

## GENERACION DE MULTIPLES HIPOTESIS UTILIZANDO INFORMACION LÉXICA

*Montse Pardàs, Antonio Bonafonte, José B. Mariño*

Dpt. de Teoria del Senyal i Comunicacions  
Universitat Politècnica de Catalunya

### Introducción.

El reconocimiento automático del habla presenta numerosos problemas debido a la gran variabilidad inherente a la señal de voz. Dependiendo del contexto, estilo del discurso, dialecto o hablante tendremos diferencias muy grandes en la pronunciación de las unidades utilizadas para el reconocimiento. Por otra parte, el habla es también estructurada y muy redundante. Por ello, para el reconocimiento de habla continua o conectada, con grandes vocabularios, resultará imprescindible utilizar la información presente en los diferentes niveles lingüísticos de la señal.

En general, las fuentes de conocimiento que podemos emplear son: conocimiento acústico-fonético, léxico, sintáctico y semántico.

En nuestro sistema hemos utilizado:

-Conocimiento acústico-fonético: Es básico disponer de una buena modelización de las unidades de reconocimiento. Las unidades utilizadas en nuestro caso son las semisílabas y se modela cada una de ellas mediante modelos ocultos de Markov [1].

-Conocimiento léxico: Mediante una gramática de estados finitos representamos cada una de las palabras como una cadena de unidades de nivel inferior (semisílabas). Es decir, una palabra se produce como consecuencia de las producciones sucesivas que tienen lugar al recorrer un camino de estados entre el estado inicial y el final.

-Conocimiento sintáctico: Representamos a su vez el lenguaje por una gramática de estados finitos en la que los estados serán las diferentes palabras que forman el lenguaje.

Las dos últimas fuentes de conocimiento se integrarán en una sola gramática que será la utilizada para el reconocimiento. Los estados de esta gramática son las semisílabas, pero incorporarán la información de la palabra a la que pertenecen.

Esta integración se presenta como un compromiso entre los dos tipos de gramáticas habituales:

-Gramáticas expandidas: Una misma unidad fonética deberá ser representada por diferentes estados del autómata cuando pertenezca a diferentes palabras, aunque una ley de minimización los compactara en un solo estado. Es decir, diferentes palabras no pueden compartir un mismo estado. Esto provocará que el espacio de búsqueda sea mucho más grande que sin utilizar un modelo del lenguaje.

-Gramáticas compactadas: Mediante un algoritmo de minimización se reduce el número de unidades sintácticas (estados de la gramática) [2], permitiendo por tanto la compartición de estados entre diferentes cadenas. En este caso, como contrapartida, perderemos la información léxica, ya que la relación entre estados y palabras no es unívoca.

En este nuevo tipo de gramáticas minimizaremos el número de unidades dentro de una misma palabra de manera que al añadir nuevas transcripciones debidas a la coarticulación entre palabras y dentro de las palabras, no habrá un aumento importante del número de unidades. Sin embargo, conservaremos la información léxica, incorporándola en las semisílabas.

Por otra parte, será posible reducir aún más el espacio de búsqueda, compactando algunas unidades por el final de palabra y conservando la información léxica sólo en las semisílabas iniciales de las palabras.

Estas gramáticas nos permitirán aplicar un algoritmo de generación de múltiples hipótesis que generará las  $n$  mejores cadenas de palabras, a las que posteriormente podríamos aplicar otras fuentes de conocimiento de nivel superior para elegir la hipótesis más probable.

Un ejemplo sencillo de aplicación de las múltiples hipótesis es el reconocimiento de una cadena de dígitos, el último de los cuales ("check-sum"), sirva de comprobación de los anteriores (por ejemplo, transmisión telefónica del N.I.F.). Para elegir la cadena correcta se examinarán las hipótesis, empezando por la primera, hasta encontrar una que cumpla la condición impuesta por el último dígito.

### Generación de una gramática de semisílabas con información léxica.

Utilizaremos gramáticas (G) regulares para describir nuestro lenguaje (L(G)). Estas gramáticas no tienen la suficiente riqueza de estructuras para generar un lenguaje natural (en particular no permiten recursividad), pero nos permiten generar subconjuntos de éste no triviales, y apropiados para tópicos específicos. Estas gramáticas las podemos representar por diagramas de transición de estados y es fácil integrarlas en un algoritmo de reconocimiento.

Para definir formalmente el autómata de estados finitos(G), a partir del cual podremos obtener todas las frases posibles de L(G), utilizaremos los siguientes conceptos:

- Unidad fonética: unidad acústica básica.
- Copia: cada estado o unidad sintáctica asociado a la misma unidad fonética. Serán necesarias para considerar los diferentes contextos en que puede aparecer una unidad.
- Unidad sintáctica: Cada estado del autómata. Se incluyen además de las copias, las unidades inicio y fin, que indican los límites de las cadenas.
- Predecesores: Un estado es predecesor de otro si éste último puede ser alcanzado desde el primero en una sola transición.

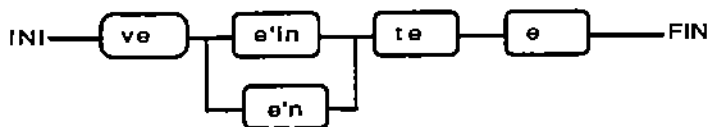
Así pues, asociaremos las observaciones a los estados, mientras que usualmente éstas son asociados a los arcos (transiciones). Pero es inmediato probar la equivalencia de estas dos representaciones. La primera nos resulta más conveniente al utilizar el algoritmo de programación dinámica en un paso [3] para el reconocimiento.

El primer paso en la creación de una gramática será generar el autómata que describa nuestro lenguaje en función de las palabras que lo forman. Usamos para ello una herramienta desarrollada en el departamento de T.S.C de la U.P.C. que genera la gramática de estados finitos a partir de la simple enumeración de las frases que componen el lenguaje. También pueden generarse gramáticas parciales que mediante operaciones de encadenado (producto cartesiano de dos gramáticas) y suma (unión de gramáticas) permitan crear la gramática global. La gramática generada por este algoritmo tendrá el mínimo número de estados posibles (aplicamos compactación de las unidades)[2].

Cada una de las palabras de nuestro vocabulario deberá ser descompuesta en unidades de reconocimiento, es decir, semisílabas. En primer lugar deberá efectuarse una transcripción fonética del texto, seguido de silabificación y semisilabificación de éste. Se aplicarán también transcripciones alternativas, tanto a nivel fonético como de grupos de semisílabas, para adecuar la descripción final de las palabras a diferentes realizaciones acústicas de éstas.

Una vez descompuestas las palabras en semisílabas, debemos crear la red de estados finitos de cada una de ellas. También aquí minimizaremos el número de estados.

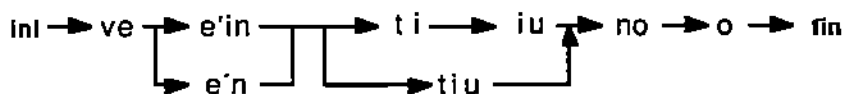
Así por ejemplo, para representar la palabra "veinte", la red sería:



La gramática que describe el lenguaje y las que describen las palabras deben integrarse para generar el autómata a utilizar por el algoritmo de reconocimiento. El problema más importante será la coarticulación entre palabras, que dará lugar a enlaces no triviales entre ellas.

Al pronunciar las palabras concatenadas cambiarán en muchos casos la unidad final de la primera palabra y la inicial de la segunda, siendo posibles también varios enlaces diferentes.

Así por ejemplo, si consideramos las palabras veinte y uno al concatenarlas para formar veintiuno obtenemos:



Para la gramática de cada palabra tendremos pues que considerar varias unidades iniciales y finales, cada una de las cuales se utilizará en diferentes contextos. Para ello deberemos cambiar la estructura de las redes de estados de las gramáticas de semislabas, generalizando el concepto de inicio y fin. A cada semisilaba que nos aparezca como final de palabra debido a las nuevas transcripciones, le asignaremos una copia diferente de la unidad "fin" y análogamente con los inicios.

Así pues, el paso previo al enlace de estas gramáticas será realizar las transcripciones alternativas correspondientes entre cada palabra y sus posibles sucesores, añadiendo a cada una de ellas las unidades que sean necesarias. Además modificaremos la gramática de palabras del lenguaje de manera que el arco que une dos palabras lleve información sobre la unión de éstas.

Una vez realizado este proceso para modelar la coarticulación, procederemos al enlace de las gramáticas de semislabas.

Las gramáticas utilizadas hasta ahora quedaban completamente definidas a través de sus estados. La información de cada uno de ellos comprende:

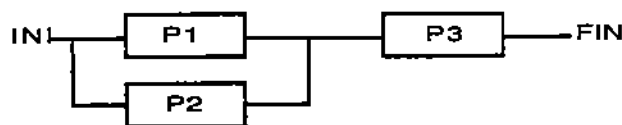
- La unidad fonética del estado y su número de copia correspondiente.
- La lista de sucesores del estado.
- La lista de predecesores (aunque esta información es redundante).

En la gramática global asignaremos a cada unidad dos parámetros más:

- Un indicador de unidad inicial y del número de copia del inicio que la precede.
- Un indicador análogo para las unidades finales.

Se tratará pues de insertar la gramática de cada una de las palabras en una gramática nueva. Basándonos en la información contenida en la gramática de palabras (información sobre qué palabras pueden ser sucesoras de otras y mediante qué semislabas se producen estos enlaces), generaremos la gramática global. Ésta tendrá un solo inicio y un solo fin (que representan el lugar donde empieza y acaba la frase a reconocer), y la información sobre los diferentes inicios y finales de las palabras estará contenida en estos nuevos campos. Procederemos entonces a crear los nuevos enlaces.

Como ejemplo, supongamos una gramática formada por tres palabras, P1, P2 y P3:

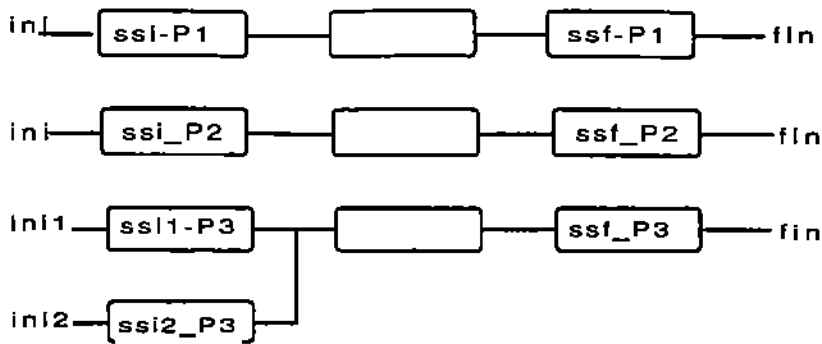


Con los enlaces:

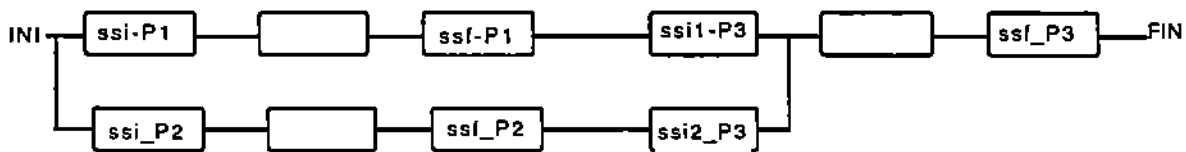
ssf1\_P1-----> ssi1\_P3

ssf1\_P2-----> ssi2\_P3

Y las gramáticas de semislabas:



Entonces la gramática resultante será:



Como hemos dicho, realizaremos una minimización parcial de las unidades resultantes. Las palabras podrán compartir unidades siempre que éstas no sean iniciales. Reduciremos aquellas unidades que aún perteneciendo a palabras diferentes, compartan el mismo conjunto de sucesores. Es decir, palabras de la misma categoría sintáctica y que tengan la misma terminación, compartirán estados de la gramática.

En la aplicación realizada, consistente en la gramática de los números del 0 al 1000 autoencadenados obtuvimos, para 77 unidades fonéticas:

- 161 estados diferentes y un número medio de predecesores por estado de 4.98.
- 239 estados y 8.27 predecesores sin aplicar esta última minimización.
- 135 estados y 4.29 predecesores aplicando una minimización total (perdiendo la información léxica).

Comprobamos pues que el aumento en el espacio de búsqueda con estas gramáticas semi-compactadas no es exagerado.

### Generación de múltiples hipótesis de cadenas léxicas.

Dados los modelos acústicos, el modelo del lenguaje y los datos acústicos observados, nuestro sistema debe proporcionar las  $n$  cadenas de palabras más probables.

Si hubiéramos compactado las gramática de semisílabas, perdiendo así la información léxica, obtendríamos en muchos casos, debido a las transcripciones mencionadas, hipótesis que aún siendo diferentes secuencias de estados del autómata (diferentes cadenas de semisílabas), representarían la misma cadena de palabras. Incorporando el conocimiento léxico, el reconocimiento puede gestionarse de forma que las diferentes hipótesis pertenezcan siempre a diferentes cadenas léxicas.

El algoritmo que presentamos es una adaptación, que utiliza la información léxica, del propuesto en [4], basado en la programación dinámica de un solo paso.

Representamos la señal de test por  $i=1, \dots, N$  tramas temporales (secciones de muestras de la señal), cada una caracterizada por un vector de características  $y(i), i=1, \dots, N$ . Cada unidad sintáctica la representamos por el Modelo Oculto de Markov [1] de la unidad fonética correspondiente, convenientemente entrenado. Cada modelo estará representado por una serie de estados, conectados mediante transiciones y desde cada uno de los cuales pueden efectuarse

observaciones de los diferentes vectores de características. Consideraremos  $K$  modelos, y cada modelo  $k$ ,  $k=1, \dots, K$  formado por  $j=1, \dots, J(k)$  estados.

La gramática de estados finitos impone que cada unidad sintáctica pueda tener como predecesores sólo a un subconjunto de unidades  $P_k$ . Llamamos  $P_0$  a las unidades permitidas como inicio de cadena y  $P_\infty$  a las que puedan finalizar una cadena.

Podemos considerar un "macromodelo" que englobaría todos los modelos, conectados según las restricciones sintácticas. Se trata entonces de encontrar las secuencias de estados a través de este macromodelo con mayor probabilidad de haber producido la secuencia observada  $Y=y(1), \dots, y(N)$ . Para ello minimizaremos una función de coste acumulativa a través de los posibles caminos. Llamaremos  $c(i, k, j, k', j')$  al coste local asociado a la probabilidad de transición del estado  $j'$  del modelo  $k'$  en la trama  $(i-1)$  al estado  $j$ , perteneciente al modelo  $k$ , en la trama  $(i)$  y a la probabilidad de producir desde este estado la observación  $y(i)$ . El coste acumulado hasta el punto  $(i, k, j)$  por el camino óptimo lo designaremos por  $C(i, k, j)$ .

El camino óptimo se determinará de forma secuencial, empezando en la trama  $i=1$  y acabando en  $y=N$ . Distinguiremos dos situaciones diferentes:

-Puntos interiores a un modelo  $k$ . Consideraremos en este caso las transiciones desde estados pertenecientes al mismo modelo  $k$ . En los modelos utilizados sólo son posibles transiciones de izquierda a derecha o sobre un mismo estado.

-El punto corresponde al primer estado ( $j=1$ ) de un modelo  $k$ . Este punto puede ser alcanzado desde el último estado de los modelos predecesores de  $k$ , es decir, el conjunto de puntos  $w(i, k, j)$  en el espacio de búsqueda desde los cuales se puede llegar al punto  $(i, k, j)$ , será:

$$w(i, k, 1) = \{(i-1, k, 1), (i-1, k', J(k'))\}; k' \in P_k\}$$

Para obtener las  $n$  mejores secuencias de unidades sintácticas deberíamos seguir los  $n$  mejores caminos en el espacio  $(i, k, j)$ , es decir, de todas las posibles alternativas de alcanzar el punto  $(i, k, j)$  considerar las  $n$  mejores. Esto supondría un aumento enorme del coste computacional. Además este método nos proporcionaría las  $n$  mejores secuencias de estados y no de unidades sintácticas. Haremos pues una aproximación: sólo en las transiciones entre modelos consideraremos  $n$  alternativas, y el resto de las decisiones serán hechas según la hipótesis del mejor camino. Es decir, la mejor hipótesis guía los caminos dentro de un modelo y decide cuándo tiene lugar una transición entre modelos.

Por otra parte, ya que buscamos diferentes cadenas de palabras y no de semisílabas, cuando tenga lugar una transición entre modelos comprobaremos si los caminos que llegan al nuevo modelo están formados por diferentes cadenas de palabras. De cada subconjunto de caminos formados por las mismas cadenas conservaremos sólo el mejor de ellos, dejando prosperar otros caminos que aún con mayor distancia al óptimo, representen diferentes cadenas.

Utilizaremos ahora:

- $C(i, k, j, m)$ : coste acumulado en el punto  $(i, k, j)$  para la hipótesis  $m$ .

- $Par\_ini(k)$ : indicador de la palabra a la cual pertenece el modelo  $k$ .

- $Par\_fin(k)$ : indica si el modelo  $k$  corresponde a una unidad final de palabra.

- $Par(i, k, j, m, p)$ : identificador de la palabra  $p$ -ésima de la cadena que pasa por el estado  $j$  del modelo  $k$  en la trama  $i$  de la hipótesis  $m$ .

- $l(i, k, m)$ : controla la longitud en palabras de la cadena a la cual pertenece el modelo  $k$  en la hipótesis  $m$  y la trama  $i$ .

La formulación del algoritmo será la siguiente:

#### Paso 1: inicialización:

Desde  $m=1$  hasta  $n$

Si  $k \in P_0$

$$C(1, k, 1, m) = c(1, k, 1, 0, 0)$$

$$Par(1, k, j, m, 1) = par\_ini(k)$$

$$l(1, k, m) = 1$$

Si  $k \notin P_0$  o  $j \neq 1$

$$C(1,k,j,m)=\infty$$

$$\text{Par}(1,k,j,m,1)=\emptyset$$

$$l(1,k,m)=0$$

Fin m

Desde  $i=2$  hasta  $N$

Desde  $k=1$  hasta  $K$

### Paso 2: Transición entre modelos.

$$C(i,k,1,1)=\min_{w(i,k,1)} (c(i,k,1,k',j') + C(i',k',j',1))$$

-Si el punto en  $w(i,k,1)$  que minimiza  $C(i,k,1,1)$  no es  $(i-1,k,1)$  tiene lugar una transición entre modelos.

En este caso el algoritmo examina el conjunto de costes acumulados:

$$\{c(i,k,1,k',j')+C(i-1,k',J(k'),m), k' \in P_k, m=1,\dots,n'\}$$

( $n'=1$  para una transición entre la primera y la segunda unidades de la secuencia y  $n'=n$  en cualquier otro caso.)

-Asignamos:

$$\text{par}(i,k,1,m,r)=\text{par}(i-1,k',J(k'),m',r) \quad 1 \leq r \leq l(i-1,k',m')$$

$$l(i,k,m)=l(i-1,k',m')$$

-Si el modelo  $k$  representa una semisílaba inicial de palabra y  $k'$  una semisílaba final (transición entre palabras), entonces:

$$l(i,k,m)=l(i,k,m)+1$$

$$\text{Par}(i,k,1,m,l(i,k,m))=\text{Par\_ini}(k)$$

-Se comparan seguidamente las hipótesis para cualquier  $m$  y  $k'$ , y en caso de que para dos  $m$  diferentes ( $m_1$  y  $m_2$ ) tengamos:

$$l(i,k,m_1)=l(i,k,m_2)$$

$$\text{Par}(i,k,1,m_1,p)=\text{Par}(i,k,1,m_2,p) \quad p=1,\dots,l(i,k,m_1)$$

asignaremos:

$$C(i,k,1,m_i)=\infty$$

siendo  $m_i$  aquella entre  $m_1$  y  $m_2$  que tenga menor coste acumulado hasta el momento.

-Buscamos, entre las hipótesis restantes las  $n$  mejores.

-Llamando  $k(m)$  y  $C(i-1,k(m),J(k(m)),m')$  el modelo que proporciona el coste  $m$ -ésimo y el correspondiente coste respectivamente, el algoritmo fijará los costes acumulados después de la transición como:

$$C(i,k,1,m)=c(i,k,1,k(m),J(k(m))) + C(i-1,k(m),J(k(m)),m')$$

### Paso 3: Transición dentro de los modelos.

$$C(i,k,j,m)=c(i,k,j,k',j',m) + C(i-1,k',j',m)$$

$$\text{par}(i,k,j,m,l)=\text{par}(i-1,k,j',m,l) \quad 1 \leq l \leq l(i,k,m)$$

Donde el punto  $(i-1,k,j')$ , minimiza  $C(i,k,j,l)$ .

Fin k

Fin i

#### Paso 4: Optimización.

En la trama N de la plantilla de test, los n mínimos costes acumulados del conjunto:

$$\{C(N,k,J(k),m); k \in P_\infty, m=1,\dots,n\}$$

son determinados.

De esta manera, obtenemos el último modelo  $k'(m)$  correspondiente a la m-ésima mejor secuencia

#### Paso 5: Backtracking.

A partir de la información de palabras almacenada:

$$\text{Par}(N,k'(m),J(k'(m)),l) \quad \text{para } 1 \leq m \leq n$$

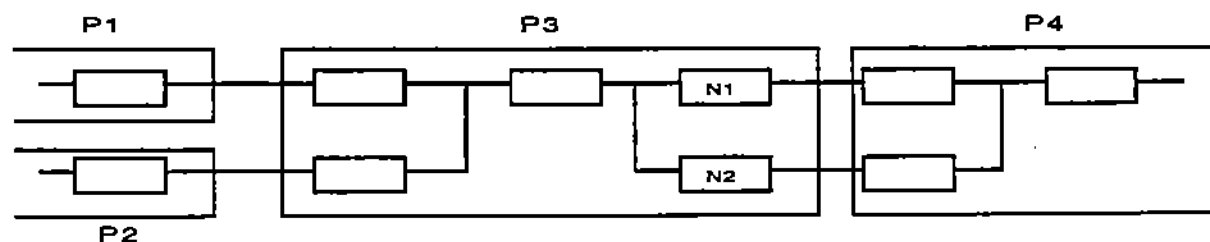
$$1 \leq l \leq l(N,k'(m),m)$$

obtendremos las n mejores cadenas de palabras.

Este algoritmo, aplicado a cadenas largas de palabras, debido al gran número de comparaciones que deben realizarse ( $\text{Par}(i,k,l,m_1,p) = \text{Par}(i,k,l,m_2,p)$ ,  $p=1,\dots,l$ ) entre los vectores de palabras, comporta un aumento de coste computacional considerable respecto al algoritmo de búsqueda de múltiples hipótesis de cadenas de semisílabas.

Consideramos pues la posibilidad de comparar para cada unidad, sólo la palabra precedente en la cadena.

Observemos en la figura un ejemplo de enlaces entre palabras:



Únicamente en el caso de que la entrada en la palabra P3 desde P1 o P2 afecte a la salida de esta palabra por la unidad N1 o N2, podemos perder alguna hipótesis al limitar las comparaciones a la última palabra. Parece lógico que el enlace de una palabra con sus predecesores no tenga influencia sobre el enlace con sus sucesores.

Así pues, modificamos las variables y el algoritmo del siguiente modo:

-Par(i,k,j,m): identificador de la palabra anterior a la actual, para la trama i en el estado j del modelo k y la hipótesis m.

-Par\_actual(i,k,j,m): palabra a la cual pertenece el estado j en esta situación. Con esta variable transmitiremos la información léxica desde las semisílabas iniciales hasta el final de palabra.

-Desaparece la variable  $l$  (no llevamos un control de la longitud de las cadenas.  
La modificación del algoritmo será:

Inicialización:

$$\begin{aligned} \text{Par\_actual}(l,k,j,m) &= \text{par\_ini}(k) && \text{si } k \in P_0 \\ \text{Par\_actual}(l,k,j,m) &= \emptyset && \text{si } k \notin P_0 \end{aligned}$$

$$\text{Par}(l,k,j,m) = \emptyset \quad \forall k$$

Paso 2:

Al examinar los costes acumulados:

$$\{(c(i,k,l,k',j') + C(i-1,k',J(k'),m)), k' \in P_k, m=1, \dots, n'\}$$

Si  $k$  representa una semisílaba inicial de palabra y  $k'$  una semisílaba final, entonces:

$$\begin{aligned} \text{Par\_actual}(i,k,l,m) &= \text{par\_ini}(k) \\ \text{Par}(i,k,l,m) &= \text{par\_actual}(i-1,k',J(k'),m) \end{aligned}$$

Sino:

$$\begin{aligned} \text{Par\_actual}(i,k,l,m) &= \text{par\_actual}(i-1,k',J(k'),m) \\ \text{Par}(i,k,l,m) &= \text{par}(i-1,k',J(k'),m) \end{aligned}$$

Comparamos entonces las hipótesis proporcionadas por cada  $k'$  con las de los otros modelos. En el caso de que dos hipótesis procedentes de diferentes modelos provengan de la misma palabra, eliminaremos la de mayor coste acumulado.

Paso 4: Backtracking.

En este caso, para recuperar las  $n$  cadenas de palabras será necesario almacenar durante el algoritmo la variable:

$$-i\_ant(i,k,J(k),m), \text{ trama donde empezó el camino en la palabra } \text{par}(i,k,J(k),m).$$

Pudimos comprobar, realizando las pruebas de reconocimiento que, efectivamente, esta simplificación no altera ninguno de los resultados, reduciendo considerablemente el tiempo de ejecución.

**Entorno del sistema.**

Las gramáticas generadas y el algoritmo de reconocimiento de múltiples hipótesis se han utilizado dentro del sistema de reconocimiento de habla continua R.A.M.S.E.S. [5].

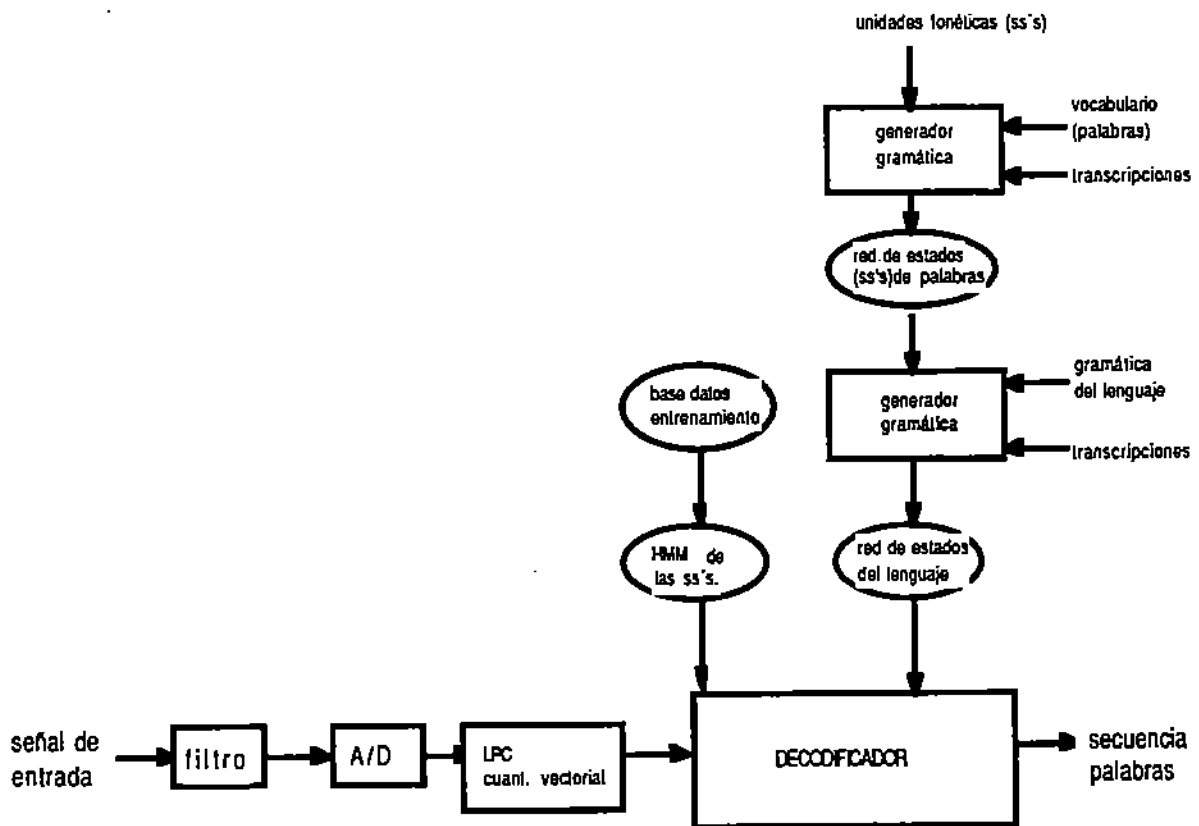
En este sistema la señal de voz es primeramente filtrada en una banda de 100 Hz.-3400 Hz. y muestreada a 8 KHz. La frase pronunciada es aislada mediante un detector de principio y fin. Después de un filtrado de pre-énfasis se le aplica una parametrización predictiva lineal: la señal es segmentada en tramas de 30 ms. a través de un enventanado Hamming, a una velocidad de 15 ms., y cada trama es caracterizada por un filtro L.P. con 8 coeficientes. Se calculan después 12 coeficientes cepstrum. La energía de la señal completa la parametrización. El sistema calcula también la diferencia espectral y de energía entre tramas (con un tiempo medio de 90 ms.). El vector espectral y las diferencias espectral y de energía son cuantificados

vectorialmente por separado. Así pues, cada trama de la señal de voz es representada por tres símbolos.

Tal como hemos dicho, la unidad básica de reconocimiento es la semisílaba y se modela mediante Modelos Ocultos de Markov (H.M.M.). La estructura de los H.M.M usada es de izquierda a derecha y permite eludir un estado en las transiciones. La emisión de símbolos es asociada a los estados, y en ellos se emiten los tres símbolos mencionados. El número de estados de cada semisílaba se determina en función de la longitud media de éstas.

Las muestras de cada semisílaba utilizadas para entrenar los H.M.M. son extraídas de la pronunciación de frases de nuestro lenguaje por algunos de los locutores de la base de datos que posteriormente utilizaremos en el reconocimiento.

El esquema global del sistema utilizado es:



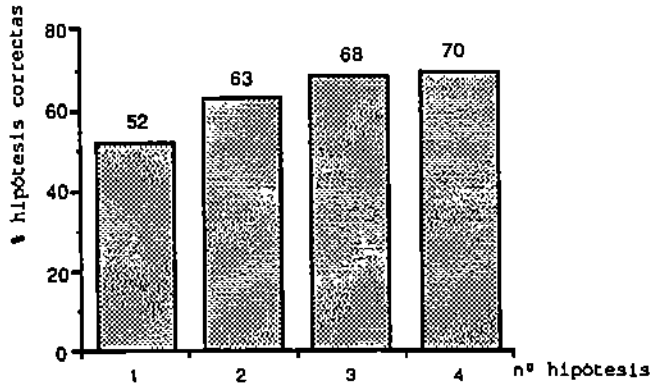
### Resultados y conclusiones.

Las pruebas se han realizado sobre cadenas de números enteros. Para formar la gramática de estas cadenas se han considerado 38 palabras diferentes. Cada cadena contiene aproximadamente entre 6 y 10 palabras. Así por ejemplo, la cadena 196/79/10 la consideramos formada por las palabras:

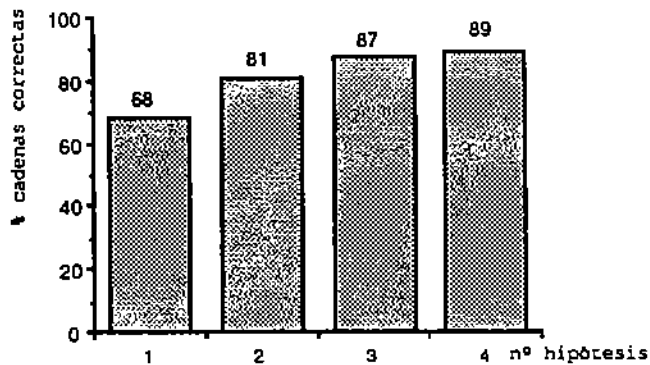
ciento/noventa/i/seis/setenta/i/nueve/diez

La base de datos utilizada está formada por 40 cadenas pronunciadas por 20 locutores diferentes. Las frases de las cuales se han extraído las semisílabas que entrenarán el sistema pertenecen a 10 de estos locutores, 5 hombres y 5 mujeres.

En la siguiente gráfica observamos el porcentaje de cadenas correctas obtenido al considerar de una a cuatro hipótesis por cadena:



Si consideramos únicamente los locutores utilizados para el entrenamiento de las semisílabas, los resultados son:



Analizando las cadenas erróneas reconocidas observamos que en la mayoría de los casos la cadena correcta difiere de éstas sólo en una de las palabras. Esto nos permite suponer que aumentando el número de hipótesis y aplicando posteriormente unos criterios de selección adecuados el porcentaje de error sería reducido considerablemente.

Por otra parte, la utilización de estas nuevas gramáticas, aún implicando un aumento del tiempo de cálculo (debido al aumento del espacio de búsqueda y al mayor número de parámetros que se utilizan en el reconocimiento), ha mejorado sensiblemente los resultados para un mismo número de hipótesis generadas y además facilitan enormemente un tratamiento posterior de las cadenas de palabras resultantes.

**Referencias.**

- [1] Lawrence R. Rabiner, "A tutorial on Hidden Markov Models and selected applications in Speech Recognition," Proceedings I.E.E.E., pag. 257-284. Febrero 1989.
- [2] José B. Mariño, Climent Nadeu, Eduardo LLeida, "Finite State Grammar Inference for Connected Word Recognition," Proceedings E.U.S.I.P.C.O. ,pag 1059-1062, 1988.
- [3] Hermann Ney, "The use of a One-Stage Dynamic Programming Algorithm for Connected Word Recognition," I.E.E.E. Trans. on Acoustics, Speech and Signal Processing, vol-32 N<sup>o</sup> 2, pag 263-271, Abril 1984.
- [4] José B. Mariño, Enric Monte, "Generation of Multiple Hypothesis in Connected Phonetic-Units Recognition by a modified One-Stage Dynamic Programming Algorithm," EUROSPEECH, pag 408-411, 1989
- [5] J.B.Mariño, C. Nadeu, A. Moreno, E. Lleida, E. Monte, A. Bonafonte, "RAMSES, a Spanish demisyllable based continuous speech recognition system," NATO-ASCI-90, Julio, 1990.

