Capítulo 4

Modelos de Markov ocultos (HMM,s)

Cuando se abordó inicialmente el problema de la etiquetación de textos en lenguaje natural, se comenzó diseñando manualmente reglas de etiquetación que intentaban describir el comportamiento del lenguaje humano. Sin embargo, en el momento en el que aparecen disponibles grandes cantidades de textos electrónicos, muchos de ellos incluso etiquetados, las aproximaciones estocásticas adquieren gran importancia en el proceso de etiquetación. Son muchas las ventajas de los etiquetadores estocásticos frente a los etiquetadores construidos manualmente. Además del hecho de que evitan la laboriosa construcción manual de las reglas, también capturan automáticamente información útil que el hombre podría no apreciar.

En este capítulo estudiaremos un método estocástico comúnmente denominado aproximación con modelos de Markov ocultos o HMM,s[1]. A pesar de sus limitaciones, los HMM,s y sus variantes son todavía la técnica más ampliamente utilizada en el proceso de etiquetación del lenguaje natural, y son generalmente referenciados como una de las técnicas más exitosas para dicha tarea. Los procesos de Markov fueron desarrollados inicialmente por Andrei A. Markov, alumno de Chebyshev, y su primera utilización tuvo un objetivo realmente lingüístico: modelizar las secuencias de letras de las palabras en la literatura rusa [Markov 1913]. Posteriormente, los modelos de Markov evolucionaron hasta convertirse en una herramienta estadística de propósito general. La suposición subyacente de las técnicas de etiquetación basadas en HMM,s es que la tarea de la etiquetación se puede formalizar como un proceso paramétrico aleatorio, cuyos parámetros se pueden estimar de una manera precisa y perfectamente definida.

Comenzaremos con un repaso de la teoría de cadenas de Markov, seguiremos con una extensión de las ideas mostradas hacia los modelos de Markov ocultos y centraremos nuestra atención en las tres cuestiones fundamentales que surgen a la hora de utilizar un HMM:

1. La evaluación de la probabilidad de una secuencia de observaciones, dado un HMM específico.

2. La determinación de la secuencia de estados más probable, dada una secuencia de observaciones específica.

3. La estimación de los parámetros del modelo para que éste se ajuste a las secuencias de observaciones disponibles.

Una vez que estos tres problemas fundamentales sean resueltos, veremos cómo se pueden aplicar inmediatamente los HMM,s al problema de la etiquetación del lenguaje natural, y estudiaremos con detalle las múltiples variantes de implementación que surgen al hacerlo.

[1] *Hidden Markov Models.*

4.1 Procesos de Markov de tiempo discreto

Consideremos un sistema que en cada instante de tiempo se encuentra en un determinado estado. Dicho estado pertenece a un conjunto finito de estados Q. De momento, para no complicar demasiado la notación, supondremos que existe una correspondencia entre el conjunto de estados Q y el conjunto de números enteros $\{1, 2, \ldots, N\}$, y etiquetaremos cada estado con uno de esos números, tal y como se ve en la figura 4.1, donde $N = 5$. Regularmente, transcurrido un espacio de tiempo discreto, el sistema cambia de estado (posiblemente volviendo al mismo), de acuerdo con un conjunto de probabilidades de transición asociadas a cada uno de los estados del modelo. Los instantes de tiempo asociados a cada cambio de estado se denotan como $t = 1, 2, \ldots, T$, y el estado actual en el instante de tiempo t se denota como q_t. En general, una descripción probabilística completa del sistema requeriría la especificación del estado actual, así como de todos los estados precedentes. Sin embargo, las cadenas de Markov presentan dos características muy importantes:

1. **Propiedad del horizonte limitado**. Esta propiedad permite truncar la dependencia probabilística del estado actual y considerar, no todos los estados precedentes, sino únicamente un subconjunto finito de ellos. En general, una cadena de Markov de orden n es la que utiliza n estados previos para predecir el siguiente estado. Por ejemplo, para el caso de las cadenas de Markov de tiempo discreto de primer orden tenemos que:

$$P(q_t = j | q_{t-1} = i, q_{t-2} = k, \ldots) = P(q_t = j | q_{t-1} = i). \qquad (4.1)$$

2. **Propiedad del tiempo estacionario**. Esta propiedad nos permite considerar sólo aquellos procesos en los cuales la parte derecha de (4.1) es independiente del tiempo[2]. Esto nos lleva a una matriz $A = \{a_{ij}\}$ de probabilidades de transición entre estados de la forma

$$a_{ij} = P(q_t = j | q_{t-1} = i) = P(j|i), \qquad 1 \leq i, j \leq N,$$

independientes del tiempo, pero con las restricciones estocásticas estándar:

$$a_{ij} \geq 0, \qquad \forall i, j,$$

$$\sum_{j=1}^{N} a_{ij} = 1, \qquad \forall i.$$

Sin embargo, es necesario especificar también el vector $\pi = \{\pi_i\}$, que almacena la probabilidad que tiene cada uno de los estados de ser el estado inicial[3]:

$$\pi_i = P(q_1 = i), \qquad \pi_i \geq 0, \qquad 1 \leq i \leq N,$$

$$\sum_{i=1}^{N} \pi_i = 1.$$

[2]Como ya hemos visto en el capítulo 1, este es el punto central del famoso argumento de Chomsky contra el uso de los modelos de Markov para el procesamiento de lenguaje natural.

[3]La necesidad de este vector podría evitarse especificando que la cadena de Markov comienza siempre en un estado inicial extra, e incluyendo en la matriz A las probabilidades de transición de este nuevo estado: las del vector π para aquellas celdas que lo tienen como estado origen, y 0 para las que lo tienen como estado destino.

A un proceso estocástico que satisface estas características se le puede llamar un *modelo de Markov observable*, porque su salida es el conjunto de estados por los que pasa en cada instante de tiempo, y cada uno de estos estados se corresponde con un suceso observable.

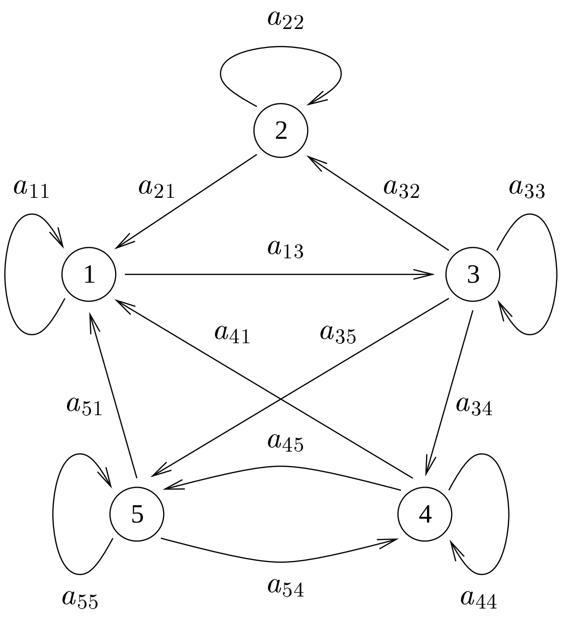


Figura 4.1: Una cadena de Markov de 5 estados con algunas transiciones entre ellos

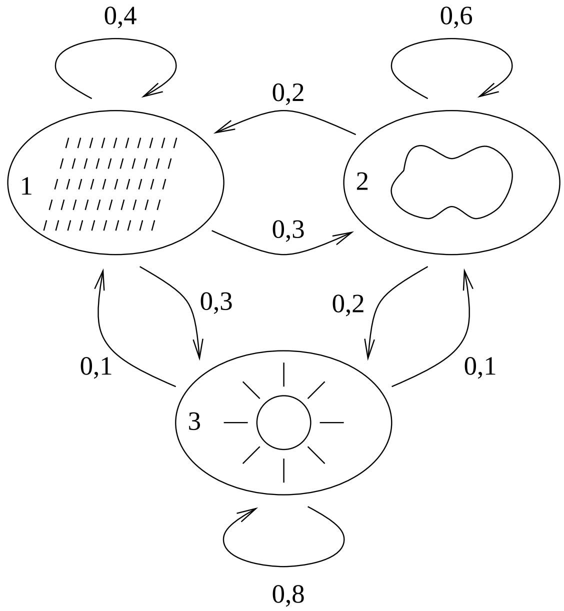


Figura 4.2: Un modelo de Markov para la evolución del clima

Ejemplo 4.1 Para remarcar las ideas, supongamos que una vez al día, por ejemplo al mediodía, se observa el clima, y las posibles observaciones resultan ser las siguientes: (1) precipitación de lluvia o nieve, (2) nublado, o (3) soleado. Supongamos también que el clima del día t se caracteriza por uno solo de esos tres estados, que la matriz A de las probabilidades de transición entre estados es

$$A = \{a_{ij}\} = \begin{bmatrix} 0,4 & 0,3 & 0,3 \\ 0,2 & 0,6 & 0,2 \\ 0,1 & 0,1 & 0,8 \end{bmatrix}$$

y que los tres estados tienen la misma probabilidad de ser el estado inicial, es decir, $\pi_i = \frac{1}{3}$, $1 \leq i \leq 3$. Tal y como se ve en la figura 4.2, no hemos hecho más que especificar un modelo de Markov para describir la evolución del clima. □

La probabilidad de observar una determinada secuencia de estados, es decir, la probabilidad de que una secuencia finita de T variables aleatorias con dependencia de primer orden, $q_1, q_2, \ldots, q_T$, tome unos determinados valores, $o_1, o_2, \ldots, o_T$, con todos los $o_i \in \{1, 2, \ldots, N\}$, es sencilla de obtener. Simplemente calculamos el producto de las probabilidades que figuran en las aristas del grafo o en la matriz de transiciones:

$$P(o_1, o_2, \ldots, o_T) =$$

$$= P(q_1 = o_1)P(q_2 = o_2|q_1 = o_1)P(q_3 = o_3|q_2 = o_2) \ldots P(q_T = o_T|q_{T-1} = o_{T-1}) =$$

$$= \pi_{o_1} \prod_{t=1}^{T-1} a_{o_t o_{t+1}}.$$

Ejemplo 4.2 Utilizando el modelo de Markov para la evolución del clima, podemos calcular la probabilidad de observar la secuencia de estados 2, 3, 2, 1, como sigue:

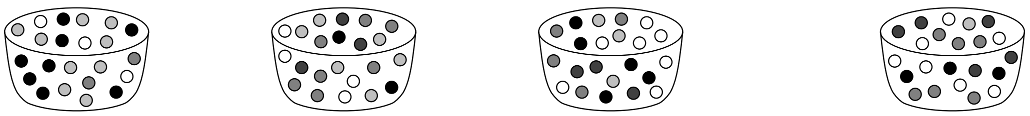
$$P(2,3,2,1) = P(2)P(3|2)P(2|3)P(1|2) = \pi_2 \times a_{23} \times a_{32} \times a_{21} = \frac{1}{3} \times 0,2 \times 0,1 \times 0,2 = 0,00133333.$$

Ésta es, por tanto, la probabilidad que corresponde a la secuencia *nublado-soleado-nublado-lluvia.* □

4.2 Extensión a modelos de Markov ocultos

En la sección anterior hemos considerado modelos de Markov en los cuales cada estado se corresponde de manera determinista con un único suceso observable. Es decir, la salida en un estado dado no es aleatoria, sino que es siempre la misma. Esta modelización puede resultar demasiado restrictiva a la hora de ser aplicada a problemas reales. En esta sección extendemos el concepto de modelos de Markov de tal manera que es posible incluir aquellos casos en los cuales la observación es una función probabilística del estado. El modelo resultante, denominado modelo de Markov oculto, es un modelo doblemente estocástico, ya que uno de los procesos no se puede observar directamente (está oculto), sino que se puede observar sólo a través de otro conjunto de procesos estocásticos, los cuales producen la secuencia de observaciones.

Ejemplo 4.3 Para ilustrar los conceptos básicos de un modelo de Markov oculto consideremos el sistema de urnas y bolas de la figura 4.3. El modelo consta de un gran número N de urnas colocadas dentro de una habitación. Cada urna contiene en su interior un número también grande de bolas. Cada bola tiene un único color, y el número de colores distintos para ellas es M. El proceso físico para obtener las observaciones es como sigue. Una persona se encuentra dentro de la habitación y, siguiendo un procedimiento totalmente aleatorio, elige una urna inicial, saca una bola, nos grita en alto su color y devuelve la bola a la urna. Posteriormente elige otra urna de acuerdo con un proceso de selección aleatorio asociado a la urna actual, y efectúa la extracción de una nueva bola. Este paso se repite un determinado número de veces, de manera que todo el proceso genera una secuencia finita de colores, la cual constituye la salida observable del modelo, quedando oculta para nosotros la secuencia de urnas que se ha seguido.



Urna 1	Urna 2	Urna 3		Urna N
$P(color\ 1) = b_{11}$	$P(color\ 1) = b_{21}$	$P(color\ 1) = b_{31}$	$\dots$	$P(color\ 1) = b_{N1}$
$P(color\ 2) = b_{12}$	$P(color\ 2) = b_{22}$	$P(color\ 2) = b_{32}$	$\dots$	$P(color\ 2) = b_{N2}$
$P(color\ 3) = b_{13}$	$P(color\ 3) = b_{23}$	$P(color\ 3) = b_{33}$	$\dots$	$P(color\ 3) = b_{N3}$
$\vdots$	$\vdots$	$\vdots$		$\vdots$
$P(color\ M) = b_{1M}$	$P(color\ M) = b_{2M}$	$P(color\ M) = b_{3M}$	$\dots$	$P(color\ M) = b_{NM}$

Figura 4.3: Un modelo de Markov oculto de N urnas de bolas y M colores

Este proceso se modeliza de forma que cada urna se corresponde con un estado del HMM, la elección de las urnas se realiza de acuerdo con la matriz de probabilidades de transición entre estados asociada al modelo, y para cada uno de esos estados existe una función de probabilidad de los colores perfectamente definida. Es importante señalar también que lo que hace que una urna sea distinta de otra es la cantidad de bolas de cada color que tiene en su interior, pero los colores de las bolas son los mismos para todas las urnas. Por tanto, una observación aislada de un determinado color de bola no nos dice automáticamente de qué urna viene esa bola. □

El ejemplo de las urnas y las bolas puede identificarse de una manera muy intuituva con el problema de la etiquetación de las palabras de un texto en lenguaje natural. En este caso concreto, las bolas representan a las palabras, las urnas representan a las distintas etiquetas o categorías gramaticales a las que pertenecen las palabras, y las secuencias de observaciones representan a las frases del texto. Una misma palabra puede estar en urnas distintas, de ahí la ambigüedad léxica, y puede estar también varias veces en la misma urna, de ahí que las probabilidades de las palabras que están dentro de una misma urna puedan ser distintas.

4.3 Elementos de un modelo de Markov oculto

Los ejemplos vistos anteriormente nos proporcionan una idea de lo que son los modelos de Markov ocultos y de cómo se pueden aplicar a escenarios prácticos reales, tales como la etiquetación de las palabras de un texto en lenguaje natural. Pero antes de tratar nuestro caso particular, vamos a definir formalmente cuáles son los elementos de un modelo de Markov oculto.

Definición 4.1 Un *HMM* se caracteriza por la 5-tupla (Q, V, π, A, B), donde:

1. Q es el conjunto de estados del modelo. Aunque los estados permanecen ocultos, para la mayoría de las aplicaciones prácticas se conocen *a priori*. Por ejemplo, para el caso de la etiquetación de palabras, cada etiqueta del juego de etiquetas utilizado sería un estado. Generalmente los estados están conectados de tal manera que cualquiera de ellos se puede alcanzar desde cualquier otro en un solo paso, aunque existen muchas otras posibilidades de interconexión. Los estados se etiquetan como $\{1, 2, \ldots, N\}$, y el estado actual en el instante de tiempo t se denota como q_t. El uso de instantes de tiempo es apropiado, por ejemplo, en la aplicación de los HMM,s al procesamiento de voz. No obstante, para el caso de la etiquetación de palabras, no hablaremos de los instantes de tiempo, sino de las posiciones de cada palabra dentro de la frase.

2. V es el conjunto de los distintos sucesos que se pueden observar en cada uno de los estados. Por tanto, cada uno de los símbolos individuales que un estado puede emitir se denota como $\{v_1, v_2, \ldots, v_M\}$. En el caso del modelo de las urnas y las bolas, M es el número de colores distintos y cada $v_k, 1 \leq k \leq M$, es un color distinto. En el caso de la etiquetación de palabras, M es el tamaño del diccionario y cada $v_k, 1 \leq k \leq M$, es una palabra distinta.

3. $\pi = \{\pi_i\}$, es la distribución de probabilidad del estado inicial. Por tanto,
$$\pi_i = P(q_1 = i), \qquad \pi_i \geq 0, \qquad 1 \leq i \leq N,$$
$$\sum_{i=1}^{N} \pi_i = 1.$$

4. $A = \{a_{ij}\}$ es la distribución de probabilidad de las transiciones entre estados, es decir,
$$a_{ij} = P(q_t = j | q_{t-1} = i) = P(j|i), \qquad 1 \leq i, j \leq N, \qquad 1 \leq t \leq T,$$
$$\sum_{j=1}^{N} a_{ij} = 1, \qquad \forall i.$$
Para el caso de un modelo con estados totalmente conexos en un solo paso, tenemos que $a_{ij} > 0$ para todo i, j. Para otro tipo de HMM,s podría existir algún $a_{ij} = 0$.

5. $B = \{b_j(v_k)\}$ es la distribución de probabilidad de los sucesos observables, es decir,

$$b_j(v_k) = P(o_t = v_k|q_t = j) = P(v_k|j), \quad b_j(v_k) \geq 0, \quad 1 \leq j \leq N,\ 1 \leq k \leq M,\ 1 \leq t \leq T.$$

$$\sum_{k=1}^{M} b_j(v_k) = 1, \qquad \forall j.$$

   Este conjunto de probabilidades se conoce también con el nombre de *conjunto de probabilidades de emisión.*

Tal y como hemos visto, una descripción estricta de un HMM necesita la especificación de Q y V, el conjunto de estados y el conjunto de los símbolos que forman las secuencias de observación, respectivamente, y la especificación de los tres conjuntos de probabilidades π, A y B. Pero dado que los dos primeros conjuntos normalmente se conocen *a priori*, y que en todo caso los tres últimos elementos del HMM ya incluyen de manera explícita al resto de los parámetros, utilizaremos la notación compacta

$$\mu = (\pi, A, B)$$

a lo largo de las siguientes secciones, y ésta seguirá siendo una representación completa de un HMM. □

Dada una especificación de un HMM, podemos simular un proceso estocástico que genere secuencias de datos, donde las leyes de producción de dichas secuencias están perfectamente definidas en el modelo. Sin embargo, es mucho más interesante tomar una secuencia de datos, suponer que efectivamente ha sido generada por un HMM, y estudiar distintas propiedades sobre ella, tales como su probabilidad, o la secuencia de estados más probable por la que ha pasado. La siguiente sección se ocupa de este tipo de cuestiones.

4.4 Las tres preguntas fundamentales al usar un HMM

Existen tres preguntas fundamentales que debemos saber responder para poder utilizar los HMM,s en aplicaciones reales. Estas tres preguntas son las siguientes:

1. Dada una secuencia de observaciones $O = (o_1, o_2, \ldots, o_T)$ y dado un modelo $\mu = (\pi, A, B)$, ¿cómo calculamos de una manera eficiente $P(O|\mu)$, es decir, la probabilidad de dicha secuencia dado el modelo?

2. Dada una secuencia de observaciones $O = (o_1, o_2, \ldots, o_T)$ y dado un modelo $\mu = (\pi, A, B)$, ¿cómo elegimos la secuencia de estados $S = (q_1, q_2, \ldots, q_T)$ óptima, es decir, la que mejor *explica* la secuencia de observaciones?

3. Dada una secuencia de observaciones $O = (o_1, o_2, \ldots, o_T)$, ¿cómo estimamos los parámetros del modelo $\mu = (\pi, A, B)$ para maximizar $P(O|\mu)$?, es decir, ¿cómo podemos encontrar el modelo que mejor *explica* los datos observados?

Normalmente, los problemas que manejaremos en la práctica no son tan sencillos como el modelo de las urnas y las bolas, es decir, lo normal es que no conozcamos *a priori* los parámetros del modelo y tengamos que estimarlos a partir de los datos observados. La secuencia de observación utilizada para ajustar los parámetros del modelo se denomina secuencia de *entrenamiento*, ya que a partir de ella se entrena el HMM. Ésta es la problemática que se aborda en la tercera pregunta.

La segunda pregunta trata el problema de descubrir la parte oculta del modelo, es decir, trata sobre cómo podemos adivinar qué camino ha seguido la cadena de Markov. Este camino oculto se puede utilizar para clasificar las observaciones. En el caso de la etiquetación de un texto en lenguaje natural, dicho camino nos ofrece la secuencia de etiquetas más probable para las palabras del texto.

La primera pregunta representa el problema de la evaluación de una secuencia de observaciones dado el modelo. La resolución de este problema nos proporciona la probabilidad de que la secuencia haya sido generada por ese modelo. Un ejemplo práctico puede ser suponer la existencia de varios HMM,s compitiendo entre sí. La evaluación de la secuencia en cada uno de ellos nos permitirá elegir el modelo que mejor encaja con los datos observados.

A continuación se describen formalmente todos los algoritmos matemáticos que son necesarios para responder a estas tres preguntas.

4.4.1 Cálculo de la probabilidad de una observación

Dada una secuencia de observaciones $O = (o_1, o_2, \ldots, o_T)$ y un modelo $\mu = (\pi, A, B)$, queremos calcular de una manera eficiente $P(O|\mu)$, es decir, la probabilidad de dicha secuencia dado el modelo. La forma más directa de hacerlo es enumerando primero todas las posibles secuencias de estados de longitud T, el número de observaciones. Existen N^T secuencias distintas. Considerando una de esas secuencias, $S = (q_1, q_2, \ldots, q_T)$, la probabilidad de dicha secuencia de estados es

$$P(S|\mu) = \pi_{q_1} \, a_{q_1 q_2} \, a_{q_2 q_3} \ldots a_{q_{T-1} q_T} \tag{4.2}$$

y la probabilidad de observar O a través de la secuencia S es

$$P(O|S,\mu) = \prod_{t=1}^{T} P(o_t|q_t,\mu) = b_{q_1}(o_1) \, b_{q_2}(o_2) \ldots b_{q_T}(o_T) \tag{4.3}$$

La probabilidad conjunta de O y S, es decir, la probabilidad de que O y S ocurran simultáneamente, es simplemente el producto de (4.2) y (4.3), es decir,

$$P(O,S|\mu) = P(S|\mu) \, P(O|S,\mu)$$

Por tanto, la probabilidad de O dado el modelo se obtiene sumando esta probabilidad conjunta para todas las posibles secuencias S:

$$P(O|\mu) = \sum_{S} P(S|\mu) \, P(O|S,\mu).$$

Sin embargo, si calculamos $P(O|\mu)$ de esta forma, necesitamos realizar exactamente $(2T-1)N^T$ multiplicaciones y $N^T - 1$ sumas, es decir, un total de $2TN^T - 1$ operaciones. Estas cifras no son computacionalmente admisibles ni siquiera para valores pequeños de N y T. Por ejemplo, para $N = 5$ estados y $T = 100$ observaciones, el número de operaciones es del orden de 10^{72}. El secreto para evitar esta complejidad tan elevada está en utilizar técnicas de programación dinámica, con el fin de recordar los resultados parciales, en lugar de recalcularlos. A continuación se presentan varios procedimientos que calculan $P(O|\mu)$ utilizando dichas técnicas.

4.4.1.1 Procedimiento hacia adelante

Consideremos la variable $\alpha_t(i)$ definida como

$$\alpha_t(i) = P(o_1, o_2, \ldots, o_t, q_t = i|\mu),$$

es decir, la probabilidad conjunta de obtener $o_1, o_2, \ldots, o_t$, la secuencia parcial de observaciones hasta el instante de tiempo t, y de estar en el estado i en ese instante de tiempo t, dado el modelo μ. Los valores de $\alpha_t(i)$, para los distintos estados y para los distintos instantes de tiempo, se pueden obtener iterativamente, y pueden ser utilizados para calcular $P(O|\mu)$ mediante los pasos del siguiente algoritmo.

Algoritmo 4.1 Cálculo hacia adelante de la probabilidad de una secuencia de observaciones:

1. Inicialización:
$$\alpha_1(i) = \pi_i \, b_i(o_1), \qquad 1 \leq i \leq N.$$

2. Recurrencia:
$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) \, a_{ij} \right] b_j(o_{t+1}), \qquad t = 1, 2, \ldots, T-1, \qquad 1 \leq j \leq N. \tag{4.4}$$

3. Terminación:
$$P(O|\mu) = \sum_{i=1}^{N} \alpha_T(i).$$

El paso 1 inicializa las N probabilidades $\alpha_1(i)$ como la probabilidad conjunta de que i sea el estado inicial y genere la primera observación o_1. El paso de recurrencia, que es el punto central de este algoritmo, se ilustra en la figura 4.4 (a). Esta figura muestra cómo el estado j se puede alcanzar en el instante de tiempo $t+1$ desde los N posibles estados i del instante de tiempo t. Para cada uno de esos estados, dado que $\alpha_t(i)$ es la probabilidad conjunta de haber observado $o_1, o_2, \ldots, o_t$, y de estar en el estado i en el instante t, $\alpha_t(i) \, a_{ij}$ será la probabilidad conjunta de haber observado $o_1, o_2, \ldots, o_t$, y de haber alcanzado el estado j en el instante $t+1$ desde el estado i del instante t. Sumando todos estos productos sobre los N posibles estados i del instante t, y multiplicando esa suma por $b_j(o_{t+1})$, la probabilidad de que el estado j emita el símbolo o_{t+1}, obtenemos $\alpha_{t+1}(j)$, es decir, la probabilidad conjunta de obtener $o_1, o_2, \ldots, o_{t+1}$, la secuencia parcial de observaciones hasta el instante de tiempo $t+1$, y de estar en el estado j en ese instante de tiempo $t+1$. El cálculo de la ecuación (4.4) se realiza para los N estados j de un mismo instante de tiempo t, y para todos los instantes de tiempo $t = 1, 2, \ldots, T-1$. Finalmente, el paso 3 obtiene $P(O|\mu)$ sumando el valor de las N variables $\alpha_T(i)$. □

Los cálculos globales involucrados en este proceso requieren del orden de N^2T operaciones, tal y como se muestra en el enrejado de la figura 4.4 (b). Más concretamente, se trata de $N(N+1)(T-1)+N$ multiplicaciones y $N(N-1)(T-1)+N-1$ sumas, es decir, un total de $2(T-1)N^2+2N-1$ operaciones. Para $N=5$ estados y $T=100$ observaciones, el número total de operaciones es aproximadamente 5.000, que comparado con 10^{72} supone un ahorro de 69 órdenes de magnitud.

4.4.1.2 Procedimiento hacia atrás

De manera similar, podemos considerar la variable $\beta_t(i)$ definida como
$$\beta_t(i) = P(o_{t+1}, o_{t+2}, \ldots, o_T | q_t = i, \mu),$$
es decir, la probabilidad de la secuencia de observación parcial desde el instante de tiempo $t+1$ hasta el final, dado que el estado en el instante de tiempo t es i y dado el modelo μ. Nuevamente, podemos resolver las variables $\beta_t(i)$ y calcular el valor de $P(O|\mu)$ de manera recurrente mediante los pasos del siguiente algoritmo.

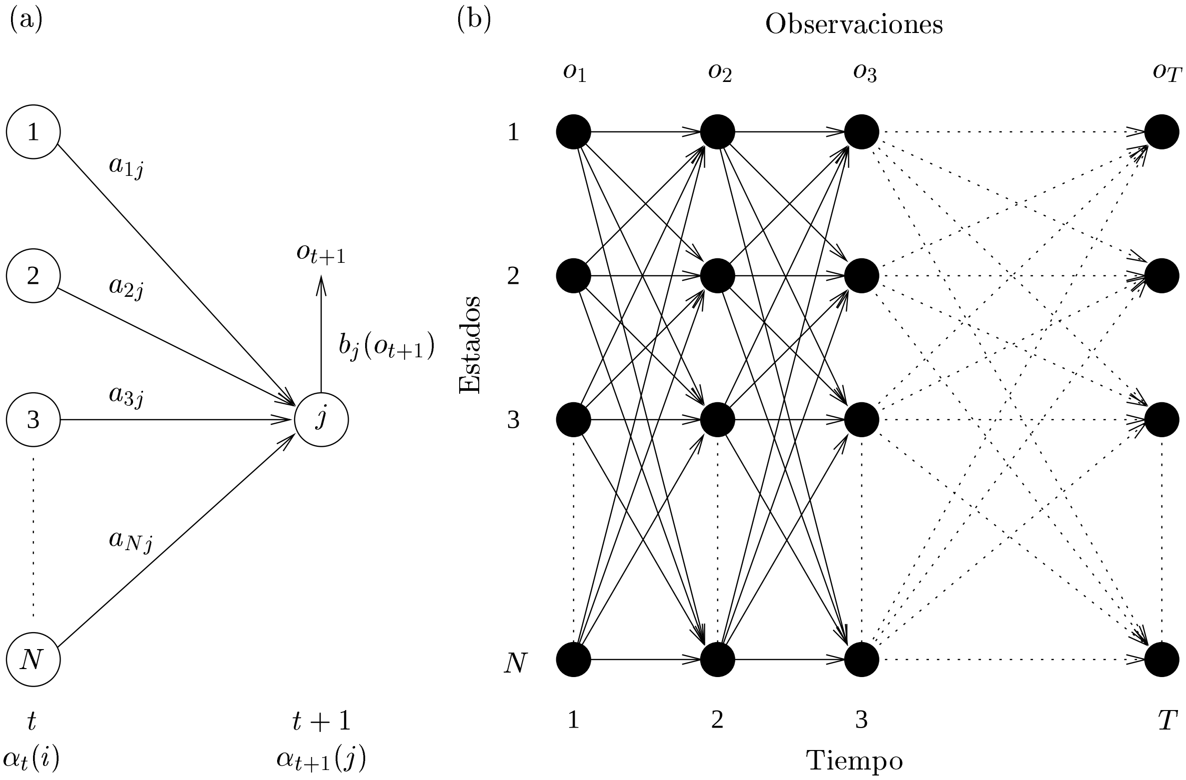


Figura 4.4: (a) Detalle de la secuencia de operaciones necesarias para el cálculo hacia adelante de la variable $\alpha_{t+1}(j)$ y (b) implementación genérica del cálculo hacia adelante de la variable $\alpha_t(i)$ mediante un enrejado de T observaciones y N estados

Algoritmo 4.2 Cálculo hacia atrás de la probabilidad de una secuencia de observaciones:

1. Inicialización:
$$\beta_T(i) = 1, \qquad 1 \leq i \leq N.$$

2. Recurrencia:
$$\beta_t(i) = \sum_{j=1}^{N} a_{ij}\,\beta_{t+1}(j)\,b_j(o_{t+1}), \qquad t = T-1, T-2, \ldots, 1, \qquad 1 \leq i \leq N.$$

3. Terminación:
$$P(O|\mu) = \sum_{i=1}^{N} \beta_1(i)\,\pi_i\,b_i(o_1).$$

El paso de inicialización asigna 1 a las N variables $\beta_T(i)$. El paso de recurrencia, tal y como se muestra en la figura 4.5, calcula el valor de $\beta_t(i)$ en función de las N variables $\beta_{t+1}(j)$ del instante siguiente, en función de las probabilidades de transición entre estados a_{ij}, y en función de las probabilidades de emisión del símbolo o_{t+1} desde los N estados j. Finalmente, el paso 3 obtiene $P(O|\mu)$ sumando el valor de las N variables $\beta_1(i)$ multiplicado por la probabilidad de que el estado i sea el estado inicial y por la probabilidad de que emita el primer símbolo de observación o_1. □

Una vez más, el cálculo de las variables $\beta_t(i), 1 \leq t \leq T, 1 \leq i \leq N$, requiere del orden de N^2T operaciones y se puede resolver sobre un enrejado similar al de la figura 4.4 (b). Por tanto,

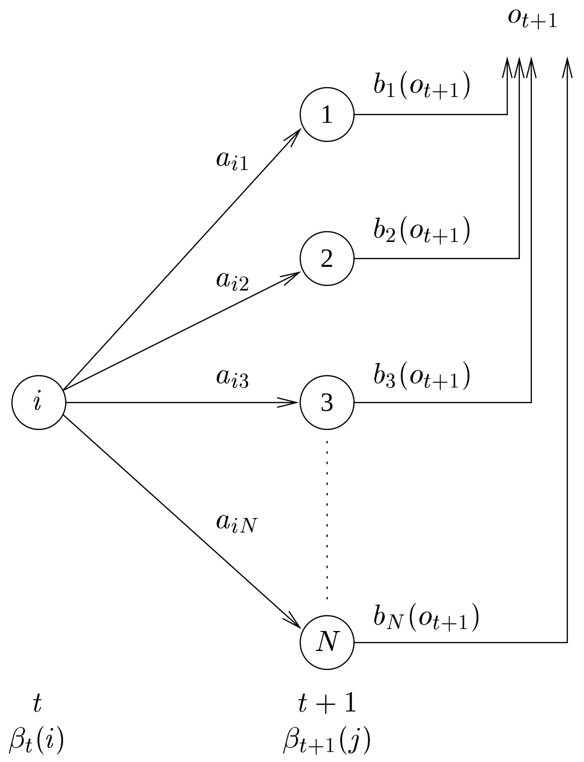


Figura 4.5: Detalle de la secuencia de operaciones necesarias para el cálculo hacia atrás de $\beta_t(i)$

la verdadera razón para haber introducido este procedimiento hacia atrás es que los cálculos involucrados en él son de vital importancia para resolver las preguntas fundamentales 2 y 3 sobre los HMM,s, es decir, el cálculo de la secuencia de estados óptima y la estimación de los parámetros del modelo, tal y como veremos a continuación.

4.4.2 Elección de la secuencia de estados más probable

El segundo problema ha sido definido vagamente como el problema de encontrar la secuencia de estados óptima que mejor *explica* las observaciones. Debido a esta definición informal del problema, podrían existir varias formas de abordarlo, es decir, se podrían considerar diferentes criterios para esa optimación. Uno de ellos podría ser la elección de los estados que son *individualmente* más probables en cada instante de tiempo. Para implementar este criterio, podemos considerar la cantidad $\gamma_t(i)$ definida como

$$\gamma_t(i) = P(q_t = i|O, \mu), \tag{4.5}$$

es decir, la probabilidad de estar en el estado i en el instante t, dada la secuencia de obsevaciones O y dado el modelo μ. Dicha cantidad se puede obtener a partir de

$$\gamma_t(i) = P(q_t = i|O, \mu) = \frac{P(q_t = i, O|\mu)}{P(O|\mu)} = \frac{P(q_t = i, O|\mu)}{\displaystyle\sum_{j=1}^{N} P(q_t = j, O|\mu)} = \frac{\alpha_t(i)\,\beta_t(i)}{\displaystyle\sum_{j=1}^{N} \alpha_t(j)\,\beta_t(j)} \tag{4.6}$$

donde $\alpha_t(i)$ almacena la probabilidad de la observación parcial $o_1, o_2, \ldots, o_t$, y $\beta_t(i)$ almacena la probabilidad de la observación parcial $o_{t+1}, o_{t+2}, \ldots, o_T$, dado el estado i en el instante t.

Utilizando $\gamma_t(i)$ podemos obtener q_t^*, el estado individualmente más probable en el instante t, como

$$q_t^* = \arg \max_{1 \leq i \leq N} [\gamma_t(i)], \qquad 1 \leq t \leq T. \tag{4.7}$$

La ecuación (4.7) maximiza el número esperado de estados correctos, eligiendo el estado más probable en cada instante t. Sin embargo, podrían surgir problemas con la secuencia de estados resultante. Por ejemplo, en un HMM con alguna transición entre estados de probabilidad cero ($a_{ij} = 0$ para algún i y algún j), podría ocurrir que esos estados i y j aparecieran contiguos en la secuencia óptima, cuando de hecho esa secuencia ni siquiera sería una secuencia válida. Esto es debido a que la solución que nos proporciona la ecuación (4.7) determina simplemente el estado más probable en cada instante, sin tener en cuenta la probabilidad de la secuencia de estados resultante.

Una posible solución al problema anterior es modificar el criterio de optimalidad para considerar, por ejemplo, la secuencia que maximiza el número esperado de pares de estados correctos (q_{t-1}, q_t), o de triplas de estados correctos (q_{t-2}, q_{t-1}, q_t), etc. Aunque estos criterios podrían ser razonables para algunas aplicaciones, el criterio más ampliamente utilizado consiste en encontrar la mejor secuencia considerando globalmente todos los instantes de tiempo, es decir, la secuencia de estados $S = (q_1, q_2, \ldots, q_T)$ que maximiza $P(S|O, \mu)$, lo cual es equivalente a maximizar $P(S, O|\mu)$. Existe un procedimiento formal y eficiente, basado también en técnicas de programación dinámica, para obtener esa secuencia S. Dicho procedimiento es el algoritmo de Viterbi [Viterbi 1967, Forney 1973].

4.4.2.1 Algoritmo de Viterbi

Para encontrar la secuencia de estados más probable, $S = (q_1, q_2, \ldots, q_T)$, dada la observación $O = (o_1, o_2, \ldots, o_T)$, consideramos la variable $\delta_t(i)$ definida como

$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P(q_1, q_2, \ldots, q_{t-1}, q_t = i, o_1, o_2, \ldots, o_t|\mu),$$

es decir, $\delta_t(i)$ almacena la probabilidad del mejor camino que termina en el estado i, teniendo en cuenta las t primeras observaciones. Se demuestra fácilmente que

$$\delta_{t+1}(j) = \left[\max_{1 \leq i \leq N} \delta_t(i)\, a_{ij} \right] b_j(o_{t+1}). \tag{4.8}$$

Una vez calculadas las $\delta_t(i)$ para todos los estados y para todos los instantes de tiempo, la secuencia de estados se construye realmente hacia atrás a través de una traza que recuerda el argumento que maximizó la ecuación (4.8) para cada instante t y para cada estado j. Esta traza se almacena en las correspondientes variables $\psi_t(j)$. La descripción completa del algoritmo es como sigue.

Algoritmo 4.3 Cálculo de la secuencia de estados más probable para una secuencia de observaciones dada (algoritmo de Viterbi):

1. Inicialización:

$$\delta_1(i) = \pi_i\, b_i(o_1), \qquad 1 \leq i \leq N.$$

2. Recurrencia:

$$\delta_{t+1}(j) = \left[\max_{1 \leq i \leq N} \delta_t(i)\, a_{ij} \right] b_j(o_{t+1}), \qquad t = 1, 2, \ldots, T-1, \qquad 1 \leq j \leq N. \tag{4.9}$$

$$\psi_{t+1}(j) = \arg \max_{1 \leq i \leq N} \delta_t(i)\, a_{ij}, \qquad t = 1, 2, \ldots, T-1, \qquad 1 \leq j \leq N.$$

3. Terminación:
$$q_T^* = \arg \max_{1 \leq i \leq N} \delta_T(i).$$

4. Construcción hacia atrás de la secuencia de estados:
$$q_t^* = \psi_{t+1}(q_{t+1}^*), \qquad t = T-1, T-2, \ldots, 1.$$

El algoritmo de Viterbi es similar al cálculo hacia adelante de la probabilidad de una observación que vimos en el algoritmo 4.1. Las únicas diferencias reseñables son que el sumatorio de la ecuación (4.4) se ha cambiado por la maximización de la ecuación (4.9), y que se ha añadido el paso final para construir hacia atrás la secuencia de estados. En todo caso, la complejidad del algoritmo es del orden de N^2T operaciones y se puede resolver también sobre un enrejado similar al de la figura 4.4 (b). □

Por supuesto, durante los cálculos del algoritmo de Viterbi se podrían obtener empates. Si esto ocurre, la elección del camino se realizaría aleatoriamente. Por otra parte, existen a menudo aplicaciones prácticas en las cuales se utiliza no sólo la mejor secuencia de estados, sino las n mejores secuencias. Para todos estos casos, se podría considerar una implementación del algoritmo de Viterbi que devolviera varias secuencias de estados.

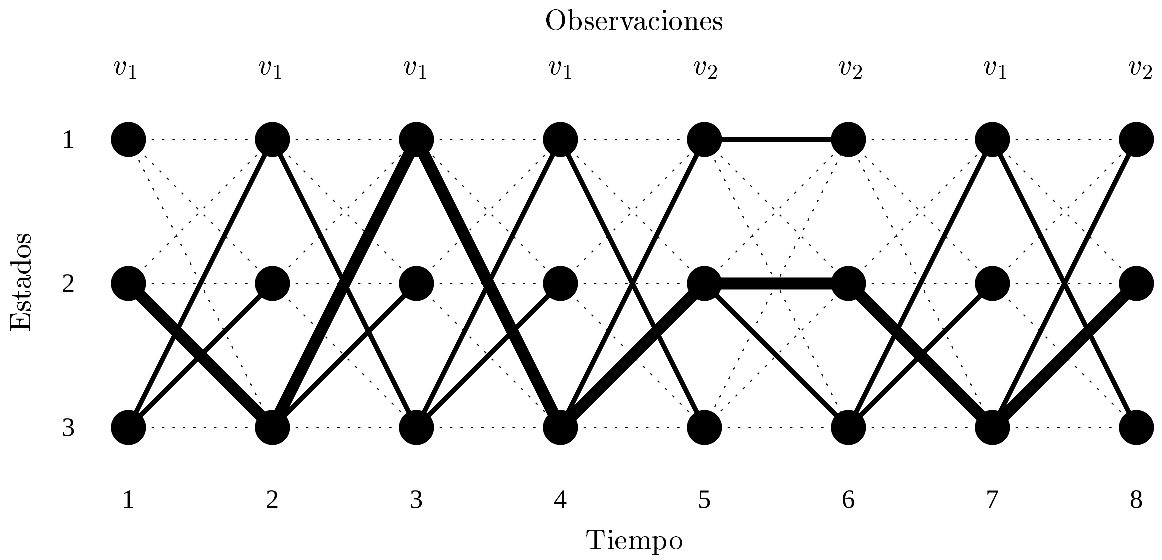


Figura 4.6: Ejemplo de ejecución del algoritmo de Viterbi

Ejemplo 4.4 A continuación presentamos un ejemplo de ejecución del algoritmo de Viterbi. Supongamos un HMM con tres estados y dos símbolos de observación, $\mu = (Q, V, \pi, A, B)$, donde
$$Q = \{1, 2, 3\}, \qquad V = \{v_1, v_2\}, \qquad \pi = \begin{bmatrix} 0,25 \\ 0,50 \\ 0,25 \end{bmatrix},$$
$$A = \begin{bmatrix} 0,25 & 0,25 & 0,50 \\ 0 & 0,25 & 0,75 \\ 0,50 & 0,50 & 0 \end{bmatrix} \quad y \quad B = \begin{bmatrix} 0,50 & 0,50 \\ 0,25 & 0,75 \\ 0,75 & 0,25 \end{bmatrix}.$$
Los cálculos para encontrar la secuencia de estados más probable dada la observación
$$O = (v_1, v_1, v_1, v_1, v_2, v_2, v_1, v_2)$$

de longitud $T = 8$, son los siguientes:

$$\begin{aligned}
\delta_1(1) &= \pi_1\, b_1(v_1) = (0,25)(0,50)\\
\delta_1(2) &= \pi_2\, b_2(v_1) = (0,50)(0,25)\\
\delta_1(3) &= \pi_3\, b_3(v_1) = (0,25)(0,75)
\end{aligned}$$

$$\begin{aligned}
\delta_2(1) &= \max\,[\delta_1(1)\,a_{11}, \delta_1(2)\,a_{21}, \underline{\delta_1(3)\,a_{31}}]\; b_1(v_1) = (0,25)\,(0,50)^2\,(0,75) && \psi_2(1) = 3\\
\delta_2(2) &= \max\,[\delta_1(1)\,a_{12}, \delta_1(2)\,a_{22}, \underline{\delta_1(3)\,a_{32}}]\; b_2(v_1) = (0,25)^2\,(0,50)\,(0,75) && \psi_2(2) = 3\\
\delta_2(3) &= \max\,[\delta_1(1)\,a_{13}, \underline{\delta_1(2)\,a_{23}}, \delta_1(3)\,a_{33}]\; b_3(v_1) = (0,25)\,(0,50)\,(0,75)^2 && \psi_2(3) = 2
\end{aligned}$$

$$\begin{aligned}
\delta_3(1) &= \max\,[\delta_2(1)\,a_{11}, \delta_2(2)\,a_{21}, \underline{\delta_2(3)\,a_{31}}]\; b_1(v_1) = (0,25)\,(0,50)^3\,(0,75)^2 && \psi_3(1) = 3\\
\delta_3(2) &= \max\,[\delta_2(1)\,a_{12}, \delta_2(2)\,a_{22}, \underline{\delta_2(3)\,a_{32}}]\; b_2(v_1) = (0,25)^2\,(0,50)^2\,(0,75)^2 && \psi_3(2) = 3\\
\delta_3(3) &= \max\,[\underline{\delta_2(1)\,a_{13}}, \delta_2(2)\,a_{23}, \delta_2(3)\,a_{33}]\; b_3(v_1) = (0,25)\,(0,50)^3\,(0,75)^2 && \psi_3(3) = 1
\end{aligned}$$

$$\begin{aligned}
\delta_4(1) &= \max\,[\delta_3(1)\,a_{11}, \delta_3(2)\,a_{21}, \underline{\delta_3(3)\,a_{31}}]\; b_1(v_1) = (0,25)\,(0,50)^5\,(0,75)^2 && \psi_4(1) = 3\\
\delta_4(2) &= \max\,[\delta_3(1)\,a_{12}, \delta_3(2)\,a_{22}, \underline{\delta_3(3)\,a_{32}}]\; b_2(v_1) = (0,25)^2\,(0,50)^4\,(0,75)^2 && \psi_4(2) = 3\\
\delta_4(3) &= \max\,[\underline{\delta_3(1)\,a_{13}}, \delta_3(2)\,a_{23}, \delta_3(3)\,a_{33}]\; b_3(v_1) = (0,25)\,(0,50)^4\,(0,75)^3 && \psi_4(3) = 1
\end{aligned}$$

$$\begin{aligned}
\delta_5(1) &= \max\,[\delta_4(1)\,a_{11}, \delta_4(2)\,a_{21}, \underline{\delta_4(3)\,a_{31}}]\; b_1(v_2) = (0,25)\,(0,50)^6\,(0,75)^3 && \psi_5(1) = 3\\
\delta_5(2) &= \max\,[\delta_4(1)\,a_{12}, \delta_4(2)\,a_{22}, \underline{\delta_4(3)\,a_{32}}]\; b_2(v_2) = (0,25)\,(0,50)^5\,(0,75)^4 && \psi_5(2) = 3\\
\delta_5(3) &= \max\,[\underline{\delta_4(1)\,a_{13}}, \delta_4(2)\,a_{23}, \delta_4(3)\,a_{33}]\; b_3(v_2) = (0,25)^2\,(0,50)^6\,(0,75)^2 && \psi_5(3) = 1
\end{aligned}$$

$$\begin{aligned}
\delta_6(1) &= \max\,[\underline{\delta_5(1)\,a_{11}}, \delta_5(2)\,a_{21}, \delta_5(3)\,a_{31}]\; b_1(v_2) = (0,25)^2\,(0,50)^7\,(0,75)^3 && \psi_6(1) = 1\\
\delta_6(2) &= \max\,[\delta_5(1)\,a_{12}, \underline{\delta_5(2)\,a_{22}}, \delta_5(3)\,a_{32}]\; b_2(v_2) = (0,25)^2\,(0,50)^5\,(0,75)^5 && \psi_6(2) = 2\\
\delta_6(3) &= \max\,[\delta_5(1)\,a_{13}, \underline{\delta_5(2)\,a_{23}}, \delta_5(3)\,a_{33}]\; b_3(v_2) = (0,25)^2\,(0,50)^5\,(0,75)^5 && \psi_6(3) = 2
\end{aligned}$$

$$\begin{aligned}
\delta_7(1) &= \max\,[\delta_6(1)\,a_{11}, \delta_6(2)\,a_{21}, \underline{\delta_6(3)\,a_{31}}]\; b_1(v_1) = (0,25)^2\,(0,50)^7\,(0,75)^5 && \psi_7(1) = 3\\
\delta_7(2) &= \max\,[\delta_6(1)\,a_{12}, \delta_6(2)\,a_{22}, \underline{\delta_6(3)\,a_{32}}]\; b_2(v_1) = (0,25)^3\,(0,50)^6\,(0,75)^5 && \psi_7(2) = 3\\
\delta_7(3) &= \max\,[\delta_6(1)\,a_{13}, \underline{\delta_6(2)\,a_{23}}, \delta_6(3)\,a_{33}]\; b_3(v_1) = (0,25)^2\,(0,50)^5\,(0,75)^7 && \psi_7(3) = 2
\end{aligned}$$

$$\begin{aligned}
\delta_8(1) &= \max\,[\delta_7(1)\,a_{11}, \delta_7(2)\,a_{21}, \underline{\delta_7(3)\,a_{31}}]\; b_1(v_2) = (0,25)^2\,(0,50)^7\,(0,75)^7 && \psi_8(1) = 3\\
\delta_8(2) &= \max\,[\delta_7(1)\,a_{12}, \delta_7(2)\,a_{22}, \underline{\delta_7(3)\,a_{32}}]\; b_2(v_2) = (0,25)^2\,(0,50)^6\,(0,75)^8 && \psi_8(2) = 3\\
\delta_8(3) &= \max\,[\underline{\delta_7(1)\,a_{13}}, \delta_7(2)\,a_{23}, \delta_7(3)\,a_{33}]\; b_3(v_2) = (0,25)^3\,(0,50)^8\,(0,75)^5 && \psi_8(3) = 1
\end{aligned}$$

En cada paso aparecen subrayados los términos máximos. El valor de i en cada uno de esos términos es el valor que se asigna a cada $\psi_t(j)$. La probabilidad máxima para la secuencia de observaciones completa se alcanza en $\delta_8(2)$, lo cual implica que $q_8^* = 2$, y al reconstruir hacia atrás la secuencia de estados obtenemos

$$S = (2, 3, 1, 3, 2, 2, 3, 2)$$

tal y como se puede observar también en el enrejado de la figura 4.6. En dicha figura, se han representado todos los caminos posibles mediante líneas de puntos. Posteriormente, para todos los instantes de tiempo t, excepto el primero, cada estado j se une con una línea continua al estado del instante anterior que indique $\psi_t(j)$. Por ejemplo, el estado 1 del instante de tiempo 4 aparece unido con una línea continua con el estado 3 del instante 3, ya que $\psi_4(1) = 3$. La explicación intuitiva de esta línea es la siguiente: aún no sabemos si el camino más probable pasará por el estado 1 del instante 4, pero si lo hace, entonces sabemos que pasará también por el estado 3 del instante 3. Esta *traza* se mantiene hasta el final para todos los estados. Y por último, cuando vemos que la probabilidad máxima en el instante 8 correponde al estado 2, reconstruimos el camino hacia atrás desde ese punto para obtener la secuencia de estados más probable, que es la que aparece marcada con una línea continua más gruesa. □

Para el caso que nos ocupa, que es el de la etiquetación de palabras, los cálculos involucrados en el algoritmo de Viterbi se realizan frase por frase sobre enrejados simplificados como el de la figura 4.7, donde en cada posición no se consideran todos los estados posibles, es decir, todas la etiquetas del juego de etiquetas utilizado, sino sólo las etiquetas candidatas que proponga el diccionario para cada palabra.

No obstante, como se puede observar, hemos añadido un estado especial, que denominaremos estado 0 ó `etiqueta 0`, para marcar el comienzo y el fin de frase. Conceptualmente, el propósito real de este nuevo estado es garantizar que el etiquetador simula un sistema que funciona indefinidamente en el tiempo, sin detenerse, tal y como exige el paradigma de los modelos de Markov. En la práctica, una vez que nuestro etiquetador está funcionando, la justificación coloquial es que si tenemos que etiquetar un conjunto de frases, podemos procesar unas cuantas, detener el proceso, apagar el ordenador, encenderlo al día siguiente, arrancar de nuevo el proceso, etiquetar el resto de frases, y el resultado obtenido será el mismo que si las frases hubieran sido etiquetadas todas juntas en bloque.

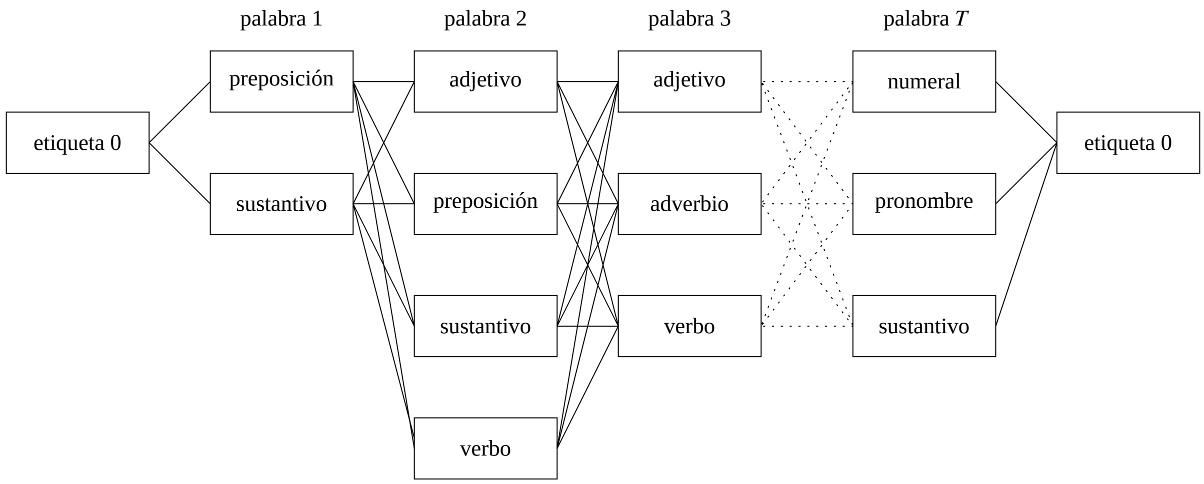


Figura 4.7: Enrejado simplificado para la etiquetación de una frase de T palabras

Los experimentos realizados por Merialdo sugieren que no existe una gran diferencia de éxito entre maximizar la probabilidad de cada etiqueta individualmente y maximizar la probabilidad de la secuencia completa, tal y como hace el algoritmo de Viterbi [Merialdo 1994]. Intuitivamente, esto es sencillo de comprender. Con el algoritmo de Viterbi, las transiciones entre estados o etiquetas son más sensibles, pero si algo va mal, se pueden obtener secuencias con varias etiquetas incorrectas. Maximizando etiqueta por etiqueta, esto no ocurre. Un fallo no desencadena otros fallos, de manera que lo que se obtiene son secuencias con errores ocasionales dispersos. No obstante, en la práctica, el método preferido para los etiquetadores basados en HMM,s sigue siendo el algoritmo de Viterbi.

Otra cuestión que podemos considerar es si el proceso de etiquetación resulta ser reversible o no, es decir, si etiquetando las frases de izquierda a derecha se obtienen los mismos resultados que etiquetando de derecha a izquierda. Si revisamos la implementación de los algoritmos de cálculo hacia adelante y hacia atrás de la probabilidad de una secuencia de observaciones dada (algoritmos 4.1 y 4.2), veremos que ambos obtienen la misma probabilidad. Así que, en definitiva, el proceso de etiquetación no es dependiente del sentido elegido. El algoritmo de Viterbi que hemos descrito aquí se basa en el algoritmo hacia adelante y por tanto etiqueta de izquierda a derecha, mientras que el etiquetador de Church, por ejemplo, lo hace en sentido contrario [Church 1988].

Sin embargo, lo que sí es importante es que hasta ahora hemos modelizado el proceso de la etiquetación del lenguaje natural utilizando siempre HMM,s de orden 1, es decir, modelos en los que la dependencia contextual de una etiqueta se establece solamente en relación a la etiqueta anterior. A este tipo de modelos se les denomina también modelos de *bigramas* de etiquetas. Si se quiere trabajar con un HMM de orden superior, por ejemplo de orden 2, sería necesario extender los cálculos del algoritmo de Viterbi para considerar las transiciones de estados desde dos posiciones antes de la etiqueta actual, tal y como podemos ver a continuación.

Algoritmo 4.4 Cálculo de la secuencia de estados más probable para una secuencia de observaciones dada (algoritmo de Viterbi para HMM,s de orden 2):

1. Inicialización:
$$\delta_1(i) = \pi_i \, b_i(o_1), \qquad 1 \leq i \leq N.$$
$$\delta_2(i,j) = \delta_1(i) \, a_{ij} \, b_j(o_2), \qquad 1 \leq i,j \leq N. \tag{4.10}$$
2. Recurrencia:
$$\delta_{t+1}(j,k) = \left[\max_{1 \leq i \leq N} \delta_t(i,j) \, a_{ijk} \right] b_k(o_{t+1}), \qquad t = 2,3,\ldots,T-1, \qquad 1 \leq j,k \leq N.$$
$$\psi_{t+1}(j,k) = \arg \max_{1 \leq i \leq N} \delta_t(i,j) \, a_{ijk}, \qquad t = 2,3,\ldots,T-1, \qquad 1 \leq j,k \leq N.$$
3. Terminación:
$$(q^*_{T-1}, q^*_T) = \arg \max_{1 \leq j,k \leq N} \delta_T(j,k).$$
4. Construcción hacia atrás de la secuencia de estados:
$$q^*_t = \psi_{t+2}(q^*_{t+1}, q^*_{t+2}), \qquad t = T-2, T-3, \ldots, 1.$$

Obsérvese que resulta necesario almacenar tanto $a_{ij} = P(j|i)$, la distribución de probabilidad de las transiciones entre cada posible par de estados, utilizada en la ecuación (4.10), segundo paso de inicialización, como $a_{ijk} = P(k|i,j)$, la distribución de probabilidad de las transiciones entre cada posible trío de estados, utilizada en el resto de pasos del algoritmo. □

Sin embargo, en lugar de realizar esta extensión, resulta mucho más sencillo cambiar el espacio de estados del modelo por la n-ésima potencia del conjunto de etiquetas, donde n es el orden del HMM. De esta manera, es posible realizar una única implementación del algoritmo de Viterbi que sirva para cualquier orden n, con sólo introducir dicho orden como un parámetro del algoritmo. Por ejemplo, en el caso de los HMM,s de orden 2, el conjunto de estados del modelo se define como el producto cartesiano del conjunto de etiquetas, siendo válidas las transiciones entre un estado (t^i, t^l) y otro (t^m, t^k), donde todas las t^j son etiquetas, sólo cuando $l = m$. En cualquier caso, esto es lo que se conoce como modelo de *trigramas* de etiquetas, ampliamente referenciado y utilizado por la mayoría de los etiquetadores estocásticos.

Por supuesto, cabría pensar que el hecho de aumentar el orden de un HMM podría desembocar en una mejora de la representación del contexto necesario para la correcta etiquetación de las palabras. Es decir, ¿por qué considerar sólo 2 etiquetas hacia atrás, en lugar de 3, ó 4, ó incluso más? Esto no se lleva a cabo en la práctica porque las unidades de información mayores que el trigrama generalmente modelizan fenómenos lingüísticos muy complejos y que afectan a muy pocos idiomas, resultando por ello suficientemente adecuado el uso de los trigramas. Por otra parte, el gran problema de aumentar el orden de un HMM es que crece desorbitadamente el número de parámetros que es necesario estimar para hacer operativo el modelo, tal y como veremos más adelante.

4.4.2.2 Implementaciones alternativas del algoritmo de Viterbi

Dado que el algoritmo de Viterbi no calcula una probabilidad exacta, sino que construye una secuencia de estados, podemos tomar logaritmos sobre los parámetros del modelo e implementarlo sólo con sumas, sin necesidad de ninguna multiplicación. De esta manera, no sólo se evita el problema de la rápida pérdida de precisión debida a las multiplicaciones de números muy pequeños, sino que además la velocidad de ejecución aumenta ya que las sumas se realizan más rápido que las multiplicaciones. En la práctica, es particularmente importante disponer de una implementación muy eficiente de este algoritmo, porque es el que realmente etiqueta las palabras que aparecen en los textos, mientras que el proceso de estimación de los parámetros del modelo, que veremos en la siguiente sección, puede realizarse de manera previa y separada, y por tanto su velocidad no es tan crítica. Los pasos del nuevo algoritmo son los siguientes.

Algoritmo 4.5 Cálculo de la secuencia de estados más probable para una secuencia de observaciones dada (algoritmo de Viterbi con logaritmos y sumas):

0. Preproceso:

$$\begin{array}{ll} \tilde{\pi}_i = \log(\pi_i), & 1 \leq i \leq N. \\ \tilde{a}_{ij} = \log(a_{ij}), & 1 \leq i,j \leq N. \\ \tilde{b}_i(o_t) = \log[b_i(o_t)], & 1 \leq i \leq N,\ 1 \leq t \leq T. \end{array}$$

1. Inicialización:

$$\tilde{\delta}_1(i) = \log[\delta_1(i)] = \tilde{\pi}_i + \tilde{b}_i(o_1), \qquad 1 \leq i \leq N.$$

2. Recurrencia:

$$\tilde{\delta}_{t+1}(j) = \log[\delta_{t+1}(j)] = \left[\max_{1 \leq i \leq N} [\tilde{\delta}_t(i) + \tilde{a}_{ij}]\right] + \tilde{b}_j(o_{t+1}), \quad t = 1, 2, \ldots, T-1, \quad 1 \leq j \leq N.$$

$$\psi_{t+1}(j) = \arg \max_{1 \leq i \leq N} [\tilde{\delta}_t(i) + \tilde{a}_{ij}], \qquad t = 1, 2, \ldots, T-1, \qquad 1 \leq j \leq N.$$

3. Terminación:

$$q_T^* = \arg \max_{1 \leq i \leq N} \tilde{\delta}_T(i).$$

4. Construcción hacia atrás de la secuencia de estados:

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \qquad t = T-1, T-2, \ldots, 1.$$

Esta implementación alternativa requiere del orden de N^2T sumas, más los cálculos necesarios para el paso de preproceso. No obstante, dado que el paso de preproceso normalmente se realiza una sola vez y se almacenan los resultados, su coste es despreciable en la mayoría de las aplicaciones. □

Por otra parte, el tiempo de procesamiento del algoritmo de Viterbi también se puede reducir introduciendo una búsqueda con corte[4]: cada estado que recibe un valor δ menor que el δ máximo hasta ese instante dividido por un cierto umbral θ queda excluido de los cálculos. Por supuesto, la incorporación de esta búsqueda con corte ya no garantiza que el algoritmo de Viterbi encuentre la secuencia de estados de máxima probabilidad. Sin embargo, para propósitos prácticos y con una buena elección del umbral θ, no existe virtualmente ninguna diferencia de precisión entre los algoritmos con y sin corte. Empíricamente, un valor de $\theta = 1.000$ puede aproximadamente doblar la velocidad de un etiquetador sin afectar a la precisión [Brants 2000].

[4] *Beam search* o *purging*.

4.4.3 Estimación de los parámetros del modelo

Dada una secuencia de observaciones O, el tercer problema fundamental relativo a los HMM,s consiste en determinar los parámetros del modelo que mejor *explican* los datos observados. Para el caso que nos ocupa, que es el de la etiquetación, dichas observaciones se corresponden con las palabras de un texto en lenguaje natural. En general, existen dos posibles métodos de estimación claramente diferenciados, en función de si el texto que observamos ha sido ya previamente etiquetado o no.

Con el fin de seguir el orden cronológico en el que han ido apareciendo las distintas técnicas, consideraremos primero el caso en el que el texto que observamos no está etiquetado. En este caso, es posible realizar un proceso de estimación, que denominaremos *no visible* o no supervisado, mediante el algoritmo de Baum-Welch. Sin embargo, veremos que un texto no etiquetado por sí solo no es suficiente para entrenar el modelo. Es necesario contar también con la ayuda de otro recurso lingüístico en forma de diccionario, que permita obtener un buen punto de inicialización para dicho algoritmo. Posteriormente, a medida que aparecieron disponibles los textos etiquetados, fue surgiendo también otro tipo de proceso de estimación que denominaremos *visible* o supervisado.

4.4.3.1 Estimación no supervisada: algoritmo de Baum-Welch

Cuando se dispone de una observación O, formada por un texto que no ha sido previamente etiquetado, este tercer problema de la estimación de los parámetros es el más complejo, ya que no se conoce ningún método analítico definitivo para encontrar un modelo $\mu = (\pi, A, B)$ que maximice $P(O|\mu)$. Sin embargo, podemos elegir un modelo que maximice localmente dicha probabilidad mediante un procedimiento iterativo tal como el algoritmo de Baum-Welch, que es un caso especial del algoritmo EM[5] [Dempster *et al.* 1977].

La idea intuitiva del algoritmo de Baum-Welch es la siguiente. En un primer momento, no sabemos cómo es el modelo, pero podemos trabajar sobre la probabilidad de la secuencia de observaciones utilizando algún modelo inicial, quizás preseleccionado o simplemente elegido de forma aleatoria. A partir de ese cálculo, identificamos qué transiciones y qué símbolos de emisión son los más probables. Incrementando la probabilidad de esas transiciones y de esos símbolos, construimos un modelo revisado, el cual obtendrá una probabilidad mayor que el modelo anterior para la secuencia de observaciones dada. Este proceso de maximización, denominado normalmente *proceso de entrenamiento*, se repite un cierto número de veces. Finalmente, el algoritmo se detiene cuando no consigue construir un modelo que mejore la probabilidad de la secuencia de observaciones dada.

Para describir formalmente este procedimiento, definimos primero $\xi_t(i,j)$, la probabilidad de estar en el estado i en el instante t y en el estado j en el instante $t+1$, dada la observación y dado el modelo, es decir,

$$\xi_t(i,j) = P(q_t = i, q_{t+1} = j | O, \mu). \tag{4.11}$$

Los caminos que satisfacen las condiciones requeridas por la ecuación (4.11) son los que se muestran en la figura 4.8. Por tanto, a partir de las definiciones de las variables hacia adelante y hacia atrás, podemos escribir $\xi_t(i,j)$ de la forma

$$\xi_t(i,j) = \frac{P(q_t = i, q_{t+1} = j, O|\mu)}{P(O|\mu)} = \frac{\alpha_t(i)\, a_{ij}\, b_j(o_{t+1})\, \beta_{t+1}(j)}{P(O|\mu)} = \frac{\alpha_t(i)\, a_{ij}\, b_j(o_{t+1})\, \beta_{t+1}(j)}{\displaystyle\sum_{k=1}^{N}\sum_{l=1}^{N} \alpha_t(k)\, a_{kl}\, b_l(o_{t+1})\, \beta_{t+1}(l)}.$$

[5] *Expectation-Maximization* (maximización de la esperanza); la parte E del algoritmo de Baum-Welch se conoce también como algoritmo *forward-backward* (hacia adelante y hacia atrás) [Baum 1972].

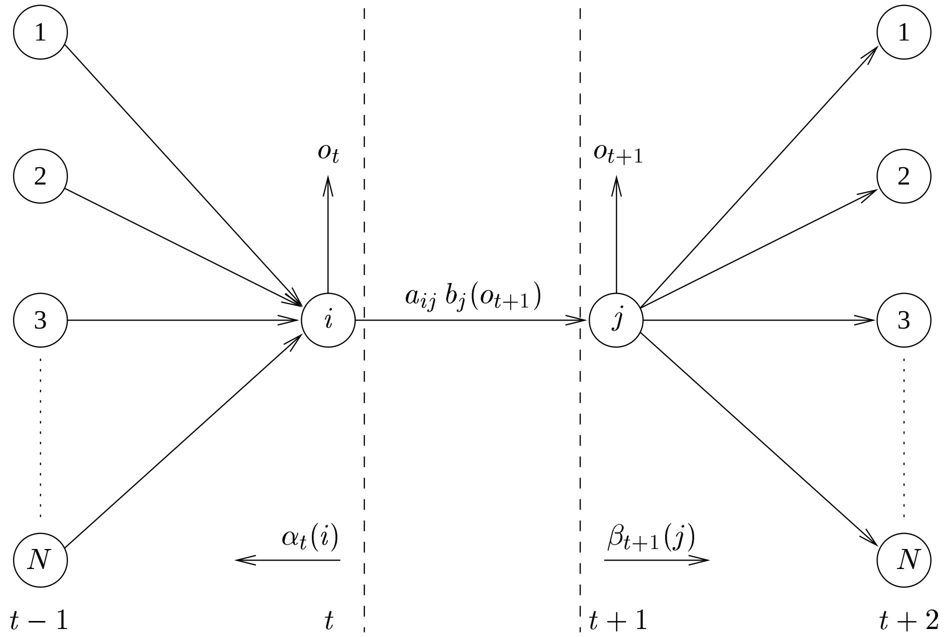


Figura 4.8: Detalle de la secuencia de operaciones necesarias para el cálculo de la probabilidad conjunta de que el sistema esté en el estado i en el instante t y en el estado j en el instante $t+1$

Previamente, en las ecuaciones (4.5) y (4.6), habíamos definido también $\gamma_t(i)$ como la probabilidad de estar en el estado i en el instante t, dada la secuencia de obsevaciones y dado el modelo. Por tanto, podemos relacionar $\gamma_t(i)$ con $\xi_t(i,j)$ mediante un sumatorio que recorre todo el espacio de estados sobre el índice j, dejando fijo el estado i, es decir,

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j).$$

Si sumamos $\gamma_t(i)$ a lo largo del tiempo, obtenemos un valor que se puede interpretar como el número esperado de veces que se visita el estado i, o lo que es lo mismo, el número esperado de transiciones hechas desde el estado i, si eliminamos del sumatorio el instante de tiempo $t = T$. De manera similar, el sumatorio de $\xi_t(i,j)$ sobre t, también desde $t = 1$ hasta $t = T-1$, se puede interpretar como el número esperado de transiciones desde el estado i al estado j. Es decir,

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{número esperado de transiciones desde el estado } i \text{ en } O,$$

$$\sum_{t=1}^{T-1} \xi_t(i,j) = \text{número esperado de transiciones desde el estado } i \text{ al estado } j \text{ en } O.$$

Utilizando estas fórmulas, se puede dar un método general para reestimar los parámetros de un HMM.

Algoritmo 4.6 Reestimación de los parámetros de un HMM (algoritmo de Baum-Welch). El conjunto de ecuaciones para la reestimación de π, A y B es el siguiente:

$$\bar{\pi}_i = \text{frecuencia esperada de estar en el estado } i \text{ en el primer instante } = \gamma_1(i),$$

$$\bar{a}_{ij} = \frac{\text{número esperado de transiciones desde el estado } i \text{ al estado } j}{\text{número esperado de transiciones desde el estado } i} = \frac{\displaystyle\sum_{t=1}^{T-1} \xi_t(i,j)}{\displaystyle\sum_{t=1}^{T-1} \gamma_t(i)},$$

$$\bar{b}_j(v_k) = \frac{\text{número esperado de veces en el estado } j \text{ observando el símbolo } v_k}{\text{número esperado de veces en el estado } j} = \frac{\displaystyle\sum_{\substack{t=1 \\ \text{tal que } o_t = v_k}}^{T} \gamma_t(j)}{\displaystyle\sum_{t=1}^{T} \gamma_t(j)}.$$

Si definimos el modelo actual como $\mu = (\pi, A, B)$ y lo utilizamos para calcular los términos que aparecen en las partes derechas de las tres ecuaciones anteriores, y definimos el modelo reestimado como $\bar{\mu} = (\bar{\pi}, \bar{A}, \bar{B})$ a partir de los valores obtenidos para las partes izquierdas de dichas ecuaciones, está demostrado por Baum que: o bien (1) el modelo inicial μ define un punto crítico para la función de probabilidad, en cuyo caso $\bar{\mu} = \mu$; o bien (2) el modelo $\bar{\mu}$ es más probable que el modelo μ en el sentido de que $P(O|\bar{\mu}) > P(O|\mu)$, esto es, hemos encontrado un nuevo modelo $\bar{\mu}$ para el cual la probabilidad de generar la secuencia de observaciones es mayor. Así pues, reemplazando μ por $\bar{\mu}$ y repitiendo la reestimación de los parámetros un cierto número de veces, podemos mejorar la probabilidad de O en cada iteración, hasta que no se aprecie ninguna ganancia significativa. $\square$

Sin embargo, este método para la reestimación de los parámetros es muy sensible a las condiciones de inicialización del modelo. Es decir, el método no garantiza que encontremos el mejor modelo, ya que, debido a una inicialización incorrecta, el proceso podría detenerse en un máximo local, y en la mayoría de los problemas reales la función de probabilidad suele ser muy compleja, no lineal y con numerosos máximos locales. Por tanto, para encontrar el máximo global, una posible aproximación es intentar inicializar el modelo en una región del espacio de parámetros cercana a ese máximo global. En definitiva, una inicialización aleatoria podría dejar el problema de la reestimación de parámetros demasiado abierto y la manera de restringir el problema es elegir unos parámetros adecuados, en lugar de fijarlos aleatoriamente.

Algunos autores consideran que en la práctica, una estimación inicial de los parámetros π y A con números aproximadamente iguales, ligeramente perturbados para evitar los máximos locales, pero que respeten las restricciones estocásticas estándar y que sean siempre distintos de cero, suele ser normalmente satisfactoria para que posteriormente el algoritmo de Baum-Welch aprenda por sí solo la regularidad de las etiquetas, es decir, todo lo relativo a los transiciones entre los estados del modelo. Sin embargo esto no es más que una confusión que parte de la creencia general de que cuando un sistema tiene la capacidad de aprender, debe aprenderlo todo por sí mismo. Esto es efectivamente falso. Cuanta más información se le proporcione inicialmente al sistema, mejor[6]. Es decir, en nuestro caso, se ha demostrado que siempre se van a obtener mejores resultados cuando se inicializan los parámetros π y A a partir de un

[6]Por supuesto, esta idea de considerar como modelo más preciso el que más se acerca a los datos disponibles, llevada al extremo, degenera en un tratamiento manual específico para cualquier nuevo corpus, lo cual no es en absoluto realista. Ésta es la razón por la cual se desarrollan modelos abstractos y técnicas de aprendizaje: para que los sistemas se adapten rápidamente a los nuevos datos y sean capaces de manejarlos adecuadamente. Por tanto, se debe encontrar un equilibrio entre: por un lado, la descripción del modelo (y la dificultad de parametrizarlo), y por otro, su posibilidad de adaptación (es decir, su aprendizaje) con respecto al volumen de datos. Desde el punto de vista de la estadística y del aprendizaje, esto se conoce como el *Bias-Variance balance* (balance del sesgo y la varianza).

texto etiquetado, por pequeño que sea. Dicho texto podría ser construido manualmente si no se dispone de ninguno. Incluso repetir unas cuantas veces la estrategia de construir un modelo inicial, etiquetar otra pequeña porción de texto, corregirla manualmente, y utilizarla para estimar de nuevo los parámetros del modelo, ofrecería mejores resultados que la aplicación a ciegas del método de Baum-Welch. En definitiva, lo que queremos señalar es que la división entre métodos de estimación visible y no visible no es en nuestro caso tan estricta como la estamos presentando aquí. O si se prefiere, podríamos decir que, en sentido estricto, la estimación totalmente no visible no existe.

En cualquier caso, cuando no se dispone de textos etiquetados, o cuando estos son muy pequeños, resulta importante obtener un buen valor inicial para el parámetro B, es decir, para las probabilidades de emisión de las palabras, lo cual puede hacerse mediante el uso de un diccionario. Existen dos métodos generales para realizar esta inicialización: el método de Jelinek y el método de Kupiec. A continuación esbozamos cada uno de ellos:

- **Método de Jelinek**. Si definimos $Q(v_k)$ como el número de etiquetas permitidas para la palabra v_k en el diccionario, $C(v_k)$ como el número de apariciones de la palabra v_k en el corpus de entrenamiento, y $b_j^*(v_k)$ como

$$b_j^*(v_k) = \begin{cases} 0 & \text{si } j \text{ no es una etiqueta permitida para la palabra } v_k \\ \frac{1}{Q(v_k)} & \text{en caso contrario,} \end{cases}$$

  entonces se inicializa $b_j(v_k)$ con el valor

$$b_j(v_k) = \frac{b_j^*(v_k)\, C(v_k)}{\sum_m b_j^*(v_m)\, C(v_m)}.$$

  Es decir, el método de Jelinek inicializa las probabilidades de emisión del modelo mediante el uso de la regla de Bayes, estimando mediante la frecuencia observada la probabilidad de aparición de una palabra y suponiendo que todas las etiquetas que aparecen en el diccionario para una palabra dada son igualmente probables [Jelinek 1985].

- **Método de Kupiec**. La otra alternativa es agrupar las palabras en clases de ambigüedad, de tal manera que todas aquellas palabras que tengan el mismo conjunto de etiquetas permitidas en el diccionario pertenezcan a la misma clase o *metapalabra* u_L, donde L es un subconjunto de Q, el conjunto de etiquetas utilizado. Es decir, si $L(v_k)$ es el conjunto de todas las etiquetas posibles para la palabra v_k, entonces

$$u_L = \{v_k \mid L = L(v_k)\}, \qquad \forall L \subseteq Q.$$

  Por ejemplo, la metapalabra $u_{\{\texttt{Scms},\texttt{P}\}}$ contendrá todas las palabras para las cuales el diccionario permite las etiquetas `Scms`, es decir, sustantivo común masculino singular, y `P`, es decir, preposición, y ninguna otra etiqueta más. Entonces, a cada una de estas metapalabras se le da un tratamiento similar al del método de Jelinek:

$$b_j^*(u_L) = \begin{cases} 0 & \text{si } j \notin L \\ \frac{1}{|L|} & \text{en caso contrario,} \end{cases}$$

  donde $|L|$ es el cardinal del conjunto L, y

$$b_j(u_L) = \frac{b_j^*(u_L)\, C(u_L)}{\sum_{L'} b_j^*(u_{L'})\, C(u_{L'})}$$

donde $C(u_L)$ (respectivamente $C(u_{L'})$) es el número de palabras pertenecientes a u_L (respectivamente $u_{L'}$) que aparecen en el corpus de entrenamiento. La implementación real utilizada por Kupiec es una variante de la presentada aquí, pero se ha expuesto de esta manera para hacer más transparente la similitud entre ambos métodos. Por ejemplo, Kupiec no incluye las 100 palabras más frecuentes dentro de las clases de ambigüedad, sino que las trata separadamente como clases de una sola palabra, con el fin de no introducir errores y mejorar así la estimación inicial [Kupiec 1992].

La ventaja del método de Kupiec sobre el de Jelinek es que no necesita afinar una probabilidad de emisión diferente para cada palabra. Mediante la definición de estas clases de ambigüedad, el número total de parámetros se reduce substancialmente y éstos se pueden estimar de una manera más precisa. Por supuesto, esta ventaja podría convertirse en desventaja si existe la suficiente cantidad de material de entrenamiento como para estimar los parámetros palabra por palabra tal y como hace el método de Jelinek.

En el algoritmo de Baum-Welch también es necesario hacer algo para evitar el problema de la pérdida de precisión en punto flotante. Sin embargo, la necesidad de realizar sumas dificulta el uso de logaritmos. Una solución bastante común es utilizar unos coeficientes auxiliares de escalado, cuyos valores crecen con el tiempo de manera que si se multiplican las probabilidades por esos coeficientes, éstas siempre se mantienen dentro del rango de punto flotante del ordenador. Al final de cada iteración, cuando efectivamente se reestiman los parámetros, se cancela el uso de esos coeficientes [Rabiner y Juang 1993, pp. 365-368]. Otra alternativa es utilizar igualmente logaritmos y utilizar la función que mostramos a continuación para sumarlos.

Algoritmo 4.7 Función para la suma de logaritmos. Dados $x = \log(x')$ e $y = \log(y')$, la función calcula $\log(x' + y')$:

```
function Sumar_Logaritmos (x, y) =
  begin
    if (y − x > log C) then
      return y
    else
      if (x − y > log C) then
        return x
      else
        return min (x, y) + log (exp (x − min (x, y)) + exp (y − min (x, y)))
  end;
```

En esta porción de pseudo-código, C representa una constante grande, del orden de 10^{30}. De esta manera, lo que estamos haciendo es calcular un factor de escalado apropiado en el momento de realizar cada suma, y lo único que resta es tener cuidado con los errores de redondeo. □

4.4.3.2 Estimación supervisada: métodos de suavización

Cuando se dispone de un texto etiquetado, la primera idea que acude a nuestra mente para abordar el proceso de estimación de los parámetros del modelo es quizás la de diseñar un sencillo mecanismo basado en el uso de frecuencias relativas. Pero antes de comentar dicho mecanismo, vamos a introducir un cambio en la notación que hace referencia a la definición formal de los HMM,s.

Hasta aquí, la notación que hemos utilizado estaba inspirada en la excelente presentación que Rabiner llevó a cabo en relación con este tipo de procesos estocásticos [Rabiner 1989].

Efectivamente, se trata de una notación genérica sobre HMM,s especialmente orientada a describir con el máximo detalle la implementación de determinados algoritmos, como pueden ser el de Viterbi o el de Baum-Welch. Sin embargo, una vez que hemos centrado la discusión sobre el tema concreto de la etiquetación, algunos aspectos de esa notación no se identifican con nuestro problema todo lo bien que nos gustaría. Por poner un ejemplo, los estados de un HMM habían sido denotados simplemente con los números enteros del conjunto $\{1, 2, \ldots, N\}$. Esto facilita mucho la expresión de operaciones tales como la indexación de las celdas de la matriz A de transiciones entre estados, pero no nos permite hacer referencia a las etiquetas concretas que se están utilizando, si no es a través de la consideración de una función correspondencia entre ambos conjuntos.

La notación que proponemos ahora está más acorde con la que se adopta en la mayoría de las referencias bibliográficas dedicadas específicamente al tema de la etiquetación.

Definición 4.2 Básicamente, la nueva notación describe cada uno de los elementos de un HMM como sigue:

1. Considerando la inicial de la palabra inglesa *tag* (etiqueta), denotaremos mediante t^i la i-ésima etiqueta del conjunto de etiquetas utilizado, y mediante t_i la etiqueta de la palabra que ocupa la posición i dentro de la frase que se está tratando. De esta manera, en el modelo de bigramas, los estados se denotan también mediante t^i, es decir, mediante el nombre de las propias etiquetas. Y si el modelo está basado en trigramas, los estados serán de la forma (t^i, t^j), donde cada uno de estos pares proviene del producto cartesiano del conjunto de etiquetas utilizado, tal y como ya habíamos esbozado.

2. Anteriormente, habíamos denotado mediante v_k cada uno de los símbolos de observación correspondientes a la parte *visible* del modelo, y habíamos visto que, en el caso de la etiquetación, dichos sucesos observables se corresponden con las palabras. Pues bien, de igual manera, considerando la inicial del vocablo inglés *word* (palabra), denotaremos ahora mediante w^i la i-ésima palabra del diccionario, y mediante w_i la palabra que ocupa la posición i dentro de la frase.

3. Habíamos indicado también que trabajaremos siempre en el nivel de frase, y que consideraremos un estado o etiqueta especial t^0 para marcar el inicio y el final de cada una de las frases. Por tanto, respecto al parámetro π, la distribución de probabilidad del estado inicial, tendremos que $\pi(t^0)$ será igual a 1, y $\pi(t^i)$ será igual a 0, para todo $i = 1, 2, \ldots, N$.

4. Por lo que respecta al parámetro A, la distribución de probabilidad de las transiciones entre estados, basta aclarar que, de acuerdo con la nueva notación, la información que realmente está almacenada en cada una de las celdas de esta matriz es $P(t^j|t^i)$ en el caso de los modelos basados en bigramas, y $P(t^k|t^i\, t^j)$ en el caso de los modelos basados en trigramas.

5. Por último, respecto a B, las probabilidades de emisión de las palabras, simplemente indicamos que la información que antes se denotaba mediante $b_j(v_k)$ se escribe ahora como $P(w^k|t^j)$.

Para terminar, completamos esta notación con algunos aspectos que nos permitirán avanzar más cómodamente con la exposición. Denotaremos mediante $C(x)$ el número de veces que el suceso x aparece en el corpus de entrenamiento. Por ejemplo, $C(t^i, t^j, t^k)$ representa el número de veces que aparece el trigrama $t^i\, t^j\, t^k$. De igual manera, $C(w^k|t^j)$ representa el número de veces que aparece la palabra w^k etiquetada como t^j. Denotaremos también mediante $f(x)$ la frecuencia del suceso x en el corpus de entrenamiento. Y finalmente denotaremos mediante $\hat{p}(x)$ nuestra estimación de la probabilidad real $P(x)$. □

Nuevo modelo probabilístico. La etiquetación se describe entonces, a través de esta nueva notación, como el proceso de encontrar la mejor secuencia de etiquetas $t_{1,n}$ para una frase dada de n palabras $w_{1,n}$. Aplicando la regla de Bayes, podemos escribir:

$$\arg\max_{t_{1,n}} P(t_{1,n}|w_{1,n}) = \arg\max_{t_{1,n}} \frac{P(w_{1,n}|t_{1,n})P(t_{1,n})}{P(w_{1,n})} = \arg\max_{t_{1,n}} P(w_{1,n}|t_{1,n})P(t_{1,n}).$$

Ahora, trabajamos sobre esta expresión para conseguir que utilice los parámetros que podemos estimar directamente desde el corpus de entrenamiento. Para ello, además de la propiedad del horizonte limitado (4.14), utilizamos dos suposiciones más acerca de las palabras:

1. Las etiquetas no son independientes unas de otras, pero las palabras sí (4.12).
2. La identidad de una palabra depende sólo de su propia etiqueta (4.13).

Así pues, tenemos que:

$$P(w_{1,n}|t_{1,n})P(t_{1,n}) =$$

$$= \left(\prod_{i=1}^{n} P(w_i|t_{1,n})\right) \times P(t_n|t_{1,n-1}) \times P(t_{n-1}|t_{1,n-2}) \times \ldots \times P(t_2|t_1) = \tag{4.12}$$

$$= \left(\prod_{i=1}^{n} P(w_i|t_i)\right) \times P(t_n|t_{1,n-1}) \times P(t_{n-1}|t_{1,n-2}) \times \ldots \times P(t_2|t_1) = \tag{4.13}$$

$$= \left(\prod_{i=1}^{n} P(w_i|t_i)\right) \times P(t_n|t_{n-1}) \times P(t_{n-1}|t_{n-2}) \times \ldots \times P(t_2|t_1) = \tag{4.14}$$

$$= \prod_{i=1}^{n} [P(w_i|t_i) \times P(t_i|t_{i-1})].$$

Por tanto, la ecuación final para determinar la secuencia de etiquetas óptima para una frase dada es

$$\hat{t}_{1,n} = \arg\max_{t_{1,n}} P(t_{1,n}|w_{1,n}) = \arg\max_{t_{1,n}} \prod_{i=1}^{n} [P(w_i|t_i) \times P(t_i|t_{i-1})]$$

para los etiquetadores basados en bigramas, y

$$\hat{t}_{1,n} = \arg\max_{t_{1,n}} P(t_{1,n}|w_{1,n}) = \arg\max_{t_{1,n}} \prod_{i=1}^{n} [P(w_i|t_i) \times P(t_i|t_{i-2}, t_{i-1})]$$

para los etiquetadores basados en trigramas. Estos cálculos son precisamente los que realiza el algoritmo de Viterbi, de tal manera que para disponer de un etiquetador totalmente operativo sólo resta especificar de qué manera se pueden obtener los parámetros de funcionamiento del HMM subyacente. La sección anterior presentaba un método para la estimación de dichos parámetros a partir de textos no etiquetados. Ahora nos ocuparemos del diseño de métodos de estimación de parámetros a partir de textos etiquetados.

Retomemos entonces la idea inicial de diseñar un mecanismo intuitivo para estimar los parámetros de nuestro modelo. Dado que en este caso partimos de un texto previamente etiquetado, podemos sugerir que el proceso de estimación de las transiciones entre estados podría realizarse en base a unos sencillos cálculos de frecuencias relativas. En el caso de un modelo basado en bigramas, tendríamos que

$$\hat{p}(t^j|t^i) = f(t^j|t^i) = \frac{C(t^i, t^j)}{C(t^i)}.$$

En el caso de un modelo basado en trigramas, tendríamos que

$$\hat{p}(t^k|t^i\,t^j) = f(t^k|t^i\,t^j) = \frac{C(t^i,t^j,t^k)}{C(t^i,t^j)}.$$

De igual manera, para la estimación de las probabilidades de emisión de las palabras, tendríamos que

$$\hat{p}(w^k|t^j) = f(w^k|t^j) = \frac{C(w^k|t^j)}{C(t^j)}.$$

Hasta aquí, lo que hacemos no es realmente estimar, sino construir un modelo de manera visible. Una vez que este modelo está preparado, podemos utilizarlo para etiquetar nuevos textos mediante la aplicación del algoritmo de Viterbi a cada una de sus frases, y la idea original de modelo oculto vuelve a tener sentido.

Sin embargo, antes de utilizar directamente un modelo construido a partir de frecuencias relativas, es necesario hacer la siguiente reflexión. Pensemos en un caso práctico real, como por ejemplo el experimento realizado con el corpus ITU[7]. El cardinal del juego de etiquetas utilizado en dicho experimento es de 373 etiquetas, tal y como puede verse en la sección A.2. Esto no quiere decir que una frase de longitud n palabras deba ser etiquetada aplicando el algoritmo de Viterbi sobre un enrejado de $373 \times n$ estados, sino más bien sobre un enrejado simplificado como el que veíamos en la figura 4.7, donde se consideran sólo las posibles etiquetas de cada palabra. Lo que sí quiere decir es que, en el caso de un modelo basado en bigramas, la matriz A contendría $373 \times 373 = 139.129$ celdas, y sin embargo, en el mayor de los corpus de entrenamiento utilizados en dicho experimento, tan solo se pueden ver 4.037 de esas transiciones (es decir, el 2,90%), y las restantes celdas quedarían a cero. En el caso de un modelo basado en trigramas, la matriz A contendría $373 \times 373 \times 373 = 51.895.117$ celdas, sólo se pueden llegar a ver 23.119 (el 0,04%), y las restantes quedarían también a cero. Esto es debido al fenómeno de *dispersión de los datos*[8], el cual puede provocar también que algunas de esas transiciones aparezcan como candidatas a la hora de etiquetar nuevas frases.

Así pues, teniendo en cuenta que los enrejados simplificados ya contemplan sólo un subconjunto reducido de todos los caminos posibles, parece muy arriesgado trabajar con transiciones nulas, cuyo uso implica multiplicaciones por cero que anularían completamente todos los caminos que pasen por ellas. Para evitar el problema de las transiciones nulas, se suelen utilizar *métodos de suavización*[9]. Estos métodos permiten que a partir de muestras pequeñas se puedan estimar unas probabilidades más representativas del comportamiento real de la población que estamos estudiando.

Una forma de implementar la suavización es mediante técnicas de *interpolación lineal*. En general, el esquema de suavizado mediante interpolación lineal funciona como sigue. La distribución $f(x)$ observada en un conjunto de E posibles sucesos se modifica añadiendo un valor muy pequeño procedente de una distribución menos específica $q(x)$ a la cual damos un peso o confianza α. Entonces, la distribución de probabilidad que nos interesa se aproxima de la siguiente forma:

$$\hat{p}(x) \approx \frac{f(x) + \alpha\, q(x)}{E + \alpha}.$$

Y si utilizamos el parámetro tradicional de interpolación $\lambda = \frac{\alpha}{E+\alpha}$, entonces también se verifica la siguiente aproximación:

$$\hat{p}(x) \approx (1-\lambda)\,\frac{f(x)}{E} + \lambda\, q(x).$$

[7] *International Telecommunications Union CCITT Handbook* (véase la sección 2.1).
[8] También denominado fenómeno de *sparse data*.
[9] También denominados métodos de *smoothing*.

Es decir, si lo que queremos estimar son las probabilidades de transición entre estados de un modelo basado en bigramas, esa distribución menos específica ponderada con α al interpolar puede ser la distribución de probabilidad de los *unigramas*, esto es, la distribución de probabilidad de cada etiqueta individualmente[10]. Por tanto, las probabilidades de los bigramas se pueden aproximar mediante

$$\hat{p}(t_i|t_{i-1}) = (1-\lambda)\, f(t_i|t_{i-1}) + \lambda\, f(t_i)$$

y las de los trigramas mediante

$$\hat{p}(t_i|t_{i-2}\, t_{i-1}) = \lambda_3\, f(t_i|t_{i-2}\, t_{i-1}) + \lambda_2\, f(t_i|t_{i-1}) + \lambda_1\, f(t_i) \tag{4.15}$$

donde todos los *pesos* λ_i deben ser no negativos y deben satisfacer la restricción $\lambda_1+\lambda_2+\lambda_3=1$. A continuación, y centrando ya toda nuestra atención en el modelo de trigramas, discutiremos cómo elegir los pesos λ_i de manera óptima.

Suavizado lineal óptimo. Un modelo de lenguaje que opere de acuerdo con la ecuación (4.15) se puede ver realmente como un HMM. La figura 4.9 muestra un fragmento de este modelo. En este punto es importante aclarar la siguiente cuestión. Existe una variante de representación de los HMM,s que lleva asociadas las probabilidades de emisión no a un único estado, sino tanto al estado origen como al estado destino de las transiciones. Es decir, dichas probabilidades de emisión están asociadas realmente a los arcos o transiciones, no a los estados individuales. Algunos autores prefieren enfocar la introducción a los HMM,s comenzando con este tipo de representación, ya que el paso de generalización desde los HMM,s con probabilidades de emisión en los arcos a los HMM,s con probabilidades de emisión en los estados resulta más natural que el paso inverso. A pesar de esto, nosotros hemos preferido manejar desde el principio HMM,s con probabilidades de emisión en los estados, porque este es el tipo de HMM,s que efectivamente se utiliza en la práctica para la etiquetación de textos, y por tanto es posible establecer analogías entre ambos conceptos desde el primer momento.

No obstante, el HMM de la figura 4.9 es un HMM con probabilidades de emisión en los arcos. Ocurre además que en este tipo de HMM,s está permitido el uso de transiciones que no emiten ningún símbolo, sino que simplemente se utilizan para cambiar de estado. Este tipo de transiciones se denominan *transiciones épsilon* o *transiciones vacías*. En la figura, no aparecen representados los símbolos de salida, sino simplemente las probabilidades de cada arco. Por tanto, para distinguir las transiciones normales de las transiciones épsilon, hemos representado estas últimas mediante líneas discontinuas. Así pues, saliendo del estado más a la izquierda (t^i,t^j) tenemos tres transiciones vacías que van a los pseudo-estados $s_1(t^i,t^j)$, $s_2(t^i,t^j)$ y $s_3(t^i,t^j)$. Las probabilidades de estas transiciones son λ_1, λ_2 y λ_3, respectivamente. A continuación, saliendo de cada uno de los tres pseudo-estados aparecen N transiciones. Cada una de ellas conduce a un estado (t^j,t^k), $k=1,2,\ldots,N$, para generar la tercera etiqueta del trigrama. Las probabilidades de estas transiciones son $f(t^k)$, $f(t^k|t^j)$ y $f(t^k|t^i\,t^j)$, $k=1,2,\ldots,N$, respectivamente. En la figura, hemos etiquetado sólo las transiciones que entran en los estados (t^j,t^1) y (t^j,t^N). Nótese que desde el estado más a la izquierda (t^i,t^j) se puede alcanzar cada uno de estos estados a través de tres caminos posibles, y que por tanto la probabilidad total de llegar a cada estado coincide con el cálculo de la ecuación (4.15).

[10]Esta fórmula viene del hecho de que $P(X|Y)=P(X)$, si X e Y son independientes. Por eso decimos que $P(X|Y)$ se estima mediante $f(X|Y)$, o en todo caso mediante una combinación lineal de $f(X|Y)$ y $f(X)$, si el bigrama $Y\,X$ aparece en el corpus de entrenamiento, y mediante $f(X)$ si no aparece, asumiendo la hipótesis de independencia para esos sucesos perdidos. Rigurosamente hablando esto no es correcto, pero constituye una buena aproximación de primer orden (considerando la hipótesis de Markov como una hipótesis de segundo orden).

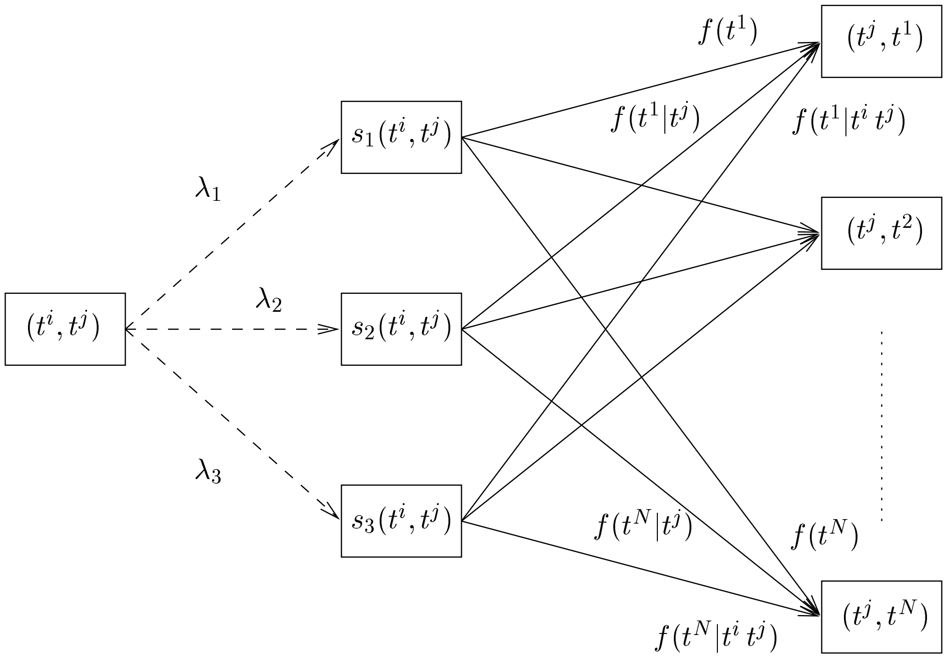


Figura 4.9: Fragmento del suavizado lineal de un HMM basado en trigramas

Por supuesto, otra forma de ver el problema podría ser considerando el HMM de la figura 4.9 todavía como un HMM con probabilidades de emisión en los estados, salvo que ahora es a los pseudo-estados $s_1(t^i, t^j)$, $s_2(t^i, t^j)$ y $s_3(t^i, t^j)$ a los que no se les permite generar ninguna palabra. En cualquier caso, tenemos que las probabilidades de las transiciones normales son conocidas, mientras que las de las transiciones épsilon, λ_1, λ_2 y λ_3, deben ser determinadas mediante algún procedimiento. El HMM completo es desde luego enorme: consta de $4 \times N^2$ estados, donde N es el tamaño del conjunto de etiquetas utilizado. Pero de acuerdo con la ecuación (4.15), para un i dado, $i \in \{1, 2, 3\}$, todas las probabilidades λ_i de las transiciones que van desde el estado más a la izquierda (t^i, t^j) hasta uno de los estados más a la derecha (t^j, t^k) tienen el mismo valor, sea cual sea la combinación de etiquetas que forma el trigrama $t^i\,t^j\,t^k$. Se dice entonces que estas probabilidades toman *valores empatados*[11].

Pues bien, dado que hemos visto que el modelo de lenguaje al que obedece la ecuación (4.15) es un HMM, se puede utilizar el algoritmo de Baum-Welch para estimar los valores λ_i óptimos. Además, como una consecuencia favorable de que el algoritmo deba *empatar* o igualar las probabilidades λ_i, independientemente de a qué subparte del HMM pertenezcan, el proceso de estimación sólo necesita manejar tres contadores, uno para cada tipo de transición épsilon, del número esperado de veces que se cruza por una transición. Para el resto de transiciones, no se necesita establecer ningún otro tipo de contador porque de hecho las probabilidades de esas transiciones están ya fijadas de antemano. Debido a todo esto, se pueden sugerir ideas prácticas que dan lugar a cálculos en este caso más sencillos que los del algoritmo de Baum-Welch, y que por tanto simplifican aún más la obtención de los λ_i óptimos [Jelinek 1997, pp. 66-69].

Pero ahora preferimos centrar nuestro interés en responder a una importante pregunta: ¿qué tipo de datos de entrenamiento deberíamos usar para determinar los pesos λ_i? Está claro que no pueden ser los mismos datos sobre los que se calculan las frecuencias $f(\ |\ )$, porque en ese caso las estimaciones darían como resultado $\lambda_3 = 1$ y $\lambda_1 = \lambda_2 = 0$. De hecho, $f(t^k|t^i\,t^j)$ es la estimación

[11] O también *tied values*.

de máxima verosimilitud de $P(t^k|t^i\,t^j)$ para los datos de entrenamiento en los que se basa dicha frecuencia $f(t^k|t^i\,t^j)$. Sin embargo, mirando de nuevo la figura 4.9, se intuye que, de los tres caminos posibles que conducen a cada uno de los estados más a la derecha, el HMM ha incluido el camino superior para poder generar la tercera etiqueta t^k, incluso aunque el bigrama $t^j\,t^k$ no haya aparecido en los datos de entrenamiento. Y de manera similar, mediante ese mismo camino superior y mediante el camino central, se hace posible la tercera etiqueta t^k, incluso aunque no exista ninguna ocurrencia del trigrama $t^i\,t^j\,t^k$. Por tanto se concluye que el conjunto total de los datos de entrenamiento debe ser dividido en dos porciones:

1. La primera, mucho más grande, denominada *datos de desarrollo*[12], se utiliza para estimar las frecuencias relativas $f(\;|\;)$.

2. Una vez que éstas están fijadas, la segunda porción de datos, mucho más pequeña ya que quedan por estimar muchos menos parámetros, se usa para estimar los pesos λ_i. Esta porción se denomina *datos extendidos* u *ocultos*[13].

Por supuesto, una vez que se ha realizado este proceso, el modelo obtenido se puede mejorar combinando ambas porciones de datos y reestimando las frecuencias $f(\;|\;)$. Esta técnica de suavizado lineal se denomina *interpolación de borrado*[14].

La interpolación de borrado se puede realizar también mediante una eliminación sucesiva de cada trigrama del corpus de entrenamiento, y una posterior estimación de los λ_i óptimos a partir del resto de n-gramas del corpus. Conocidos los contadores de frecuencias para unigramas, bigramas y trigramas, los pesos λ_i se pueden determinar eficientemente a través del siguiente algoritmo.

Algoritmo 4.8 Cálculo de los parámetros λ_1, λ_2 y λ_3 de un esquema de interpolación lineal, conocidas las frecuencias de unigramas, bigramas y trigramas, y conocido N, el tamaño del corpus de entrenamiento [Brants 2000]:

1. Inicializar $\lambda_1 = \lambda_2 = \lambda_3 = 0$.

2. Para cada trigrama $t_1\,t_2\,t_3$ con $C(t_1,t_2,t_3) > 0$, localizar el máximo de los tres valores siguientes y realizar la acción correspondiente:

   - $\frac{C(t_1,t_2,t_3)-1}{C(t_1,t_2)-1}$: incrementar λ_3 en $C(t_1,t_2,t_3)$ unidades.
   - $\frac{C(t_2,t_3)-1}{C(t_2)-1}$: incrementar λ_2 en $C(t_1,t_2,t_3)$ unidades.
   - $\frac{C(t_3)-1}{N-1}$: incrementar λ_1 en $C(t_1,t_2,t_3)$ unidades.

3. Normalizar los valores λ_1, λ_2 y λ_3.

Abusando un poco de la notación, con el fin de no complicar en exceso las ecuaciones con la presencia de tantos subíndices, hemos denotado el trigrama $t_{i-2}\,t_{i-1}\,t_i$ mediante $t_1\,t_2\,t_3$, donde de manera obvia los números 1, 2 y 3 hacen referencia a la posición de cada etiqueta dentro del trigrama. Si el denominador de alguna de las expresiones es 0, se define el resultado de esa expresión como 0. Restar 1 en este algoritmo es la manera de tener en cuenta los datos no observados. Sin esta resta, el modelo efectivamente sobreestimaría los datos de entrenamiento, generando $\lambda_3 = 1$ y $\lambda_1 = \lambda_2 = 0$ y produciendo peores resultados. □

[12] *Development data.*
[13] *Held-out data.*
[14] O también *deleted interpolation.*

Hemos visto que la ecuación (4.15) propone valores constantes para los parámetros de interpolación λ_1, λ_2 y λ_3. Es cierto que quizás no es una buena idea utilizar siempre los mismos valores, pero es cierto también que la consideración de un conjunto de valores λ_i, $i \in \{1,2,3\}$, para cada posible par de etiquetas eleva muchísimo el número de parámetros del modelo, lo cual no sólo no introduce ninguna mejora en relación con el fenómeno de los datos dispersos, sino que empeora el problema.

No obstante, algunos autores han sugerido que al menos sí sería conveniente una agrupación de los bigramas en un número moderado de clases y una posterior estimación de un conjunto de valores λ_i distinto para cada clase, aunque no queda claro cuál sería un buen criterio para la definición de esas clases.

Brants es quizás el único autor que proporciona datos concretos sobre diferentes experimentos de agrupación. Uno de ellos incluía un conjunto de valores λ_i distinto para cada frecuencia. Otra de las posibilidades estudiadas fue la definición de dos clases, frecuencias altas y frecuencias bajas, a ambos extremos de la escala, y la posterior aplicación de diferentes criterios de agrupación, también basados en frecuencias, para definir las clases intermedias. Pero finalmente observó que la mayoría de los resultados eran equivalentes, obteniendo incluso peores cifras con algunas de las agrupaciones consideradas. Por esta razón, Brants utiliza en su etiquetador interpolación lineal independiente del contexto, es decir, valores constantes para los parámetros λ_i [Brants 2000].

Otros autores, sin embargo, proponen una reducción del número de parámetros a estimar mediante la especificación de que ciertos sucesos son absolutamente improbables, es decir, tienen probabilidad 0, lo cual permite introducir *ceros estructurales* en el modelo. Efectivamente, el hecho de hacer que algunos sucesos no sean posibles, por ejemplo un trigrama formado por tres preposiciones seguidas, añade gran cantidad de estructura al modelo, mejora el rendimiento del proceso de entrenamiento, y por tanto facilita en gran medida la labor de estimación de parámetros. Pero esto resulta apropiado sólo en algunas circunstancias y no con espacios de estados tan grandes.

En cualquier caso, la interpolación lineal no es la única manera de enfrentarse al problema de la poca frecuencia de determinados sucesos en los datos de entrenamiento. Existen otros métodos de estimación que, aunque son también dependientes del número de veces que aparecen los sucesos en el corpus de entrenamiento, no se basan en absoluto en frecuencias relativas. Como consecuencia, se han sugerido muchas fórmulas que son capaces de asignar probabilidades distintas de cero a los sucesos no observados, permitiendo así hacer una importante distinción entre este tipo de *ceros no observados*, y los *ceros reales* que serían los correspondientes a los sucesos efectivamente no posibles.

Entre todas ellas, la más conocida es la fórmula de Good-Turing[15]. La sección 4.5 está especialmente dedicada al estudio detallado de ésta y otras aproximaciones, incluida la que actualmente comparte el estado del arte con la interpolación lineal, en lo que se refiere al tratamiento de datos dispersos dentro del ámbito de los sistemas de etiquetación: el método de *marcha atrás* o *back-off* [16].

[15] La idea intuitiva del método de estimación de Good-Turing es que, en lugar de estimar $P(X)$, intentamos estimar $P(X|C)$, donde C es una clasificación de sucesos definida *a priori* (en nuestro caso, los sucesos X que aparecen en el corpus de entrenamiento exactamente C veces). De esta manera, es más sencillo estimar $P(X)$, y en concreto se puede estimar $P(X|0)$, que es precisamente la probabilidad de los sucesos que no aparecen en el corpus. En resumen, el método de Good-Turing corrige la estimación de máxima verosimilitud en base al número de veces que algo ocurre en el corpus.

[16] La idea general del método de estimación de marcha atrás o *back-off* es confiar en las frecuencias si hay un número suficiente de apariciones, utilizar Good-Turing si no lo hay, pero el suceso está todavía presente, y utilizar una hipótesis de menor nivel (como el suavizado lineal) si no está presente en absoluto. La estimación de menor nivel esta basada en la suposición que hemos visto anteriormente, de que $P(X|Y) = P(X)$ cuando X e Y son independientes.

4.4.3.3 Estimación combinada mediante métodos híbridos

Los etiquetadores basados en HMM,s trabajan bien cuando se dispone de un corpus de entrenamiento suficientemente grande. A menudo éste no es el caso. Suele ocurrir con frecuencia que queremos etiquetar un texto de un dominio especializado donde las probabilidades de emisión de las palabras son diferentes de las que se pueden observar en los textos de entrenamiento. Otras veces necesitamos etiquetar textos de idiomas para los cuales simplemente no existen *corpora* etiquetados. En estos casos, hemos visto anteriormente que se puede utilizar un procedimiento de estimación no visible o no supervisado a partir de una inicialización pseudo-aleatoria de los parámetros del modelo y de la posterior aplicación del algoritmo de Baum-Welch.

Los fundamentos teóricos sugieren que el algoritmo de Baum-Welch debe detenerse cuando no se puede construir un modelo que mejore la probabilidad de la secuencia de entrenamiento dada. Sin embargo, ha sido demostrado que, para el caso de la etiquetación, este criterio a menudo produce como resultado un sobreentrenamiento del modelo. Este fenómeno ha sido estudiado en profundidad por Elworthy, quien entrenó diferentes HMM,s considerando una gran variedad de condiciones de inicialización y de distintos números de iteraciones [Elworthy 1994]:

- En ocasiones, el proceso de entrenamiento se mostraba estable y siempre producía mejoras del rendimiento después de cada iteración, demostrando que el criterio probabilístico era apropiado para esos casos.

- Pero otras veces, la mejora aparecía sólo durante algunas iteraciones, normalmente 2 ó 3, y después el proceso de entrenamiento no hacía más que degradar el rendimiento.

Esto último se producía en lo que inicialmente se consideraban como los escenarios típicos de aplicación de los HMM,s: cuando se dispone de un diccionario, pero no se dispone de ningún corpus etiquetado. Por tanto, en este tipo de situaciones, los cuidados se deben extremar al máximo con el fin de no sobreentrenar el modelo. Una forma de conseguir esto es validar dicho modelo después de cada iteración sobre un conjunto de datos separado (*held-out data*), y parar el entrenamiento cuando el rendimiento empieza a decrecer.

Elworthy también confirmó los resultados encontrados por Merialdo, que demuestran que si se dipone inicialmente de un corpus etiquetado, la aplicación del algoritmo de Baum-Welch puede degradar el rendimiento ya desde la primera iteración [Merialdo 1994]. Sin embargo, una peculiaridad interesante es el hecho de que si el texto de entrenamiento y el de validación son muy diferentes, unas pocas iteraciones podrían producir mejoras. Además, esto suele ocurrir con frecuencia en la práctica, ya que a menudo tenemos que enfrentarnos con tipos de textos para los cuales no se dispone de *corpora* de entrenamiento etiquetados similares. En resumen:

- Si existe un texto de entrenamiento suficientemente grande y similar a los textos con los que se va a trabajar, entonces se debería utilizar un procedimiento de estimacion totalmente visible o supervisado.

- Si no existe ningún texto de entrenamiento disponible, o si los textos de entrenamiento y validación son muy diferentes, entonces se debería aplicar el algoritmo de Baum-Welch durante unas pocas iteraciones.

- Sólo se debería de aplicar durante un número grande de iteraciones, 10 ó más, cuando no hay ningún tipo de información léxica disponible.

Y en este último caso, no cabe esperar un buen rendimiento. Esto no se debe a ningún defecto del algoritmo de Baum-Welch, sino al hecho de que dicho algoritmo tan sólo realiza ajustes de los parámetros del HMM con el fin de maximizar la probabilidad de los datos de entrenamiento.

Los cambios que realiza para reducir la entropía cruzada de esos datos podrían no estar de acuerdo con nuestro objetivo real, que es el de asignar a las palabras etiquetas pertenecientes a un conjunto predefinido. Por tanto, esta técnica no siempre es capaz de optimizar el rendimiento para la tarea particular de la etiquetación.

4.4.3.4 Integración de diccionarios

Un problema similar al de las transiciones entre estados ocurre también con las palabras. En un primer momento podemos pensar que si estamos hablando de entrenamiento puramente supervisado, las probabilidades de generación de las palabras son lo único que sí se puede estimar perfectamente. Sin embargo, en la práctica, podemos encontrarnos con palabras que no aparecen en los textos de entrenamiento, pero que quizás sí las tenemos presentes en un diccionario. Es decir, se trata de palabras para las cuales conocemos sus posibles etiquetas, pero que al no haber sido vistas durante el entrenamiento, no tienen definidas sus probabilidades de emisión. Una vez más, no resulta conveniente dejar a cero esas probabilidades.

Por tanto, se hace necesario el uso de métodos que permitan la correcta integración de la información aportada por el diccionario con los datos proporcionados por el texto de entrenamiento. Existen dos métodos generales para realizar esta integración:

- **Método *Adding One* de Church**. El método *Adding One* o método *de sumar uno* utiliza el diccionario como si se tratara de un corpus etiquetado, de manera que cada uno de los pares palabra-etiqueta presentes en el diccionario aparece una sola vez en dicho corpus [Church 1988]. Por supuesto, este nuevo corpus se utilizará sólo para estimar las probabilidades de emisión de las palabras, pero no las probabilidades de transición entre estados, ya que el orden que siguen las palabras y las etiquetas dentro del diccionario no tiene nada que ver con el orden que siguen en las frases reales. De esta manera, para todo par (w^k, t^j) presente en el diccionario, la probabilidad de emisión se estima mediante

$$\hat{p}(w^k|t^j) = \frac{C(w^k|t^j) + 1}{C(t^j) + K_j}$$

  donde K_j es el número de palabras etiquetadas con t^j que aparecen en el diccionario, mientras que para el resto de pares palabra-etiqueta que aparezcan en el corpus de entrenamiento y no en el diccionario la estimación se realiza mediante la misma ecuación, pero sin sumar 1 en el numerador. Intuitivamente, volviendo al ejemplo de las urnas y las bolas, el método opera como si todas y cada unas de las bolas o palabras del diccionario fueran colocadas en sus urnas o etiquetas correspondientes una sola vez, como paso previo a la estimación de las probabilidades de emisión. Con esto se produce el efecto deseado de que las probabilidades de emisión de las palabras del diccionario no queden a cero, incluso aunque no hayan aparecido en el corpus de entrenamiento.

- **Método de Good-Turing**. Como ya hemos comentado anteriormente, éste es un método de estimación que no está basado en frecuencias relativas, pero que también es capaz de asignar probabilidades distintas de cero a los sucesos no observados. En este contexto, un suceso posible pero no observado es cualquier par palabra-etiqueta que aparece en el diccionario y no aparece en el corpus de entrenamiento (cero no observado), mientras que un suceso no posible es cualquier par palabra-etiqueta que no aparece ni en el diccionario ni en el corpus de entrenamiento (cero real). La sección 4.5 describe detalladamente la obtención de las fórmulas de Good-Turing.

Es importante recordar que uno de nuestros principales objetivos es comprobar de qué manera el uso de un diccionario externo puede ayudar a incrementar el rendimiento del proceso de etiquetación. En el capítulo 7 se detallan las cifras de rendimiento para todos los etiquetadores que se han evaluado en el presente trabajo. Siempre que ha sido posible, dicha evaluación se ha realizado con y sin el uso de diccionarios externos.

4.4.3.5 Tratamiento de palabras desconocidas

Hemos visto anteriormente cómo estimar las probabilidades de emisión para las palabras que aparecen o bien en el corpus, o bien en nuestro diccionario, o bien en ambos. Pero es seguro que al intentar etiquetar nuevas frases encontraremos multitud de palabras que no han aparecido previamente en ninguno de esos recursos. Hemos visto también que un conocimiento *a priori* de la distribución de etiquetas de una palabra, o al menos de la proporción para la etiqueta más probable, supone una gran ayuda para la etiquetación. Esto significa que las palabras desconocidas son el mayor problema de los etiquetadores, y en la práctica la diferencia de rendimiento entre unos y otros puede ser debida a la proporción de palabras desconocidas y al procedimiento de adivinación y tratamiento de las mismas que cada etiquetador tenga integrado.

La manera más simple de enfrentarse a este problema es suponer que toda palabra desconocida puede pertenecer a cualquier categoría gramatical, o quizás a una clase de categorías *abiertas*, por ejemplo, sustantivos, adjetivos, verbos, etc., pero no preposiciones, ni artículos, ni pronombres, que suponemos que pertenecen a otra clase de categorías *cerradas* y que todas las palabras correspondientes a ellas son conocidas y están ya en el diccionario. Aunque esta aproximación podría ser válida en algunos casos, en general, el no hacer uso de la información léxica de las palabras (prefijos, sufijos, etc.) para proponer un conjunto más limitado de posibles etiquetas degrada mucho el rendimiento de los etiquetadores. Es por ello que existen numerosos trabajos orientados a explorar este tipo de características, y mejorar así la estimación de las probabilidades de emisión para las palabras desconocidas:

- Por ejemplo, Weischedel estima las probabilidades de generación de las palabras en base a tres tipos de información: la probabilidad de que una categoría genere una palabra desconocida[17], la probabilidad de que una categoría genere una palabra cuya inicial es mayúscula o minúscula, y la probabilidad de generación de sufijos particulares:

$$\hat{p}(w^k|t^j) = \frac{1}{Z}\, P(\text{desconocida}|t^j)\; P(\text{mayúscula}|t^j)\; P(\text{sufijo}|t^j)$$

  donde Z es una constante de normalización. Este modelo reduce la proporción de error de palabaras desconocidas de más de un 40% a menos de un 20% [Weischedel *et al.* 1993]. Charniak propuso un modelo alternativo que utiliza tanto las raíces como los sufijos [Charniak *et al.* 1993].

- La mayor parte del trabajo que se realiza en relación con las palabras desconocidas asume que las propiedades consideradas son independientes. Dicha independencia no siempre es una buena suposición. Por ejemplo, las palabras en mayúsculas tienen más probabilidad de ser desconocidas, y por tanto las propiedades *mayúscula* y *desconocida* del modelo de Weischedel no son realmente independientes. Franz desarrolló un modelo que tiene en cuenta este tipo de dependencias [Franz 1996, Franz 1997].

- Actualmente, el método de manejo de palabras desconocidas que parece ofrecer mejores resultados para los lenguajes flexivos es el análisis de sufijos mediante aproximaciones basadas en inferencias Bayesianas [Samuelsson 1993]. En este método, las probabilidades de las etiquetas propuestas para las palabras no presentes en el diccionario se eligen en

[17] Esta probabilidad puede ser cero para algunas categorías, como por ejemplo las preposiciones.

función de las terminaciones de dichas palabras. La distribución de probabilidad para un sufijo particular se genera a partir de todas las palabras del corpus de entrenamiento que comparten ese mismo sufijo. El término sufijo tal y como se utiliza aquí significa *secuencia final de caracteres de una palabra*, lo cual no coincide necesariamente con el significado lingüístico de sufijo. Por esta razón, el método requiere una definición previa de la longitud máxima de sufijos que se va a considerar.

Las probabilidades se suavizan mediante un procedimiento de abstracción sucesiva. Esto permite calcular $P(t|l_{n-m+1},\ldots,l_n)$, la probabilidad de una etiqueta t dadas las últimas m letras l_i de una palabra, a través de una secuencia de contextos más generales que omite un caracter del sufijo en cada iteración, de forma que la suavización se lleva a cabo en base a $P(t|l_{n-m+2},\ldots,l_n)$, $P(t|l_{n-m+3},\ldots,l_n)$, $\ldots$, $P(t)$. La fórmula de la recursión es:

$$P(t|l_{n-i+1},\ldots,l_n) = \frac{\hat{p}(t|l_{n-i+1},\ldots,l_n) + \theta_i\, P(t|l_{n-i+2},\ldots,l_n)}{1+\theta_i}$$

para $i = m,\ldots,1$, utilizando la estimación de máxima verosimilitud para un sufijo de longitud i calculada a partir de las frecuencias del corpus de entrenamiento como

$$\hat{p}(t|l_{n-i+1},\ldots,l_n) = \frac{C(t,l_{n-i+1},\ldots,l_n)}{C(l_{n-i+1},\ldots,l_n)},$$

utilizando también unos determinados pesos de suavización θ_i, y utilizando como caso base $P(t)=\hat{p}(t)$. Por supuesto, para el modelo de Markov, necesitamos las probabilidades condicionadas inversas $P(l_{n-i+1},\ldots,l_n|t)$, las cuales se obtienen por la regla de Bayes.

Como se puede ver, la definición de este método no es rigurosa en todos y cada uno de sus aspectos, lo cual da lugar a diferentes interpretaciones a la hora de aplicarlo. Por ejemplo:

- Es necesario identificar un buen valor para m, la longitud máxima utilizada para los sufijos. Brants elige para su etiquetador el criterio de que m depende de cada palabra en cuestión [Brants 2000], y utiliza para dicha palabra el sufijo más largo de entre todos los que hayan sido observados al menos una vez en el corpus de entrenamiento, pero como máximo $m=10$ caracteres.
- Brants utiliza también una elección independiente del contexto para los pesos de suavización de sufijos θ_i, al igual que hacía con los parámetros de interpolación lineal de trigramas λ_i. Es decir, en lugar de calcular los θ_i tal y como propuso Samuelsson, a partir de la desviación estándar de las probabilidades de máxima verosimilitud de los sufijos, Brants asigna a todos los θ_i la desviación estándar de las probabilidades de máxima verosimilitud no condicionadas de las etiquetas en el corpus de entrenamiento:

  $$\theta_i = \frac{1}{s-1}\sum_{j=1}^{s}(\hat{p}(t^j)-\bar{P})^2$$

  para todo $i=m,\ldots,1$, donde s es el cardinal del conjunto de etiquetas utilizado, y la media $\bar{P}$ se calcula como

  $$\bar{P} = \frac{1}{s}\sum_{j=1}^{s}\hat{p}(t^j).$$

- Otro grado de libertad del método es el que concierne a la elección de las palabras del corpus de entrenamiento que se utilizan para la extracción de los sufijos. ¿Deberíamos utilizar todas las palabras, o algunas son más adecuadas que otras? Dado que las palabras desconocidas son probablemente las más infrecuentes, se puede argumentar

> que el uso de los sufijos de las palabras menos frecuentes del corpus constituye una aproximación mejor para las palabras desconocidas que los sufijos de las palabras frecuentes. Brants, por tanto, realiza la extracción de sufijos sólo sobre aquellas palabras cuya frecuencia sea menor o igual que un cierto umbral de corte, y fija dicho umbral empíricamente en 10 apariciones. Además, mantiene dos pruebas de sufijos separadas, en función de si la inicial de la palabra es mayúscula o minúscula.

En general, para las palabras desconocidas, $P(w|t) = 0$ estrictamente hablando. En caso contrario, la palabra no sería desconocida. Pero tal y como hemos dicho, una de las principales habilidades que debe incluir un buen etiquetador es asignar una probabilidad $P(w|t)$ distinta de cero para las palabras no presentes en el diccionario. La única restricción que hay que respetar es que, para toda etiqueta t del conjunto de etiquetas, debería cumplirse

$$\sum_{w} P(w|t) = 1, \tag{4.16}$$

sobre todas las palabras w: las conocidas y las, en teoría, posiblemente desconocidas. Ésta es la clave del problema, porque implica que deberíamos conocer *a priori* todas las palabras desconocidas. En la práctica, esto no es así, y por tanto no conocemos completamente el conjunto de las probabilidades que estamos sumando. Una forma de afrontar el problema es considerar que no todas las palabras desconocidas juegan el mismo papel. Algunas de ellas serán efectivamente palabras nuevas para las cuales queremos probabilidades de emisión distintas de cero, pero otras podrían ser palabras con errores ortográficos para las cuales queremos probabilidades igual a cero o, en todo caso, las probabilidades de las correspondientes palabras corregidas.

Así pues, si el conjunto de palabras que van a ser aceptadas como desconocidas es *de alguna manera* conocido, entonces se puede dar un significado a la probabilidad $P(w|t)$, ya que la restricción (4.16) podría ser calculada y verificada. Si esta restricción se respeta, entonces podemos elegir la definición de $P(w|t)$ que queramos, y una definición basada en la morfología es, por supuesto, una buena elección. En otras palabras, definir $P(w|t)$ mediante $P(\mathit{sufijo}(w)|t)$ para las palabras desconocidas es una buena idea siempre que se cumpla la restricción (4.16).

Para ello, se podrían combinar las probabilidades de los sufijos con las probabilidades de emisión de las palabras conocidas, reservando una parte de la masa de probabilidad de éstas últimas. Es decir, si conocida la etiqueta t y para toda palabra w del diccionario,

$$S = \sum_{w} P(w|t),$$

donde $S < 1$, entonces $1 - S$ sería la masa de probabilidad dedicada a las palabras desconocidas, conocida t. $P(\mathit{sufijo}|t)$ debería ser una distribución de probabilidad, de tal forma que

$$\sum_{\mathit{sufijo}} P(\mathit{sufijo}|t) = 1, \tag{4.17}$$

para toda etiqueta t del conjunto de etiquetas utilizado. Y los sufijos deberían estar correctamente definidos, de tal manera que la función $\mathit{sufijo}(w)$ fuera una aplicación. Es decir, el sufijo de w debería ser único para cada palabra w. De esta forma, $P(w|t)$ se define para las palabras desconocidas como

$$P(w|t) = (1 - S)\, P(\mathit{sufijo}(w)|t),$$

lo cual no sólo es una definición formal, sino también sensible a las propiedades morfológicas de cada palabra. Las dificultades, por supuesto, radican en la identificación de los sufijos válidos para las palabras y en la estimación de un buen valor para S.

Tal y como vimos antes, la implementación que hace Brants del método de Samuelsson no mezcla las probabilidades de los sufijos con las probabilidades de las palabras conocidas, y por tanto no respeta la restricción (4.16). Sin embargo, dicho método construye un conjunto de probabilidades de emisión distinto para cada longitud de sufijo, y cada uno de esos conjuntos verifica la restricción (4.17). Como se puede apreciar, el enfoque es totalmente distinto, pero el método también genera probabilidades directamente utilizables por los modelos de Markov.

En definitiva, el tratamiento de palabras desconocidas[18] representa el punto más débil de la aplicación de los HMM,s al proceso de etiquetación del lenguaje natural, y no se incluye normalmente en el estado del arte de los HMM,s. En esta sección hemos esbozado sólo algunos de los métodos de tratamiento de palabras desconocidas que han sido desarrollados e integrados formalmente dentro del marco de los HMM,s.

4.5 Estimación de probabilidades a partir de contadores

En las secciones previas hemos asumido tácitamente que las frecuencias relativas de los sucesos son una buena estimación de sus probabilidades. La conocida ley de los grandes números nos asegura que hasta cierto punto esto es cierto. En particular, si $\mathcal{X}$ es el conjunto de posibles sucesos al repetir N experimentos distribuidos idénticamente, y si $C_N(x)$ es el número de veces que el suceso $x \in \mathcal{X}$ fue observado, entonces

$$P\left[\lim_{N\to\infty} \frac{C_N(x)}{N} = P(x)\right] = 1, \qquad \forall x \in \mathcal{X},$$

donde $P(x)$ denota la probabilidad de que un experimento dado dé como resultado el suceso x.

Ya que nuestra definición de frecuencia relativa $f(x)$ es la proporción $\frac{C_N(x)}{N}$, podría parecer que nuestro método de estimación de $P(x)$ es el mejor posible. Pero tenemos al menos tres problemas:

1. La estimación $\frac{C_N(x)}{N}$ para una determinada cantidad finita N de datos de entrenamiento puede no ser suficiente.

2. En muchas situaciones de interés, el espacio de sucesos $\mathcal{X}$ no se conoce en toda su totalidad. Es decir, el conjunto de sucesos que tienen lugar en los datos de entrenamiento puede ser sólo un subconjunto de $\mathcal{X}$.

3. Incluso cuando $\mathcal{X}$ es conocido, puede ser tan grande[19] que para muchos de los $x \in \mathcal{X}$ la probabilidad verdadera satisface $0 < P(x) \ll \frac{1}{N}$, de manera que el hecho de que x no haya sido observado en los datos de entrenamiento no implica en absoluto que su probabilidad deba ser 0.

Y todavía podrían surgir otras muchas cuestiones. Por ejemplo, el tercero de los problemas anteriores nos lleva a preguntarnos no sólo si un suceso no observado debería tener probabilidad 0 o no, sino también cuánto más grande debería ser la probabilidad de los sucesos observados una vez respecto a los no observados ninguna vez, o, en general, si la proporción de las probabilidades de dos sucesos observados n y m veces, respectivamente, debería ser realmente n/m.

En esta sección introduciremos primero un método de estimación de las probabilidades deseadas que emplea una aproximación basada en *datos extendidos* o *held-out data*. Posteriormente, una variante de esta aproximación nos llevará a las conocidas fórmulas de Good-Turing [Church y Gale 1991]. Y finalizaremos con una explicación detallada del método de

[18] También denominado *word guessing*.

[19] Por ejemplo, el conjunto de los posibles trigramas de etiquetas $t^i t^j t^k$ de un juego de etiquetas dado.

estimación de *marcha atrás* o *back-off* [Katz 1987], el cual constituye la base del funcionamiento de muchos de los modelos actuales de procesamiento automático del lenguaje.

4.5.1 Método de estimación mediante *held-out data*

La idea básica. En este método de estimación, la estrategia a seguir consiste en dividir los datos de entrenamiento $\mathcal{X}$ en dos partes [Jelinek y Mercer 1985]. La primera parte, llamada $\mathcal{D}$ (*datos de desarrollo* o *development data*), se usa para recolectar los contadores $C_d(x)$ de los sucesos x. La segunda parte, llamada $\mathcal{H}$ (*datos extendidos* o *held-out data*), se utilizará para estimar los parámetros adicionales que determinarán el valor final de los $\hat{p}(x)$, nuestras estimaciones de las probabilidades $P(x)$. Dichas estimaciones tendrán la siguiente estructura:

$$\hat{p}(x) = \begin{cases} q_i & \text{para todo } x \text{ tal que } C_d(x) = i, \text{donde } i = 0, 1, \ldots, M, \\ \alpha f_d(x) & \text{para todo } x \text{ tal que } C_d(x) > M, \text{donde } f_d(x) = \frac{C_d(x)}{N_d}, \end{cases} \tag{4.18}$$

y donde N_d es el tamaño del conjunto de datos $\mathcal{D}$. Los datos de $\mathcal{H}$ se utilizarán para encontrar los valores óptimos de los parámetros q_i y α que satisfagan la normalización

$$\sum_{x \in \mathcal{X}} \hat{p}(x) = 1. \tag{4.19}$$

La idea intuitiva que subyace en la ecuación (4.18) es que todos los sucesos observados el mismo número de veces i, con $i \in \{0, 1, \ldots, M\}$, deberían tener la misma probabilidad. El segundo caso, cuando $C_d(x) > M$, representa la suavización. Más adelante discutiremos cómo elegir un valor apropiado para el umbral de confianza M.

Por el momento, denotemos mediante n_i el número de símbolos diferentes $x \in \mathcal{X}$ para los cuales $C_d(x) = i$. Es decir,

$$n_i = \sum_{x \in \mathcal{X}} \delta(C_d(x), i)$$

donde la expresión $\delta(a, b)$ es la función de Kronecker definida como:

$$\delta(a, b) = \begin{cases} 1 & \text{si } a = b, \\ 0 & \text{en caso contrario.} \end{cases}$$

Entonces podemos dar una interpretación basada en clases a la ecuación (4.18). Un suceso x pertenecerá a la clase Φ_i si $C_d(x) = i$, para $i \in \{0, 1, \ldots, M\}$, y pertenecerá a la clase Φ_{M+1} si $C_d(x) > M$. Entonces

$$\hat{p}(x) = P(\Phi(x))\, P(x|\Phi(x)) \tag{4.20}$$

donde

$$P(x|\Phi(x)) = \begin{cases} \frac{1}{n_i} & \text{si } \Phi(x) = \Phi_i, \text{ donde } i = 0, 1, \ldots, M, \\ \frac{C_d(x)}{\sum_{j>M} j\, n_j} & \text{si } \Phi(x) = \Phi_{M+1}, \end{cases} \tag{4.21}$$

y utilizaremos los datos de $\mathcal{H}$ para estimar las probabilidades de ocupación $P(\Phi_i)$. Por supuesto, $q_i = P(\Phi_i)/n_i$ y $\alpha = P(\Phi_{M+1}) N_d / \sum_{j>M} j\, n_j$.

La estimación. Para determinar los valores de máxima verosimilitud de los q_i y de α, necesitamos notaciones adicionales. Denotaremos mediante:

- $C_h(x)$, el número de veces que el símbolo x aparece en $\mathcal{H}$,

- r_i, el número de veces que se han visto en $\mathcal{H}$ símbolos x tales que $C_d(x) = i$, es decir,

$$r_i = \sum_{x \in \mathcal{X}} C_h(x)\, \delta(C_d(x), i),$$

- r^*, el número de veces que $\mathcal{H}$ contiene símbolos x tales que $C_d(x) > M$, es decir,

$$r^* = \sum_{i>M} r_i,$$

- $|\mathcal{H}|$, el tamaño total del conjunto $\mathcal{H}$, es decir,

$$|\mathcal{H}| = \left(\sum_{i=0}^{M} r_i \right) + r^*. \tag{4.22}$$

La probabilidad de los datos de $\mathcal{H}$ calculada según la ecuación (4.18) es entonces igual a

$$P(\mathcal{H}) = \left(\prod_{i=0}^{M} (q_i)^{r_i} \right) \alpha^{r^*} K \tag{4.23}$$

donde

$$K = \prod_{x:C_d(x)>M} f_d(x)^{C_h(x)}.$$

Ahora podemos elegir los valores de q_i y α que maximizan la ecuación (4.23) y mantienen la restricción de la ecuación (4.19) mediante el método de los multiplicadores indeterminados de Lagrange. Esto es, lo que buscamos son las soluciones del siguiente conjunto de ecuaciones:

$$\frac{\partial}{\partial q_j} \left[P(\mathcal{H}) - \lambda \left(\sum_{i=1}^{M} n_i q_i + \alpha P_M \right) \right] = 0 \qquad j = 0, 1, \ldots, M$$

$$\frac{\partial}{\partial \alpha} \left[P(\mathcal{H}) - \lambda \left(\sum_{i=1}^{M} n_i q_i + \alpha P_M \right) \right] = 0$$

$$\sum_{i=1}^{M} n_i q_i + \alpha P_M = 1 \tag{4.24}$$

donde

$$P_M = \sum_{x:C_d(x)>M} f_d(x) = \frac{1}{N_d} \sum_{i>M} i\, n_i.$$

Calculando las derivadas parciales se sigue que

$$\frac{r_j}{q_j} P(\mathcal{H}) = \lambda n_j, \qquad j = 0, 1, \ldots, M,$$

y

$$\frac{r^*}{\alpha} P(\mathcal{H}) = \lambda P_M$$

y substituyéndolas en la ecuacion (4.24), se obtiene

$$\frac{P(\mathcal{H})}{\lambda} \left(\left(\sum_{i=0}^{M} r_i \right) + r^* \right) = 1.$$

Utilizando la definición (4.22), la solución final es

$$q_j = \frac{1}{n_j}\frac{r_j}{|\mathcal{H}|}, \qquad j = 0, 1, \ldots, M, \tag{4.25}$$

y

$$\alpha = \frac{1}{P_M}\frac{r^*}{|\mathcal{H}|}. \tag{4.26}$$

Es interesante resaltar en particular que

$$q_0 = \frac{1}{n_0}\frac{r_0}{|\mathcal{H}|} > 0$$

y que

$$\frac{q_{j+1}}{q_j} = \frac{r_{j+1}}{r_j}\frac{n_j}{n_{j+1}}, \qquad j = 0, 1, \ldots, M,$$

lo cual no es igual a $(j+1)/j$, el valor que habríamos obtenido si las probabilidades hubieran sido estimadas directamente mediante frecuencias relativas.

Elección del valor de M. Claramente, lo que queremos es que $\hat{p}(x)$ sea una función creciente de los contadores $C_d(x)$. Por tanto, debemos elegir M de manera que $q_j < q_{j+1}$ para $j = 0, 1, \ldots, M$. Esto quiere decir que debe verificarse

$$r_j\, n_{j+1} < r_{j+1}\, n_j, \qquad j = 0, 1, \ldots, M-1,$$

y

$$\frac{r_M}{n_M} < \frac{\sum_{j>M} r_j}{\sum_{j>M} j\, n_j}(M+1).$$

Utilizando los valores de (4.25) y (4.26), M debería ser elegido de forma que

$$\frac{r_M}{M\, n_M} \cong \frac{\sum_{j>M} r_j}{\sum_{j>M} j\, n_j}.$$

Universalidad de la estimación *held-out*. Para que la estimación de (4.18) con valores (4.25) y (4.26) que acabamos de obtener tenga una aplicabilidad universal sobre el tipo de datos en los que está basado el conjunto de entrenamiento, sólo necesitamos hacer uso de la interpretación de clases de las ecuaciones (4.20) y (4.21) y reescribir la fórmula (4.18) como sigue:

$$\hat{p}(x) = \begin{cases} \lambda_i \frac{1}{n_i} & \text{para todo } x \text{ tal que } C(x) = i, \text{donde } i = 0, 1, \ldots, M, \\ \lambda_{M+1} \frac{C(x)}{\sum_{i>M} i\, n_i} & \text{para todo } x \text{ tal que } C(x) > M. \end{cases} \tag{4.27}$$

De esta manera, los coeficientes λ_i vienen a ser los pesos de un esquema de interpolación de borrado[20].

Comparando (4.18) y (4.27) obtenemos las siguientes relaciones:

$$\lambda_i = q_i n_i^d = \frac{r_i}{|\mathcal{H}|}, \qquad i = 0, 1, \ldots, M, \tag{4.28}$$

y

$$\lambda_{M+1} = \alpha \frac{\sum_{i>M} i\, n_i^d}{N_d} = \frac{r^*}{|\mathcal{H}|}$$

[20] *Deleted interpolation.*

donde se ha utilizado una d diacrítica en n_i^d y N_d para indicar que estamos tratando con las cantidades correspondientes a un conjunto particular $\mathcal{D}$. Por tanto, primero utilizamos los contadores de la parte $\mathcal{D}$ para calcular los λ_i, y una vez que estos valores están fijados mejoramos las estimaciones finales $\hat{p}(x)$ mediante los cálculos de la fórmula (4.27), la cual utiliza los contadores $C(x)$ obtenidos no sólo de la parte $\mathcal{D}$, sino a lo largo de todo el corpus de entrenamiento disponible ($\mathcal{D} \cup \mathcal{H}$).

4.5.2 Método de estimación de Good-Turing

Existe otra fórmula particular para estimar probabilidades a partir de contadores, que no depende directamente de una división del corpus de entrenamiento en los conjuntos de datos $\mathcal{D}$ (*development*) y $\mathcal{H}$ (*held-out*). Se trata de la estimación de Good-Turing, la cual se puede derivar mediante varios métodos. Una de las posibles derivaciones, la que veremos a continuación, está fuertemente relacionada con el concepto de universalidad que hemos visto anteriormente [Nadas 1991].

Primero creamos una serie de conjuntos pseudo-*held-out* mediante el método de *dejar uno fuera*[21] [Quenouille 1949, Ney *et al.* 1995]. Esto es, creamos N pares de conjuntos diferentes $\mathcal{D}_j$ y $\mathcal{H}_j$ a partir del conjunto de entrenamiento $\mathcal{T}$ de tamaño N, simplemente omitiendo cada uno de los elementos x_j de $\mathcal{T} = \{x_1, x_2, \ldots, x_N\}$. Por tanto, nuestros pares de conjuntos *development/held-out* serán de la forma:

$$\mathcal{D}_j = \mathcal{T} - x_j, \qquad \mathcal{H}_j = \{x_j\}, \qquad j = 1, 2, \ldots, N.$$

Lo que haremos ahora será predecir $\mathcal{H}_j$ en base a las estimaciones obtenidas a partir de $\mathcal{D}_j$, y elegiremos nuestros parámetros de manera que se maximice el producto

$$\prod_{j=1}^{N} P_j(\mathcal{H}_j) = \prod_{j=1}^{N} P_j(x_j)$$

donde $P_j(x_j)$ denota la probabilidad de x_j en base a los contadores obtenidos a partir del conjunto $\mathcal{D}_j$.

Sean $C(x)$ y n_i los contadores obtenidos a partir del conjunto completo de entrenamiento $\mathcal{T}$. Es decir, n_i es el número de símbolos diferentes x para los cuales $C(x) = i$. Denotemos también mediante $C_j(x)$ el contador de x en $\mathcal{D}_j$, y mediante r_i el número de veces que se han visto en la totalidad de los conjuntos *held-out* $\mathcal{H}_j$, $j = 1, 2, \ldots, N$, símbolos x tales que $C_j(x) = i$. Entonces,

$$r_i = (i+1)\, n_{i+1}. \tag{4.29}$$

En efecto, si $\mathcal{H}_j = x$, entonces $C_j(x) = C(x) - 1$. Por tanto, si $C_j(x) = i$ entonces $C(x) = i+1$. Y además, en $\mathcal{T}$ existen n_{i+1} de esos elementos x, y por tanto existen $(i+1)\, n_{i+1}$ valores diferentes de $j \in \{1, 2, \ldots, N\}$ para los cuales siendo $\mathcal{H}_j = x$ se verifica que $C_j(x) = i$, lo cual demuestra la certeza de la ecuación (4.29).

Dado que en este caso $|\mathcal{H}| = N$, obtenemos a partir de (4.25) que

$$q_j = \frac{n_{j+1}}{n_j} \frac{j+1}{N}, \qquad j = 0, 1, \ldots, M. \tag{4.30}$$

El valor de α se obtiene a partir de la normalización $\sum \hat{p}(x) = 1$. Esto es, a partir de

$$\sum_{j=0}^{M} q_j\, n_j + \alpha \sum_{j>M} \frac{j}{N}\, n_j = 1$$

[21] También denominado método *leave-one-out*.

y de (4.30) obtenemos

$$\alpha = \frac{\sum_{j>M+1} j\, n_j}{\sum_{j>M} j\, n_j}. \tag{4.31}$$

De esta manera, hemos derivado en (4.30) y (4.31) las conocidas fórmulas de Good-Turing, y la restricción de monotonía natural $q_{j-1} < q_j$ requiere ahora la elección de un umbral de confianza M que satisfaga:

$$(n_j)^2 < \frac{j+1}{j} n_{j-1}\, n_{j+1}, \qquad j = 1, 2, \ldots, M, \tag{4.32}$$

y

$$\frac{n_{M+1}}{n_M} < \frac{\sum_{j>M+1} j\, n_j}{\sum_{j>M} j\, n_j}. \tag{4.33}$$

Queremos llamar ahora la atención sobre el hecho de que aunque los números de aparición n_j para $j = 1, 2, \ldots, M+1$ son los que realmente se observan en el conjunto de datos $\mathcal{T}$, el número n_0 es diferente. Este número se infiere como el tamaño total del espacio de sucesos $\mathcal{X}$, que presumiblemente conocemos, menos el tamaño del subespacio $\mathcal{X}_{\mathcal{T}}$ observado realmente en $\mathcal{T}$. Es decir,

$$n_0 = |\mathcal{X}| - \sum_{i=1}^{\infty} n_i.$$

Por ejemplo, si $\mathcal{X}$ es el conjunto de todos los trigramas de $t^i t^j t^k$ pertenecientes a un juego de etiquetas dado de cardinal t, entonces $|\mathcal{X}| = t^3$, mientras que $\mathcal{X}_{\mathcal{T}}$ consistiría simplemente en el conjunto de todos los trigramas diferentes observados realmente en los datos de entrenamiento $\mathcal{T}$. Es interesante entonces destacar a partir de la ecuación (4.30) que la probabilidad total $q_0\, n_0$ asignada por las fórmulas de Good-Turing a los sucesos no observados es igual a n_1/N, esto es, la probabilidad total que sería asignada a los sucesos observados una sola vez mediante una fórmula de frecuencias relativas.

Observamos también a partir de la ecuación (4.31) que, en comparación con la fórmula de frecuencias relativas, el método de Good-Turing descuenta a los sucesos de frecuencia alta la cantidad

$$\frac{(M+1)\, n_{M+1}}{\sum_{j>M} j\, n_j}.$$

De hecho, la manera en que se dota de probabilidad a los sucesos no observados es a través de una secuencia de frecuencias relativas desplazadas desde los contadores más altos a los más bajos.

Finalmente, dado que M es normalmente un número entero pequeño, es importante introducir la fórmula de optimización

$$\sum_{j>M} j\, n_j = N - \sum_{i=1}^{M} i\, n_i$$

que indica que el sumatorio de la parte izquierda de la igualdad se puede calcular rápidamente mediante un sumatorio mucho más sencillo, el de la parte derecha, y una resta posterior.

4.5.3 Aplicabilidad de las estimaciones *held-out* y Good-Turing

Está claro que el método de estimación de Good-Turing fue diseñado para situaciones en las cuales $n_0 \gg n_1 \gg n_2$, es decir, para casos en los que hay muchos sucesos no observados y los datos aparecen realmente dispersos. De hecho, la fórmula (4.30) no es aplicable para $j = 0$ cuando todos los elementos de $\mathcal{X}$ han sido observados en los datos de entrenamiento, es decir, cuando $n_0 = 0$. Por otra parte, el estimador *held-out* siempre es aplicable: si $n_0 = 0$, entonces necesariamente $r_0 = 0$, pero la fórmula (4.28) permanece válida. No obstante, es fácil incurrir en usos abusivos de este método, por ejemplo cuando se considera un conjunto de datos *held-out* excesivamente pequeño, o simplemente cuando la partición entre los datos de desarrollo y los datos extendidos no está correctamente balanceada.

En todo caso, si el método de estimación mediante *held-out data* es siempre válido, ¿por qué no lo es el método de Good-Turing, que ha sido derivado a partir del estimador *held-out*? La razón es que hemos asumido desde el principio una cierta regularidad en nuestras observaciones de sucesos. Suponemos que los sucesos que fueron observados una sola vez son muy raros y perfectamente podían no haber sido observados en absoluto. O, en general, que los sucesos que fueron observados $j+1$ veces también podían haber sido observados sólo j veces, y esta situación queda perfectamente representada por los cálculos del método de *dejar uno fuera*, más robustos en este sentido. Por supuesto, esto es asumible sólo para grandes espacios de sucesos, donde además los sucesos son, por así decirlo, anónimos. Es decir, la frecuencia de aparición de un suceso no tiene nada que ver con la frecuencia de aparición de otro suceso diferente.

Cuando trabajamos con modelos de lenguaje basados en trigramas, estamos interesados en probabilidades de la forma $P(t_i|t_{i-2}, t_{i-1})$, que, una vez más, abusando un poco de la notación, y para no complicar en exceso las ecuaciones con la presencia de tantos subíndices, denotaremos mediante $P(t_3|t_1, t_2)$, donde de manera obvia los números 1, 2 y 3 hacen referencia a la posición de cada etiqueta dentro del trigrama. En este caso, existe una tentación obvia de utilizar el método de estimación de Good-Turing directamente, es decir, de usar contadores $n_i(t_1, t_2)$ del número de etiquetas diferentes t_3 que han seguido al bigrama de etiquetas $t_1\, t_2$ exactamente i veces en los datos de entrenamiento. Sin embargo, debemos resistir esta tentación porque en la mayoría de los casos esos contadores podrían estar basados en datos muy reducidos. Para modelos de lenguaje basados en trigramas lo que realmente necesitaríamos calcular es

$$P(t_3|t_1, t_2) = \frac{P_T(t_1, t_2, t_3)}{\sum_{t_3'} P_T(t_1, t_2, t_3')} \tag{4.34}$$

donde sólo las probabilidades P_T serían el resultado de una estimación Good-Turing. La fórmula (4.34) sería computacionalmente bastante factible, ya que el denominador se puede calcular de manera previa:

$$\sum_{t_3'} P_T(t_1, t_2, t_3') = \sum_{i=0}^{M} q_i\, n_i(t_1, t_2) + \frac{\alpha}{N} \sum_{i=M+1}^{\infty} i\, n_i(t_1, t_2)$$

donde q_i y α vendrían dados por las ecuaciones (4.30) y (4.31), respectivamente.

4.5.4 Mejora de los métodos de estimación

Ambos métodos de estimación, *held-out* y Good-Turing, están basados en la estructura (4.18), la cual dice que la probabilidad de cada elemento de $\mathcal{X}$ debería determinarse mediante su frecuencia en el conjunto de datos de desarrollo. Pero podría ocurrir que tuviéramos algún tipo de información disponible que nos permitiera hacer distinciones entre elementos con la misma frecuencia de aparición. Church y Gale apuntaron ideas sobre cómo se puede hacer uso de esta información adicional [Church y Gale 1991].

Supongamos que el espacio de sucesos $\mathcal{X}$ es el conjunto de los bigramas de $t^i\,t^j$ pertenecientes a un juego de etiquetas dado. ¿Existe alguna razón para pensar que dos bigramas diferentes, ninguno de los cuales ha sido observado, puedan tener probabilidades diferentes? He aquí un posible argumento: si tanto t^i como t^j son frecuentes, entonces es muy probable que no haya sido simplemente mala suerte el hecho de que por ejemplo el bigrama $t^i\,t^j$ no aparezca; por otra parte, si t^i y t^j no son frecuentes, su no aparición no debería implicar automáticamente una *alergia* entre ellos.

Por tanto, una manera de obtener una estimación mejor es categorizar o clasificar cada bigrama $t^i\,t^j$, no sólo por su frecuencia de aparición, sino también por su producto de probabilidad $\hat{p}(t^i)\,\hat{p}(t^j)$, donde normalmente la estimación $\hat{p}(t) = f(t)$, porque dado que todos los unigramas aparecen, la frecuencia $f(t)$ sí estará basada en datos suficientes como para constituir una estimación precisa de esa probabilidad. Sin embargo, no será posible basar las categorías de sucesos en el valor exacto $\hat{p}(t^i)\,\hat{p}(t^j)$. Es cierto que se podría realizar una ponderación utilizando una función continua sobre esos valores. Pero en este caso nos interesa establecer una clasificación, de manera que habrá que particionar el intervalo $[0, 1]$ en subintervalos y clasificar $t^i\,t^j$ mediante el subintervalo en el que cae $\hat{p}(t^i)\,\hat{p}(t^j)$. Church y Gale recomiendan que esos subintervalos sean logarítmicos y que su tamaño exacto se determine de manera que la clase o subintervalo correspondiente contenga un número suficiente de bigramas diferentes. Resulta intuitivamente obvio que los contadores $C(t^i, t^j)$ bajos deberían dividirse en más subclases que los contadores altos.

Para encontrar las fórmulas de esta estimación mejorada, basta pensar en una interpretación basada en clases de la estructura (4.18), que ya habíamos presentado en las ecuaciones (4.20) y (4.21). En esta interpretación, dos sucesos x e y pertenecían a una misma clase Φ_i, si $C(x) = C(y) = i$. Pues bien, considerando de nuevo esta interpretación, encontramos que todas las ecuaciones finales, (4.25) y (4.26) para la estimación *held-out*, y (4.30) y (4.31) para la estimación Good-Turing, pueden permanecer invariables con sólo definir una partición de clases Φ basada en otro criterio de clasificación diferente [Jelinek 1997, pp. 268-270]. Es decir, considerando una partición que clasifique un suceso x mediante un criterio más acorde con las ideas expuestas en el párrafo anterior, y no basado simplemente en los contadores de aparición $C(x)$, las fórmulas de estimación ya vistas continuarán siendo válidas y dichas estimaciones mejorarán, en la medida en que Φ agrupe mejor los sucesos.

Debido a esto, basta cambiar Φ para aplicar diferentes criterios de clasificación, y por tanto existirán muchas posibles mejoras. La siguiente sección presta atención a la principal de todas ellas: el método de estimación de *marcha atrás* o *back-off*.

4.5.5 Método de estimación *back-off*

Tal y como hemos apuntado ya desde las primeras secciones del presente capítulo, cualquier estimación de las probabilidades de un modelo de lenguaje que dependan de una cierta *historia*, es decir, no basados simplemente en los unigramas, necesariamente sufre el problema de los datos dispersos. Hasta ahora, habíamos intentado aliviar este problema mediante interpolaciones lineales. Por ejemplo, para los trigramas de etiquetas, definíamos las probabilidades del modelo mediante

$$P(t_3|t_1, t_2) = \lambda_3\, f(t_3|t_1, t_2) + \lambda_2\, f(t_3|t_2) + \lambda_1\, f(t_3) \tag{4.35}$$

donde los pesos λ_i se estimaban mediante datos *held-out*. Sin embargo, Katz argumentó que si el contador $C(t_1, t_2, t_3)$ es suficientemente grande, entonces $f(t_3|t_1, t_2)$ es en sí mismo un criterio de categorización que porporciona una estimación mucho mejor de $P(t_3|t_1, t_2)$ que la mostrada anteriormente en la ecuación (4.35) [Katz 1987], de manera que el esquema

$$\hat{p}(t_3|t_1,t_2) = \begin{cases} f(t_3|t_1,t_2) & \text{si } C(t_1,t_2,t_3) \geq K \\ \alpha\, Q_T(t_3|t_1,t_2) & \text{si } 1 \leq C(t_1,t_2,t_3) < K \\ \beta(t_1,t_2)\,\hat{p}(t_3|t_2) & \text{en caso contrario} \end{cases} \tag{4.36}$$

constituye una aproximación superior, una vez que los coeficientes α y $\beta(t_1,t_2)$ hayan sido convenientemente elegidos[22]. En el esquema (4.36), $Q_T(t_3|t_1,t_2)$ es una función de tipo Good-Turing, tal y como veremos más adelante, y $\hat{p}(t_3|t_2)$ es la probabilidad del bigrama estimada mediante un esquema similar:

$$\hat{p}(t_3|t_2) = \begin{cases} f(t_3|t_2) & \text{si } C(t_2,t_3) \geq K \\ \alpha'\, Q_T(t_3|t_2) & \text{si } 1 \leq C(t_2,t_3) < K \\ \beta(t_2)\, f(t_3) & \text{en caso contrario.} \end{cases} \tag{4.37}$$

Por lo tanto, la definición de la ecuación (4.36) es una definición recursiva[23].

¿Cómo determinamos las funciones Q_T? Si el umbral de confianza K fuera infinito, Katz sugiere que $\alpha = 1$, y tenemos por definición que

$$Q_T(t_3|t_1,t_2) = \frac{P_T(t_1,t_2,t_3)}{f(t_1,t_2)} \tag{4.38}$$

donde el numerador es la probabilidad Good-Turing obtenida a partir de la colección de contadores $C(t_1,t_2,t_3)$ mediante las fórmulas de la sección 4.5.2. La definición (4.38) de Q_T no constituye una probabilidad porque no está normalizada adecuadamente, lo cual ocurriría sólo si $f(t_1,t_2)$ fuera reemplazado por $\sum_{t'_3} P_T(t_1,t_2,t'_3)$. No obstante, la fórmula (4.38) es simple y permite una elección sencilla de los coeficientes α y β. Nos ocuparemos de la normalización de los $\hat{p}(t_3|t_1,t_2)$ resultantes al final de nuestro desarrollo.

Sin embargo, dado que K es finito (Katz recomienda $K \cong 6$), los cálculos son un poco más complejos. Nos interesa determinar α de manera que satisfaga el siguiente principio básico: la probabilidad total reservada a todos los trigramas no observados[24] debería ser n_1/N, tal y como prescribe el método de Good-Turing cuando $K = \infty$. Entonces tenemos que

$$n_1 = \sum_{t_1,t_2,t_3} \delta(C(t_1,t_2,t_3),1)$$

y

$$N = \sum_{t_1,t_2,t_3} C(t_1,t_2,t_3).$$

Ahora, mediante la fórmula (4.30),

$$P_T(t_1,t_2,t_3) = q_i = \frac{n_{i+1}}{n_i}\frac{i+1}{N}, \qquad \text{donde } i = C(t_1,t_2,t_3),$$

[22]Entre otras restricciones, se les exige que aseguren la normalización estándar de las probabilidades $\hat{p}(t_3|t_1,t_2)$; obsérvese también que el coeficiente α es constante, pero el coeficiente β es variable y depende del bigrama t_1t_2 que constituye la *historia* del trigrama $t_1t_2t_3$ que se está tratando.

[23]En ambos esquemas, (4.35) y (4.36), se asume que $f(t_1,t_2) > 0$; si esto no es así, entonces en (4.35) el primer sumando es 0, mientras que (4.36) simplemente no es aplicable y por definición tenemos que $\hat{p}(t_3|t_1,t_2) = \hat{p}(t_3|t_2)$, estando este último término definido por (4.37); nótese finalmente que, dado que utilizaremos siempre textos de entrenamiento en los cuales cada etiqueta estará siempre presente al menos una vez, $f(t_3) > 0$ para todo t_3 perteneciente al juego de etiquetas utilizado.

[24]Nos referimos realmente a la probabilidad de los trigramas, no a la probabilidad condicionada de la tercera etiqueta dadas las dos primeras.

y

$$f(t_1,t_2,t_3) = f_i = \frac{i}{N}, \qquad \text{donde } i = C(t_1,t_2,t_3),$$

y, por tanto, considerando la restricción $\sum \hat{p}(t_1,t_2,t_3) = 1$ y el principio citado anteriormente, la obtención de α es como sigue:

$$\sum_{i=K}^{\infty} f_i\, n_i + \alpha \sum_{i=1}^{K-1} q_i\, n_i + \frac{n_1}{N} = 1 \quad \Rightarrow \quad \sum_{i=K}^{\infty} \frac{i}{N}\, n_i + \alpha \sum_{i=1}^{K-1} \frac{n_{i+1}}{n_i} \frac{i+1}{N}\, n_i + \frac{n_1}{N} = 1 \quad \Rightarrow$$

$$\Rightarrow \quad \sum_{i=K}^{\infty} i\, n_i + \alpha \sum_{i=1}^{K-1} n_{i+1}\,(i+1) + n_1 = N \quad \Rightarrow \quad \sum_{i=K}^{\infty} i\, n_i + \alpha \underbrace{\sum_{i=1}^{K-1} (i+1)\, n_{i+1}}_{\substack{\parallel \\ \sum_{i=2}^{K} i\, n_i}} = \underbrace{N - n_1}_{\substack{\parallel \\ \sum_{i=2}^{\infty} i\, n_i}} \quad \Rightarrow$$

$$\Rightarrow \quad \alpha = \frac{\displaystyle\sum_{i=2}^{\infty} i\, n_i - \sum_{i=K}^{\infty} i\, n_i}{\displaystyle\sum_{i=2}^{K} i\, n_i} \quad \Rightarrow \quad \alpha = \frac{\displaystyle\sum_{i=2}^{K-1} i\, n_i}{\displaystyle\sum_{i=2}^{K} i\, n_i}.$$

Finalmente, para que los $\hat{p}(t_3|t_1,t_2)$ estén normalizados, es decir, para que $\sum_{t_3} \hat{p}(t_3|t_1,t_2) = 1$, $\beta(t_1,t_2)$ ha de satisfacer

$$\beta(t_1,t_2) \sum_{t_3 \in \varphi_0} \hat{p}(t_3|t_2) = 1 - \sum_{t_3 \in \varphi_K} f(t_3|t_1,t_2) - \alpha \sum_{t_3 \in \varphi_*} Q_T(t_3|t_1,t_2)$$

donde $\varphi_0 = \{t_3 : C(t_1,t_2,t_3) = 0\}$, $\varphi_K = \{t_3 : C(t_1,t_2,t_3) \geq K\}$, y $\varphi_* = \{t_3 : 1 \leq C(t_1,t_2,t_3) < K\}$.

De manera similar, a partir de las restricciones $\sum \hat{p}(t_2,t_3) = 1$ y $\sum_{t_3} \hat{p}(t_3|t_2) = 1$, es posible derivar las fórmulas para calcular el valor de α' y de $\beta(t_2)$, respectivamente, para el esquema (4.37).

Dichas fórmulas aparecen en el resumen que mostramos a continuación, el cual describe de nuevo los cálculos involucrados en los esquemas (4.36) y (4.37), pero esta vez con todos los detalles que resultan necesarios para poder realizar una correcta implementación del método de estimación *back-off* a partir de los contadores observados en los datos de entrenamiento. Así pues, de acuerdo con el esquema (4.36), $\hat{p}(t_3|t_1,t_2)$ se define como sigue:

- Si $C(t_1,t_2,t_3) \geq K$, $\hat{p}(t_3|t_1,t_2) = f(t_3|t_1,t_2)$, donde

$$f(t_3|t_1,t_2) = \frac{C(t_1,t_2,t_3)}{C(t_1,t_2)}.$$

- Si $1 \leq C(t_1,t_2,t_3) = i < K$, $\hat{p}(t_3|t_1,t_2) = \alpha\, Q_T(t_3|t_1,t_2)$, donde

$$\alpha = \frac{\displaystyle\sum_{j=2}^{K-1} j\, n_j}{\displaystyle\sum_{j=2}^{K} j\, n_j}$$

siendo n_j el número de trigramas diferentes cualesquiera que han aparecido j veces en los datos de entrenamiento, y donde

$$Q_T(t_3|t_1,t_2) = \frac{P_T(t_1,t_2,t_3)}{f(t_1,t_2)} = \frac{q_i}{f(t_1,t_2)} = \frac{\frac{n_{i+1}}{n_i}\frac{i+1}{N_3}}{\frac{C(t_1,t_2)}{N_2}}$$

siendo N_3 y N_2 el número total de trigramas y bigramas, respectivamente, que han sido observados en los datos de entrenamiento.

- Si $C(t_1,t_2,t_3) = 0$, $\hat{p}(t_3|t_1,t_2) = \beta(t_1,t_2)\,\hat{p}(t_3|t_2)$, donde

$$\beta(t_1,t_2) = \frac{1 - \displaystyle\sum_{t_3\in\varphi_K} f(t_3|t_1,t_2) - \alpha \sum_{t_3\in\varphi_*} Q_T(t_3|t_1,t_2)}{\displaystyle\sum_{t_3\in\varphi_0} \hat{p}(t_3|t_2)}$$

siendo $\varphi_K = \{t_3 : C(t_1,t_2,t_3) \geq K\}$, $\varphi_* = \{t_3 : 1 \leq C(t_1,t_2,t_3) < K\}$ y $\varphi_0 = \{t_3 : C(t_1,t_2,t_3) = 0\}$, y donde $\hat{p}(t_3|t_2)$ viene dado por los cálculos del esquema (4.37).

De acuerdo con el esquema (4.37), $\hat{p}(t_3|t_2)$ se define como sigue:

- Si $C(t_2,t_3) \geq K$, $\hat{p}(t_3|t_2) = f(t_3|t_2)$, donde

$$f(t_3|t_2) = \frac{C(t_2,t_3)}{C(t_2)}.$$

- Si $1 \leq C(t_2,t_3) = i < K$, $\hat{p}(t_3|t_2) = \alpha'\,Q_T(t_3|t_2)$, donde

$$\alpha' = \frac{\displaystyle\sum_{j=2}^{K-1} j\,n'_j}{\displaystyle\sum_{j=2}^{K} j\,n'_j}$$

siendo n'_j el número de bigramas diferentes cualesquiera que han aparecido j veces en los datos de entrenamiento, y donde

$$Q_T(t_3|t_2) = \frac{P_T(t_2,t_3)}{f(t_2)} = \frac{q'_i}{f(t_2)} = \frac{\frac{n'_{i+1}}{n'_i}\frac{i+1}{N_2}}{\frac{C(t_2)}{N_1}}$$

siendo N_2 y N_1 el número total de bigramas y unigramas, respectivamente, que han sido observados en los datos de entrenamiento.

- Si $C(t_2,t_3) = 0$, $\hat{p}(t_3|t_2) = \beta(t_2)\,f(t_3)$, donde

$$\beta(t_2) = \frac{1 - \displaystyle\sum_{t_3\in\varphi'_K} f(t_3|t_2) - \alpha' \sum_{t_3\in\varphi'_*} Q_T(t_3|t_2)}{\displaystyle\sum_{t_3\in\varphi'_0} f(t_3)}$$

siendo $\varphi'_K = \{t_3 : C(t_2,t_3) \geq K\}$, $\varphi'_* = \{t_3 : 1 \leq C(t_2,t_3) < K\}$ y $\varphi'_0 = \{t_3 : C(t_2,t_3) = 0\}$, y donde $f(t_3) = \frac{C(t_3)}{N_1}$.

Finalmente, queremos destacar que aunque el valor del coeficiente K es constante, no tiene porqué ser el mismo en ambos esquemas de estimación. Es decir, resulta perfectamente factible la consideración de un coeficiente K para el esquema (4.36), y de otro coeficiente K' distinto para el esquema (4.37). Es más, al igual que en el método de estimación Good-Turing calculábamos el valor del umbral de confianza M, ambos coeficientes K y K' también deberían ser convenientemente estimados directamente sobre los datos de entrenamiento mediante las restricciones (4.32) y (4.33), para poder efectuar una aplicación totalmente rigurosa del método de estimación *back-off*.

4.6 Variantes de implementación de los HMM,s

Las predicciones condicionadas sobre una historia o contexto de gran longitud no son siempre una buena idea. Por ejemplo, normalmente no existen dependencias sintácticas entre las palabras que aparecen antes y después de las comas. Por tanto, el conocimiento de la etiqueta que aparece antes de una coma no siempre ayuda a determinar correctamente la que aparece después. De hecho, ante este tipo de casos, un etiquetador basado en trigramas podría hacer predicciones peores que otro basado en bigramas, debido una vez más al fenómeno de los datos dispersos[25]. La técnica de interpolación lineal de las probabilidades de los unigramas, bigramas y trigramas presentada en las secciones anteriores, y el método de *back-off* que ya hemos descrito también detalladamente, son algunas de las aproximaciones que tratan de paliar este problema, pero no son las únicas.

Algunos autores han optado por aumentar de manera selectiva un modelo de Markov de orden bajo, tomando como base para ello el análisis de los errores cometidos y algún otro tipo de conocimiento lingüístico previo. Por ejemplo, Kupiec observó que un HMM de orden 1 se equivocaba sistemáticamente al etiquetar la secuencia de palabras `the bottom of` como artículo-adjetivo-preposición. Entonces extendió el modelo con una red especial para esta construcción, con el propósito de que el etiquetador pudiera aprender la imposibilidad de que una preposición siga a la secuencia artículo-adjetivo [Kupiec 1992]. En general, este método selecciona manualmente determinados estados del modelo y aumenta su orden, para casos donde la *memoria* de orden 1 no es suficiente.

Otro método relacionado es el de los Modelos de Markov de memoria variable o VMMM,s[26] [Schütze y Singer 1994, Triviño y Morales 2000]. Los VMMM,s presentan estados de diferentes *longitudes*, en lugar de los estados de *longitud fija* de los etiquetadores basados en bigramas o en trigramas. Un VMMM podría transitar desde un estado que recuerda las dos últimas etiquetas (correspondiente, por tanto, a un trigrama) a un estado que recuerda las tres últimas etiquetas (correspondiente a un cuatrigrama) y después a otro estado sin memoria (correspondiente a un unigrama). El número de símbolos a recordar para una determinada secuencia se determina durante el proceso de entrenamiento, en base a criterios teóricos. En contraste con la interpolación lineal, los VMMM,s condicionan la longitud de memoria utilizada durante la predicción a la secuencia actual, en lugar de considerar una suma ponderada fija para todas las secuencias. Los VMMM,s se construyen con estrategias descendentes mediante métodos de división de estados[27]. Una posible alternativa es construirlos con estrategias ascendentes mediante métodos de fusión de modelos[28] [Stolcke y Omohundro 1994, Brants 1998].

Otra aproximación que resulta todavía más potente es la constituida por los modelos

[25] Muchas veces, las probabilidades de transición de los trigramas se estiman en base a sucesos de rara aparición, y por tanto la posibilidad de obtener estimaciones malas es mayor.

[26] *Variable Memory Markov Models.*

[27] *State splitting.*

[28] *Model merging.*

de Markov jerárquicos no emisores [Ristad y Thomas 1997]. Mediante la introducción de transiciones sin emisiones[29], estos modelos pueden almacenar también dependencias de longitud arbitraria entre los estados.

4.7 Otras aplicaciones de los HMM,s

La teoría matemática relativa a los HMM,s fue desarrollada por Baum y sus colaboradores a finales de los años sesenta y principios de los setenta [Baum *et al.* 1970]. Los HMM,s se aplicaron al procesamiento de la voz o reconocimiento del discurso hablado[30] en los años setenta por Baker [Baker 1975], y por Jelinek y sus colegas de IBM [Jelinek *et al.* 1975, Jelinek 1976]. Fue posteriormente cuando los HMM,s se utilizaron para modelizar otros aspectos de los lenguajes humanos, tales como el proceso de etiquetación.

Dentro del contexto del reconocimiento de voz, existen muy buenas referencias sobre los HMM,s y sus algoritmos [Levinson *et al.* 1983, Charniak 1993, Jelinek 1997]. Particularmente conocidas son también [Rabiner 1989, Rabiner y Juang 1993]. Todas ellas estudian en detalle tanto los HMM,s continuos, donde la salida es un valor real, como los HMM,s discretos que nosotros hemos considerado aquí. Aunque se centran en la aplicación de los HMM,s al reconocimiento de voz, resulta también muy recomendable su consulta para la obtención de directrices y de otras muchas referencias acerca del desarrollo y uso general de los HMM,s.

En este capítulo, hemos asumido en todo momento que la arquitectura de nuestro HMM era fija, y hemos centrado la discusión en torno al problema de la obtención de los parámetros óptimos para esa arquitectura. Sin embargo, ¿cuál es la forma y el tamaño que deberíamos elegir para nuestros HMM,s cuando nos enfrentamos a un nuevo problema? Normalmente, como en el caso de la etiquetación, es la naturaleza del problema la que determina de alguna manera la arquitectura del modelo. Para circunstancias en las que no es ese el caso, existen trabajos que estudian la construcción automática de la estructura de los HMM,s. Dichos trabajos se basan en el principio de intentar encontrar el HMM más compacto posible que describa adecuadamente los datos [Stolcke y Omohundro 1993].

Los HMM,s han sido también ampliamente utilizados en aplicaciones bioinformáticas para el análisis de secuencias de genes [Baldi y Brunak 1998, Durbin *et al.* 1998]. Es cierto que puede parecer increíble que el manejo de un alfabeto de sólo cuatro símbolos, las bases nitrogenadas del ADN, entrañe grandes dificultades, pero la bioinformática es un dominio perfectamente establecido, y en el que efectivamente se plantean problemas muy serios que pueden requerir el uso de modelizaciones complejas.

Y quizás la aplicación más reciente de los HMM,s dentro del marco del procesamiento de lenguaje natural es su uso directo como motor de búsqueda en sistemas de recuperación de información [Miller *et al.* 1999]. La idea básica de esta aproximación consiste una vez más en considerar la unión de todas las palabras que aparecen en el corpus como el conjunto de símbolos de salida del modelo, e identificar un estado distinto, en este caso no para cada etiqueta, sino para cada uno de los posibles mecanismos de generación de palabras en las consultas: términos temáticos, términos dependientes del documento, palabras que aparecen frecuentemente en las consultas, mecanismos de sinonimia, etc. Posteriormente, se construye un sencillo HMM individual para cada documento, a través del cual se puede calcular la probabilidad de que dicho documento genere las palabras involucradas en la consulta que efectúa el usuario. Esa probabilidad indica si el documento es relevante o no, y en función de ella se establece la lista ordenada de documentos que el sistema muestra al usuario como respuesta a su consulta. Los

[29] Transiciones entre estados que no emiten ninguna palabra o que, de manera equivalente, emiten la palabra vacía ϵ.

[30] *Speech recognition.*

resultados obtenidos con esta técnica son muy prometedores, y refuerzan la perspectiva de que los HMM,s continúan siendo aplicables incluso en temas de verdadera actualidad como es el de la recuperación de información.