia entre el coste de palabra y el coste de basura ponderado*** (*Diferencia*) es:

$$Diferencia = CostePalabra - CosteBasura \cdot Q \qquad [4.22]$$

Una *Diferencia* muy positiva indica que la palabra reconocida es muy probable que haya sido la pronunciada, mientras una muy negativa indica que es muy probable que lo pronunciado no tenga nada que ver con lo reconocido.

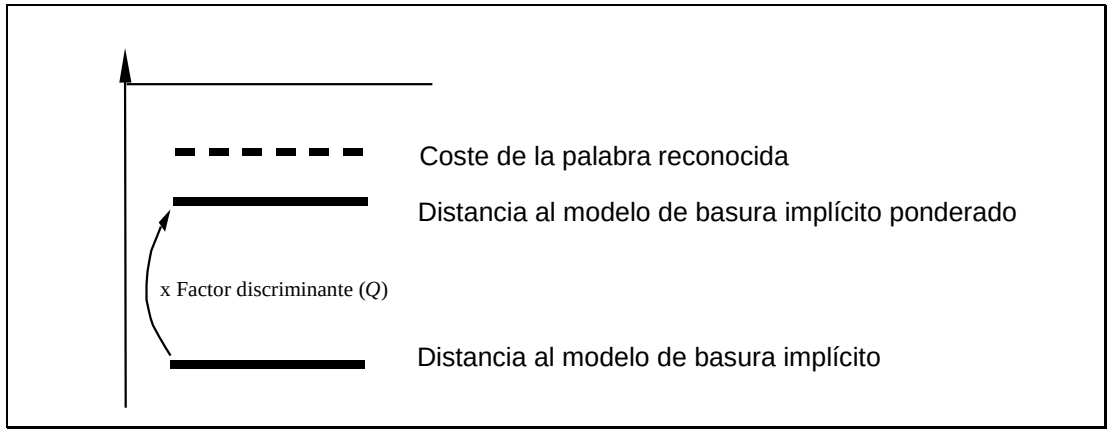

Figura 4-13. Esquema de lo que ocurre cuando se pronuncia una palabra dentro del vocabulario activo. Aunque el modelo de basura se multiplique por un valor de discriminación, éste suele ser menor que el del modelo de basura y, por tanto, no debe ser rechazada dicha palabra.

En la Figura 4-13 se muestra lo que ocurre cuando la diferencia es positiva. En este caso, ni siquiera al multiplicar el coste del modelo de basura por el discriminante se puede alcanzar el coste de la palabra reconocida. Esta medida indicaría que la palabra reconocida se corresponde con la palabra pronunciada.

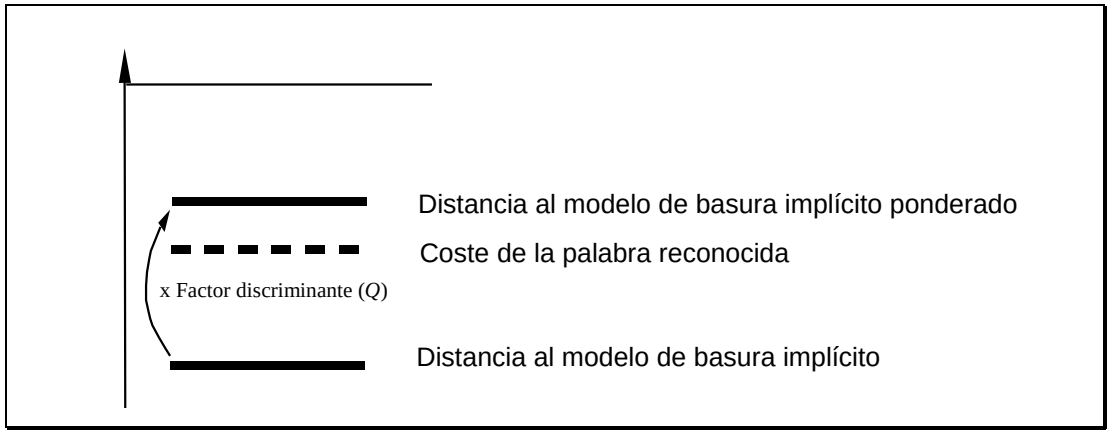

Figura 4-14. **Esquema de lo que ocurre cuando se pronuncia una palabra fuera de vocabulario. Al multiplicarse el modelo de basura por el factor discriminante, éste suele superar a la distancia del modelo de palabra y, por tanto, se puede utilizar para rechazar dicha palabra.**

Por otro lado, en la Figura 4-14 se muestra lo que ocurre cuando la diferencia es negativa. En este caso, al multiplicar el coste del modelo de basura por el discriminante, *Diferencia* supera el coste de la palabra reconocida. Esta medida indica que lo reconocido no se corresponde con lo pronunciado y la palabra se debería rechazar.

Sin embargo, la medida así obtenida no resulta muy útil, pues palabras con una duración larga darán valores absolutos mayores que con palabras cuya duración sea menor. Es necesario tener una medida que sea independiente del número de tramas de la palabra reconocida, es decir, una ***diferencia normalizada*** (*DifNor*) para que los umbrales que se impongan posteriormente para el rechazo de las pronunciaciones puedan ser generales. Para conseguir esto, lo más sencillo es dividir el valor obtenido de *Diferencia* entre el número de tramas que contiene la palabra reconocida (*NTramas*). Esto es,

$$DifNor = \frac{CostePalabra - CosteBasura \cdot Q}{NTramas} \qquad [4.23]$$

Finalmente, hay que mencionar que los valores de K y Q están ajustados para obtener unas medidas de diferencias lo más pronunciadas posibles, es decir, cuando a la red se le presente una palabra del vocabulario, el valor *DifNor* sea lo más positivo posible y, en el caso de presentarse palabras fuera de vocabulario o ruidos sostenidos de fondo, fuese lo más negativa posible.

Tabla 4-V. **Valores típicos de los parámetros con los que se ajusta el valor de diferencia entre el modelo de basura y el modelo de palabra.**

K	*Q*
40	**0,95**

En la Tabla 4-V se muestran los valores de ajuste obtenidos y se utilizan para calcular las diferencias entre el coste de palabra y el coste de basura. El primero representa el número de distancias a modelos de estados (K) y el segundo el factor de discriminación (Q).

4.5.2.3 Cálculo del Valor de Puntuación (Score).

En el apartado anterior se describió cómo obtener un valor de diferencia entre un modelo de palabra y un modelo de basura, ahora se explica cómo usar aquellas diferencias para rechazar las palabras reconocidas.

Al aplicar el algoritmo de reconocimiento y realizar la búsqueda hacia atrás, se extrae la secuencia de palabras reconocidas y su localización en el tiempo. Todas las tramas fuera de las palabras reconocidas pertenecientes a silencios o a palabras comodín, como los de la estructura de red descrita en el apartado 3.7 para nombres y apellidos, no se consideran, ya que únicamente se tienen en cuenta las tramas que pertenezcan a las palabras que se quieren validar.

A partir de estos valores se puede establecer un criterio para determinar si una determinada secuencia de palabras reconocida tiene que ser rechazada o no, dándole una determinada puntuación, y estableciendo para cada conjunto de palabras un umbral que marque cuando han sido correctamente reconocidas y cuando no.

Entre las aplicaciones de una centralita automática, lo más normal es reconocer una palabra (como *sí* o *no*, los dígitos) o bien una secuencia fija (como es el caso de nombres y apellidos). Resaltan dos tipos de secuencias de palabras: las formadas por un único modelo de palabra y las compuestas por dos modelos de palabras.

Para dar una puntuación al reconocimiento de un único modelo de palabra ($score_1$) se utiliza la expresión:

$$score_1 = (DifNor - offset_1) * gain_1 \qquad [4.24]$$

donde $DifNor$ indica la diferencia entre el modelo de palabra y de basura acumulada y normalizada por el número de tramas, $offset_1$ es una constante para fijar lo estricto que se desea ser con el rechazo y $gain_1$ una constante de ganancia para obtener unos valores de $score_1$ ajustados alrededor del valor 100 cuando la palabra ha sido correctamente reconocida.

De la misma manera, se puede obtener un valor de puntuación cuando se reconocen dos conjuntos de palabras, como nombre y apellido, ($score_2$). En este caso, la expresión quedaría como:

$$score_2 = (DifNor1 + DifNor2 - offset_2) * gain_2 \qquad [4.25]$$

donde $DifNor1$ y $DifNor2$ indican la diferencias entre el modelo de palabra y de basura acumulada y normalizada por el número de tramas de cada una de las palabras reconocidas, $offset_2$ es una constante para fijar lo estricto que se desea ser con el rechazo y $gain_2$ una constante de ganancia para obtener unos valores de $score_2$ ajustados alrededor del valor 100 cuando la palabra ha sido correctamente reconocida.

4.6 Medidas de Distancias entre Modelos Fonéticos

4.6.1 Introducción.

Un aspecto adicional para aumentar la robustez del reconocedor de voz es el de la reducción del número de modelos a reconocer mediante el agrupamiento con un grado de discriminación muy bajo. Si dos modelos son muy similares y aportan poca información, podrían entrenarse juntos. Así, se consigue un sistema más eficiente con un aumento del error no muy elevado.

Cualquier técnica de reducción del número de parámetros tiene la ventaja de que al compartir el número de datos utilizados para entrenar, cada uno de ellos aumenta. Es algo similar a lo que ocurre cuando se entrenan modelos semicontinuos [Hua90] donde las gaussianas son compartidas por todos los modelos.

El tipo de reducción considerado es el de la compartición a nivel subfonético, utilizando una medida de distancia o distorsión entre modelos. En [Hwa93a] para el entrenamiento de los llamados *senones* se utilizan técnicas de medida de entropía para determinar los modelos de estado similares con el fin de agruparlos y, posteriormente, reentrenarlos juntos.

Pero existe un problema, y es que esas medidas se basan en métodos automáticos, por lo que no es posible conocer qué es lo que está pasando o por qué se están agrupando. Una manera de solucionarlo consiste en intentar visualizar las distribuciones de probabilidad. Sin embargo, los vectores de características están formados por vectores de n componentes y es muy difícil percibir directamente cómo son esas estructuras. La solución es realizar una proyección desde ese espacio n-dimensional a otro bi-dimensional donde ya sí es posible extraer información. Para ello, se necesita una técnica de proyección no lineal que tenga la propiedad de que distancias cercanas en el espacio n-dimensional también lo estén en el espacio bi-dimensional, ya que interesa que las estructuras locales (aquellos puntos más cercanos en el espacio n-dimensional) sean mantenidas por la proyección para poder discriminar semejanzas entre modelos.

La técnica de proyección considerada fue propuesta por Sammon en 1.969 [Sam69].

4.6.2 La Proyección de Sammon.

El objetivo de la proyección de Sammon es aproximar las distancias de todas las parejas de puntos d_{ij} en el espacio bi-dimensional, donde se lleva a cabo la proyección, a las distancias entre esas mismas parejas de puntos en el espacio n-dimensional d_{ij}.

Para conseguir esto, Sammon proponía la utilización de una función de error que hay que minimizar. Esta medida está basada en la comparación de las medidas entre

dos puntos cualesquiera en el espacio n-dimensional d_{ij} con las del espacio bi-dimensional d_{ij}. La expresión es la siguiente:

$$E_{s1} = \frac{1}{\sum_{i=1}^{n-1}\sum_{j=n+1}^{n} d_{ij}} \sum_{i=1}^{n-1}\sum_{j=n+1}^{n} \frac{\left(d_{ij} - d_{ij}\right)^2}{d_{ij}} \qquad [4.26]$$

4.6.2.1 Descripción del Algoritmo de Proyección.

La proyección de Sammon es un algoritmo iterativo. Empieza por una estimación inicial de la proyección, y en cada nueva iteración mueve los puntos en el espacio bi-dimensional a partir de la distribución de la iteración anterior. Así, para obtener la distribución de puntos en la iteración $m + 1$, después de llevar m iteraciones, la fórmula de redistribución está dada por:

$$y_{pq}(m+1) = y_{pq}(m) - (MF) \cdot \Delta_{pq}(m) \qquad [4.27]$$

donde:

$$\Delta_{pq}(m) = \frac{\partial E(m)}{\partial y_{pq}(m)} \Bigg/ \left| \frac{\partial^2 E(m)}{\partial y_{pq}(m)^2} \right| \qquad [4.28]$$

y MF es un factor que fue calculado de forma empírica y al que se le suele asignar un valor de $0{,}3$.

Cuando el incremento producido de una iteración a la siguiente es menor que un determinado umbral, el algoritmo acaba.

Se puede observar, analizando las fórmulas [4.27] y [4.28], que realmente se trata de un algoritmo de gradiente. De una iteración a otra los puntos se mueven en la dirección de máxima variación, por lo que además de ser un algoritmo de convergencia muy rápido también suele llegar a resultados óptimos. Puesto que la condición de ser un espacio bidimensional no se encuentra en ninguna de las fórmulas, es posible utilizar el algoritmo para reducir datos en cualquier otro número de dimensiones, aunque sería difícil analizarlo.

Un aspecto importante es la manera de realizar la distribución inicial de los puntos en el espacio bidimensional. Existen dos alternativas:

- ***Distribución aleatoria***: Las posiciones iniciales de los puntos se localizan aleatoriamente sobre el plano, pero todos ellos muy juntos entre sí para que el algoritmo empiece a colocarlos.
- ***Proyección lineal tras una descomposición en componentes principales***: Se trata de realizar una descomposición en componentes principales y proyectar linealmente todos los puntos sobre el plano caracterizado por las dos direcciones principales.

La proyección lineal converge mucho más rápidamente que la aleatoria porque los puntos ya tienen una cierta distribución y, por tanto, tienen que desplazarse menos que la aleatoria, que puede tardar más en converger hasta que encuentra una distribución adecuada.

4.6.3 Aplicación a la Reducción del Número de Modelos.

Gracias a la proyección de Sammon es posible discriminar de manera visual el conjunto de modelos subfonéticos que se encuentran realmente cercanos entre sí y, por tanto, el grado de discriminación es muy reducido. Basta con realizar una proyección de las funciones densidad de probabilidad y comprobar el grado de solapamiento.

Analizando la señal de voz se puede comprobar que los fonemas oclusivos presentan una duración muy breve, normalmente menor a los $10\ ms$ que, con respecto a la señal enventanada (que es de $256\ ms$), es muy pequeña. La variabilidad de estas tramas puede ser tan alta que no aporten ninguna información. Si esto es así, no hay necesidad de entrenar estas unidades fonéticas por separado.

Por ejemplo, si tras una comparación inicial de los fonemas *p*, *t* y *d*, que presentan un grado de discriminación muy pequeño en tasa de error, y si se comprueba que son muy similares, podrían agruparse sus estados formando una nueva unidad fonética denominada *oclusiva sorda*.

4.6.3.1 Descripción de la Técnica.

La proyección de todos los vectores de medias de los modelos fonéticos permite discriminar las parejas de puntos más cercanos que se entrenarán de forma conjunta.

Para ver realmente la similitud entre dos modelos de estado se proyecta de forma conjunta las funciones densidad de probabilidad. Para ello, utilizando los vectores de medias y el vector de varianza se genera un conjunto de vectores de forma sintética que representen a la distribución correspondiente.

Si al proyectar las distribuciones de dos unidades fonéticas diferentes las nubes se solapan, entonces se puede decir que son coincidentes. Si no ocurre lo anterior, no se agrupan y siguen siendo modelos separados.

Capítulo

5

Resultados Experimentales

5.1 Introducción

En este capítulo se describen las pruebas experimentales realizadas sobre el reconocedor fonético de voz descrito en la presente tesis, así como los resultados obtenidos. El punto de partida es el de las prestaciones del reconocedor con las características descritas en el capítulo 3, para luego incorporar las técnicas de robustez que aparecen en el capítulo 4.

Los resultados experimentales se pueden dividir en tres categorías:

- ***Resultados de robustez de laboratorio***: Son los resultados que se obtienen cuando se hacen medidas de tasas de error en el reconocedor de voz con condiciones restrictivas, por ejemplo, sin posibilidad de utilizar palabras comodín ni detector de extremos.

- ***Resultados de robustez en tiempo real***: Las medidas realizadas en el apartado anterior no se corresponden con las condiciones prácticas que se pueden encontrar en situaciones reales. Se necesitan medidas que utilicen una red de reconocimiento robusta, detección de extremos o llamadas telefónicas a través de una centralita automática. Estos nuevos aspectos aumentan la incertidumbre del reconocedor, lo que supone un incremento en la tasa de error. En este apartado se mide la influencia de estos aspectos, así como el funcionamiento de la técnica denominada *CMN Recursivo*.
- ***Medidas de capacidad y eficiencia***: Aunque el principal aspecto tratado en esta tesis es el diseño e implementación de un conjunto de técnicas que permitan a un reconocedor de voz para aplicaciones telefónicas mantener sus prestaciones cuando las condiciones de funcionamiento son muy variadas, existen otros aspectos de suma importancia para que sea un desarrollo viable y no sólo de laboratorio. Estos aspectos son las prestaciones que el reconocedor presenta en cantidad de memoria necesaria y las capacidades de procesado en función del tamaño del vocabulario y del número de reconocedores activos.

El orden de exposición del presente capítulo es: en el apartado 5.2 se muestran los resultados experimentales de robustez en condiciones de laboratorio; en el apartado 5.3 los obtenidos en situaciones realistas; y finalmente, en el apartado 5.4, medidas sobre la eficiencia del sistema de reconocimiento en tiempo real.

5.2 Resultados de Robustez en Condiciones de Laboratorio

5.2.1 Introducción. Diseño de los Experimentos.

En un reconocedor de voz y, en general, en cualquier sistema donde exista una gran cantidad de variables, resulta fundamental un desarrollo organizado de las medidas para poder analizar por separado lo que cada una de ellas aporta al sistema. En caso contrario, aún cuando el número de pruebas fuera muy exhaustivo, sería difícil obtener unas conclusiones claras y directas.

El punto de partida es el análisis del reconocedor cuando no se tienen en cuenta ninguno de los aspectos de robustez, para luego ir introduciéndolos poco a poco. Sin embargo, hay ocasiones en las que no es posible separar dos variables porque para que se dé una es necesaria la otra, por lo que hay que analizarlas a la vez (por ejemplo, dadas las características del reconocedor de voz, no es posible utilizar la segmentación automática con una red de saltos simples, porque los modelos así entrenados divergen).

Se comienza con unos modelos entrenados a partir de la segmentación manual de la base de datos del proyecto Albayzín, con una red de reconocimiento de saltos simples y sin ninguna técnica de robustez incorporada. Los modelos de unidades fonéticas libres de contexto (*monofonemas*) se utilizan para ver la eficacia de las técnicas propuestas. La red de reconocimiento es una de las más sencillas posibles, sin la posibilidad de introducir palabras comodín (como la estructura de la Figura 3-24, en la que aparece el detalle de los nombres y apellidos, o los modelos de dígitos de la Figura 3-21b).

En el reconocedor de voz existen dos aspectos básicos, uno la robustez frente al locutor y el otro frente al canal de comunicaciones. Por esta razón, se analizan primero las técnicas de robustez de independencia de locutor, eligiendo los parámetros que proporcionan mejores resultados, para posteriormente analizar los resultados de la robustez frente al canal de comunicaciones. A continuación, se analizan otras técnicas como el modelo de rechazo de palabras fuera de vocabulario, el estudio sobre alternativas de transcripción fonética de palabras breves, el funcionamiento cuando se introducen unidades dependientes de contexto (en este caso sólo *bifonemas izquierdos de vocales*) y la reducción de modelos subfonéticos para una mejora en su entrenamiento usando la proyección de Sammon.

5.2.1.1 Bases de Datos de Reconocimiento.

Para medir las tasas de error que presenta el reconocedor de voz y comprobar la manera en que cada una de las técnicas de aumento de la robustez afectan al mismo, se dispone de las siguientes bases de datos:

- ***NOM_100_TEL:*** Es una base de datos compuesta por 100 nombres y apellidos españoles pronunciados por 40 locutores femeninos y masculinos (en total 4.000 palabras). Fue grabada a través de múltiples canales telefónicos con diferentes niveles de ruido y de distorsión, incluso teléfonos móviles. La frecuencia de muestreo es de $8\ kHz$.
- ***DIG_13_MIC***: Fue grabada por 12 locutores masculinos y 9 femeninos, con 16 repeticiones por locutor (es decir, $2.496 + 1.872 = 4.368$ palabras). Está formada por los dígitos del *cero* al *nueve* y las palabras *información*, *sí* y *no*. La frecuencia de muestreo es de $8\ kHz$ y la grabación se realizó a través de un micrófono.
- ***DIG_15_TEL***: Grabada por 200 locutores masculinos y femeninos, con una repetición por locutor. Está formada por los dígitos del *cero* al *nueve* y las palabras *cancelar*, *terminar*, *ayuda*, *sí* y *no*. La frecuencia de muestreo es de $8\ kHz$ y se captó a través del canal telefónico. Es una base de datos representativa de una gran cantidad de canales de comunicación, con distorsiones, niveles de ruido y señales interferentes importantes.

La base de datos *NOM_100_TEL* se utiliza como referencia para analizar las diferentes técnicas de robustez. Las otras dos bases sirven para la validación de aspectos específicos como la independencia de locutor y el estudio de las alternativas de transcripción fonética en palabras breves.

5.2.2 Resultados con las Técnicas de Independencia de Locutor.

En este apartado se exponen los resultados obtenidos al aplicar técnicas que permiten aumentar la independencia del locutor según el apartado 4.2.

5.2.2.1 Resultados con Saltos Dobles y Entrenamiento Automático.

Los aspectos que se analizan sobre la independencia de locutor son:

- *Estructura de red de saltos simples (SS) vs. saltos dobles (SD)*.
- *Entrenamiento manual (EM) vs. entrenamiento automático (EA)*.

Realmente, no todas las posibilidades son factibles. Por ejemplo, como se menciona en el apartado 4.2.4.2, el entrenamiento automático con saltos simples no es posible porque los modelos entrenados divergen en cada iteración debido a que la segmentación no es capaz de acomodarse a velocidades de pronunciación elevadas.

Tabla 5-I. Tasas de error del reconocedor fonético de voz utilizando técnicas de robustez para conseguir la independencia de locutor.

	Entrenamiento manual		***Entrenamiento automático***	
	Saltos simples	***Saltos dobles***		
	BD_ALB_1	***BD_ALB_1***	***BD_ALB_2***	***BD_ALB_2 + NOM***
NOM_100_TEL	**6,27 %**	**6,14 %**	**7,19 %**	**5,31 %**
DIG_15_TEL	**19,50 %**	**22,48 %**	**19,35 %**	**19,42 %**
DIG_13_MIC	**8,59 %**	**10,60 %**	**11,03 %**	**10,01 %**

En la Tabla 5-I se muestran los resultados correspondientes a la introducción de las diferentes técnicas de robustez frente a la variabilidad del locutor. *BD_ALB_1* se corresponde con la parte de la base de datos del proyecto Albayzín que está segmentada, *BD_ALB_2* es la base de datos Albayzín completa y *BD_ALB_2 + NOM* es el entrenamiento con la base de datos Albayzín completa al que se ha incorporado una pequeña base de datos de nombres a través de canal telefónico (formada por un conjunto de nombres grabados en llamadas reales con un total de 200 grabaciones).

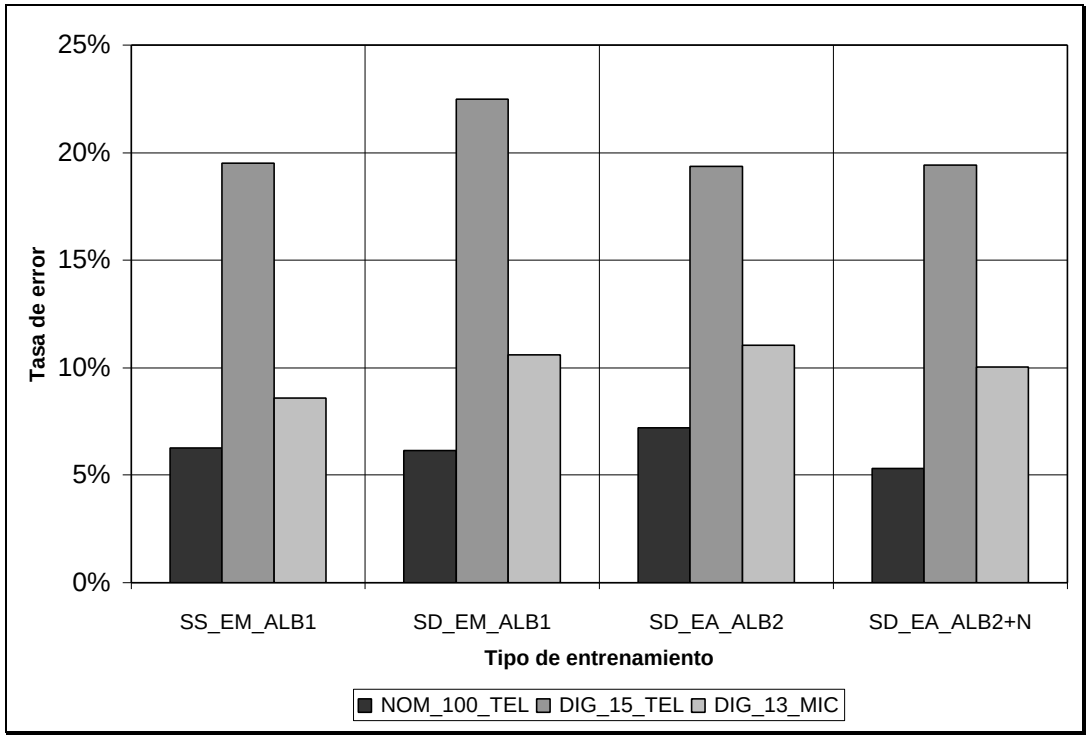

Figura 5-1. Gráfico con los resultados de tasa de error en función del tipo de entrenamiento para las tres bases de datos de reconocimiento.

En la Figura 5-1 se observan gráficamente los resultados de tasa de error de la Tabla 5-I para las tres bases de datos de reconocimiento, utilizando cuatro maneras diferentes de entrenar los modelos para mejorar la independencia de locutor.

Para la base de datos de reconocimiento *NOM_100_TEL*, el efecto de los saltos dobles sobre la tasa de error es una ligera reducción frente al obtenido usando saltos simples. Esto es así porque la velocidad de pronunciación en ciertas palabras es muy elevada, consiguiendo los saltos dobles una mejora.

Sin embargo, cuando el entrenamiento se realiza mediante la segmentación automática de Albayzín, se obtiene una tasa de error mayor que con la segmentada manualmente (que consta únicamente de un 25 % de las frases del corpus de aprendizaje). Esto se debe a que existen tres fenómenos:

- Se aumenta el número de locutores, por lo que los modelos presentan mayor variabilidad.
- Se producen confusiones debido a los errores de transcripción y de segmentación que aparecen. Esta descompensación es mayor que la mejora por introducir más modelos.
- Sigue existiendo desadaptación entre las bases de datos en entrenamiento y en reconocimiento.

Al añadir una pequeña base de datos de nombres pronunciados a una velocidad normal a través de canal telefónico se obtiene una disminución en la confusión de los modelos, lo que produce una mejora en la tasa de error por encima de la que se obtenía con los modelos entrenados a partir de la base de datos segmentada manualmente. Esta pequeña base de datos sirve para ajustar en cierto grado las diferencias ocasionadas por la desadaptación del canal de comunicaciones.

Los resultados para la base de datos *DIG_15_TEL* son algo diferentes. En primer lugar, la tasa de error con saltos simples es menor que con saltos dobles. Realmente, la mayor parte de las palabras obtienen tasas de error menor, pero existen tres palabras que la empeoran de forma sustancial, que son *ayuda*, *ocho* y *no*. En el caso de *ayuda* se pasa de un 52 % de error de reconocimiento con saltos simples a un 83 % con saltos dobles. Esto puede ser debido a que los fonemas que contienen (como el caso del fonema *Z* en *ayuda* o del *tS* en *ocho*) no están bien entrenados con la base de datos utilizada por no tener suficiente número de repeticiones, por lo que la introducción de saltos dobles aumenta el grado de confusión entre palabras similares. Sin embargo, al pasar al entrenamiento automático se obtienen mejores resultados, ya que se aumenta el número de apariciones de dichos fonemas en el entrenamiento.

Con la base de datos *DIG_13_MIC* se produce el mismo fenómeno al pasar de saltos simples a saltos dobles, al no estar ciertos modelos fonéticos bien entrenados la introducción de más posibilidades en el algoritmo de Viterbi redunda en un aumento de la tasa de error. Con el entrenamiento automático los resultados son algo diferentes, pues las tasas de error mejoran respecto al de saltos dobles, pero son ligeramente peores que las de saltos simples.

5.2.2.2 Resultados de Usar Saltos Dobles en Entrenamiento y Simples en Reconocimiento de Dígitos.

Como se puede constatar en la Tabla 5-I para el caso del reconocimiento de dígitos, el funcionamiento con saltos simples es mejor que con saltos dobles utilizando la segmentación manual de las frases. A continuación, se muestran los resultados comparativos cuando se utilizan saltos simples en reconocimiento, aún cuando el entrenamiento se haya realizado con saltos dobles.

Tabla 5-II. Tasas de error del reconocedor fonético de voz utilizando técnicas de robustez para conseguir la independencia de locutor.

	BD_ALB_2		***BD_ALB_2 + NOM***	
	Saltos simples	***Saltos dobles***	***Saltos simples***	***Saltos dobles***
DIG_15_TEL	**15,80 %**	**19,35 %**	**16,23 %**	**19,42 %**
DIG_13_MIC	**7,83 %**	**11,03 %**	**7,67 %**	**10,01 %**

Como se puede apreciar en la Tabla 5-II, la utilización de los saltos dobles en el reconocimiento de dígitos, no sólo no produce mejora, sino que incluso resulta perjudicial al producirse un incremento notable en la tasa de error.

La razón de este incremento en la tasa de error puede ser debida a que al tratarse de palabras tan breves, la introducción de un mayor número de posibilidades en el algoritmo de Viterbi provoca una disminución del grado de discriminación superior a la adaptación que se consigue al permitir una mejor adaptación a diferentes velocidades de elocución.

Por este motivo, el resto de medidas realizadas en este capítulo con las bases de datos de dígitos utilizan red de reconocimiento con saltos simples, mientras que con la base de datos de nombres se sigue utilizando la red con saltos dobles por obtenerse una mejora clara en este caso.

5.2.3 Resultados con Técnicas de Robustez del Canal de Comunicaciones.

Una vez analizadas las técnicas para aumentar la robustez frente a diferentes locutores, es el momento de analizar los resultados que se obtienen con las técnicas para aumentar la robustez del canal de comunicaciones. Para ello, se tienen en cuenta las siguientes posibilidades:

- *Filtrado RASTA con diferentes constantes de filtrado.*
- *Técnica de restado cepstral* (*CMN*).

Se parte del reconocedor fonético de voz entrenado automáticamente a partir de la base de datos *BD_ALB_2 + NOM* usando saltos dobles, por ser la configuración que mejor resultados ofrece. En el caso de la base de datos *NOM_100_TEL* en el reconocimiento se utiliza también saltos dobles, mientras que en la *DIG_15_TEL* es de saltos simples.

Tabla 5-III. Tasas de error del reconocedor fonético de voz utilizando el filtrado *RASTA*, con diferentes valores de la constante *K*, y el procesado *CMN* comparándolos con los resultados que se obtienen al no utilizar ninguna de ellas.

	RASTA K=0,75	***RASTA K=0,85***	***RASTA K=0,92***	***SIN RASTA K=1,00***	***CMN***
NOM_100_TEL	**7,54 %**	**5,76 %**	**4,71 %**	**5,31 %**	**4,48 %**
DIG_15_TEL	**19,29 %**	**16,55 %**	**14,52 %**	**16,23 %**	**12,57 %**

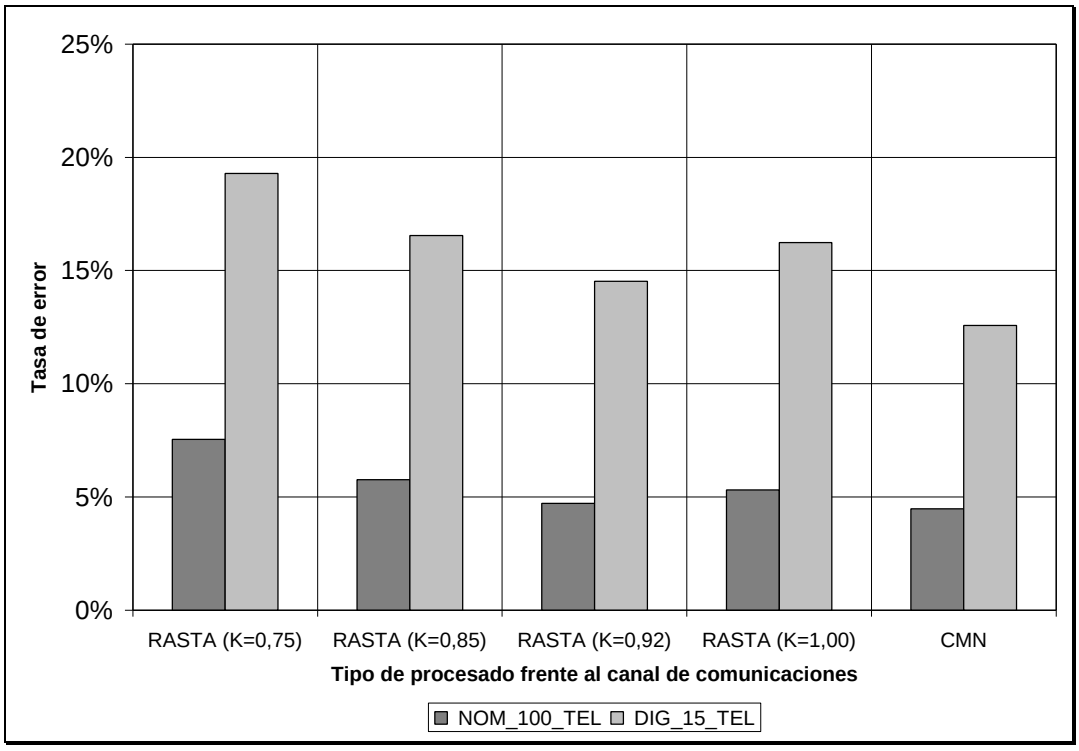

Figura 5-2. Gráfico con los resultados de reconocimiento en función del procesado realizado a los coeficientes cepstrum para las bases de datos de reconocimiento por canal telefónico.

En la Tabla 5-III (de forma gráfica en la Figura 5-2) se muestran los resultados obtenidos tras aplicar las técnicas de robustez de adaptación del canal telefónico: los del filtrado *RASTA* con diferentes valores de constante K (comprobándose que el óptimo se corresponde con K=0,92), los correspondientes a no aplicar nada (poner en el filtrado *RASTA* K=1 equivale a no realizar filtrado alguno) y el procesado *CMN* sobre cada una de las frases de entrenamiento y reconocimiento.

Se comprueba que para todas las bases de datos el filtrado *RASTA* óptimo se obtiene con K igual a 0,92. Cuando la constante es menor, el filtrado de la señal es muy elevado y se elimina información, por lo que la tasa de error aumenta. Si se compara con los resultados que ofrece el procesado *CMN*, se comprueba que son algo mejores, pero en ambos casos se constata una mejora en la tasa de reconocimiento.

5.2.4 Resultados con el Modelo de Basura y las Técnicas de OOV.

A continuación, se analizan los resultados que se obtienen al incorporar el modelo de basura como medida de calidad de las palabras reconocidas para comprobar el grado de rechazo en función del nivel de fiabilidad deseado.

En todos los casos, se parte de un reconocedor de voz que tiene como base el entrenamiento de unos modelos fonéticos libres de contexto a partir de la segmentación automática de la base de datos Albayzín junto con una pequeña base de datos de nombres a través de canal telefónico, con una red de reconocimiento de saltos dobles.

En este apartado se quiere comprobar cómo afecta el grado de exigencia de calidad de la palabra reconocida a la tasa de error y a la tasa de rechazo, utilizando el modelo de basura y la puntuación (*score*) descrita en el apartado 4.5, para las diferentes técnicas de robustez frente al canal de comunicaciones.

Para realizar las pruebas se ha utilizado un valor de $offset_2$ de 3 y un valor de $gain_2$ de 15, por tratarse de frases que constan de dos palabras.

5.2.4.1 Rechazo de Palabras en Función del Nivel de Calidad Exigido.

Primeramente, se analiza la influencia que la puntuación mínima exigida tiene sobre la tasa de error y sobre la tasa de rechazo.

Tabla 5-IV. Tasas de error y de rechazo del reconocedor cuando se imponen diferentes niveles de calidad de la palabra reconocida en función de la técnica de robustez frente al canal de comunicaciones para la base de datos *NOM_100_TEL*.

		Nada	*Score 0*	*Score 10*	*Score 20*	*Score 30*	*Score 40*
RASTA K=0,92	***T.E.***	**4,71 %**	**4,38 %**	**3,95 %**	**3,36 %**	**2,65 %**	**1,91 %**
	T.R.	**0,00 %**	**0,82 %**	**1,99 %**	**4,56 %**	**9,58 %**	**16,17 %**
CMN	***T.E.***	**4,48 %**	**4,15 %**	**3,85 %**	**3,26 %**	**2,68 %**	**2,04 %**
	T.R.	**0,00 %**	**0,61 %**	**1,48 %**	**3,72 %**	**7,21 %**	**12,64 %**

En la Tabla 5-IV se indican las diferentes tasas de error y de rechazo cuando se utilizan distintos umbrales de puntuación para la base de datos NOM_100_TEL, tanto para *RASTA* con un factor de 0,92, como para CMN.

En ambos casos, se comprueba que al aumentar el umbral de rechazo la tasa de error disminuye, al igual que la tasa de reconocimiento, pero a costa de un incremento mayor en la tasa de rechazo. Para ciertas aplicaciones puede resultar conveniente que la tasa de error sea muy baja, aunque disminuya también la tasa de acierto. De todas las configuraciones, la que resulta más interesante es la de *score* 20 ya que permite tener una tasa de error por debajo del 4 %, con una tasa de error del mismo orden.

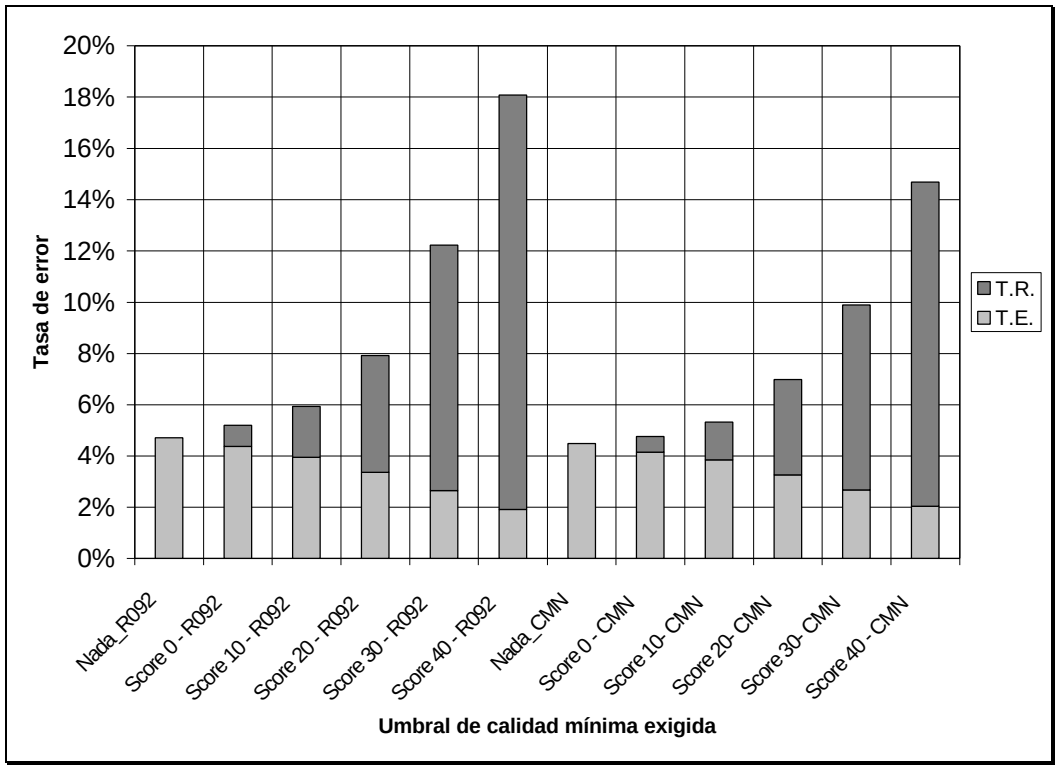

Figura 5-3. Gráfico con los resultados de tasa de error y de rechazo en función del valor de corte de la puntuación de la palabra reconocida para *RASTA* con *K* igual a 0,92 y para *CMN*.

En la Figura 5-3 se muestran los mismos resultados gráficamente.

5.2.4.2 Pruebas de Detección de Palabras Fuera de Vocabulario.

A continuación, se exponen los resultados obtenidos en la evaluación del nivel de rechazo de palabras que no se corresponden con las del vocabulario indicado. Para ello, se utiliza el vocabulario de nombres de la base de datos *NOM_100_TEL* para la base de datos de dígitos *DIG_15_TEL*.

Las pruebas están realizadas con modelos de unidades independientes de contexto y la técnica *CMN*.

Tabla 5-V. Tasas de rechazo del reconocedor fonético de voz utilizando la técnica *CMN*, con diferentes valores de puntuación de rechazo para la base de datos *DIG_15_TEL* con un vocabulario de 100 nombres.

	0	***10***	***20***	***30***	***40***
DIG_15_TEL	**49,76 %**	**60,67 %**	**70,36 %**	**78,52 %**	**85,57 %**

En la Tabla 5-V se muestran los valores de rechazo de palabras fuera de vocabulario en función del valor de puntuación de corte. Se observa que utilizando un

valor de 20 se rechaza más del 70 % de las palabras con una tasa de error baja y una tasa de rechazo con palabras dentro del vocabulario no muy elevada.

5.2.5 Resultados de Robustez Frente a Palabras Breves.

En el apartado 4.4, dedicado al estudio para aumentar la robustez en el reconocimiento de palabras breves usando reconocedores de voz fonéticos mediante la utilización de alternativas fonéticas, se vio que una manera de aumentar el grado de discriminación entre varios modelos era la incorporación de varias transcripciones fonéticas para una misma palabra. En este apartado se analizan los resultados que se obtienen en el reconocimiento de dígitos y palabras como *sí* y *no* al incorporar esta metodología.

Tabla 5-VI. Tasas de error utilizando todas las palabras de la base de datos *DIG_15_TEL* en función del tipo de modelos utilizados (fonético y fonético con transcripciones alternativas) con *RASTA* y *CMN*.

	Fonético	***Alternativas fonéticas***
RASTA K = 0,92	**14,52 %**	**12,29 %**
CMN	**12,57 %**	**10,65 %**

En la Tabla 5-VI se muestran los resultados de tasa de error al utilizar la concatenación de los modelos fonéticos según su transcripción fonética, y utilizando para algunos de ellos más de una (las enumeradas en la Tabla 4-IV). Tanto en los modelos entrenados utilizando *RASTA* con factor 0,92 como con *CMN*, se obtiene una mejora significativa en la tasa de error cuando se aplica a todas las palabras que contiene la base de datos *DIG_15_TEL*, es decir, los dígitos y las palabras *sí*, no, *cancelar*, *ayuda* y *terminar*.

Tabla 5-VII. Tasas de error utilizando los números de la base de datos *DIG_15_TEL* en función del tipo de modelos utilizados (fonético y fonético con transcripciones alternativas) con *RASTA* y *CMN*.

	Fonético	***Alternativas fonéticas***
RASTA K = 0,92	**11,09 %**	**8,19 %**
CMN	**9,32 %**	**6,62 %**

En la Tabla 5-VII se muestran los resultados de tasa de error con y sin transcripciones fonéticas, pero únicamente para los dígitos que aparecen en la base de datos *DIG_15_TEL*. Los resultados obtenidos con las transcripciones fonéticas

alternativas y el procesado *CMN* permiten afirmar que los modelos fonéticos de palabras aisladas breves pueden utilizarse para reconocer con unos valores no muy elevados. Mediante la utilización de modelo de rechazo de palabras según la calidad exigida, los resultados son incluso mejores.

Tabla 5-VIII. Tasas de error utilizando las palabras *sí* y *no* de la base de datos *DIG_15_TEL* en función del tipo de modelos utilizados (fonético y fonético con transcripciones alternativas) con *RASTA* y *CMN*.

	Fonético	***Alternativas fonéticas***
RASTA K = 0,92	**6,33 %**	**2,64 %**
CMN	**1,85 %**	**0,26 %**

Finalmente, en la Tabla 5-VIII se muestran los resultados de la tasa de error cuando se desea únicamente discriminar entre las palabras *sí* y *no*. En este caso, se observa una mejora elevada gracias a la utilización del procesado *CMN* y al uso de transcripciones fonéticas alternativas, obteniéndose una tasa de error de sólo un 0,26 %.

Como conclusión, se obtienen unos resultados muy significativos al utilizar las transcripciones fonéticas alternativas y, principalmente, cuando el número de palabras que se desea discriminar es más reducido. Esta técnica permite la utilización de modelos fonéticos con palabras breves con unos resultados ligeramente peores que los que se obtienen con modelos de palabras completas.

5.2.6 Resultados con Unidades Fonéticas Dependientes de Contexto.

Hasta ahora todas las pruebas realizadas se han llevado a cabo utilizando modelos de unidades fonéticas independientes de contexto (*monofonemas*), puesto que el tamaño de la base de datos de entrenamiento está bastante limitado aún cuando se utilice el entrenamiento con segmentación automática. Sin embargo, estas unidades así entrenadas no presentan mucha resolución, al utilizar vectores de características procedentes de fonemas en multitud de contextos diferentes. A continuación se analiza qué ocurre cuando se introducen algunas unidades dependientes de contexto.

Como se menciona en el apartado 2.3.4.4 sobre modelado de unidades fonéticas, es posible entrenar uno o los dos contextos del fonema a la vez. En este caso, se ha optado por considerar un conjunto limitado de *bifonemas izquierdos* correspondientes a los fonemas vocálicos. La razón reside en el hecho de que las vocales son las que presentan una mayor duración y, por tanto, en la base de datos están mejor representadas. Entre todos los bifonemas izquierdos de vocales se ha optado por considerar aquellos que por lo menos aparecen 100 veces en la base de datos. Las unidades consideradas son las siguientes:

- a[<,*] a[B,*] a[D,*] a[J,*] a[i,*] a[k,*] a[l,*] a[m,*] a[n,*] a[p,*] a[r,*] a[s,*] a[t,*] a[*,*]
- e[&,*] e[<,*] e[B,*] e[D,*] e[L,*] e[T,*] e[j,*] e[l,*] e[m,*] e[n,*] e[p,*] e[r,*] e[s,*] e[t,*] e[w,*] e[x,*]
- i[B,*] i[l,*] i[m,*] i[n,*] i[r,*] i[t,*]
- o[D,*] o[J,*] o[L,*] o[j,*] o[k,*] o[l,*] o[m,*] o[n,*] o[p,*] o[r,*] o[s,*] o[t,*]
- u[<,*] u[G,*] u[m,*] u[t,*]

Se utiliza el contexto izquierdo porque en castellano el número de sílabas *CV* es mayor al de otras configuraciones y, como en los grupos silábicos es donde se produce el mayor grado de coarticulación, su entrenamiento puede ayudar a aumentar el grado de reconocimiento.

Estas unidades son entrenadas usando el segmentador automático de saltos dobles. Durante el entrenamiento, aquellas unidades que no se corresponden con ninguno de estos modelos se sustituyen por monofonemas (como en el caso de todas las consonantes y un subconjunto de las vocales), que no son reentrenados.

Estas nuevas unidades suponen 153 nuevos modelos de estado frente a los 82 de los monofonemas, por lo que no supone un valor excesivo que disminuya de forma importante las prestaciones en eficiencia.

5.2.6.1 Unidades Dependientes de Contexto y las Técnicas de Independencia del Canal de Comunicaciones.

En primer lugar, se analiza el funcionamiento del reconocedor cuando se utilizan las unidades dependientes de contexto con diferentes técnicas de robustez frente al canal de comunicaciones.

Tabla 5-IX. Tasas de error dependiendo del tipo de unidades fonéticas utilizadas. Por un lado, sólo monofonemas (*M*), por el otro *M* con bifonemas izquierdos de vocales (*BIV*)

	34 M (82 E)		***34 M + 51 BIV (235 E)***	
	RASTA 0,92	***CMN***	***RASTA 0,92***	***CMN***
NOM_100_TEL	**4,71 %**	**4,48 %**	**3,21 %**	**2,85 %**
DIG_15_TEL	**14,52 %**	**12,57 %**	**14,14 %**	**12,31 %**

En la Tabla 5-IX se muestra la tasa de error de la base de datos de reconocimiento de nombres y apellidos con unidades independientes de contexto, y cuando se incluyen ciertas unidades dependientes de contexto como son las vocales en

contexto izquierdo (*M + BIV*) para el caso de utilizar filtrado *RASTA* con *K* igual a 0,92 y *CMN*. Para ambas bases de datos, *NOM_100_TEL* y *DIG_15_TEL*, se obtiene una mejora gracias a la utilización de modelos dependientes de contexto, aunque en el caso de la segunda la mejora es algo inferior debida a que el número de unidades nuevas utilizadas es más reducido en el caso de los dígitos que en el de los nombres.

La reducción sobre la base de datos *NOM_100_TEL* y la técnica *CMN* usando bifonemas permite disponer de una tasa de error similar a la que se obtiene con monofonemas y rechazo de palabras con *score* de 30, pero con un 7,21 % más de acierto al no haber eliminación de palabras con calidad mala.

Para la base de datos *DIG_15_TEL* la mejora es mucho más ligera porque el número de bifonemas utilizados por el vocabulario de dígitos es más reducido.

5.2.6.2 Unidades Dependientes de Contexto y Rechazo de Palabras.

Una vez constatado que la utilización de unidades dependientes de contexto permite una disminución en la tasa de error similar a la que se obtiene con rechazo de palabras y monofonemas, pero sin tener un factor adicional de palabras rechazadas, es interesante analizar lo que ocurre cuando se utiliza rechazo y unidades dependientes de contexto a la vez.

Tabla 5-X. Tasas de error y de rechazo del reconocedor cuando se imponen diferentes niveles de calidad de la palabra reconocida en función de la técnica de robustez frente al canal de comunicaciones para la base de datos *NOM_100_TEL* y *M + BIV*.

		Nada	*Score 0*	*Score 10*	*Score 20*	*Score 30*	*Score 40*
RASTA K=0,92	***T.E.***	**3,20 %**	**2,93 %**	**2,78 %**	**2,32 %**	**1,68 %**	**1,20 %**
	T.R.	**0,00 %**	**0,74 %**	**1,20 %**	**3,21 %**	**7,11 %**	**12,20 %**
CMN	***T.E.***	**2,85 %**	**2,62 %**	**2,47 %**	**2,22 %**	**1,76 %**	**1,35 %**
	T.R.	**0,00 %**	**0,64 %**	**1,12 %**	**2,57 %**	**5,17 %**	**9,96 %**

En la Tabla 5-X se observan los resultados cuando se impone un nivel de calidad mínimo a partir de un modelo de basura obtenido de las distancias a los modelos de estado de los monofonemas para *RASTA* con *K* igual a 0,92 y para *CMN*. En ambos casos se constata una sensible reducción en la tasa de error al incrementarse el nivel de calidad mínimo exigido. Estos resultados son mucho mejores que los obtenidos con monofonemas, porque se puede llegar a tener un error del 1,20 % con sólo un rechazo del 12,23 % para aplicaciones que requieran un nivel de fiabilidad muy elevado con *RASTA*, en el caso *CMN* una tasa de error de 1,35 % con un rechazo del 9,96 %.

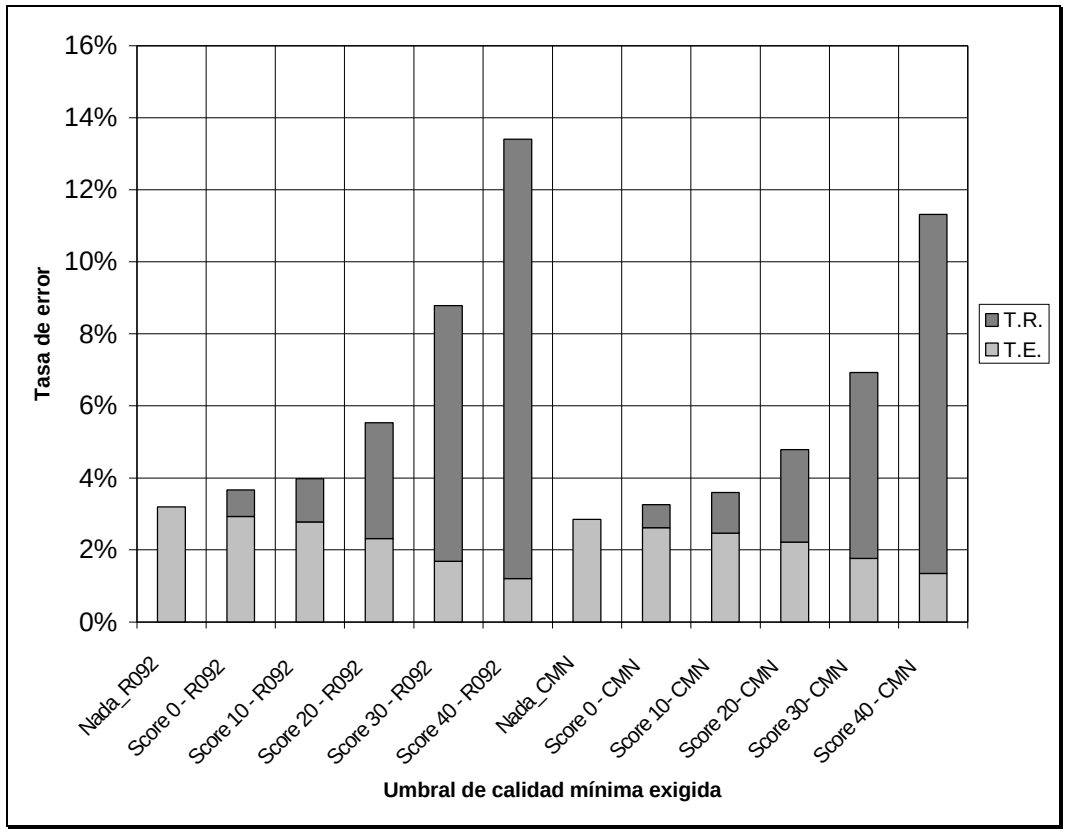

Figura 5-4. Gráfico con los resultados de tasa de error y de rechazo en función del valor de corte de la puntuación de la palabra reconocida para *RASTA* con *K* igual a 0,92 y para *CMN* y con *M + BIV*.

En la Figura 5-4 se muestran los mismos resultados de la Tabla 5-X pero de forma gráfica. Se observa más claramente la influencia que la variación del *score* provoca sobre la tasa de error y de rechazo. Un valor demasiado exigente de la calidad permite una reducción del error, pero a costa de un incremento cada vez mayor de la tasa de rechazo.

5.2.7 Resultados de Agrupamiento de Modelos a partir de Sammon.

Como se menciona en el apartado 4.6 sobre distancias entre modelos de estado fonético, la proyección de Sammon puede utilizarse para evaluar en dos fases la similitud de modelos de estado que son susceptibles de agruparse. En primer lugar, se extrae un conjunto inicial de agrupaciones a partir de la proyección de todas las medias de los modelos fonéticos. Posteriormente, se comprueba cuáles de esas agrupaciones son válidas mediante la proyección simultánea de vectores sintéticos de los modelos que forman cada una de ellas.

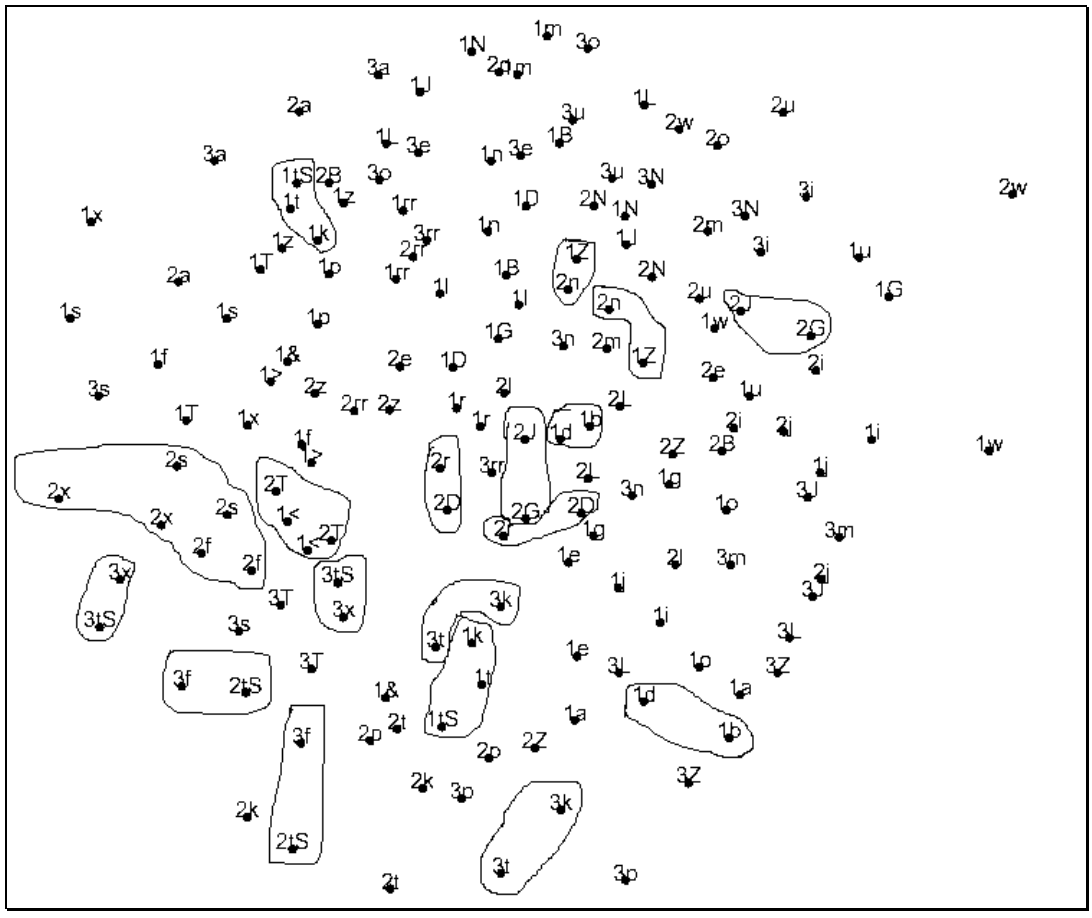

Figura 5-5. Proyección de Sammon de los 164 vectores de medias de los 34 monofonemas (82 modelos de estado en total) para el castellano.

Al realizar la proyección de Sammon sobre los vectores de medias de los modelos de estado de las unidades fonéticas, se tiene una distribución aproximada de semejanza de estos modelos entre sí. Analizando detenidamente el mapa, existe un conjunto de modelos que a priori parecen cercanos, por lo que sería posible agruparlos. En la Figura 5-5 se muestra dicha proyección, indicándose el conjunto inicial de agrupaciones (se marca como agrupación aquella en la que las dos medias de ambos modelos de estado están cercanas).

Tabla 5-XI. Tabla con el conjunto inicial de modelos de estado con posibilidades de ser similares al situarse sus medias cercanas con la proyección de Sammon.

1*tS* + 1*t* + 1*k*	1< + 2*T*
3*tS* + 3*x*	2*s* + 2*x* + 2*f*
3*f* + 2*tS*	2*r* + 2*D*
3*t* + 3*k*	1*d* + 1*b*
1*Z* + 2*n*	2*G* + 2*J*

En la Tabla 5-XI se indica la lista de agrupaciones resultante de la proyección de Sammon, que se corresponden con las de la Figura 5-5. Analizando esta lista se comprueba que los agrupamientos no son casuales sino que tienen su explicación, por ejemplo, en el caso del fonema *tS*, ya que sus tres estados son similares a los del primer estado de la *t*, el segundo de la *f* y el tercero de la *x*, respectivamente. Destaca el agrupamiento del primer estado de la *tS* con el primero de la *t* y el primero de la *k* (los tres modelan la parte inicial de un fonema oclusivo sordo).

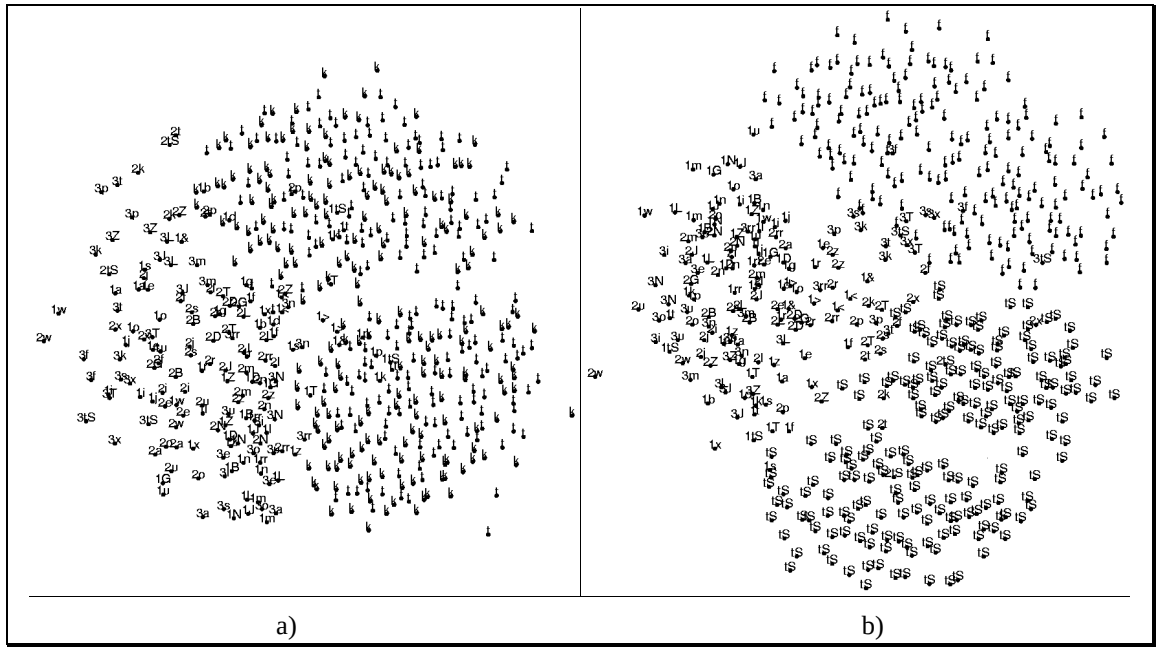

Figura 5-6. Proyecciones de Sammon para: a) los vectores sintéticos del primer estado de la *t* y de la *k* junto con todos los vectores de medias; b) los vectores sintéticos del tercer estado de la *f* y el segundo de la *tS*.

Se observa en la Figura 5-6 las proyecciones de Sammon para dos casos diferentes. Por un lado, está la proyección de los vectores sintéticos de dos estados (el primero de la *t* y el primero de la *k*) donde se comprueba que existe un claro solapamiento entre ambos conjuntos. Esto indica que son modelos de estado realmente cercanos y que pueden agruparse en el entrenamiento. Además, se comprueba que el primer estado de la *tS* también está dentro de las nubes de puntos generadas por los vectores sintéticos. Así, en este caso, los tres modelos de estado se pueden agrupar en un mismo senón.

Por otro lado, está el caso del tercer estado de la *f* con el segundo estado de la *tS*. Se comprueba que aún cuando en la proyección de la Figura 5-5 parecía que los modelos estaban muy cercanos, no lo están, lo que realmente ocurre es que no existe ningún modelo intermedio entre ambos y dicha proyección los sitúa cerca.

Realizando esta proyección para todos los casos enumerados en la Tabla 5-XI se comprueba que sólo 4 de las 10 posibilidades de agrupamiento son aceptables.

Tabla 5-XII. Tabla con el conjunto de modelos de estado que realmente están cercanas y pueden ser agrupadas por ser muy similares entre sí.

1*tS* + 1*t* + 1*k*	2*r* + 2*D*
3*t* + 3*k*	1*d* + 1*b*

En la Tabla 5-XII se muestra el conjunto de las 4 agrupaciones que pueden realizarse por la gran similitud y coincidencia de los modelos de estado.

Tabla 5-XIII. Tasas de error del reconocedor para la base de datos *NOM_100_TEL* cuando se utilizan 34 monofonemas (que se corresponde con 82 modelos de estado) y cuando se utiliza la proyección de Sammon para reducir su número (en total, 77 modelos de estado)

	Sin Sammon	***Con Sammon***
RASTA K=0,92	**4,71 %**	**4,81 %**

En la Tabla 5-IX se muestra el resultado obtenido al realizar el entrenamiento con el agrupamiento de los modelos de estado que resultan similares según la proyección de Sammon. En este caso, se obtiene un aumento en la tasa de error, al producirse un ligero aumento en el grado de confusión entre modelos fonéticos. Sin embargo, el incremento en el error no es excesivo para un conjunto de unidades que ha sido reducido (se ha pasado de 82 modelos de estado a 77).

5.2.8 Conclusiones sobre los Resultados.

El uso del entrenamiento automático de la base de datos Albayzín junto con una pequeña base de datos de nombres de canal telefónico y la introducción de filtrado *RASTA* y procesado *CMN*, facilita el entrenamiento de unos modelos más robustos frente a la variabilidad del locutor y del canal de comunicaciones en aplicaciones de reconocimiento de nombres y dígitos.

La utilización de una puntuación (*score*) para valorar la calidad de una palabra permite disminuir la tasa de error de reconocimiento a cualquier valor a expensas de un aumento en la tasa de rechazo. Este nivel de rechazo puede resultar muy útil para algunas aplicaciones telefónicas donde se requiera un grado de exactitud muy elevado.

La utilización de transcripciones fonéticas alternativas permite una disminución apreciable en la tasa de error, principalmente cuando el número de palabras a discriminar es más reducido. Esta técnica permite poder utilizar modelos fonéticos de palabras breves con bajas tasas de error.

Finalmente, el entrenamiento de modelos dependientes del contexto izquierdo para los fonemas vocálicos, permite una reducción importante de la tasa de error con un

número controlado de modelos entrenados. La utilización del rechazo de palabras permite disponer de un reconocedor con una tasa de error muy baja y un rechazo controlado, con un incremento moderado en el cómputo necesario.

La proyección de Sammon es capaz de dar información sobre la semejanza entre modelos con el fin de agruparlos para reducir el número total de unidades. Se ha probado que en el caso de monofonemas esto supone una disminución de la carga computacional con un ligero incremento en la tasa de error. Con un número mayor de unidades, la reducción que se puede conseguir en memoria y cómputo es más significativa.

5.3 Resultados en Condiciones Reales

5.3.1 Introducción. Diseño de los Experimentos.

Los resultados de este apartado muestran cómo afecta la incorporación de un reconocedor de voz con unas prestaciones medidas en condiciones de laboratorio cuando se realizan pruebas en un sistema funcionando en tiempo real. Por ello, se analizan ciertos procesados adicionales como son: el procesado *CMN Recursivo* para poder implementar el *CMN* en tiempo real con un bajo retardo, la utilización del detector de extremos, una red de reconocimiento incluyendo la posibilidad de palabras comodín y la introducción de poda de los caminos menos probables durante el algoritmo de Viterbi (*beam search*). En todos ellos se utilizan modelos entrenados a partir de la segmentación automática de la base de datos de Albayzín junto con una pequeña base de datos de nombres de canal telefónico.

El orden de exposición de resultados es el siguiente: en el apartado 5.3.2 se estudia el funcionamiento del algoritmo *CMN Recursivo* (*CMNR*); en el apartado 5.3.3 se analiza la influencia del detector de extremos sobre la tasa de error del reconocedor; en el apartado 5.3.4 la influencia de la estructura de la red gramatical en la tasa de error; en el apartado 5.3.5 el fenómeno de poda de los caminos; en el apartado 5.3.6 los resultados con *bifonemas izquierdos de vocales*; finalmente, en el apartado 5.3.7 se exponen unas conclusiones sobre los resultados en condiciones reales.

5.3.2 Utilización del Procesado CMN Recursivo (CMNR).

El procesado *CMN* no se puede utilizar en un sistema en tiempo real porque para aplicarlo es necesario tener la frase completa pronunciada. Por eso, han aparecido alternativas como el *CMN Recursivo* que, como se explica en el apartado 4.3.4.1, consiste en ir calculando de forma progresiva el valor medio a partir de una ventana de la señal de voz con un retardo de N tramas (que es el parámetro de diseño).

Tabla 5-XIV. Tasas de error del reconocedor fonético de voz al aplicar la técnica *CMN Recursivo* para diferentes tamaños de ventana sobre la base de datos *NOM_100_TEL*.

	CMN	*N=63*	*N=51*	*N=46*	*N=41*	*N=36*	*N=31*	*N=26*	*N=21*	*N=16*	*N=9*
34M (82E)	4,48%	4,84%	4,31%	4,28%	4,28%	4,30%	4,31%	4,39%	4,30%	4,62%	4,82%
34M+51BIV	2,85%	3,18%	2,70%	2,88%	2,70%	2,80%	3,04%	2,93%	2,88%	3,16%	3,25%

En la Tabla 5-XIV se reflejan los resultados obtenidos al aplicar el *CMN Recursivo* con diferentes tamaños de ventana, comparándolos con los del *CMN Puro* para la base de datos *NOM_100_TEL* con unos modelos entrenados utilizando *CMN Puro*. Para todas las pruebas se utiliza un l de $0{,}955$ como constante de actualización.

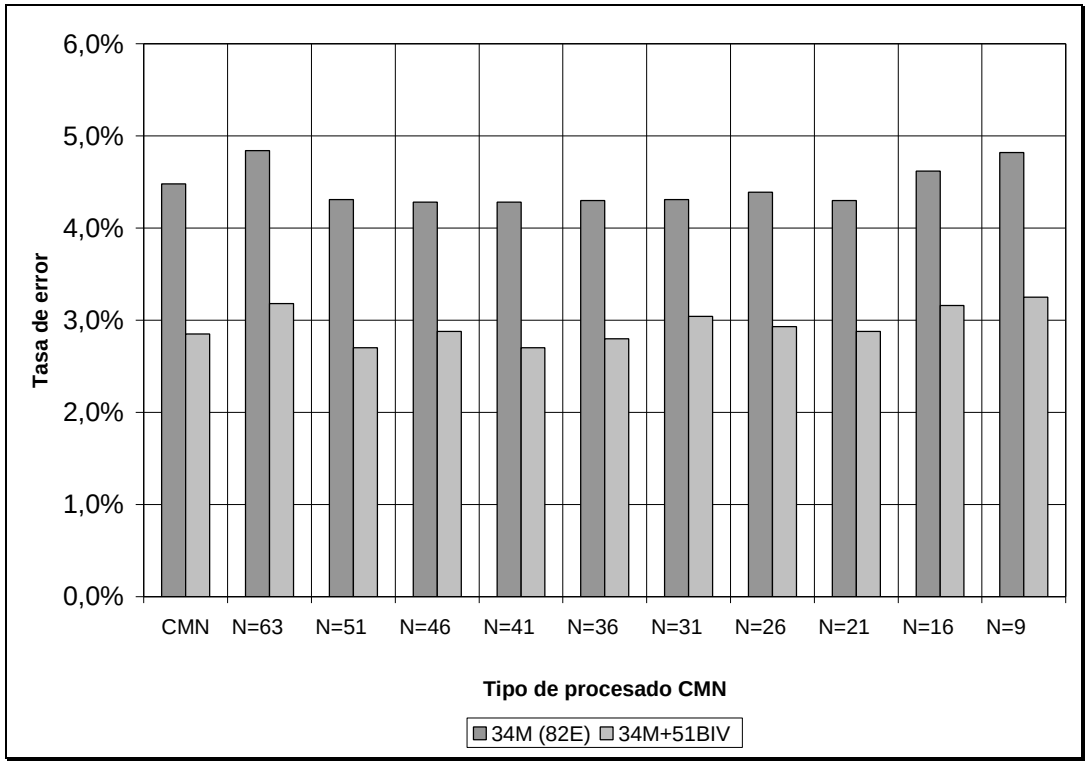

Figura 5-7. Gráfico con los resultados relativos a la variación de la tasa de error con el tamaño de la ventana de análisis del procesado *CMN Recursivo*. Además, se compara con la tasa de error del *CMN Puro*, que considera la señal completa.

Como se puede observar en la Tabla 5-XIV y en la Figura 5-7, es posible utilizar una ventana de entre $0{,}33$ *sg* (N = 16 tramas) y $0{,}82$ *sg* (N = 51 tramas) para aplicar el procesado *CMN Recursivo*, con unos resultados que mejoran los del filtrado *RASTA*. El hecho de utilizar una ventana con un valor N comprendido entre 16 y 51 permite eliminar el peso que el silencio previo pudiera tener en la estimación del vector de medias durante la pronunciación de las frases, por lo que incluso la tasa de error es menor que la que se obtiene con *CMN* de toda la frase.

Para las subsiguientes medidas y pruebas se ha utilizado un valor de ventana de 0,66 *sg* (*N* = 41 tramas), aunque con cualquier otro valor comprendido dentro del rango anterior los resultados son similares.

Tabla 5-XV. Tasas de error del reconocedor fonético de voz para la base de datos *NOM_100_TEL* para las diferentes técnicas de robustez frente al canal de comunicaciones y del número de unidades fonéticas entrenadas.

	RASTA (0,92)	***CMN***	***CMNR (N=41)***
M (82E)	**4,71 %**	**4,41 %**	**4,28 %**
M+BIV (235E)	**3,21 %**	**2,85 %**	**2,70 %**

En la Tabla 5-XV se indican los resultados que se obtienen cuando se aplican las tres técnicas de robustez frente al canal de comunicaciones (*RASTA* con factor 0,92, *CMN* y *CMNR* con una ventana de 660 *ms* que son 41 tramas). Se puede apreciar que para los dos tipos de unidades fonéticas utilizadas se obtiene una mejora gracias a la utilización del proceso *CMNR*.

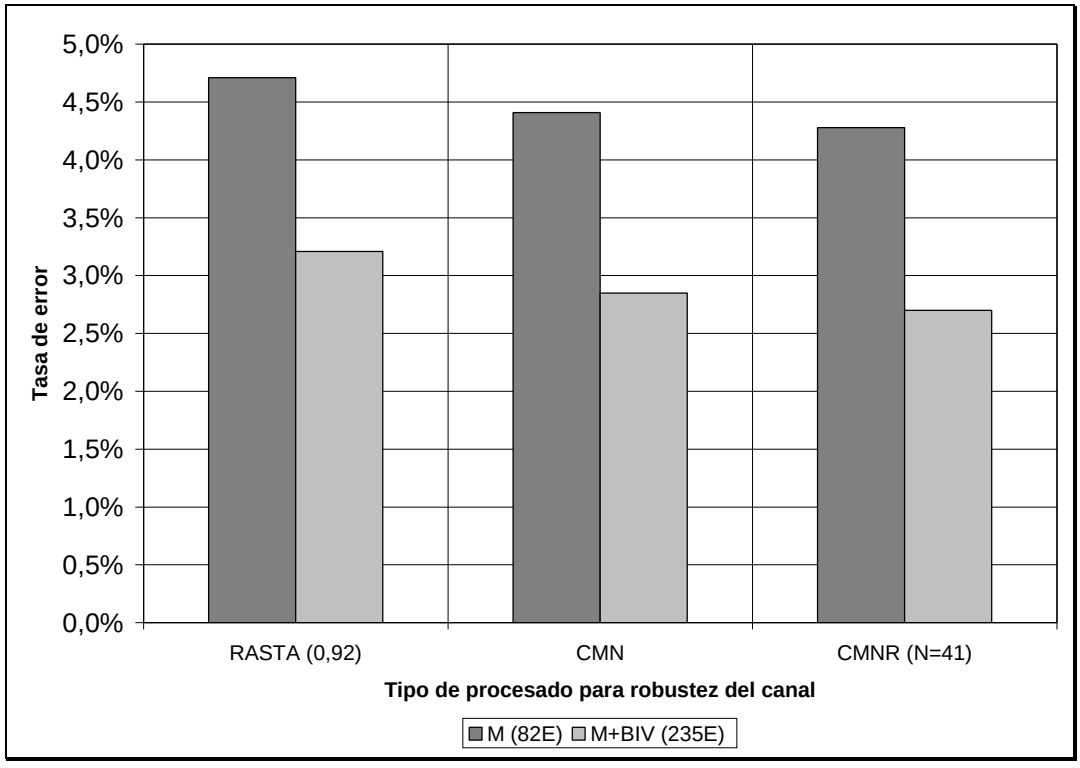

Figura 5-8. Gráfico sobre tasas de error obtenidas al aplicar las diferentes técnicas de robustez frente al canal de comunicaciones con diferentes tipo de unidades fonéticas para la base de datos de reconocimiento *NOM_100_TEL*.

En la Figura 5-8 se presentan los mismos valores de la Tabla 5-XV pero de forma gráfica. Se puede observar que la mejora en la tasa de error al utilizar *CMNR*

junto con bifonemas izquierdos de vocales es elevada comparada con *RASTA* y monofonemas.

5.3.3 Influencia de la Detección de Extremos en el Reconocimiento.

El detector de extremos es un elemento fundamental para que un reconocedor de voz pueda ser utilizado en tiempo real. Determina el fin de una palabra para realizar la búsqueda hacia atrás en la red de reconocimiento y extraer la palabra reconocida. En este apartado se muestra la influencia sobre la tasa de error del detector de extremos.

Tabla 5-XVI. Tasas de error del reconocedor fonético de voz para la base de datos *NOM_100_TEL* en función de la utilización del detector de extremos para diferentes técnicas de procesado de canal para monofonemas.

	Red sin comodines	
	Sin detección	***Con detección***
***RASTA* (*0,92*)**	**4,71 %**	**4,89 %**
CMN	**4,48 %**	**4,18 %**
***CMNR* (*N=41*)**	**4,28 %**	**4,36 %**

En la Tabla 5-XVI se observa la influencia de la detección de extremos sobre la tasa de error. Se comprueba que para el caso del filtrado *RASTA* el detector de extremos produce un incremento en la tasa de error.

Para *CMN* se produce el efecto de una disminución de la tasa de error al usar la detección de extremos. Esto se debe a que la estimación del valor medio de la frase se realiza a partir de todas las tramas analizadas, incluidas las de silencio. En el caso de usar el detector de extremos, el silencio posterior se acorta de forma importante obteniendo una estimación del vector de medias que tiene en cuenta más vectores de pronunciación y menos de silencio, por lo que el *CMN* resulta de esta manera más efectivo. Es un efecto similar al que se produce cuando se aplica el *CMN Recursivo*.

Para el caso del *CMN Recursivo* se produce también un ligero aumento de la tasa de error debido a que el silencio no afecta de forma tan importante al funcionamiento de esta técnica.

5.3.4 Influencia de Distintas Estructuras de Redes de Reconocimiento.

Otro aspecto no analizado hasta ahora es el de la utilización de una estructura de red gramatical que posibilita la introducción de modelos de palabras comodín, tal como se muestra en la Figura 3-23. Esta estructura así definida afecta al reconocedor, al introducirse dos nodos de silencio para permitir la inclusión de dichos modelos.

A continuación, se analiza la influencia de esta nueva estructura sobre la tasa de error del reconocedor para diferentes técnicas de robustez de canal y, adicionalmente, con el detector de extremos usando monofonemas.

Tabla 5-XVII. Tasas de error del reconocedor fonético de voz para la base de datos *NOM_100_TEL* cuando se incorpora una red de reconocimiento que permite modelos comodín con monofonemas.

	Red con comodines	
	Sin detección	***Con detección***
RASTA (0,92)	**5,20 %**	**5,33 %**
CMN	**4,54 %**	**4,48 %**
CMNR (N=41)	**4,67 %**	**4,87 %**

En la Tabla 5-XVII aparecen los resultados obtenidos al incorporar el efecto de la estructura de la red de reconocimiento sobre la tasa de error. Se observa que se produce un incremento en la tasa de error respecto de los valores correspondientes sin utilizar red de reconocimiento, con el mismo efecto debido al detector de extremos.

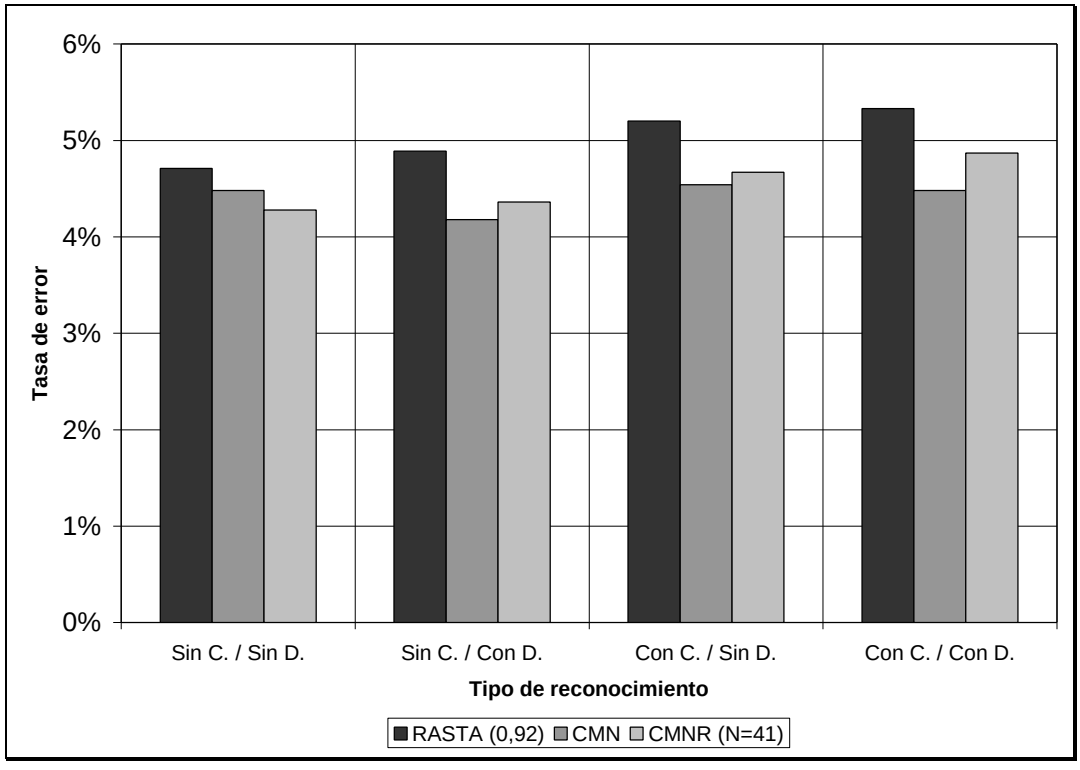

Figura 5-9. Gráfico con los resultados relativos a la variación de la tasa de error en función de la utilización del detector de extremos y de la red de reconocimiento con comodines.

Los resultados de la Tabla 5-XVI y de la Tabla 5-XVII se observan gráficamente en la Figura 5-9. La detección de extremos y la estructura de red de reconocimiento con

comodines presentan una tasa de error ligeramente superior que en condiciones sin detección y sin comodines.

5.3.5 Influencia de la Poda sobre el Reconocimiento de Voz.

La aplicación de una técnica de podado sobre los caminos que se recorren a la hora de aplicar el algoritmo de Viterbi no mejora las prestaciones del reconocedor en lo que se refiere en un aumento de la robustez, pero en determinadas circunstancias sí posibilita un incremento en el tamaño del vocabulario al mejorar la eficiencia del sistema.

Como se menciona en el apartado 3.6.3, es posible no recorrer los caminos que menos probabilidades presentan al avanzar. A continuación, se muestran los resultados de aplicar esta técnica en función de dicho $d(t)$. En este caso, al tener los valores de probabilidad en log-verosimilitudes, sólo se avanza por aquellos caminos en los que el coste acumulado sea mayor al de la trama anterior menos una determinada cantidad (*factor*). Cuanto mayor sea este *factor* la poda realizada es menos restrictiva.

Tabla 5-XVIII. Tasas de error del reconocedor fonético de voz cuando se realiza una poda de los peores caminos con el algoritmo de Viterbi para aumentar la eficiencia del sistema, en una red que no permite comodines y sin utilizar detector de extremos para la base de datos de reconocimiento *NOM_100_TEL* con monofonemas.

	35	*50*	*75*	*100*	*INF*
RASTA (0,92)	**7,08 %**	**5,28 %**	**4,81 %**	**4,71 %**	**4,71 %**
CMN	**8,46 %**	**5,45 %**	**4,56 %**	**4,48 %**	**4,48 %**
***CMNR* (*N=41*)**	**8,31 %**	**5,38 %**	**4,38 %**	**4,28 %**	**4,28 %**

Tabla 5-XIX. Tasas de error del reconocedor fonético de voz cuando se realiza una poda de los peores caminos con el algoritmo de Viterbi para aumentar la eficiencia del sistema, en una red que permite comodines y con detector de extremos para la base de datos de reconocimiento *NOM_100_TEL* con monofonemas.

	35	*50*	*75*	*100*	*INF*
RASTA (0,92)	**7,59 %**	**5,91 %**	**5,40 %**	**5,33 %**	**5,33 %**
CMN	**8,20 %**	**5,42 %**	**4,61 %**	**4,48 %**	**4,48 %**
***CMNR* (*N=41*)**	**8,87 %**	**5,91 %**	**4,97 %**	**4,87 %**	**4,87 %**

En la Tabla 5-XVIII y en la Tabla 5-XIX se muestran los resultados de cómo afecta el grado de poda en el algoritmo de reconocimiento a la tasa de error. En el

primer caso, para la base de datos *NOM_100_TEL* sin detección de extremos y la red sin permitir comodines, en el segundo, con detección de extremos y red con comodines.

Se observa que la tasa de error aumenta al utilizar un factor menor. Sólo cuando se utiliza un factor de 50 en adelante, podría compensarse desde un punto de vista de robustez la introducción de la poda.

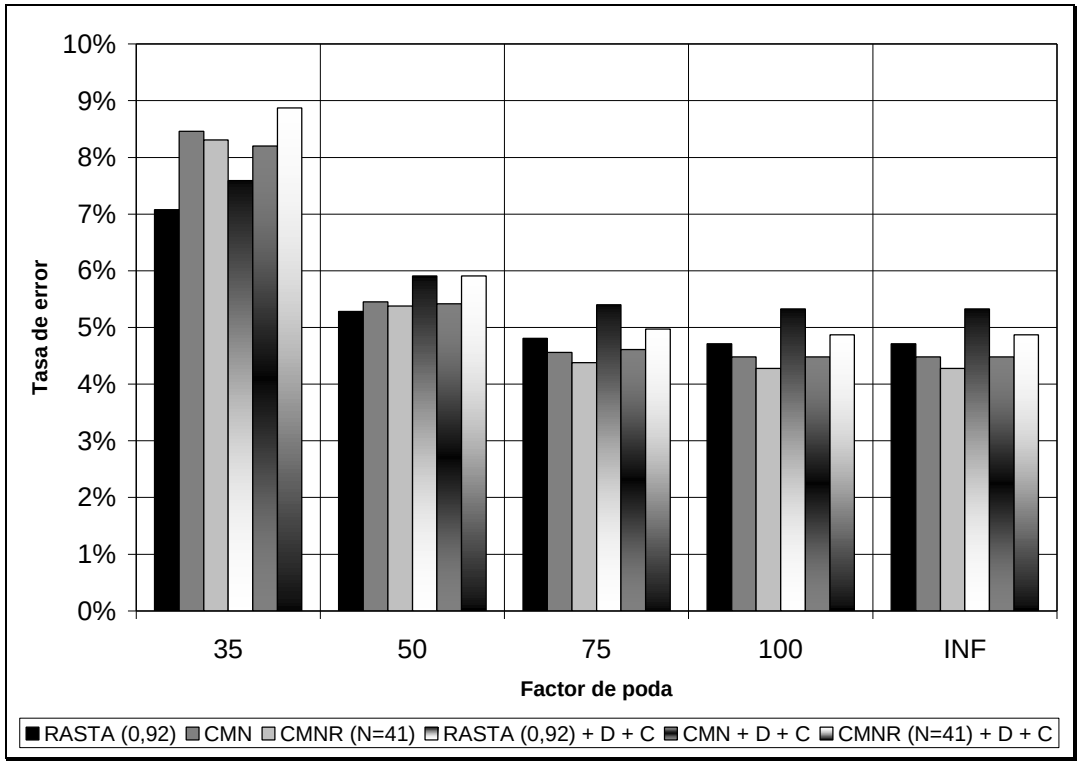

Figura 5-10. Gráfico con los resultados de la variación de la tasa de error en función del factor de poda, para diferentes técnicas de compensación de canal y considerando la detección de extremos (D) y de la red de reconocimiento con comodines (C) para monofonemas.

En la Figura 5-10 se reflejan a modo de resumen las tasas de error en función del factor de poda, de las diferentes técnicas de compensación del canal y de la utilización o no del detector de extremos y de la red de reconocimiento con comodines.

5.3.5.1 Comparación de las Tasas de Error con Monofonemas y BIV.

Un aspecto interesante consiste en comparar los resultados que la poda de caminos tiene sobre las tasas de error al utilizarse monofonemas con bifonemas izquierdos.

Tabla 5-XX. Tasas de error del reconocedor fonético de voz para diferentes valores del factor de poda, con la técnica *CMNR* para un *N* de 41, usando la base de datos *NOM_100_TEL* para monofonemas y para monofonemas con bifonemas izquierdos de vocales.

	25	*35*	*50*	*75*	*100*	*1000*	*INF*
M (82E)	**13,66 %**	**8,31 %**	**5,38 %**	**4,38 %**	**4,28 %**	**4,28 %**	**4,28 %**
M+BIV (235E)	**12,69 %**	**7,44 %**	**4,08 %**	**3,21 %**	**2,78 %**	**2,78 %**	**2,78 %**

En la Tabla 5-XX se observan los resultados comparativos de las tasas de error con diferentes factores de poda. En todos los casos la utilización de bifonemas proporciona mejores resultados en comparación con la utilización de únicamente monofonemas (excepto a partir de un factor de 35). Si el ahorro en cómputo que se obtiene al reducir el procesado en la red de reconocimiento compensa al que se obtiene con monofonemas sin poda, resulta una configuración bastante interesante.

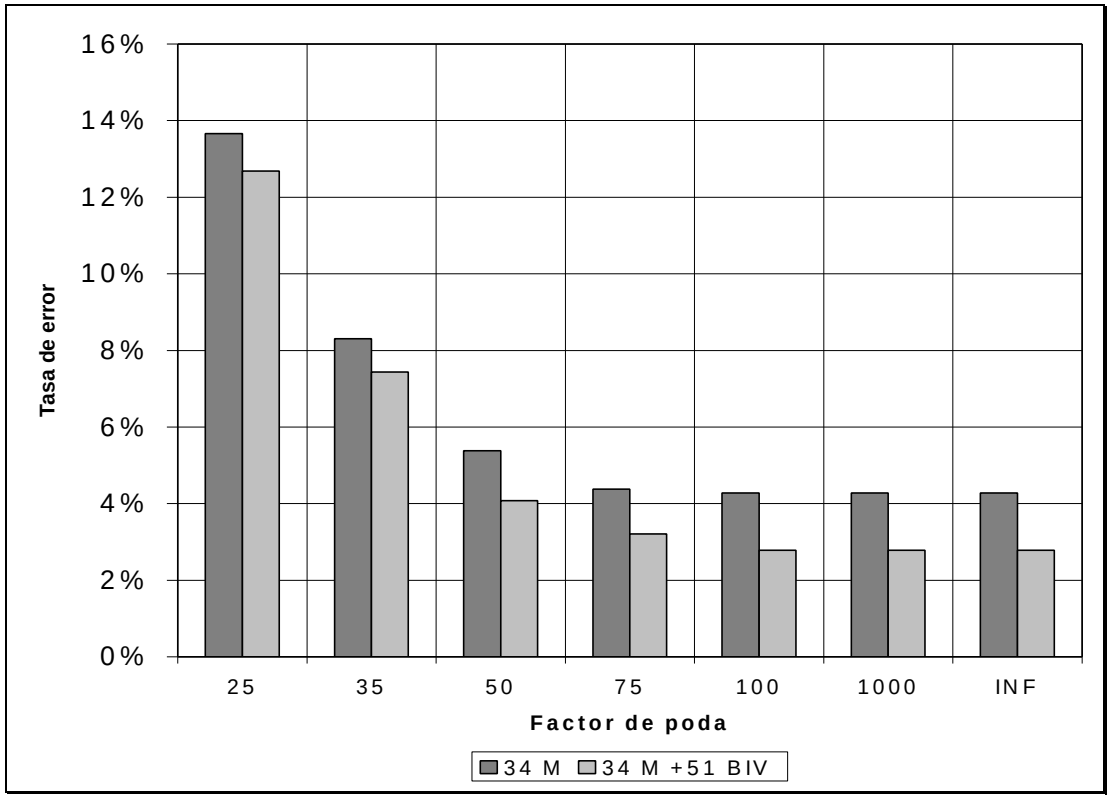

Figura 5-11. Gráfico con los resultados de variación de la tasa de error en función de la utilización de monofonemas o de monofonemas con bifonemas izquierdos de vocales en una red sin comodines y sin detección de extremos.

En la Figura 5-11 se muestran los mismos resultados de la Tabla 5-XX de forma gráfica.

5.3.6 Influencia de la Poda, la Detección de Extremos y la Red de Reconocimiento al Usar Modelos de Monofonemas con BIV.

Una vez vista la influencia que la red de reconocimiento, la detección de extremos y la poda tienen sobre la tasa de error al utilizar monofonemas, falta ver cómo estos aspectos afectan cuando se incorporan bifonemas izquierdos de ciertas vocales.

Tabla 5-XXI. Tasas de error al considerar los diferentes tipos de red de reconocimiento, la detección de extremos y la poda con un valor de 75 para la base de datos *NOM_100_TEL* utilizando la técnica *CMN Recursivo* con un valor de *N* de 41.

	Red sin comodines		***Red con comodines***	
	Sin detección	***Con detección***	***Sin detección***	***Con detección***
SIN PODA	**2,70 %**	**3,32 %**	**2,85 %**	**3,39 %**
PODA 75	**2,78 %**	**3,34 %**	**2,93 %**	**3,42 %**

En laTabla 5-XXI se muestra de forma comparativa la influencia de la detección de extremos y de la estructura de la red de reconocimiento cuando se utilizan monofonemas con ciertos bifonemas izquierdos de vocales prescindiendo de la poda y cuando se aplica con un factor de poda de 75. En todos los casos, la medida es sobre la base de datos *NOM_100_TEL* para *CMN Recursivo* con un N igual a 41.

Los resultados ofrecidos muestran que la detección de extremos afecta ligeramente a la tasa de error, mientras que la estructura de reconocimiento para permitir la introducción de palabras comodín resulta afectada de forma más significativa. Para un valor de poda de 75, las pérdidas que se producen no son significativas si se compara con los otros dos aspectos considerados.

5.3.7 Conclusiones sobre los Resultados.

Una vez analizadas las pruebas en este apartado destinado a aspectos relacionados con el funcionamiento en tiempo real, se obtienen las siguientes conclusiones:

- El método *CMNR* demuestra ser muy eficaz en la adaptación del canal de comunicaciones con un bajo retardo, incluso mejora el del *CMN*, gracias a que se disminuye la influencia de las tramas de silencio sobre la estimación del vector de medias.
- La detección de extremos no afecta de forma importante al reconocimiento, incluso para *CMN* mejora en cierta medida por el mismo efecto de reducción del número de tramas de ruido utilizadas para estimar el vector de medias.

- La introducción de palabras comodín afecta ligeramente a la tasa de error, pero permite que el locutor pueda utilizar diferentes maneras de dialogar con la máquina de forma más robusta.
- La utilización de la poda de los mejores caminos no afecta mucho a la tasa de error a partir de un determinado factor. En función de la ganancia en procesado, del tamaño del vocabulario y de las restricciones en cómputo se pueden particularizar las prestaciones del reconocedor para adaptarse a esas circunstancias.

5.4 Medidas de Capacidad del Sistema de Reconocimiento en Tiempo Real

5.4.1 Introducción.

Aunque la presente tesis está centrada en el diseño e implementación de reconocedores robustos, la eficiencia del sistema nunca ha dejado de considerarse, como se muestra en [Pue98] y [Lop99b]. En este apartado, aparecen resultados sobre capacidades en memoria y procesado que permiten tener una idea de la eficiencia del sistema de reconocimiento.

Las medidas de capacidad de procesado de un sistema siempre resultan complicadas porque dependen de las prestaciones de los equipos utilizados. Aún cuando se pueda establecer de manera relativa la potencia entre dos equipos diferentes, no se puede hacer una extrapolación directa, porque dicho incremento de potencia depende del tipo de procesado a realizar. Por ello, y para dar unas medidas de referencia, éstas se realizan sobre un ordenador con prestaciones intermedias de entre todos los que se encuentran disponibles. El equipo utilizado tiene las siguientes:

- ***Ordenador personal***: *Pentium* 200 *MMX*^{TM}, 64 *MB* de memoria *RAM* con *Windows NT 4.0*.
- ***Tarjeta de procesado de señal***: Integrada con un *DSP* de la familia *C31X* de *Texas Instruments* a una velocidad de 60 *MHz*, con 256 *kw* de memoria. Comunicación a través de un bus *ISA* de 8 *bits*.

En este apartado se estudian tres aspectos principales:

- Variación de la memoria en función del vocabulario.
- Tamaño del vocabulario en función del número de reconocedores activos a la vez y del cómputo necesario.
- Influencia de la poda sobre el incremento del vocabulario.

Un análisis más detallado de la implementación eficiente sobre este reconocedor de voz fonético puede estudiarse en [Lop00].

5.4.2 Variación de la Memoria en Función del Vocabulario y los Modelos.

Hay dos parámetros que determinan el tamaño de la memoria necesaria, uno son los modelos y otro el tamaño del vocabulario, el resto es un valor constante de memoria que es lo que ocupa el programa y otras variables adicionales del procesado sin considerar los modelos fonéticos ni las palabras del vocabulario. En todos los casos se considera una longitud máxima de frase de $4{,}8\ s$.

La memoria constante que presenta es de $118.784\ bytes$ para el código del programa, y de $4.008\ bytes$ de memoria asociada con la extracción de características y otros procesos adicionales diferentes del cálculo de distancias a modelos y de la red de reconocimiento. En total son $122.792\ bytes$.

El coste por modelo de estado es de $196\ bytes\ /\ estado$, con un coste fijo de $384\ bytes$. Así, si hay N modelos se tendría un total de $384 + N * 196\ bytes$.

El coste por nombre es aproximadamente de $3.000\ bytes\ /\ nombre$, por lo que si hay M palabras en el vocabulario se tendría un total de $M * 3.000\ bytes$.

La memoria total vendría dada por la siguiente expresión:

$$Mem\_Total = 123.176 + 196 * N + 3.000 * M\ bytes \qquad [5.1]$$

donde N representa el número de modelos de estado o senones y M el número de palabras diferentes en el vocabulario.

Tabla 5-XXII. Ejemplos de memoria consumida por el reconocedor de voz en función del número de modelos de estado y del número de nombres en el vocabulario.

Nº de modelos de estado	*Memoria por modelos*	*Nº de nombres*	*Memoria por nombres*	*Memoria fija*	*Memoria total*
M (82E)	**16.072 *B***	**10**	**30.000 *B***	**123.176 *B***	**169.248 *B***
M (82E)	**16.072 *B***	**100**	**300.000 *B***	**123.176 *B***	**439.248 *B***
M+BIV (235E)	**46.060 *B***	**100**	**300.000 *B***	**123.176 *B***	**469.236 *B***

Como se puede apreciar según la fórmula [5.1], el coste de introducir un nombre en el vocabulario es superior al de un modelo de estado. Esto se debe a que para cada nombre hay que ir almacenando para cada uno de sus estados los mejores tiempos acumulados y los mejores costes. Además, en los nodos intermedios dentro de un

nombre hay que ir guardando información sobre los mejores arcos que llegan a ese nodo para todas las tramas, con el fin de poder realizar la búsqueda hacia atrás.

Esto se comprueba mejor en la Tabla 5-XXII donde se observa que la introducción de un mayor número de estados al pasar de monofonemas a monofonemas junto con algunos bifonemas izquierdos de vocales no supone un gran incremento. La mayor parte se corresponde con el número de nombres en el vocabulario.

5.4.3 Variación del Procesado en Función del Vocabulario y los Modelos.

En el procesado se pueden distinguir tres partes: una es el cómputo de la distancia a los modelos de estado; otra es el coste de recorrer la red; y por último, está la parte fija asociada con la extracción del vector de características y otros procesados adicionales (cómputo de basura, preparación de la red, detección de extremos, etc). De los tres procesados los dos primeros son los principales; el primero depende de número de modelos fonéticos, mientras que el segundo del número de nombres en el vocabulario. Además, estos dos procesados principales ocupan la mayor parte del tiempo de la *CPU*, mientras que la extracción de características y el resto de procesado supone un tiempo muy reducido.

Si la parte de cómputo de distancias a los modelos de estado se realiza en la tarjeta de procesado *DSP*, el coste del procesado en el *PC* se reduce prácticamente a un coste variable sin fijo que depende sólo del tamaño del vocabulario.

El cómputo de la extracción de características y de otros procesados adicionales, como son las distancias a los modelos de estados y de recorrer la red, es de $1{,}86\ \%$.

El cómputo de la distancia a un modelo de estado es de $0{,}024\ \%$ de la *CPU*, hay N modelos de estado y, por tanto, se tiene un total de $N * 0{,}024\ \%$ de la *CPU* para la realización del cálculo de distancias a modelos.

El coste de recorrer la red de reconocimiento es de $0{,}0986\ \%$[4] de la *CPU* y hay M palabras en el vocabulario, por cual, se necesita un total de $M * 0{,}0986\ \%$ de la *CPU*.

El procesado necesario para el reconocimiento de voz entero en el *PC* se puede obtener, una vez conocido el número de modelos de estado y de palabras del vocabulario, como:

[4] Mientras que el coste de distancias a los modelos de estado varía de forma proporcional al aumentar su número, con los nombres no es así. Los resultados mencionados en este apartado y en el anterior se corresponden a nombres del estilo *"José Antonio Núñez"*, con nombres más cortos el vocabulario podría ser mayor, y con más largos menor.

$$Uso\_CPU = 1{,}86 + 0{,}024 * N + 0{,}0986 * M\ \% \qquad [5.2]$$

Si la parte de cálculo de distancias a modelos de estado se realiza en la tarjeta con *DSP*, entonces queda como:

$$Uso\_CPU = 0{,}0986 * M\ \% \qquad [5.3]$$

Tabla 5-XXIII. Ejemplos de memoria consumida por el reconocedor de voz en función del número de modelos de estado y de nombres en el vocabulario con todo el procesado realizado en el PC.

Nº de modelos de estado (N)	*Coste procesado distancias*	*Nº de nombres (M)*	*Coste procesado red*	*Coste procesado fijo*	*Coste procesado total*
M (82E)	**1,97 %**	**10**	**0,99 %**	**1,86 %**	**4,82 %**
M (82E)	**1,97 %**	**100**	**9,86 %**	**1,86 %**	**13,69 %**
M+BIV (235E)	**5,64 %**	**100**	**9,86 %**	**1,86 %**	**17,20 %**

Como se observa en la Tabla 5-XXIII, un reconocedor de voz para el equipo anteriormente mencionado tiene un coste fijo mínimo de un $1{,}86\ \%$ cuando todo el procesado se realiza en el *PC*.

Con este esquema de cálculo se puede llegar a la conclusión de que con un ordenador con esta configuración es posible hacer funcionar un reconocedor con un vocabulario de 975 nombres cuando todo el procesado se hace en el *PC*, o cuatro reconocedores con 215 nombres usando los 34 monofonemas.

En el caso de utilizar bifonemas, el número de nombres en el vocabulario puede llegar a ser de 940 y 175 nombres, con un reconocedor y con cuatro, respectivamente.

Si la parte de cálculo de las distancias a los modelos de estado se realiza en una tarjeta con *DSP*, con unas ligeras variaciones en función del número de modelos de estado (para un número elevado, el coste de lectura y almacenaje en la red de los mismos no sería tan despreciable), los vocabularios pueden llegar a ser de 1.000 y 250 nombres, respectivamente.

5.4.4 Influencia de la Poda sobre el Tamaño del Vocabulario.

Finalmente, se menciona la influencia que el factor de poda tiene sobre el cómputo total necesario para realizar el reconocimiento de voz, tal como se menciona en el apartado 5.3.5 sobre variación de la tasa de error en función del factor de poda.

Tabla 5-XXIV. Ejemplos de coste computacional del reconocedor de voz en función del factor de poda para unos modelos de estado de monofonemas y bifonemas izquierdos de vocales sobre los 100 nombres de la base de datos *NOM_100_TEL* cuando todo el procesado se realiza en el PC.

	35	*50*	*75*	*100*	*INF*
M + BIV (235 E)	**10,79 %**	**11,84 %**	**14,20 %**	**15,86 %**	**17,12 %**

Tabla 5-XXV. Factor de ganancia computacional (γ) en el cálculo de recorrer la red de reconocimiento en función del factor de poda.

	35	*50*	*75*	*100*	*INF*
M + BIV (235 E)	**0,358**	**0,465**	**0,704**	**0,872**	**1,000**

En la Tabla 5-XXIV se observa el coste computacional en porcentaje sobre el total disponible del reconocedor de voz utilizando monofonemas y bifonemas para la base de datos *NOM_100_TEL* en función del factor de poda. Por otro lado, en la Tabla 5-XXV se muestra el factor de ganancia del coste computacional de recorrer la red respecto a la no-utilización de la poda.

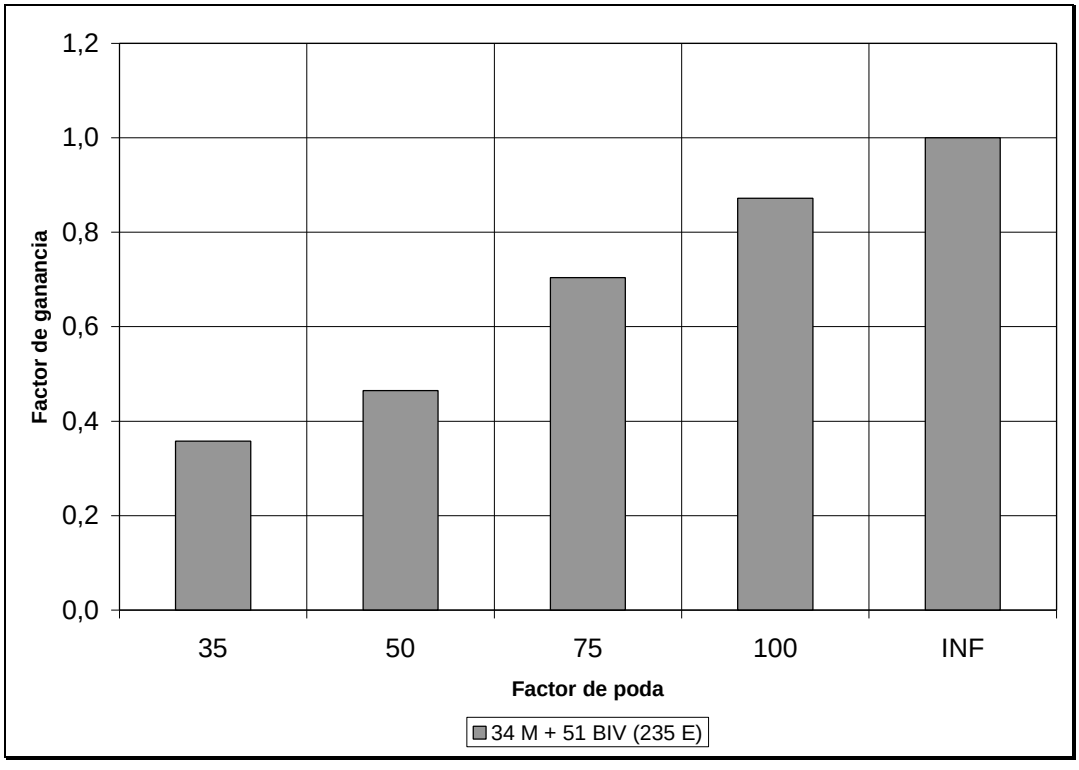

Figura 5-12. Gráfico con la variación del factor de ganancia (γ) en función del factor de poda para un vocabulario de 100 nombres con modelos de monofonemas y bifonemas izquierdos de vocales, sin detección de extremos, ni red con comodines.

En la Figura 5-12 aparece la variación del factor de ganancia del coste de recorrer la red de reconocimiento en función del factor de poda. Se observa que incluso con un factor de poda elevado como es 100, se obtiene una reducción en tiempo bastante significativa sin ningún incremento en la tasa de error. Disminuyendo aún más este factor de poda, la reducción obtenida en cómputo es cada vez menor, con un incremento en la tasa de error que hasta 50 no es muy elevada.

Con estos valores se puede reescribir la estimación del coste computacional del reconocedor de voz de la siguiente manera:

$$Uso\_CPU = 1{,}86 + 0{,}024 * N + 0{,}0986 * g * M \ \% \qquad [5.4]$$

donde N indica el número de modelos de estado, M el número de nombres del vocabulario y g el factor de ganancia de la poda.

Con los resultados así obtenidos, considerando que el reconocedor tenga distribuido el procesado entre una tarjeta con *DSP* y un *PC*, sólo se realice la parte de recorrer la red de reconocimiento en el *PC* y se utilice un factor de poda de 50, se puede manejar un vocabulario de hasta 2.200 nombres, con un ligero incremento en la tasa de error sobre el reconocedor sin poda.

Capítulo

6

Conclusiones y Desarrollos Futuros

En este último capítulo se exponen resumidamente las conclusiones obtenidas del diseño, desarrollo y puesta en marcha de un reconocedor fonético de voz robusto para aplicaciones telefónicas. Posteriormente, se enumera un conjunto de líneas de investigación que la presente tesis deja abiertas.

6.1 Conclusiones

Llegado este nivel de desarrollo, se considera que los objetivos planteados han sido alcanzados con creces.

El objetivo de diseñar una arquitectura de reconocimiento de voz altamente flexible para adaptarse a cualquier vocabulario se ha conseguido gracias al modelado de unidades fonéticas. Estas unidades, ya sean dependientes o independientes de contexto, proporcionan una elevada flexibilidad y permiten modelar cualquier palabra o frase del idioma castellano. Los aspectos relativos a la organización modular del

reconocedor, así como a la estructura eficiente de cálculo de la distancia a los modelos, aparecen reflejados en el capítulo 3 de la presente tesis.

Otro de los objetivos cumplidos es el del análisis e implementación de un conjunto de técnicas lo más sencillas posibles para mantener el grado de robustez del reconocedor de voz cuando las condiciones de reconocimiento difirieran de las diseñadas. En el capítulo 4 se muestra la descripción de cada uno de los objetivos particulares de mejora de la robustez.

El objetivo de mejorar la independencia de locutor del reconocedor se ha conseguido gracias a la introducción de saltos dobles en la red de reconocimiento, que posibilita la adaptación a diferentes velocidades de pronunciación por parte del usuario. Adicionalmente, permite el entrenamiento de modelos fonéticos utilizando la segmentación automática de las frases, por lo que es posible utilizar cualquier base de datos con la única condición de disponer de la transcripción fonética de la misma, pero sin necesidad de tener la segmentación temporal en fonemas.

Las técnicas de filtrado de los cepstrum, tanto el *CMN* como el filtrado *RASTA*, han demostrado ser muy eficaces y a la vez muy eficientes, con un procesado adicional muy bajo. La posibilidad de introducir un filtrado *CMN* con un bajo nivel de retardo, como es el *CMNR* (*CMN Recursivo*), facilita la utilización de cualquiera de las técnicas en tiempo real, incluso con una mejora sustancial respecto de la utilización directa del *CMN* gracias a la reducción de la influencia del silencio.

La utilización de un modelo de basura explícito como modo de medir la calidad de la palabra reconocida, siendo ésta independiente del vocabulario activo, posibilita la disminución de la tasa de error del reconocedor con un incremento muy reducido del número de palabras rechazadas. Además, la exigencia en la calidad de las palabras puede ser fácilmente modificada y permite ajustar la detección de palabras fuera de vocabulario de manera eficaz.

La utilización de transcripciones fonéticas alternativas para el reconocimiento de palabras breves permite incorporar dentro del reconocedor una mejora en la tasa de error, mediante el ajuste de los modelos de las palabras con mayor grado de confusión.

Gracias a una implementación modular del reconocedor de voz, ha sido posible integrar el mismo dentro de un sistema *ITO* funcionando en tiempo real, simulando llamadas a través de una centralita automática. Esto hace posible la realización de pruebas en condiciones reales, comprobando que aspectos tales como la detección de extremos o la poda de caminos afectan al funcionamiento del reconocedor de voz.

El desarrollo de la presente tesis ha dejado abiertas varias líneas de desarrollo para el futuro que se exponen en el siguiente apartado.

6.2 Desarrollos Futuros

El reconocimiento de voz es un campo de investigación que tiene todavía mucho por desarrollar. El presente trabajo representa una parte, la de reconocedores fonéticos para aplicaciones telefónicas, que deja abiertos una serie de desarrollos futuros.

Un desarrollo a realizar sería el de ***aumentar la resolución de los modelos fonéticos*** dando mayor importancia a la dependencia de los contextos, pero de manera controlada, para evitar que un número excesivo de modelos convierta al sistema en inviable. Un estudio minucioso y con medidas adecuadas puede indicar el conjunto de unidades fonéticas más interesante, ya que el presente trabajo se ha limitado a monofonemas y ciertos bifonemas izquierdos de las vocales con mayor aparición en la base de datos de entrenamiento. Este aumento de la resolución proporcionaría una mayor disminución de la tasa de error con un incremento moderado en el número de modelos.

La conversión del reconocedor de voz en un ***sistema multilingua***, capaz de reconocer con la misma estructura palabras entre un conjunto de idiomas, puede permitir agrupar varios reconocedores en uno único sin incrementar de forma lineal el número de modelos. Lo único que se requiere es tener transcriptores fonéticos para cada idioma y obtener un conjunto de modelos fonéticos válidos para todos los idiomas considerados. En [Bon97] se muestran algunos experimentos en este sentido, agrupando modelos entre diferentes idiomas mediante la utilización de la entropía cruzada para reducir su número. En [Pue99b] se indica una alternativa basada en el código *SAMPA* y la proyección de Sammon para identificar los modelos similares entre distintos idiomas. Esta técnica, utilizada ya en la presente tesis para buscar similitudes entre unidades fonéticas para el castellano, se ha mostrado de gran utilidad.

El creciente desarrollo de ***sistemas empotrados*** (*embedded systems*) puede resultar interesante en la integración del reconocedor de voz en los mismos, con todas las ventajas que aporta en el desarrollo de aplicaciones y en la utilización de reconocimiento de voz prácticamente en cualquier aparato, sea un teléfono, un televisor o incluso una lavadora.

Referencias Bibliográficas

Las referencias bibliográficas consultadas y mencionadas a lo largo de todo el texto de la presente tesis doctoral se detallan a continuación. Además, se incorporan todos los artículos publicados en conferencias por el autor.

Referencias

[Ace90a] *Acero, A. and Stern, R. M.;* **"Environmental Robustness in Automatic Speech Recognition"**, in Proc. ICASSP 90, vol. II, pp. 849-852, Alburquerque, April 1.990.

[Ace90b] *Acero, A.;* **"Acoustical and Environmental Robustness in Automatic Speech Recognition"** Carnegie Mellon University, Ph. D Thesis, 1.990.

[Ace91] *Acero, A. and Stern, R.;* **"Robust Speech Recognition by Normalization of the Acoustic Space"**; in Proc. ICASSP 91, vol. II, pp. 893-896, Toronto, May 1.991.

[All97] *Alleva, F.; Huang, X.; Hwang, M.-Y. and Jiang, L.;* **"Can Continuous Speech Recognizers Handle Isolated Speech"**, in EUROSPEECH 1997, vol. 2, pp. 911-914, Rhodes, September 1.997.

[Ave96] *Avendano, C.; Vuuren, S.; Hermansky, H.;* **"Data Based Filter Design for RASTA-like Channel Normalization in ASR"** in Proc. Int. Conf. on Spoken Language Processing, pp. 2087-2090, Philadelphia, PA, October 1.996.

[Bak75] *Baker, J. K.* **"Stochastic Modeling for Automatic Speech Understanding"** in D. R. Reddy, ed., Speech Recognition, New York, Academic Press, pp. 521-542, 1.975.

[Bah88] *Bahl, L. R.; Brown, P.F.; Souza, P. V.; Mercer, R. L. and Picheny, M. A.;* **"Acoustic Markov Models Used in the Tangora Speech Recognition System"** in Proc. ICASSP 88, vol. I, pp. 497-500, 1.988.

[Bea90] *Beale, R.; Jackson, T;* **"Neural Computing: an introduction",** Ed. Adam Hilger, 1.990.

[Bec98] *Becerra-Yoma, N.; McInnes, F. R.; Jack, M. A.;* **"Improving Performance of Spectral Substraction in Speech Recognition Using a Model for Additive Noise"**; in IEEE Trans. on Speech and Audio Processing, vol. 6, Nº 6, pp. 579-582, November 1.998.

[Boc93] *Bocchieri, E.;* **"Vector Quantization for Efficient Computation of Continuous Density Likelihoods"** in Proc. ICASSP 93, vol. II, pp. 692-695, Minneapolis, 1.993.

[Bon97] *Bonaventura, P.; Gallocchio, F. and Micca, G.;* **"Multilingual Speech Recognition for Flexible Vocabularies"**, in EUROSPEECH 97, vol. 1, pp. 355-358, Rhodes, September 1.997.

[Bou96] *Bourlard, H.; Hermansky, H.; Morgan, N.;* **"Toward Increasing Speech Recognition Error Rates"**, in Speech Communication, vol. 18, pp. 205-231, 1.996.

[Cox88] *Cox, S. J.;* **"Hidden Markov Models for Automatic Speech Recognition: Theory and Application"**, in Br Telecom Technol J, vol. 6, Nº 2, pp. 105-115, April 1.988.

[Dav80] *Dav, S.; Mermelstein, P.;* **"Comparison of Parametric Representations for Monosyllabic Word Recognition"**; IEEE Trans. on Acoustics, Speech and Signal Processing, vol. ASSP-28, pp. 357-366, 1.980.

[Dav96] *Davis, A. W.;* **"Speech Recognition Technology Backgrounder"**, in TechOnLine Review, www.techchonline.com , vol. 1, Nº 2, November 1.996.

[Del93] *Deller Jr, J.R.; Proakis, J.G., Hausen J.H.L.;* **"Discrete-Time Processing of Speech Signals"**. Ed. MacMillan. 1.993.

[Dem96] *Demuynck, K.; Duchateau, J. and VanCompernolle, D.;* **"Reduced Semi-Continuous Models for Large Vocabulary Continuous Speech Recognition in Dutch"**, in Proc. ICSLP 96, vol. 4, SuA2P1, October 1.996.

[Dig96] *Digalakis, V. V.; Monaco, P.; Murveit, H.;* **"Genones: Generalized Mixture Tying in Continuous Hidden Markov Model-Based Speech Recognizers"**; in IEEE Trans. on Speech and Audio Processing, vol. 4, Nº 4, pp. 281-289, July 1.996.

[Fur86] *Furui, S;* **"Speaker-Independent Isolated Word Recognition Using Dynamic Features of Speech Spectrum"**, in IEEE Trans. on Acoustics, Speech and Signal Processing, vol. ASSP-34, Nº 1, pp. 52-59, February 1.986.

[Gal99] *Gales, M. J. F.; Knil, K. M.; Young S. J.;* **"State-Based Gaussian Selection in Large Vocabulary Continuous Speech Recognition Using HMM's"**; in IEEE Trans. on Speech and Audio Processing, vol. 7, Nº 2, pp. 152-161, March 1.999.

[Gar99] *García-Gómez, R.; López-Barquilla, R.; Puertas-Tera, J.I. and others;* **"Speech Training for Deaf and Hearing-Impaired People"**, in EUROSPEECH 1999, vol. III, pp. 1067-1070, Budapest, September 1.999.

[Her92] *Hermansky, H.; Morgan, N.; Bayya, A.; Kohn, P.;* **"RASTA-PLP Speech Analysis Technique"**; in Proc. ICASSP 92, vol. I, pp. 121-124, San Francisco, March 1.992.

[Her93] *Hermansky, H.; Morgan, N. and Hirsch, H.-G.;* **"Recognition of Speech in Additive and Convolutional Noise Based on RASTA Spectral Processing"**, in Proc. ICASSP 93, vol. II, pp. 83-86, Minneapolis, April 1.993.

[Her94] *Hermansky, H. and Morgan, N.;* **"RASTA Processing of Speech"**, in IEEE Trans. on Speech and Audio Processing, vol. 2, Nº 4, pp. 578-589, October 1.994.

[Hua90] *Huang, X.; Lee, K.-F. and Hon, H.-W.;* **"On Semi-Continuous Hidden Markov Modeling"**, in Proc. ICASSP 90, vol. II, pp 689-692, Alburquerque, April 1.990.

[Hua93] *Huang, X.; Alleva, F.; Hon, H.; Hwang, M.; Lee, K.; Rosenfeld, R.;* **"The SPHINX-II Speech Recognition System: An Overview"**. Computer Speech and Language, in Press, 1.993.

[Hua95] *Huang, X.; Acero, A.; Alleva, F.; Hwang, M.-Y.; Jiang, L. and Mahajan, M.;* **"Microsoft Windows Highly Intelligent Speech Recognizer: WHISPER"**, in Proc. ICASSP 95, Detroit, 1.995.

[Hwa92] *Hwang, M.-Y.; Huang, X.;* **"Subphonetic Modeling with Markov States – Senone"**, in Proc. ICASSP 92, vol. I, pp. 33-36, 1.992.

[Hwa93a] *Hwang, M.-Y.;* **"Subphonetic Acoustic Modeling for Speaker-Independent Continuous Speech Recognition"**; Ph. D. Thesis, CMU-CS-93-230, Carnegie Mellon University, 1.993.

[Hwa93b] *Hwang, M.-Y.; Huang, X.; Alleva, F.;* **"Predicting Unseen Triphones with Senones"**, in Proc. ICASSP 93, vol. II, pp. 311-314, 1.993.

[Jel76] *Jelinek, F.;* **"Continuous Speech Recognition by Statistical Methods"**, in Proc. of the IEEE, vol. 64, pp. 532-556, April 1.976.

[Jua90] *Juang, B. H.; Rabiner, L. R.;* **"The Segmental k-Means Algorithm for Estimating Parameters of Hidden Markov Models"**, IEEE Trans. Acoustic, Speech and Signal Proc., 38(9), pp. 1639-1641, September 1.990.

[Lam81] *Lamel, L. F.; Rabiner,L. R.; Rosenberg, A. E. and Wilpon, J. G.;* **"An Improved Endpoint Detector for Isolated Word Recognition"**, in IEEE Trans. on Acoustics, Speech and Signal Processing, vol. ASSp-29, nº 4, August 1.981.

[Lee88] *Lee, C.-H. and Rabiner, L.R.;* **"A Network-Based Frame-Synchronous Level Buildfing Algorithm for connected Word Recognition"**, in Proc. ICASSP 88, vol. II, pp. 410-413, 1.988.

[Lee89] *Lee, C.-H.; Rabiner, L.-R.;* **"A Frame-Synchronous Network Search Algorithm for Connected Word Recognition"**, IEEE Trans. on Acoustics, Speech and Signal Processing, vol. 37, nº11, pp. 1649-1658, November 1.989.

[Lip96] *Lippmann, R., P.;* **"Recognition by Humans and Machines: Miles to Go before We Sleep"**; in Speech Communication, vol. 18, pp. 247-248, 1.996.

[Liu93] *Liu, F.-H.; Stern, R.; Huang, X.; Acero, A.;* **"Efficient Cepstral Normalization for Robust Speech Recognition"**, in Proc. of ARPA Speech and Natural Language Workshop, pp. 69-74, Pricenton, March 1.993.

[Liu94] *Liu, F.-H.; Moreno, P. J.; Stern, R. M.; Acero, A.;* **"Signal Processing for Robust Speech Recognition"**, in Proc. of the Spoken Language Processing, Yokohama, Japan, September 1.994.

[Lop98] *López-Barquilla, R.; Puertas-Tera, J. I.; Parera, J.; García-Gómez, R.;* **"A Phonetic Aided HMM Recognizer for Multilingual Applications through Telephone Channels",** SIP'98 (Signal & Image Processing), IASTED International Conference, pp. 531-534, Las Vegas (Nevada), October 1.998.

[Lop99a] *López-Barquilla, R.; Puertas-Tera, J. I.; García-Gómez, R.;* **"Visual Tests of Subphonetic Models in a Hidden Markov Model Speech Recognizer"**, Talking to Computers II, pp. 4.i-4-v, University of Sheffield, July 1.999.

[Lop99b] *López-Barquilla, R.; Puertas-Tera, J. I.; García-Gómez, R.;* **"Reconocedor de Voz Robusto y Eficiente",** URSI'99, XIV Simposium Nacional de la Unión Científica Internacional de Radio, pp. 393-394, Universidad de Santiago de Compostela, Septiembre 1.999.

[Lop00] *López-Barquilla, R;* "**Reconocimiento de Voz Flexible y Eficiente para Aplicaciones ITO"**, Tesis doctoral, UPM, 2.000.

[Mor95] *Moreno, P. J.; Raj, B.; Gouvêa,E.; Stern, R. M.;* **"Multivariate-Gaussian-Based Cepstral Normalization for Robust Speech Recognition (RATZ)"**; in Proc. ICASSP 95, pp. 137-140, Detroit, Michigan, 1.995.

[Mor96] *Moreno, P. J.;* **"Speech Recognition in Noisy Environments"**; Ph. D. Thesis, ECE Department, CMU, May 1.996.

[Mye81] *Myers, C.S.; Rabiner, L. R.;* **"A Level Building Dynamic Time Warping Algorithm for Connected Word Recognition"**, IEEE Trans. on Acoustics, Speech and Signal Proc, ASSP-29(2), pp. 284-297, April 1.981.

[Pon97] *Ponting, K. M.;* **"Channel Adaptation"**, St. Andrew's Road, Great Malvern, Worcs, UK, June 1.997.

[Pro92] **"Proyecto Albayzín, Base de Datos en Español"**, Documento de definición de la base de datos. 1.992.

[Pue98] *Puertas-Tera, J. I.; López-Barquilla, R.; García-Gómez, R.;* **"Reconocimiento Robusto de Vocabularios",** URSI'98. XIII Simposium Nacional de la Unión Científica Internacional de Radio, pp. 393-394, Universidad Pública de Navarra de Pamplona, Septiembre 1.998.

[Pue99a] *Puertas-Tera, J. I.; López Barquilla, R.; García-Gómez, R.;* **"Application of Visual Representation of The Speech to the Hearing-Impaired People"**, Talking to Computers II, University of Sheffield, pp. 9.i-9-v, July 1.999.

[Pue99b] *Puertas-Tera, J. I.; López-Barquilla, R.; García-Gómez, R.;* **"Multilingual Training for Speech Recognition"**, in Proc. ICSPAT 99, Speech Recognition Session, Orlando (Florida), 1-4 November, 1.999.

[Rab86] *Rabiner, L. R.; Huang, B.-H.;* **"An Introduction to Hidden Markov Models"**, in IEEE ASSP Magazine, pp. 4-16, January 1.986.

[Rab93] *Rabiner, L. W.; Juang, B.-H.;* **"Fundamentals of Speech Recognition"**; Prentice Hall Signal Processing Series, Alan V. Oppenheim, Series Editor, 1.993.

[Rah94] *Rahim, M. G. and Juang, B.-H.;* **"Signal Bias Removal for Robust Telephone Based Speech Recognition in Adverse Environments"**, in Proc. ICASSP 94, vol. I, pp. 445-448, Adelaide, April 1.994.

[Rah96] *Rahin, M. G.; Juang, B.-H.;* **"Signal Bias Removal by Maximum Likelihood Estimation for Robust Telephone Speech Recognition"**; in IEEE Trans. on Speech and Audio Processing, vol. 4, Nº 1, pp. 19-30, January 1.996.

[Raj97] *Raj, B.; Parikh, V. N. and Stern, R. M.;* **"The Effects of Background Music on Speech Recognition Accuracy"**, in Proc. IEEE ICASSP 97, vol. II, pp. 851-854, Munich, 1.997.

[Rec92] *Recuero, M.;* **"Acústica Arquitectónica. Soluciones Prácticas"**; Ed. Paraninfo, Madrid, 1.992.

[Riv96] *Rivlin, Z.; Cohen, M.; Abrasg, V. and Chung, T.;* **"A Phone-Dependent Confidence Measure for Utterance Rejection"**, in Proc. ICASSP 96, pp. 515-517, Atlanta, 1.996.

[Sak79] *Sakoe, H.;* **"Two-level DP Matching – A Dynamic Programming- Based Pattern Matching Algorithm for Connected Word Recognition"**; IEEE Trans. on Acoustics, Speech and Signal Proc, ASSP-27(6); pp.588-595; December 1.979.

[Sam69] *Sammon, J. W., Jr;* **"A Nonlinear Mapping for Data Structure Analysis"**, on IEEE Trans. On Computers, C-18(5): 401-409, May 1.969.

[Ste96] *Stern, R.M.; Acero, A.; Liu, F.-H-; Ohshima, Y.;* **"Signal Processing for Robust Speech Recognition"**, invited chapter in "Speech Recognition", pp 351-378, Eds. Boston, Kluwer Academic Publishers, 1.996.

[Vas96] *Vaseghi, S. V.;* **"Advanced Signal Processing and Digital Noise Reduction"**, Ed. Wiley Teubner, 1.996.

[Ver96] *Vergin, R.; O'Shaughnessy, D.; Gupta, V.;* **"Compensate Mel Frequency Cepstrum Coefficients"**, in Proc. ICASSP 96, vol. II, pp. 323-326, Atlanta, May 1.996.

[Vet97] *Veth, J.; Boves, L.;* **"Phase-Corrected RASTA for Automatic Speech Recognition over the Phone"**, in Proc. ICASSP 97, Vol. II, pp. 1239-1242, Munich, April 1.997.

[Vii98] *Viikki, O.; Bye, D.; Laurila, K.;* **"A Recursive Feature Vector Normalization Approach for Robust Speech Recognition in Noise"**, in Proc. ICASSP 98, vol. II, pp. 733-736, Seattle, WA, May 1.998.

[Vin68] *Vinstyuk, T. K.;* **"Speech Discrimination by Dynamic Programming"**; Kibernetika (Cybernetics), 4(2), pp. 81-88, January-February, 1.968.

[Vin71] *Vinstyuk, T. K.;* **"Element-Wise Recognition of Continuous Speech Consisting of Word from a Specified Vocabulary"**; Kibernetika (Cybernetics), Nº 2, pp. 133-134, March-April, 1.971.

[Whi81] *White, G. M.;* **"Dynamic Programming, the Viterbi Algorithm and Low Cost Speech Recognition"**; On Acoustic, Speech and Signal Processing, Vol. ASSP-29, nº 4, August 1.981.

[Wil84] *Wilpon, J. G.; Rabiner, L. R.; Martin, T. B.;* **"An Improved Word-Detection Algorithm for Telephone-Quality Speech Incorparting Both Syntactic and Semantic Constraints"**, AT&T Tech, J, 63(3), pp. 479-498, March 1.984.