Proyecto Final: Reconocedor de Habla

Pablo Peiretti - 103592

1er Cuatrimestre 2022

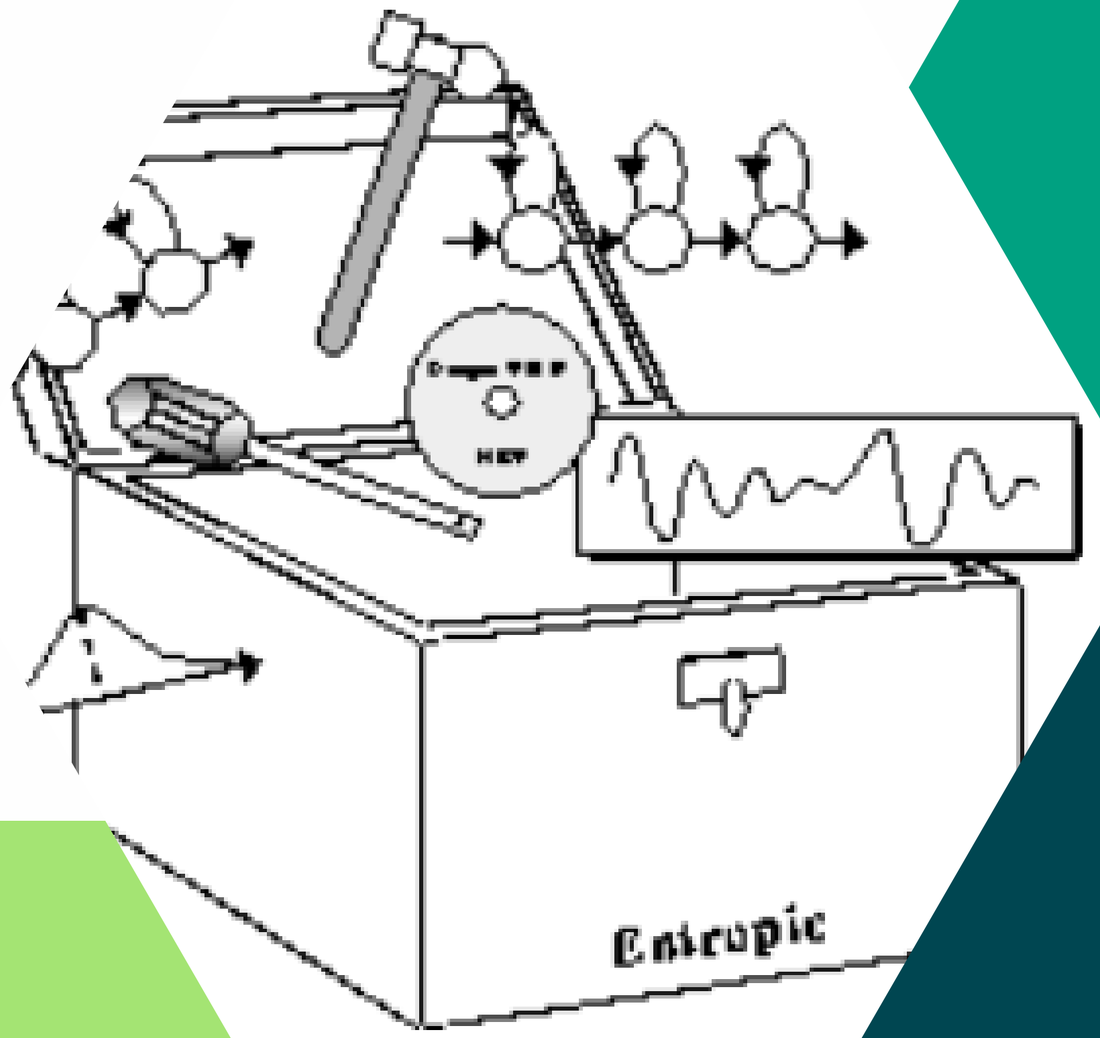
Agenda

- Introducción
- Análisis Acústico
- Modelo de Lenguaje
- Modelo Acústico
- Resultados

HTK

- Desarrollado por la Universidad de Cambridge.

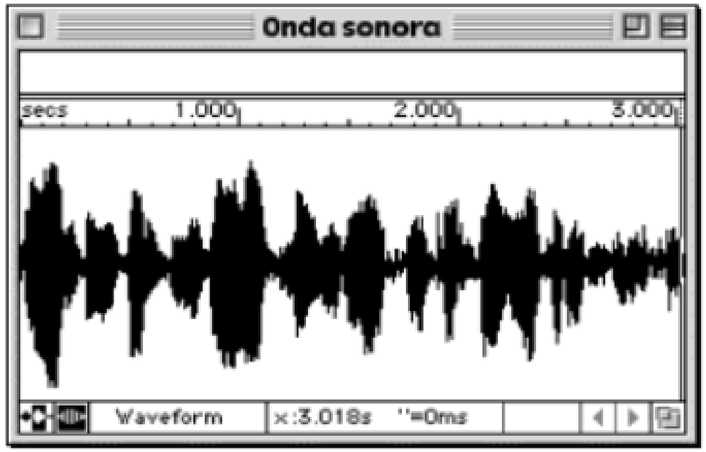
- Es un conjunto de herramientas que permite construir modelos ocultos de Markov (HMM) y estimar sus parámetros.



Reconocedor

- Dada una señal de audio, el reconocedor entrega como salida el conjunto de palabras que los modelos estadísticos encuentran más probable

Del habla …



… al texto

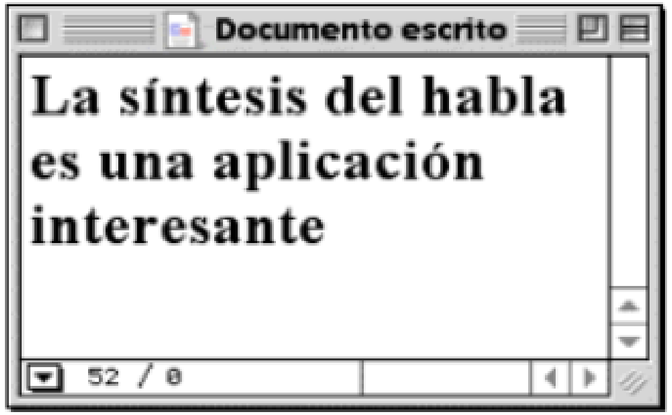
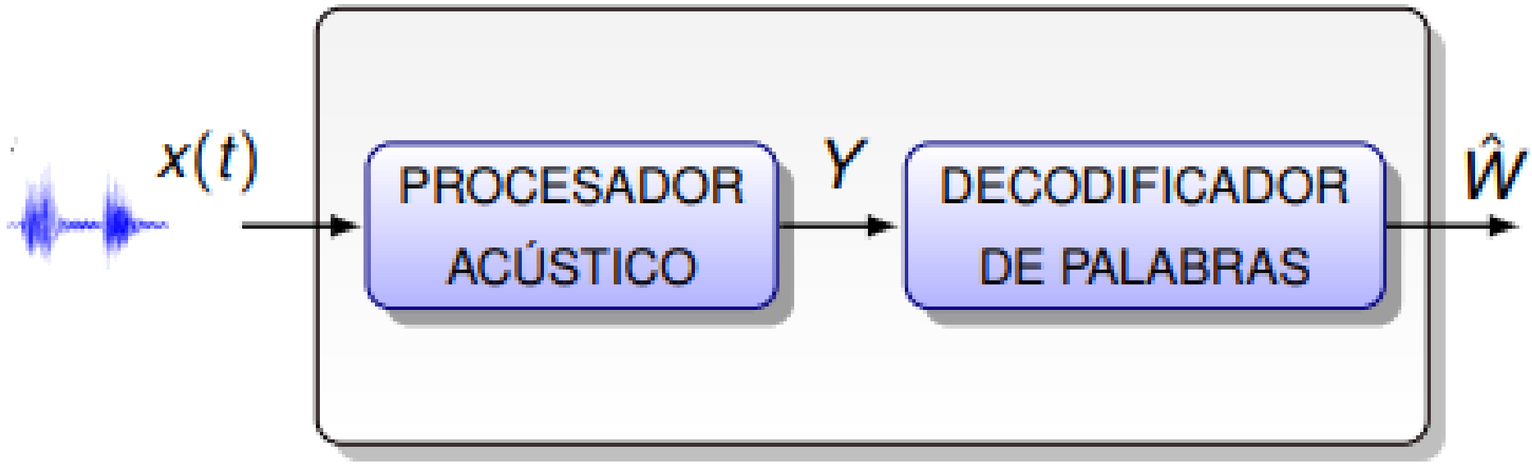


Diagrama del Reconocedor



El objetivo del **Decodificador de Palabras** es encontrar la secuencia de palabras del **vocabulario** más probable de acuerdo a la **evidencia acústica**, es decir,

$$\hat{W} = \arg\max_{W} P(W|Y)$$

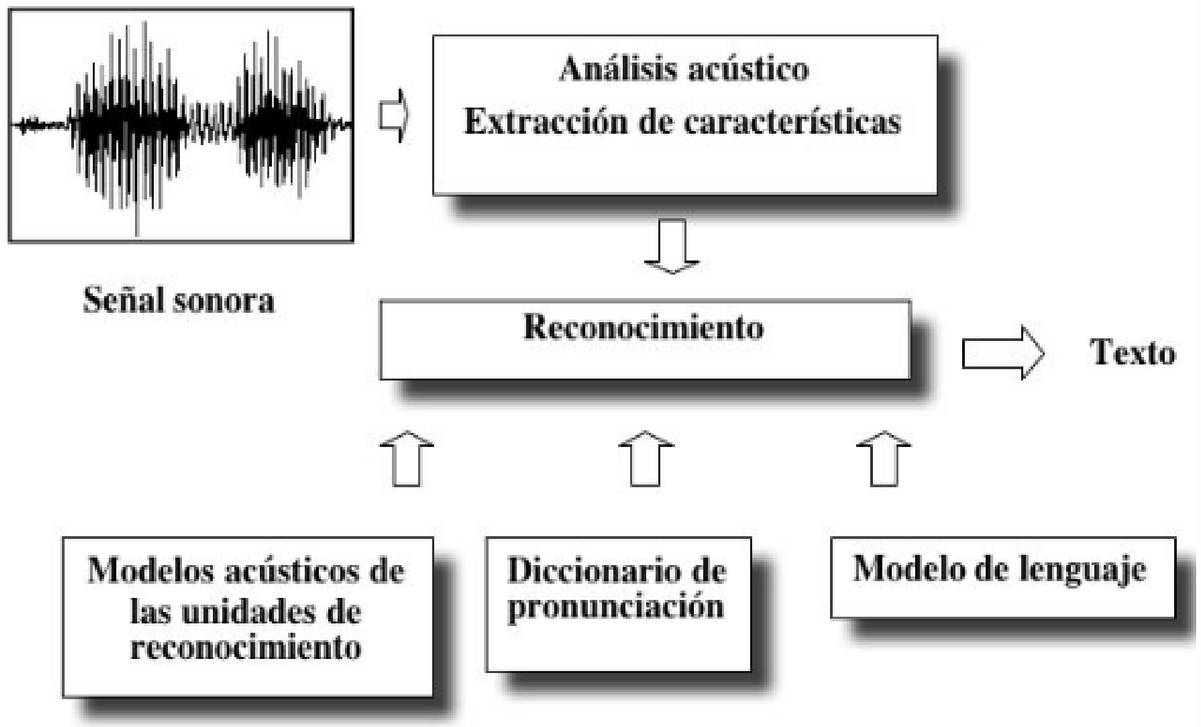
Regla de decisión bayesiana

$$\hat{W} = \operatorname*{argmax}_{W} P(W)P(\mathbb{X} \mid W)$$

Donde:

- $P(W)$ corresponde al modelo de lenguaje
- $P(\mathbb{X} \mid W)$ corresponde al modelo acústico

¿Que se necesita?



Análisis Acústico

Definir un Vocabulario

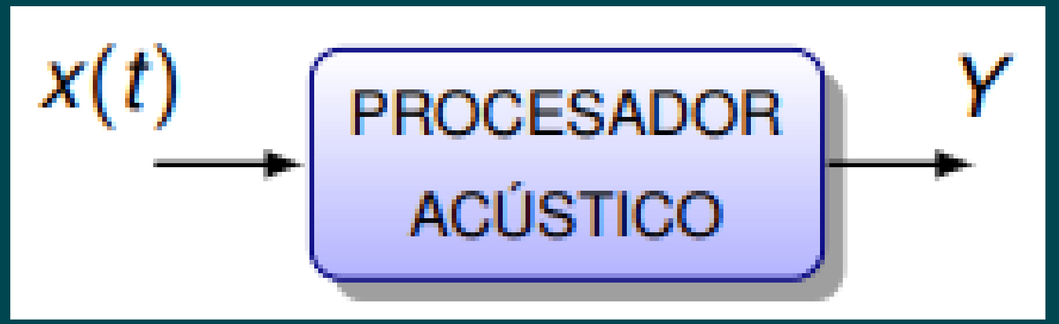
Definir el modelo fonético de cada palabra del vocabulario

Definir la red de palabras ya sea con una gramática finita o un modelo de lenguaje

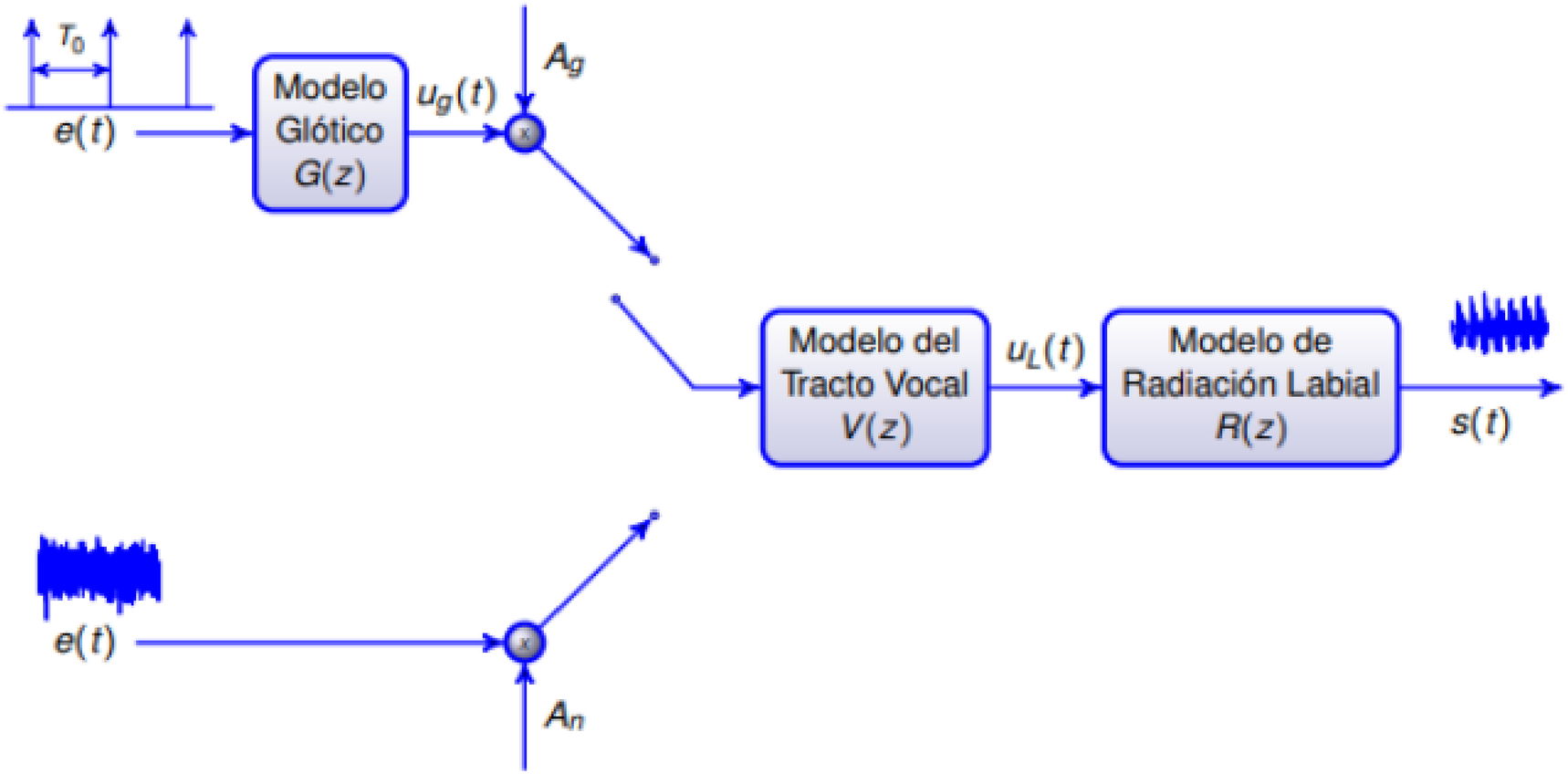
Encontrar la secuencia de estados óptima de estados de la red para la observación dada

Parametrización de los Datos

- Se cuenta con emisiones acusticas de diferentes hablantes en formato wav y su transcripcion a nivel texto.
- A partir de las grabaciones (wav), se busca obtener los coeficientes MFCC correspondientes.
- Mediante los coeficientes MFCC, se logra extraer información fonetica

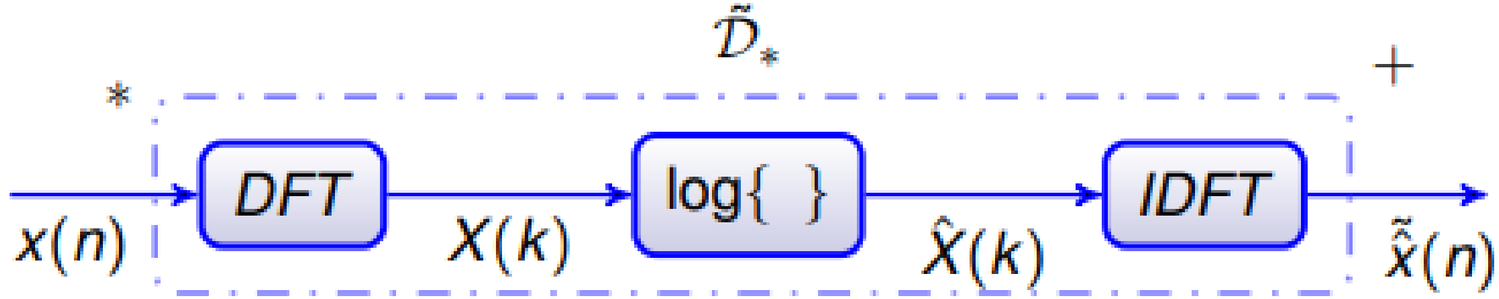


Modelo producción del habla



Cepstrum

- Implementación práctica del cepstrum



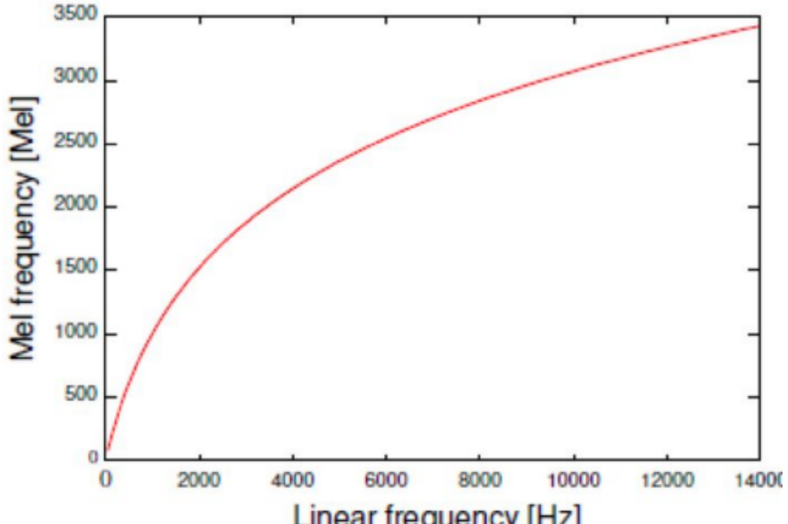
- La parte del cepstrum relacionada con el tracto vocal se concentra en la región de bajas quefrencias.
- La parte del cepstrum relacionada con la exitación glótica se concentra en las quefrencias altas.
- La deconvolución se logra multiplicando el cepstrum por una ventana que separe ambas zonas.

Coeficientes MFCC

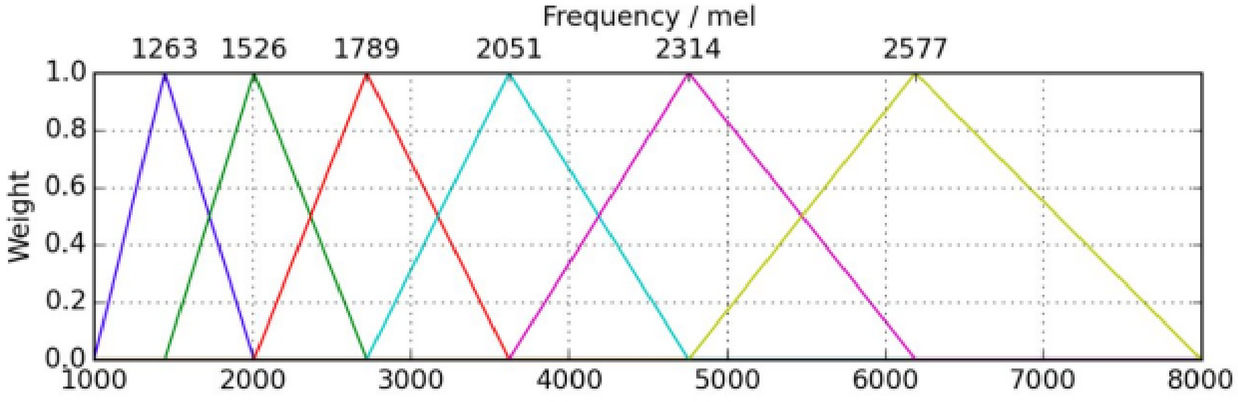
- Los coeficientes mel-cepstrum son un derivado del procesamiento Cepstrum, donde se agrega información biológica del sistema auditivo para obtener mejores resultados.

Información Biologica

- La escala en la que el oído percibe frecuencias no es lineal.



- El rango de discriminación de diferentes frecuencias no es igual para distintas frecuencias.



Coeficientes MFCC

Para cada ventana de señal $x_t(n)$:

- Hacer la DFT $X_t(k)$ de N puntos.
- Agrupar los coeficientes de la DFT en bandas críticas
- Ponderar los coeficientes con los correspondientes filtros triangulares W_m con $m = 1, \cdots, M$
- Obtener el módulo del logaritmo de la salida de los filtros y realizar la transformada coseno inversa

Coeficientes Dinamicos:

- Señal inherentemente dinámica, se agregan coeficientes de velocidad y aceleración.

$$\Delta_t(l) = \frac{Y_{t+1}(l) - Y_{t-1}(l)}{2} \quad l = 1, \cdots, L$$

$$\Delta_t^2(l) = \frac{\Delta_{t+1}(l) - \Delta_{t-1}(l)}{2} \quad l = 1, \cdots, L$$

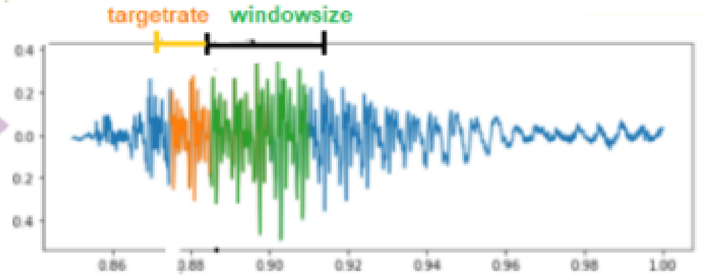
Coeficientes MFCC en HTK

```
# Coding parameters
TARGETKIND = MFCC_0
#TARGETKIND = MFCC_0_D_A
TARGETRATE = 100000.0
SAVECOMPRESSED = T
SAVEWITHCRC = T
WINDOWSIZE = 250000.0
USEHAMMING = T
PREEMCOEF = 0.97
NUMCHANS = 26
CEPLIFTER = 22
NUMCEPS = 12
ENORMALISE = F
SOURCEFORMAT = NIST
```

Usa C0 coeficiente de energia : $C_0 = \sqrt{\frac{2}{N}} \sum_{j=1}^{N} log(Y(i))$

Coeficientes dinamicos

Tamaño y desplazamiento de la ventana



Tipo de ventana

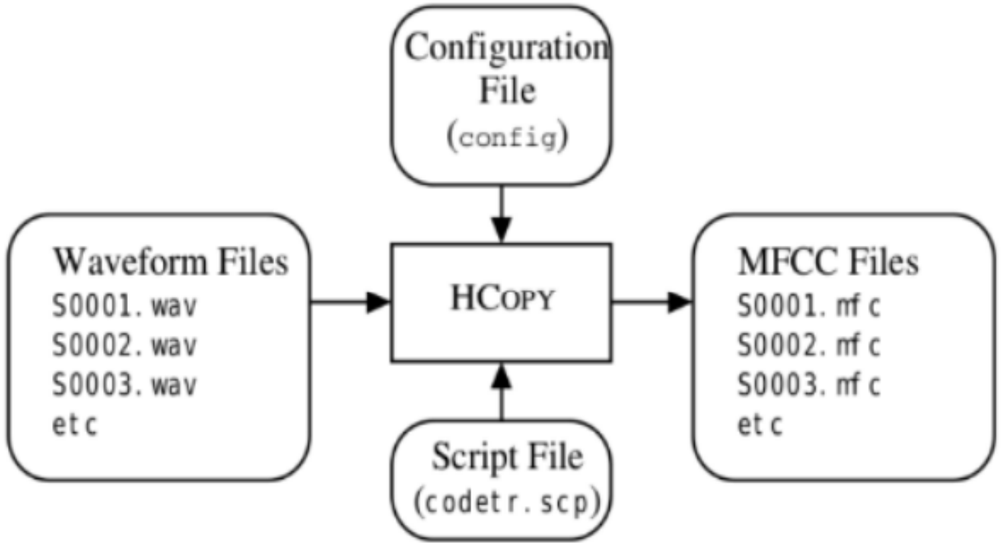
Cantidad de canales mel-cepstrum

Cantidad de coeficientes

Multiplica cada uno de los doce coeficientes

$$c'_n = \left(1 + \frac{L}{2} sin\frac{\pi n}{L}\right) c_n$$

```
176   HCopy -T 1 -C ../config/config.hcopy -S genmfc.train
177   HCopy -T 1 -C ../config/config.hcopy -S genmfc.test
```



Modelo de Lenguaje

- En el contexto de reconocimiento de habla, pueden ser descriptos como un método que asigna probabilidades a la ocurrencia de párrafos en un texto.

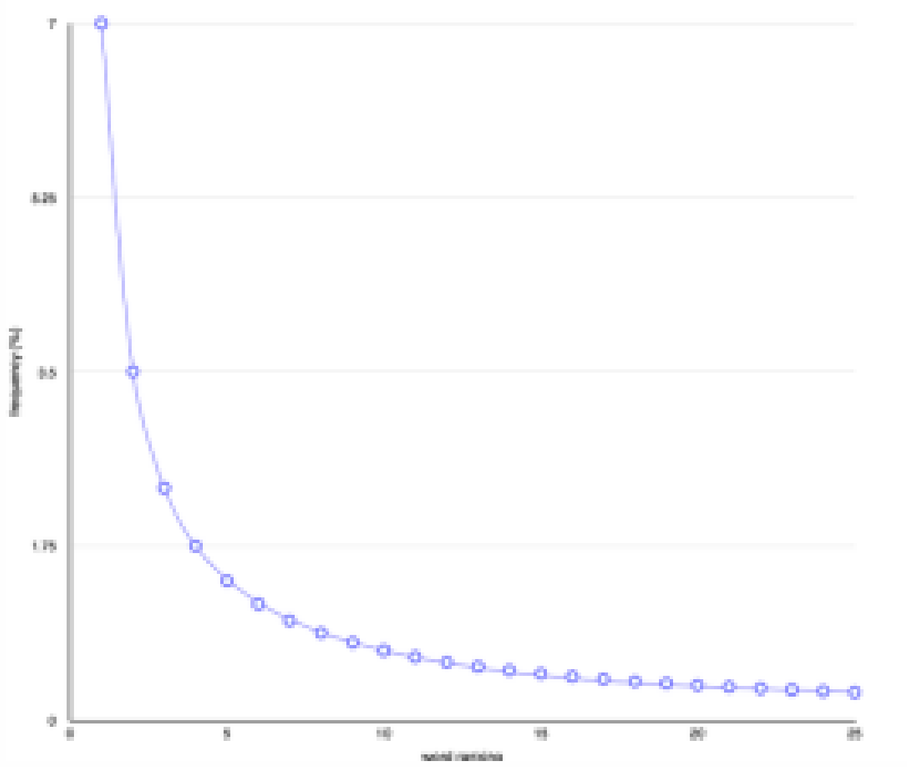
- En este proyecto se asignan las probabilidades mediante frecuencias relativas.

- Se emplearan modelos de bi-gramas, es decir, un conjunto de dos palabras consecutivas.

$$P(viento \mid el) = \frac{N(el, viento)}{N(el)}$$

Problema:

- Ley de Zipf:



- $N(respuesta, sopla) = 0$ en el texto de entrenamiento en cuyo caso $P(sopla \mid respuesta) = 0$. Overfitting

Solución: Suavizado

- Consiste en asignar probabilidades pequeñas a eventos de probabilidad nula.

Metodos de interpolación

$$P_I(z \mid h) = \lambda_h P_{ML}(z \mid h) + (1 - \lambda_h) P_I(z \mid h')$$

Metodos de back-off

$$P_{bo}(z \mid h) = \begin{cases} \alpha(z \mid h) & \text{si } N(z,h) > 0 \\ \gamma(h) P_{bo}(z \mid h') & \text{si } N(z,h) = 0 \end{cases}$$

- En que se diferencian?
  Cuando $N(z,h) > 0$ los métodos de back-off no usan la información de las distribuciones de órden inferior, y los métodos interpolados siempre utilizan dicha información.

Solución: Suavizado

Método de Kneser-Ney

$$P_{KN}(z \mid h) = \begin{cases} \dfrac{N(z,h) - d}{N(h)} & \text{para } N(z,h) > 0 \\ \gamma(h) P_{KN}(z \mid h') & \text{para } N(h,z) = 0 \end{cases}$$

$$P_{KN}(z \mid h') = \frac{N_+(\bullet, h', z)}{N_+(\bullet, h', \bullet)}$$

$$d = \frac{n_1}{n_1 + 2n_2}$$

$$\gamma(h) = \frac{1 - \sum_{z:N(z,h)>0} \dfrac{N(z,h) - d}{N(h)}}{1 - \sum_{z:N(z,h)>0} P_{KN}(z \mid h')}$$

Formato DARPA:

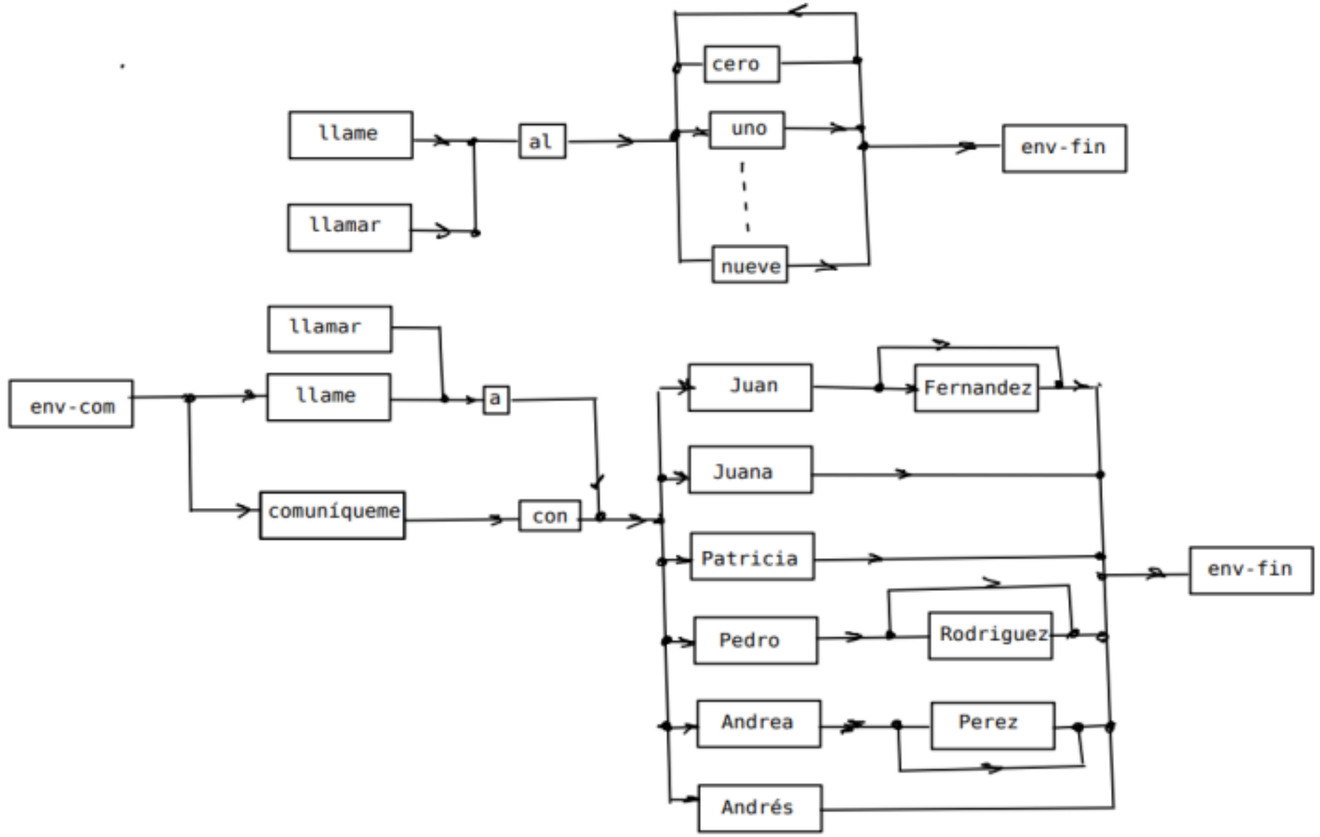
```
\data\
ngram 1=2723
ngram 2=9359

\1-grams:
-0.8753251      </s>
-99     <s>     -0.9254504
-1.679226       a       -0.753336
-7.196967       abajo
-7.196967       abandonada
-4.459543       abandonado      -0.321929
-7.196967       abarcan
-4.158513       abastecimiento  -0.6194938
-4.459543       abatir  -0.3392441
```

```
669  /usr/local/speechapp/srilm/bin/i686-m64/ngram-count
-order 2 -text train.txt -lm lm140 -ukndiscount2
-vocab vocab
```

```
1093  HBuild -n lm140 -s '<s>' '</s>' vocab wdnet
```

Gramática Finita:




```
1731   HParse ../etc/gram wdnet.gf
```

Modelo acústico

- Estimar palabras trae aparejado los problemas de que se requieren muchos ejemplos de cada palabra (ley de zipf) y que es poco flexible a la incorporacion de nuevas palabras.

- Se utilizo un modelo de sub-unidades (fonemas) .

Ventajas

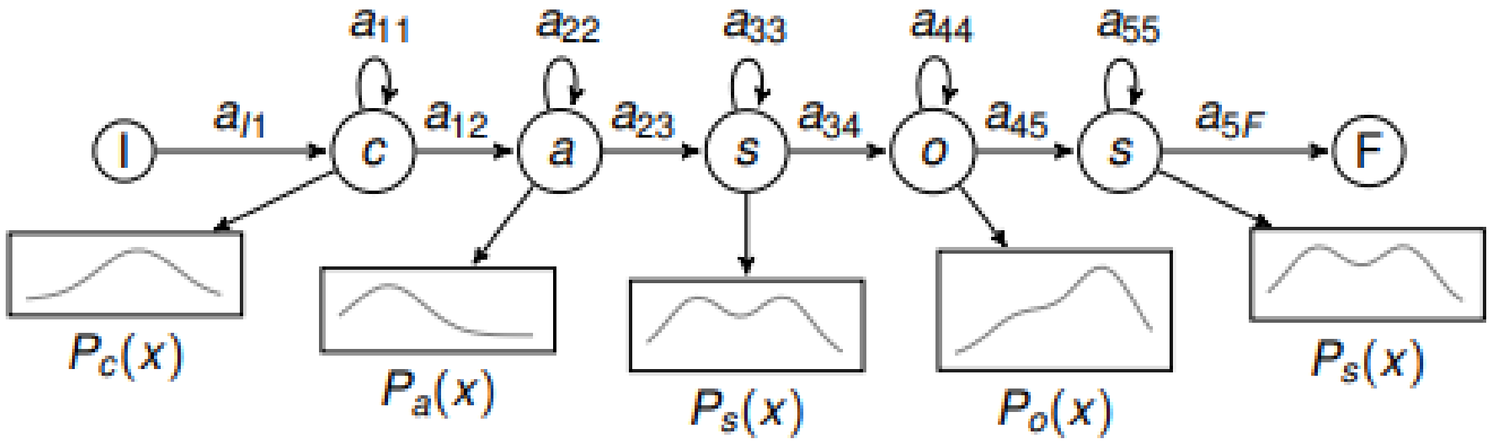
- Practicamente no hay fonemas “raros”.
- Si entrenamos modelos de fonemas en lugar de palabras tendremos suficientes datos para entrenar todos los HMMs
- Se podra crear cualquier modelo de palabra, aun de aquellas que nunca ocurrieron en los datos de entrenamiento

Modelos ocultos de Markov

- Una cadena de markov es una secuencia de variables aleatorias Z que adopta un conjunto discreto de valores llamados estados.

- Los modelos ocultos son una extensión de CM en los cuales se introduce un proceso aleatorio en cada estado que genera símbolos de observación.

.



Modelos ocultos de Markov

Parámetros

- Matriz de transición: $A = [a_{ij}]$ con $a_{ij} = P(z_t = j | z_{t-1} = i)$
- Distribuciones de observaciones por estado: $B = [b_j(x)]$ con $b_j(x) = P(x | z_t = j)$
- Distribución del estado inicial: $\Pi = [\pi_i]$ con $\pi_i = P(z_1 = i)$

Hipótesis

Primera suposición fundamental: Hipótesis de Markov

La probabilidad de que una cadena de Markov se encuentre en un estado particular en un determinado instante depende solamente del estado de la cadena en el instante anterior. Es decir,

$$P(z_t | z_{t-1}, \cdots, z_1) = P(z_t | z_{t-1})$$

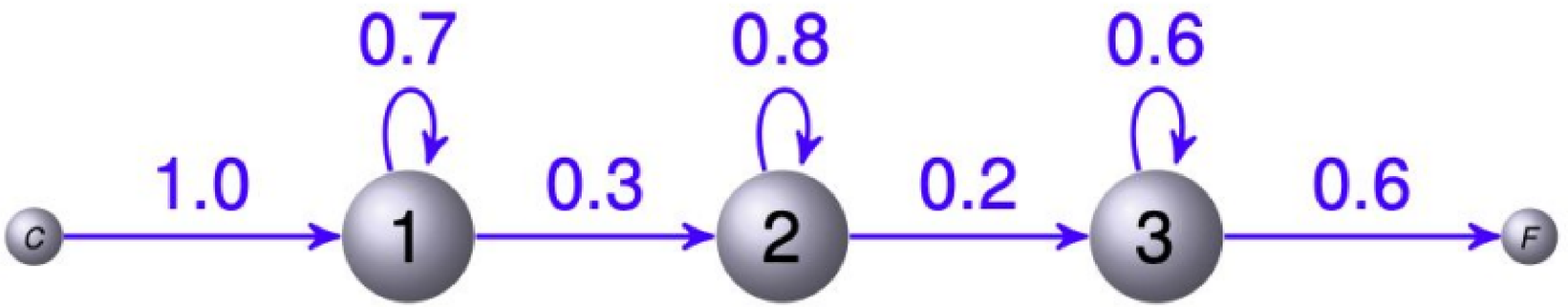
Segunda suposición fundamental: Hipótesis de independencia condicional

La probabilidad de que una observación x_t sea emitida en el instante t depende solo del estado z_t y es independiente de las observaciones y los estados anteriores, es decir, para una secuencia parcial de estados $\mathbb{Z}^t = \{z_1, \ldots, z_t\}$ que emite una secuencia parcial de observaciones $\mathbb{X}^t = \{x_1, \ldots, x_t\}$, se cumplirá que:

$$P(x_t | \mathbb{X}^{t-1}, \mathbb{Z}^t) = P(x_t | z_t)$$

En el reconocedor:

- Topología del modelo: En general se usa una topología de Bakis de tres estados. Esto es un compromiso entre eficiencia computacional y complejidad del modelo.



- Cantidad de gausianas por estado. Depende de la cantidad de datos de entrenamiento disponible. Se seleccionan por prueba y error con un mínimo de cuatro.

Primer Problema:

Determinar la probabilidad de generar una secuencia de observaciones

Dada una secuencia de observaciones $\mathbb{X} = \{x_1, \ldots, x_T\}$ y un modelo $\lambda = \{A, B, \Pi\}$ determinar la probabilidad de que dicho modelo emita esa secuencia, es decir calcular $P(\mathbb{X} \mid \lambda)$

$$P(\mathbb{X}) = P(x_1, \ldots, x_T) = \sum_{\mathbb{Z}} \prod_{t} P(x_t \mid z_t) P(z_t \mid z_{t-1})$$ ➡ N^T posibles secuencias

Tiene su aplicación en reconocimiento de palabras aisladas:

Elegir la palabra cuyo modelo sea mas verosimil con $\mathbb{X}$

$$\hat{W} = \underset{W}{\operatorname{argmax}}\, P(\mathbb{X} \mid \lambda_W)$$

Recursión forward

Se define como el likelihood parcial de todos los caminos que terminan en el estado j en el instante t

$$\alpha_t(j) \triangleq P(x_1, \cdots, x_t, z_t = j)$$

Caso general:

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(x_t) \text{ con } 1 \leq j \leq N \text{ y } 1 < t \leq T$$

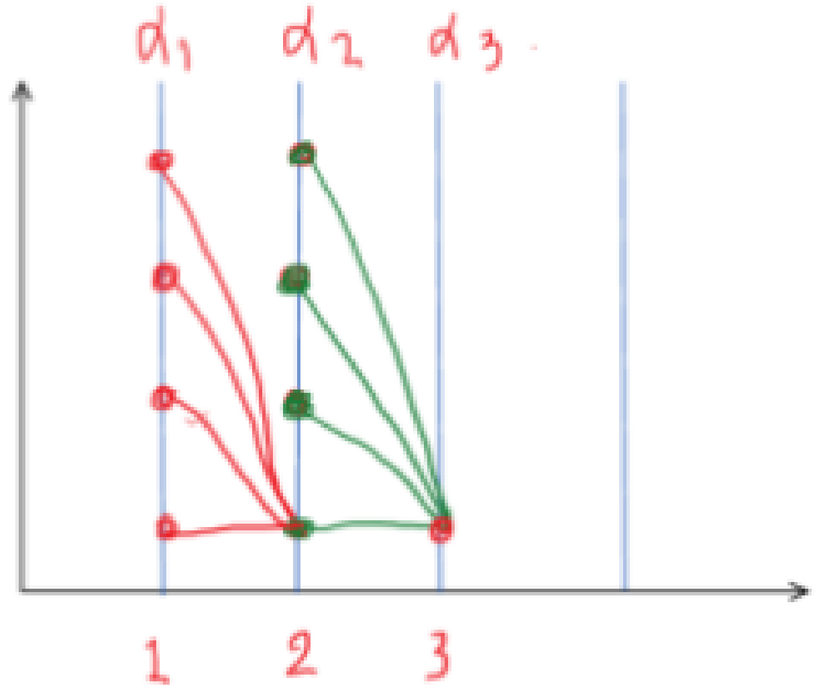
$$\alpha_1(i) = \pi_i b(x_1) \text{ con } 1 \leq j \leq N$$

Para la topologia Bakis:

$$\alpha_t(j) = \sum_{i=2}^{N-1} \alpha_{t-1}(i) a_{ij} b_j(x_t) \text{ con } 1 < j < N \text{ y } 1 < t \leq T$$

$$\alpha_0(1) = 1$$

$$\alpha_1(i) = \pi_i b_i(x_1) \text{ con } 1 < i < N$$



➡ N * T operaciones

Recursión Backward

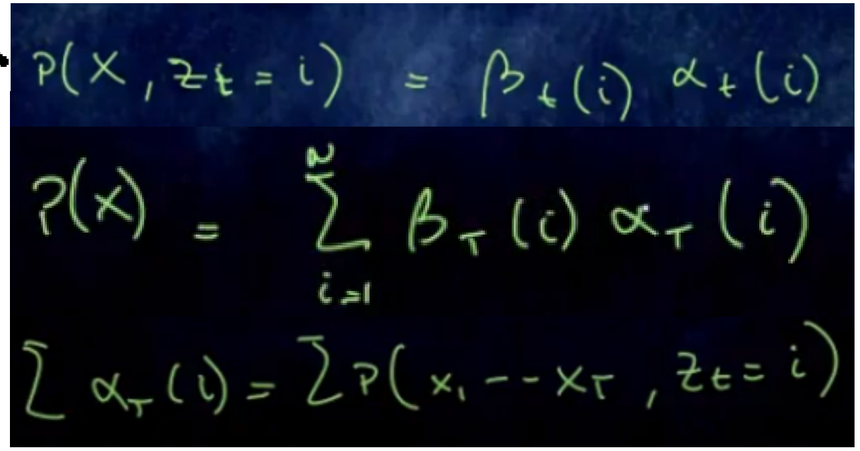
Se define como el likelihood parcial de los caminos que en el instante t se encontraban en el estado i.

$$\beta_t(i) \triangleq P(x_{t+1}, \cdots, x_T | z_t = i)$$

Caso general:

$$\beta_t(i) = \sum_{j=1}^{N} \beta_{t+1}(j) a_{ij} b_j(x_{t+1}) \text{ con } 1 \leq i \leq N \text{ y } 1 \leq t \leq T-1$$

$$\beta_T(i) = 1/N \text{ con } 1 \leq i \leq N$$



Para la topologia Bakis:

$$\beta_t(i) = \sum_{j=2}^{N-1} \beta_{t+1}(j) a_{ij} b_j(x_{t+1}) \text{ con } 1 < i < N \text{ y } T > t \geq 1$$

$$\beta_T(i) = a_{iN} \text{ con } 1 < i < N$$

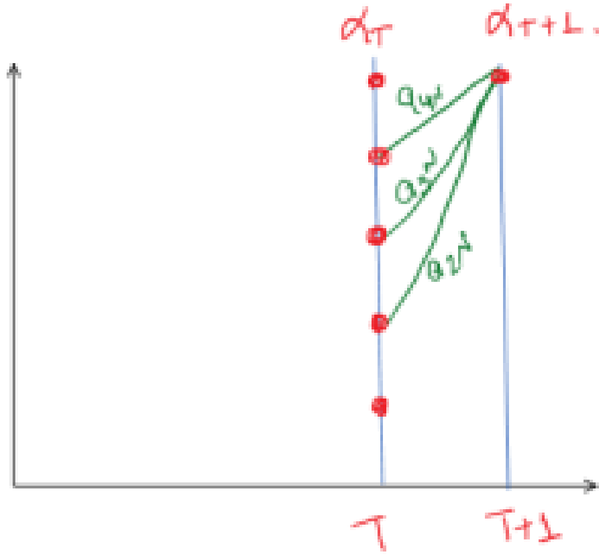
Calculo de la probabilidad de Obs:

$$P(\mathbb{X}) = \sum_{j=1}^{N} P(x_1, \cdots, x_T, z_T = j)$$

Para la topologia Bakis:

$$P(\mathbb{X}) = \alpha_{T+1}(N) = \sum_{i=2}^{N-1} \alpha_T(i) a_{iN}$$

$$P(\mathbb{X}) = \beta_0(1) = \sum_{i=2}^{N-1} \beta_1(i) \pi_i b_i(x_1)$$



Probabilidad a posteriori de un estado:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}$$

Probabilidad de transición entre estados:

$$\xi_t(i,j) = \frac{\alpha_t(i)\beta_{t+1}(j)b_j(x_{t+1})a_{ij}}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}$$

Tercer Problema:

Estimar los parámetros del modelo

Como ajustar los parámetros del modelo $\lambda = \{A, B, \Pi\}$ a partir de secuencias de observaciones de entrenamiento.

Solución: Algoritmo de Baum-Welch

- El objetivo será maximizar el log-likelihood $\ln p(\mathbf{X}|\theta)$ dado por:

$$\ln p(\mathbf{X}|\theta) = \ln \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\theta)$$

- Dado que no podemos hallar los parámetros de la distribución $p(\mathbf{X}, \mathbf{Z}; \theta)$ hallamos los parámetros que maximizan la expectación:

$$\mathcal{Q}(\theta, \theta^{old}) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \theta^{old}) \ln p(\mathbf{X}, \mathbf{Z}|\theta)$$

Algoritmo de Baum-Welch

Implementación del algoritmo de Baum Welch

1. Inicializar las μ_j, Σ_j y a_{ij} y calcular el valor inicial del log-likelihood.
2. Paso E. Calcular las responsabilidades $\gamma_t(j)$ y las probabilidades de transición de estados $\xi_t(i,j)$ usando dichos valores iniciales.
3. Paso M. Calcular los nuevos valores de los parámetros μ_j, Σ_j y a_{ij}.
4. Calcular el log-likelihood con los nuevos valores de los parámetros, y compararlo con el anterior para chequear la convergencia. Si no se satisface volver al paso 2 y repetir.

$$\mu_j = \frac{\sum_{t=1}^{T} \gamma_t(j) x_t}{\sum_{t=1}^{T} \gamma_t(j)} \qquad \Sigma_j = \frac{\sum_{t=1}^{T} (x_t - \mu_j)(x_t - \mu_j)^T \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \qquad a_{ij} = \frac{\sum_{t=1}^{T} \xi_t(i,j)}{\sum_{t=1}^{T} \gamma_t(i)}$$

Mezcla de Gaussianas:

$$\gamma_t(j,k) = P(k, z_t = j|\mathbb{X}) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)} \frac{c_{jk} b_{jk}(x_t)}{\sum_{m=1}^{K} c_{jm} b_{jm}(x_t)}$$

Entrenamiento de modelos

Para estimar los modelos fonéticos necesitamos:

1. Un conjunto de emisiones acústicas de habla separadas en oraciones con diferentes hablantes y pronunciaciones
2. Transcripciones a nivel texto de dichas emisiones
3. Un diccionario de pronunciaciones

- Parametrización de las emisiones acústicas: Las emisiones acústicas de cada oración son convertidas en secuencias de features (por ejemplo coeficientes MFCC). El resultado es un conjunto de secuencias de observaciones una por oración.
- Conversión del texto de las emisiones en secuencias de fonemas. Usando el diccionario de pronunciaciones convertimos cada palabra del texto de cada emisión en una secuencia de fonem

- Con la secuencia de fonemas y el HMM asociado a cada uno obtenemos la secuencia de estados asociados a cada secuencia de observación.
- Entrenamiento de los HMMs de los fonemas usando Baum-Welch

Entrenamiento de modelos: HTK

Construcción de los HMMs:

```
1184   HHEd -C ../config/config -T 1 -H hmm7/macros -H
hmm7/hmmdefs -M hmm2g.0 editf2g   ../etc/monophones+sil+sp
```

Estimación:

```
457   HERest -T 1 -C ../config/config -I ../etc/phones0.mlf -t
250.0 150.0 1000.0 -S train.scp -H hmm0/macros -H
```

Segundo Problema:

Determinar la secuencia de estados subyacente

Dada una secuencia de observaciones $\mathbb{X} = \{x_1, \ldots, x_T\}$ y un modelo $\lambda = \{A, B, \Pi\}$, determinar la correspondiente secuencia de estados subyacente $\mathbb{Z} = \{z_1, \ldots, x_T\}$ que sea óptima en algún sentido.

Solución: Algoritmo de Viterbi

Definimos:

$$\phi_t(j) \triangleq \max_{\forall z_1, \cdots, z_{t-1}} P(\mathbb{Z}_{t-1}, z_t = j, \mathbb{X}_t)$$

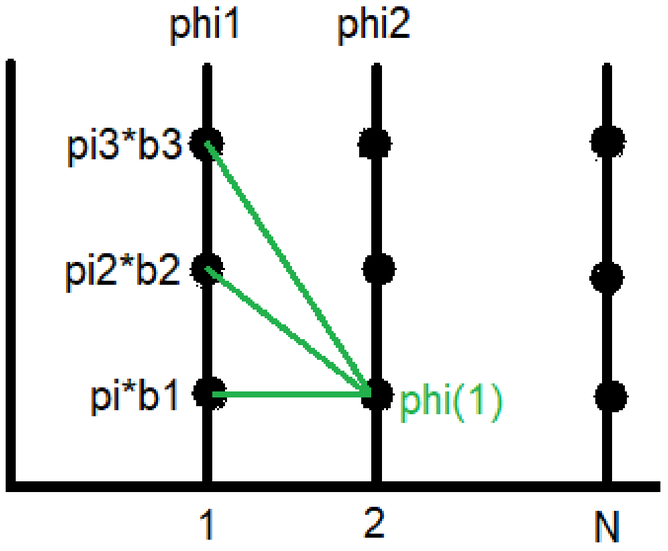
De todas las secuencias que en el instante t se encuentran en el estado j nos dice cual es la más probable para la secuencia de observaciones $\{x_1, \cdots, x_t\}$.

Algoritmo de Viterbi

Caso general:

$$\phi_t(j) = b_j(x_t) \max_{1 \leq i \leq N} \phi_{t-1}(i) a_{ij} \text{ con } 1 \leq j \leq N \text{ y } 2 \leq t \leq T$$

$$\phi_1(i) = \pi_i b_i(x_1) \text{ con } 1 \leq i \leq N$$



Para la topologia Bakis:

$$\phi_t(j) = b_j(x_t) \max_{1 < i < N} \phi_{t-1}(i) a_{ij} \text{ con } 1 < j < N \text{ y } 2 \leq t \leq T$$

$$\phi_0(1) = 1$$

$$\phi_1(i) = \pi_i b_i(x_1) \text{ con } 1 < i < N$$

Algoritmo de Viterbi

Por el principio de Bellman, la secuencia de estados óptima deberá estar formada por estados óptimos, es decir:

$$\hat{\mathbb{Z}} = \{\hat{z}_1, \cdots, \hat{z}_T\}$$

El estado óptimo en el instante T será:

$$\hat{z}_T = \underset{1 \leq j \leq N}{\operatorname{argmax}}\, \phi_T(j)$$

y el valor de la probabilidad de la secuencia total de estados más probable será:

$$\underset{\forall \mathbb{Z}}{\text{máx}}\, P(\mathbb{Z}, \mathbb{X}) = \underset{1 \leq j \leq N}{\text{máx}}\, \phi_T(j)$$

Si almacenamos el estado anterior del cual proviene el estado actual de máxima probabilidad:

$$\psi_t(j) = \underset{1 \leq i \leq N}{\operatorname{argmax}}\, \phi_{t-1}(i) a_{ij} \text{ para } 1 \leq j \leq N \text{ y } t = T, \cdots, 2$$

Algoritmo de Viterbi

1. Inicialización:

$$\phi_1(i) = \pi_i b_j(x_1) \text{ para } 1 \leq i \leq N \quad \psi_1 i = 0$$

2. Recursión:

$$\phi_t(j) = b_j(x_t) \max_{1 \leq i \leq N} \phi_{t-1}(i) a_{ij} \text{ con } 1 \leq j \leq N \text{ y } 2 \leq t \leq T$$

$$\psi_t(j) = \operatorname*{argmax}_{1 \leq i \leq N} \phi_{t-1}(i) a_{ij} \text{ para } 1 \leq j \leq N \text{ y } t = 2, \cdots, T$$

3. Terminación:

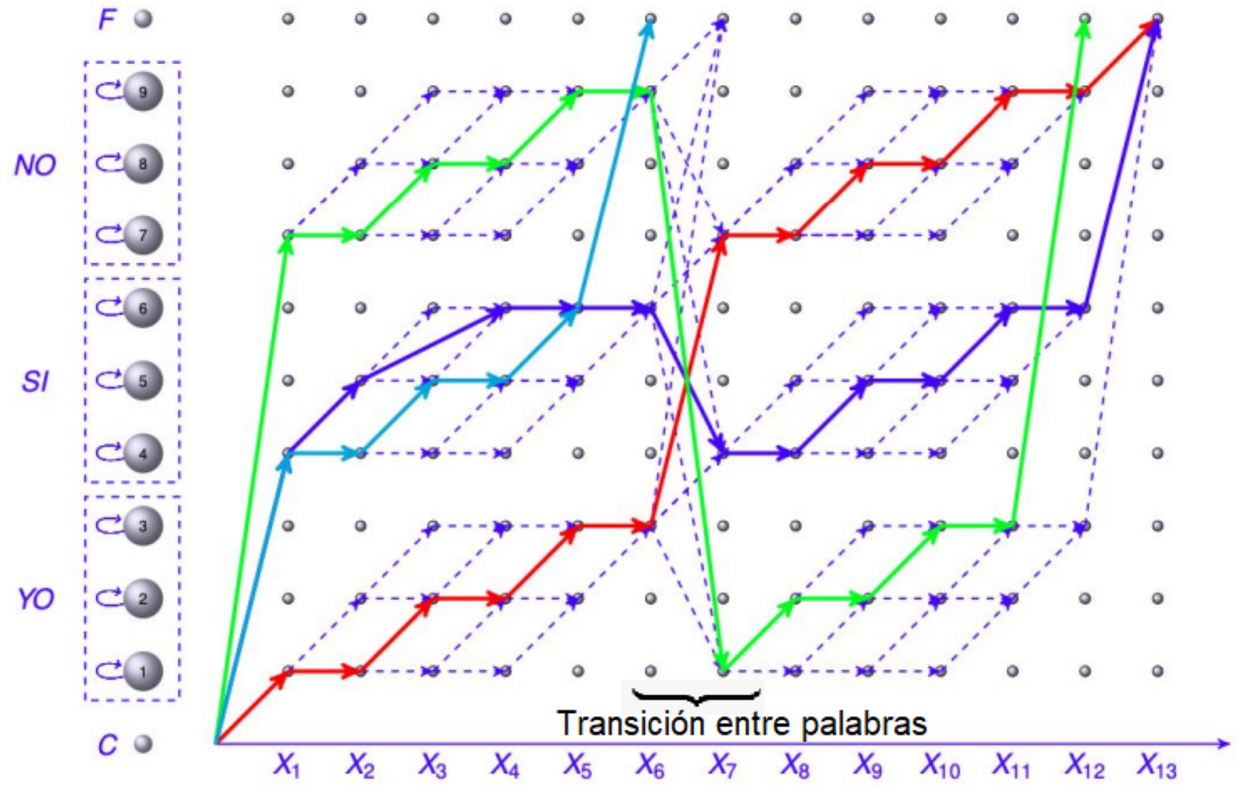
$$\max_{\forall \mathbb{Z}} P(\mathbb{Z}, \mathbb{X}) = \max_{1 \leq j \leq N} \phi_T(j) \Rightarrow \hat{z}_T = \operatorname*{argmax}_{1 \leq j \leq N} \phi_T(j)$$

4. Back-tracking del camino:

$$\hat{z}_t = \psi_{t+1}(\hat{z}_{t+1}) \text{ para: } t = T - 1, \cdots, 1$$

Reconocimiento de palabras conectadas

$$\hat{W} = \underset{W}{\operatorname{argmax}}\, P(W) \underset{\mathbb{Z}_W}{\text{máx}}\, P(\mathbb{X} \mid \mathbb{Z}) P(\mathbb{Z})$$



C – 1 – 1 – 2 – 2 – 3 – 3 – 7 – 7 – 8 – 8 – 9 – 9 – F
(Yo) (No)

C – 7 – 7 – 8 – 8 – 9 – 9 – 1 – 2 – 2 – 3 – 3 – F
(No) (Yo)

C – 4 – 5 – 6 – 6 – 6 – 4 – 5 – 5 – 6 – 6 – F
(Si) (Si)

C – 4 – 4 – 5 – 5 – 6 – F
(Si)

Reconocimiento de palabras: HTK

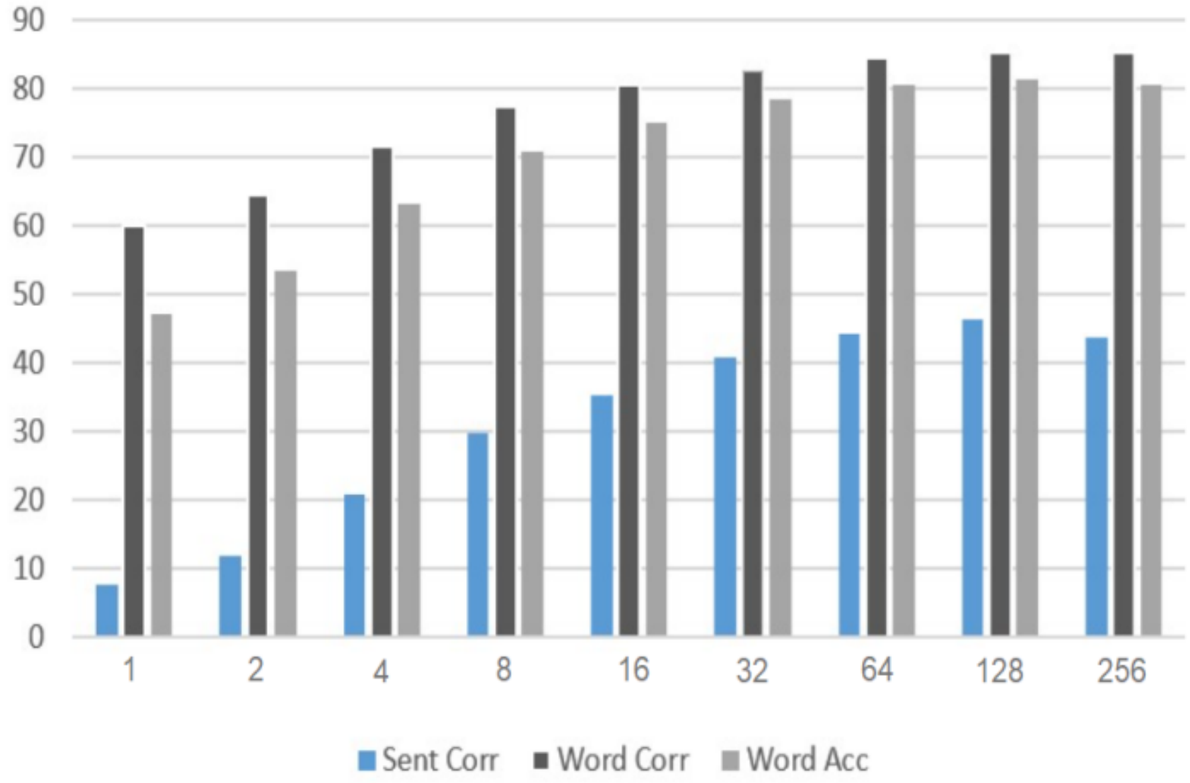
Algoritmo de Viterbi:

```
1146  HVite -C ../config/config -H ../modelos/hmm7/macros
-H ../modelos/hmm7/hmmdefs -S test.scp -l '*' -i recout.mlf
-w ../lm/wdnet -p 0.0 -s 5.0 ../etc/dict140
../etc/monophones+sil+sp &
```

Metricas del resultado:

```
1169  HResults -t -f -I testref.mlf ../etc/monophones+sil+sp
recout.mlf
```

Resultados: 3000 palabras



Mejor resultado: 128 gaussianas

```
---------------------- Overall Results -----------------------
SENT: %Correct=46.30 [H=457, S=530, N=987]
WORD: %Corr=85.45, Acc=81.44 [H=6759, D=200, S=951, I=317, N=7910]
================================================================
```

Resultados: gramática finita

```
---------------------- Overall Results -----------------------
SENT: %Correct=75.50 [H=151, S=49, N=200]
WORD: %Corr=96.78, Acc=90.35 [H=1505, D=3, S=47, I=100, N=1555]
================================================================
```

- Al disminuir la cantidad de palabras del vocabulario e incorporar restricciones al modelo de lenguaje se obtienen mejores resultados.