DISEÑO E IMPLEMENTACIÓN DE UNA INTERFAZ GENÉRICA PARA LA ESTIMACIÓN DE AUTÓMATAS DE ESTADOS FINITOS PONDERADOS

Mikel Penagarikano, German Bordel, Luis Javier Rodriguez, Maider Zamalloa

Grupo de Trabajo en Tecnologías del Software.
Departamento de Electricidad y Electrónica.
Facultad de Ciencia y Tecnología. Universidad del País Vasco.
Barrio Sarriena s/n. 48940 Leioa. Bizkaia.
mikel.penagarikano@ehu.es
http://gtts.ehu.es

RESUMEN

En este artículo se define y desarrolla una aproximación unificada al problema de estimar autómatas de estados finitos ponderados (AEFP), mediante una interfaz genérica válida para todo tipo de autómatas, cualquiera que sea su estructura interna. La metodología propuesta se ha integrado en *Sautrela*, un paquete de software desarrollado como código abierto en Java™, altamente modular y escalable, orientado al procesamiento de señal en general aunque muy enfocado al reconocimiento del habla (véase http://sautrela.org). Una vez definida la aplicación que enlaza los estados y transiciones de un autómata con los parámetros de la interfaz, ésta permite no sólo procesar secuencias de datos y obtener la decodificación óptima, sino también estimar los parámetros del autómata de acuerdo al criterio de optimización que se establezca. Esta aproximación permite reducir drásticamente el esfuerzo necesario para añadir nuevos módulos a un sistema de procesamiento del habla basado en AEFP.

1. INTRODUCCIÓN

Desde el inicio de las investigaciones sobre reconocimiento automático del habla hasta hoy se han desarrollado muchos sistemas, con muy diferentes funcionalidades, determinadas sobre todo por la tecnología disponible [1, 2, 3]. Inicialmente esos sistemas estaban orientados a la investigación de algunos de los elementos que conforman el proceso de reconocimiento y su diseño estaba optimizado para la consecución de dicho objetivo. Las implementaciones apenas permitían la migración a nuevas tareas o la integración de nuevas tecnologías. Por lo común, cualquier innovación implicaba el desarrollo de un sistema completo desde cero. En los últimos años la flexibilidad y la escalabilidad se han impuesto como principales criterios de diseño, de modo que los motores de reconocimiento han pasado a ser plataformas modulares que por un lado incorporan el estado del arte de las tecnologías del habla, y por otro, permiten el desarrollo y la integración de nuevas áreas de investigación. Sin embargo, los motores de última generación están pensados básicamente como máquinas decodificadoras, por lo que la estimación de modelos acústicos, léxicos o de lenguaje debe realizarse mediante paquetes de software externos.

Sautrela [4] es un paquete de software de código abierto ideado para facilitar el desarrollo de aplicaciones de procesamiento de señal, especialmente de procesamiento del habla. Sautrela es altamente modular y escalable, por lo que se le pueden añadir módulos o *plugins* de manera sencilla. Se ha desarrollado mediante tecnología Java™, lo que asegura su portabilidad a una extensa variedad de plataformas. A diferencia de los sistemas mencionados más arriba, Sautrela define un único formalismo para decodificar señales y estimar modelos, es decir, no es simplemente un motor de reconocimiento alimentado con modelos externos, sino que incorpora los algoritmos y criterios de estimación necesarios para producirlos internamente.

Este trabajo se centra en el problema de la estimación de modelos que, independientemente de su estructura interna, puedan representarse como Autómatas de Estados Finitos Ponderados (AEFP), a través de una simple interfaz. Esta aproximación permite reducir drásticamente el esfuerzo necesario para añadir nuevos módulos a un sistema de procesamiento del habla basado en AEFP.

El resto del artículo se estructura como sigue. En el apartado 2 se presenta la jerarquía de interfaces necesaria para representar y manejar AEFP. En el apartado 3 se introducen las interfaces que generalizan la estimación de parámetros para cualquier modelo representado mediante AEFP. En el apartado 4 se revisan los dos criterios de estimación más habituales y se describen sus implementaciones. En el apartado 5 se muestran ejemplos de implementación de las interfaces de estimación para tres tipos diferentes de modelos. Por último, en el apartado 6 se resumen las principales aportaciones de este trabajo.

Este trabajo ha sido parcialmente financiado por el Gobierno Vasco, dentro del programa SAIOTEK, a través de los proyectos S-PE04UN26 y S-PE05UN32.

(a) Interfaz básica AEFP

```
// Devuelve el nombre del AEFP
String getName();

// Devuelve el símbolo de nombre name
Symbol getSymbolByName(String name);

// Devuelve el estado inicial del AEFP
State getIniState();

// Devuelve la probabilidad de que el estado
// state sea final
double getFinProb(State state);

// Devuelve todas las transiciones posibles
// desde el estado state
Transition[] getTrans(State state);
```

(b) Interfaz específica AEFP-d (extensión de AEFP)

```
// Devuelve la transición correspondiente al
// par (state, sy)
Transition getTrans(State state, Symbol sy);
```

(c) Interfaz específica AEFP-nd (extensión de AEFP)

```
// Devuelve el conjunto de transiciones
// correspondiente al par (state, sy)
Transition[] getTrans(State from, Symbol sy);
```

Figura 1. La interfaz básica AEFP (a) consta del conjunto mínimo de métodos que permiten manejar un autómata. Cada una de las sub-interfaces (b y c) añade un único método, de acuerdo al carácter determinista o no determinista de la función de transición.

2. INTERFACES BÁSICAS PARA LA MANIPULACIÓN DE AEFP

2.1. Jerarquía de interfaces

Sautrela utiliza una jerarquía de interfaces muy simple, que permite unificar los diferentes tipos de modelos estocásticos que se aplican habitualmente en reconocimiento automático del habla. En la parte superior de dicha jerarquía se encuentra la interfaz básica de acceso a AEFP (véase la Figura 1), más dos interfaces diferenciadas para representar AEFP deterministas (AEFP-d) y AEFP no deterministas (AEFP-nd), que constan de un único método que permite manejar las transiciones en ambos tipos de formalismos. Básicamente, un AEFP consta de un estado inicial, un alfabeto de entrada y una función de transición probabilística que asigna al par formado por un estado y un símbolo de entrada otro estado (AEFP-d) o conjunto de estados (AEFP-nd). De hecho, *Estado*, *Símbolo* y *Transición* son interfaces flexibles del sistema que permiten manejar varios tipos de objetos, como por ejemplo cadenas de caracteres (para los modelos de lenguaje) o vectores de componentes reales (para los modelos acústicos).

Sautrela es capaz de manejar cualquier modelo que pueda ser representado mediante estas interfaces, independientemente de su estructura interna. No obstante, los sistemas de reconocimiento constan de varios modelos distintos, uno por cada fuente de conocimiento, que es necesario integrar en un único formalismo. Precisamente, para dar respuesta a esta necesidad, en un trabajo previo se han propuesto los *Modelos de Markov por Capas* (MMC) [5].

2.2. Integración de distintas fuentes de conocimiento mediante MMC

Un MMC consta de varias capas, cada una correspondiente a una fuente de conocimiento. Cada capa, formada por un número finito de AEFP, modela sus unidades en términos de unidades pertenecientes a capas inferiores. Independientemente del número de capas y de los tipos de AEFP que sea necesario utilizar, un MMC es equivalente a un AEFP-nd.

No hay limitaciones en cuanto al número de capas, modelos, estados o símbolos que conforman un MMC, por lo que puede utilizarse para integrar diferentes fuentes de conocimiento en un único autómata. Tal es precisamente el problema planteado por los sistemas de reconocimiento del habla, que utilizan Modelos Ocultos de Markov (MOM), diccionarios de pronunciaciones o, en su defecto, modelos léxicos en forma de grafo, gramáticas estocásticas (n-gramas), etc. Todos estos modelos pueden ser representados como AEFP a través de las interfaces mostradas en la Figura 1, de modo que el acoplamiento *ad-hoc* presente todavía en muchos sistemas de reconocimiento, que dificulta o impide completamente la incorporación de mejoras tecnológicas, puede formalizarse de manera mucho más sencilla, flexible y eficiente mediante un MMC. Otros autores han presentado aproximaciones similares (véanse, por ejemplo, [1] y [6]), pero están diseñadas para la decodificación de señales y carecen de herramientas genéricas para la estimación de modelos.

3. LA INTERFAZ DE ENTRENAMIENTO

La interfaz AEFP, junto con las sub-interfaces AEFP-d y AEFP-nd, conforman el conjunto mínimo de métodos necesario para afrontar el problema de la decodificación. Los procedimientos de entrenamiento empleados habitualmente en los sistemas de reconocimiento comienzan con una primera fase de decodificación en la que se calcula una función objetivo; tras ella sigue una segunda fase en la que los parámetros de los modelos son reestimados con objeto de optimizar dicha función. De igual modo, los parámetros de un AEFP (probabilidades de transición y probabilidades finales) pueden reestimarse con objeto de optimizar una función objetivo. No obstante, la información de dichos parámetros debe ser transferida a la representación interna del AEFP. De esta forma, es posible reestimar externamente los parámetros de un AEFP, mientras que el propio autómata será el responsable de la conversión de los parámetros ya reestimados a su representación interna.

```
// Inicializa las cuentas de entrenamiento
void initTrCounts();

// Incrementa las cuentas asociadas a una
// transición
void incTrTransCount(Transition tr, double c);

// Incrementa las cuentas asociadas a un
// estado final.
void incTrFinalCount(State state, double c);

// Vuelca las cuentas a un modelo,
// actualizando sus parámetros
void dumpTrCounts();
```

Figura 2. La interfaz de entrenamiento permite calcular las cuentas externamente, dejando en manos del modelo transferir a su representación interna la información contenida en dichas cuentas.

Para facilitar la transferencia de los parámetros reestimados, éstos son transmitidos en forma de cuentas incrementales. Como se verá más adelante, lo que llamamos *cuenta* no es más que la probabilidad de una transición en un instante t, dada la secuencia de entrada y los parámetros actuales del autómata. El conjunto de métodos que conforman la interfaz de entrenamiento permite inicializar, transmitir y volcar las cuentas para actualizar los parámetros de un AEFP (véase la Figura 2).

4. CRITERIOS DE ESTIMACIÓN

Haciendo uso de la presente metodología, pueden aplicarse diversos criterios de estimación (y sus correspondientes transformaciones). En los siguientes dos párrafos se revisan dos de los criterios más utilizados: Máxima Verosimilitud y Máxima Información Mutua.

4.1. Máxima Verosimilitud

La Máxima Verosimilitud (MV) es el criterio de estimación más extendido. Sea $\mathbf{X}$ una variable aleatoria con función de distribución $p_\phi(x)$, donde ϕ pertenece a un espacio de parámetros Φ. Dada una secuencia de observaciones independientes $\mathbf{x} = \{x_1, ..., x_n\}$, la función de verosimilitud viene dada por:

$$L = p_\phi(\mathbf{x}) = \prod_{i=1}^{n} p_\phi(x_i)$$

La estimación MV de ϕ se obtiene maximizando L:

$$\phi_{\mathrm{ML}} = \arg\max_{\phi} p_\phi(\mathbf{x})$$

El teorema de Baum-Sell para transformaciones crecientes [7] es de aplicación en dicha maximización, pudiendo obtenerse una fórmula de reestimación para todos los parámetros. En el caso de un AEFP, los parámetros se restringen a las probabilidades de transición y finales, las cuales deben sumar 1 para todo estado origen. Sea $p_\phi(\tau)$ la probabilidad de la transición $\tau = (s, d, y)$, con estado origen s, estado destino d y símbolo de emisión y. La probabilidad de ser final de un estado puede formalizarse como una transición externa especial con emisión nula, $p_\phi(s, out, null)$, y por tanto, puede ser integrada dentro de la función de transición. Del teorema de Baum-Sell se deduce que la reestimación de $p_\phi(\tau)$ viene dada por:

$$p_{\tilde{\phi}}(\tau) = \frac{p_\phi(\tau) \cdot \left(\frac{\partial L}{\partial P(\tau)}\right)_\phi}{\sum_{\tau'=(s,d',y')} p_\phi(\tau') \cdot \left(\frac{\partial L}{\partial P(\tau)}\right)_\phi} \qquad (1)$$

Esta expresión puede ser reescrita en términos de cuentas incrementales calculadas para todas las muestras de entrenamiento. Sea $count(\tau)_{t,i}$ la probabilidad de la transición τ en el instante t y dada la observación x_i. Estas cuentas pueden ser eficientemente calculadas haciendo uso de los denominados *coeficientes forward* y *backward* (véase [8]). Finalmente, la expresión (1) puede reescribirse como sigue:

$$p_{\tilde{\phi}}(\tau) = \frac{\sum_{i=1}^{n} \sum_{t=1}^{m_i} count(\tau)_{t,i}}{\sum_{\tau'=(s,d',y')} \sum_{i=1}^{n} \sum_{t=1}^{m_i} count(\tau')_{t,i}} \qquad (2)$$

4.2. Máxima Información Mutua

Cualquier problema de reconocimiento de patrones puede ser formalizado haciendo uso de la metáfora del *canal ruidoso*. Dado un par de variables Ω y X, la Información Mutua $I(X;\Omega)$ da cuenta de la reducción de la incertidumbre de X ante el conocimiento de Ω y viceversa. Si suponemos que el vector de muestras $(\mathbf{x}, \omega) = \{(x_1, \omega_1), ..., (x_n, \omega_n)\}$ es suficientemente representativo, la Información Mutua puede aproximarse mediante la siguiente expresión:

$$I(X;\Omega) \approx \log \frac{\prod_{i=1}^{n} p_\phi(x_i \mid \omega_i)}{\prod_{i=1}^{n} p_\phi(x_i)} = \log \frac{\prod_{i=1}^{n} p_\phi^{sup}(x_i)}{\prod_{i=1}^{n} p_\phi^{nosup}(x_i)}$$

Nótese que el numerador $p_\phi^{sup}(x_i)$ hace referencia a la probabilidad de la observación x_i cuando la clase ω_i es conocida (existe una supervisión). Por otro lado, el denominador $p_\phi^{nosup}(x_i)$ da cuenta de la probabilidad de x_i en ausencia de supervisión. Dado un conjunto de clases, cada una representada por un AEFP, ambas probabilidades pueden ser calculadas integrando el conjunto de autómatas en un MMC que contará con una capa adicional que dará cuenta de la posibilidad o no de supervisión. La estimación por Máxima Información Mutua (MIM) de ϕ se obtiene de la maximización de $I(X;\Omega)$:

$$\phi_{\mathrm{MIM}} = \arg\max_{\phi} \frac{\prod_{i=1}^{n} p_\phi^{sup}(x_i)}{\prod_{i=1}^{n} p_\phi^{nosup}(x_i)}$$

El teorema de Gopalakrishnan para funciones racionales [9] demuestra que existe una constante C, tal

que la siguiente expresión es una transformación creciente:

$$p_{\tilde{\phi}}(\tau) = \frac{p_\phi(\tau) \cdot \left(\left(\frac{\partial I}{\partial P(\tau)} \right)_\phi + C \right)}{\sum_{\tau'=(s,d',y')} p_\phi(\tau') \cdot \left(\left(\frac{\partial I}{\partial P(\tau)} \right)_\phi + C \right)} \quad (3)$$

Del mismo modo que en el caso de la estimación por MV, las cuentas pueden calcularse de manera eficiente haciendo uso de los coeficientes *forward* y *backward*. En este caso, definiremos dos cuentas parciales, $count(\tau)_{t,i}^{sup}$ y $count(\tau)_{t,i}^{nosup}$, y la cuenta resultante será la diferencia de las mismas:

$$count(\tau)_{t,i} = count(\tau)_{t,i}^{sup} - count(\tau)_{t,i}^{nosup}$$

La constante C mencionada anteriormente garantiza la convergencia incluso en presencia de cuentas negativas[1]. Finalmente, la ecuación (3) puede escribirse como sigue:

$$p_{\tilde{\phi}}(\tau) = \frac{C \cdot p_\phi(\tau) + \sum_{i=1}^{n} \sum_{t=1}^{m_i} count(\tau)_{t,i}}{\sum_{\tau'=(s,d',y')} C \cdot p_\phi(\tau') + \sum_{i=1}^{n} \sum_{t=1}^{m_i} count(\tau')_{t,i}} \quad (4)$$

5. EJEMPLOS DE IMPLEMENTACIÓN DE LAS INTERFACES

5.1. Modelos de Markov por Capas

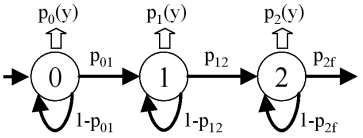
El meollo de la presente metodología de entrenamiento mediante interfaces reside en la posibilidad de reestimar un MMC compuesto por diversas capas organizadas jerárquicamente. La reestimación de un MMC implica el entrenamiento simultáneo de todos los modelos que lo componen, es decir, el entrenamiento de un conjunto de modelos. Como se ha comentado en la Sección 2, un MMC puede ser visto como un AEFP-nd. Por tanto, dado un criterio de entrenamiento, sus parámetros (desde el punto de vista de la interfaz) pueden ser reestimados. Para ello, basta calcular las cuentas para todas las muestras de entrenamiento y transmitirlas mediante la interfaz de entrenamiento mostrada en la Figura 2. Sin embargo, surge la pregunta de cómo transmitir dicha información a los modelos internos que conforman un MMC.

La respuesta es bastante simple. Una transición τ de un MMC consiste en una secuencia de transiciones $\Delta_\tau = \{\delta_1^\tau, ..., \delta_k^\tau\}$ dentro de las capas internas. Es por ello que las cuentas de dicha transición, $count(\tau)_{t,i}$, deben ser transmitidas a todas las transiciones internas que componen τ. En otras palabras, la cuenta $count(\delta)_{t,i}$ de una transición interna δ, resulta de la suma de todas las transiciones τ del MMC tales que $\delta \in \Delta_\tau$. Sea $T(\delta) = \{\tau \mid \delta \in \Delta_\tau\}$. Entonces,

$$count(\delta)_{t,i} = \sum_{\tau \in T(\delta)} count(\tau)_{t,i}$$

[1] Aunque la expresión (3) proporciona una transformación creciente para valores de C suficientemente grandes, los experimentos muestran que valores pequeños de C ofrecen una convergencia más rápida.

(a) Un MOM-d de 3 estados



(b) El AEFP equivalente

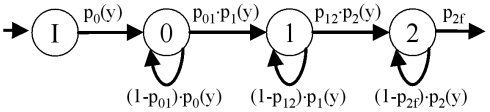


Figura 3. Cada transición en el MOM genera en el AEFP equivalente tantas transiciones como número de emisiones en el estado destino del MOM. El estado inicial adicional del modelo equivalente contiene transiciones a todos los estados iniciales del MOM.

5.2. Modelos Ocultos de Markov Discretos

En los MOM discretos las emisiones están ligadas a los estados, mientras que en los AEFP aquí descritos las emisiones ocurren en las transiciones. No obstante, el AEFP equivalente a un MOM discreto puede obtenerse fácilmente suponiendo que las emisiones ocurren en las transiciones, justo antes de llegar al estado (véase la Figura 3).

Para implementar la interfaz de entrenamiento, cada cuenta del AEFP debe ser transferida a la representación interna de los MOM discretos (cuenta inicial, de transición y de emisión para cada uno de los estados). Sea $T_A(s,d) = \bigcup_{\forall y} \{\tau = (s,d,y)\}$ el conjunto de transiciones del AEFP con estado origen s y estado destino d, $T_B(d,y) = \bigcup_{\forall s} \{\tau = (s,d,y)\}$ el conjunto de transiciones del AEFP con estado destino d y símbolo de emisión y, y s_I el estado inicial del AEFP. Entonces, las cuentas internas del MOM discreto vienen dadas por:

$$count_{init}(s)_{t,i} = \sum_{\tau \in T_A(s_I,s)} count(\tau)_{t,i}$$

$$count_{trans}(s,d)_{t,i} = \sum_{\tau \in T_A(s,d)} count(\tau)_{t,i}$$

$$count_{emit}(s,y)_{t,i} = \sum_{\tau \in T_B(s,y)} count(\tau)_{t,i}$$

Es decir, cada cuenta del AEFP es transferida bien a la cuenta inicial, bien a la cuenta de transición del MOM (dependiendo del estado origen), y siempre a la cuenta de emisión del estado destino. Cabe destacar que para el

caso del entrenamiento por MV, las cuentas así obtenidas son exactamente las mismas que se obtendrían mediante el algoritmo de Baum-Welch [8].

5.3. Modelos Ocultos de Markov Continuos

El AEFP equivalente a un MOM continuo es análogo al presentado previamente, a excepción de que existirá un conjunto continuo (es decir, infinito) de transiciones. Esto no supone problema alguno para el formalismo de los AEFP aquí descrito, ya que no impone condición alguna sobre la naturaleza continua o no de los simbolos emitidos, y por tanto, puede manejar conjuntos teóricamente infinitos de transiciones.

6. CONCLUSIONES

En este artículo se ha presentado una metodología novedosa para el entrenamiento de un AEFP. Todo modelo que implemente una sencilla interfaz puede ser entrenado independientemente de su estructura y sin conocimiento alguno de su representación interior. Para el caso del entrenamiento de MOM por MV, la reestimación es exactamente la misma que se obtendría mediante el algoritmo de Baum-Welch. Por otra parte, los MMC constituyen un formalismo extremadamente útil tanto para la decodificación como para el entrenamiento, ya que permiten la reestimación simultánea de un conjunto de AEFP. Partiendo de esta metodología, se ha desarrollado un módulo de entrenamiento para el sistema *Sautrela*, que simplifica considerablemente el esfuerzo necesario para incorporar nuevos formalismos de modelado a un sistema de procesamiento del habla basado en AEFP.

7. BIBLIOGRAFÍA

[1] J.K. Baker, "The DRAGON System - An Overview," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 23, no. 1, pp. 24–29, 1975.

[2] S. Young, "The HTK Hidden Markov Model Toolkit: Design and Philosophy," Tech. Rep. TR.153, Department of Engineering, Cambridge University, 1993.

[3] K. Lee, H. Hon, y R Reddy, "An Overview of the SPHINX Speech Recognition System," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 38, no. 1, pp. 35 – 45, January 1990.

[4] M. Penagarikano y G. Bordel, "Sautrela: A Highly Modular Open Source Speech Recognition Framework," in *Proceedings of the ASRU Workshop*, 2005.

[5] M. Penagarikano y G. Bordel, "Layered Markov Models: A New Architectural Approach to Automatic Speech Recognition," in *Proceedings of the MLSP Workshop*, 2004.

[6] M. Mohri, F. Pereira, y M. Riley, "Weighted Finite-State Transducers in Speech Recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 69–88, 2002.

[7] Leonard. E. Baum y George R. Sell, "Growth Transformations for Functions on Manifolds," *Pacific Journal of Mathematics*, vol. 27, pp. 211–227, 1968.

[8] Leonard. E. Baum, "An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of Markov Processes," *Inequalities*, vol. 3, pp. 1–8, 1972.

[9] P. S. Gopalakrishnan, D. Kanevsky, A. Nádas, y D. Nahamoo, "An Inequality for Rational Functions with Applications to Some Statistical Estimation Problems," *IEEE Trans. Information Theory*, vol. 37, pp. 107–113, 1991.