Capítulo 5

Inferencia de lenguajes regulares

La probabilidad a priori *de que se produzca un acontecimiento particular entre todos los posibles en el universo está próxima a cero... Nuestro número salió en el juego de Montecarlo.*
Jaques Monod, El azar y la necesidad

En este capítulo, estudiaremos la identificación de lenguajes regulares estocásticos de cadenas. Como resultado se propondrá un conjunto de algoritmos que comparan de forma sistemática los estados del árbol de prefijos de la muestra. Esta clase de algoritmos permite identificar en el límite el conjunto de cadenas que forman el lenguaje y además calcular con gran precisión sus probabilidades de aparición en el lenguaje. El tiempo necesario para que el algoritmo proporcione una hipótesis aumenta sólo linealmente con el tamaño de la muestra y, experimentalmente, la implementación desarrollada se revela como extraordinariamente rápida con vistas a su posible aplicación en tareas de reconocimiento.

5.1 Antecedentes

La identificación de lenguajes estocásticos de cadenas ha sido tratada en numerosa ocasiones anteriormente. Por ejemplo, Maryanski y Booth (1977) usaron un test del tipo χ^2 para filtrar gramáticas regulares que eran generadas mediante procedimientos heurísticos. Con este método se obtenían gramáticas razonables, pero sin garantizar la identificación en el límite de la gramática correcta. Cook, Rosenfel y Aronson (1976) definieron una función de similitud entre gramáticas estocásticas basada en la teoría de la información para buscar gramáticas sencillas que minimicen la distancia a la muestra.

El método de van der Mude y Walker (1978) fusiona variables de una gramática estocástica regular y utiliza criterios bayesianos para permitir estas fusiones. Los autores no demuestran que su algoritmo converja a la gramática correcta y además su implementación es extremadamente lenta como para poder ser aplicada en la práctica.

Recientemente, se han utilizado modelos con redes neurales para identificar lenguajes regulares (Smith & Zipser 1989, Pollack 1991, Giles 1992, Watrous & Kuhn 1992) y en algunos casos se han aplicado al caso de muestras estocásticas (Castaño, Casacuberta & Vidal 1993). Sin embargo, estos métodos comparten la desventaja de que requieren largos tiempos de cálculo y con frecuencia muestras muy grandes.

Los modelos ocultos de Markov, esencialmente equivalentes a los autómatas finitos estocásticos, han sido utilizados también con frecuencia en el tratamiento de lenguajes estocásticos. Su mayor desventaja reside en que es necesario prever de antemano el número de estados del modelo e introducir alguna información adicional sobre su estructura para que el entrenamiento no conduzca a un mínimo local. Un sistema para determinar automáticamente el número de estados del modelo de Markov ha sido desarrollado por Stolcke y Omohundro (1993). Su método intenta maximizar la probabilidad de la muestra utilizando a la vez una probabilidad *a priori* que penaliza los tamaños grandes del autómata. Tampoco estos procedimientos garantizan la

convergencia al modelo correcto.

Recientemente, Oncina y García (1992) han propuesto un algoritmo que permite la identificación correcta en el límite de cualquier lenguaje regular siempre y cuando se disponga de muestras completas. Además, el algoritmo funciona en tiempo polinómico con el tamaño de la muestra. En la práctica, el tiempo que requiere para producir su hipótesis crece sólo linealmente con el número de ejemplos en los experimentos. En este capítulo, se siguen las mismas líneas y se presenta un algoritmo (`rlips`) que construye el árbol de prefijos de la muestra y evalúa en cada nodo las probabilidades de transición de los arcos que salen de ese nodo. A continuación, se comparan parejas de nodos siguiendo un orden bien establecido (básicamente el de los niveles de profundidad en el árbol). Se acepta que dos nodos son equivalentes cuando generan —dentro de la incertidumbre estadística— el mismo sublenguaje estocástico. El proceso continúa hasta que no es posible establecer más equivalencias. Las definiciones necesarias para este capítulo se encuentran en la sección 5.2. El algoritmo es descrito en detalle en la sección 5.3 y se demuestra que es correcto en la sección 5.4. Finalmente, los resultados y su discusión son el contenido de la última sección de este capítulo.

5.2 Formalismo

Sea $\mathcal{A}$ un alfabeto finito, $\mathcal{A}^*$ el monoide libre de cadenas generadas a partir de $\mathcal{A}$ y la operación de concatenación y λ el elemento neutro o cadena vacía. Denotaremos la longitud de la cadena $w \in \mathcal{A}^*$ como $|w|$.

Dadas $x, y \in \mathcal{A}^*$, si $w = xy$ entonces escribiremos también $y = x^{-1}w$. A su vez, la expresión $x\mathcal{A}^*$ denotará al conjunto de cadenas que contienen a x como prefijo.

Diremos que x precede a y en *orden lexicográfico*, y lo denotaremos como $x < y$, en cualquiera de los siguientes casos:

1. $|x| < |y|$,

2. $|x| = |y|$ y x precede a y en orden alfabético.

Un *lenguaje estocástico* L se define mediante una distribución de probabilidad $p(w|L)$ sobre todas las cadenas w de $\mathcal{A}^*$. La probabilidad de un subconjunto de cadenas $X \subset \mathcal{A}^*$ viene dada por la suma de las probabilidades de sus cadenas:

$$p(X|L) = \sum_{x \in X} p(x|L). \tag{5.1}$$

La *identidad* de dos lenguajes estocásticos debe ser entendida de la forma siguiente:

$$L_1 = L_2 \Leftrightarrow p(w|L_1) = p(w|L_2)\ \forall w \in \mathcal{A}^*. \tag{5.2}$$

Es decir, para que dos lenguajes estocásticos sean idénticos, deben ser idénticas las probabilidades asignadas a todas y cada una de las cadenas.

A continuación vamos a presentar dos métodos para definir lenguajes estocásticos que, aunque distintas, son equivalentes: las gramáticas regulares estocásticas y los autómatas finitos estocásticos.

Una *gramática regular estocástica* (SRG), $G = (\mathcal{A}, V, S, R, p)$, consta de un alfabeto finito $\mathcal{A}$, un conjunto finito de variables V —una de las cuales, S, es el llamado símbolo inicial de la gramática—, un conjunto finito R de reglas de derivación cuya estructura es una de las siguientes:

$$\begin{array}{lcl} X & \rightarrow & aY \\ X & \rightarrow & \lambda \end{array} \tag{5.3}$$

(donde $a \in \mathcal{A}$, $X, Y \in V$), y una función real $p : R \rightarrow [0, 1]$ que representa la probabilidad de aplicación de cada regla. Obviamente, la suma de las probabilidades de las reglas asociadas a una misma variable X debe ser igual a 1. La definición que hemos tomado en (5.3), aunque aparentemente distinta a la habitualmente usada, como la de Fu (1982), es totalmente equivalente a ella. Una gramática estocástica G es *determinista* si por cada $X \in V$ y por cada $a \in \mathcal{A}$ existe como máximo una variable $Y \in V$ tal que $p(X \rightarrow aY) \neq 0$.

Toda gramática estocástica regular determinista G define un *lenguaje estocástico determinista regular* (SDRL) por medio de las probabilidades $p(w|L_G) = p(S \Rightarrow w)$. La probabilidad $p(S \Rightarrow w)$ de que la gramática G produzca la cadena $w \in \mathcal{A}^*$ se calcula de forma recursiva

$$\begin{aligned} p(X \Rightarrow \lambda) &= p(X \to \lambda) \\ p(X \Rightarrow aw) &= p(X \to aY)p(Y \Rightarrow w) \end{aligned} \tag{5.4}$$

donde Y es la única variable que satisface $p(X \to aY) \neq 0$. Si no existe tal variable, entonces $p(X \Rightarrow aw) = 0$.

Un *autómata finito determinista estocástico* (SDFA), se define como $A = (Q^A, \mathcal{A}, \delta^A, q_I^A, p^A)$, y consta de:

- un alfabeto finito $\mathcal{A}$;
- un conjunto finito de estados $Q^A = \{q_1, q_2, \ldots q_n\}$;
- un estado inicial $q_I^A \in Q^A$;
- una función de transición $\delta^A : Q^A \times \mathcal{A} \to Q^A$;
- una función de probabilidad $p^A : Q^A \times \mathcal{A} \to [0, 1]$.

La función $p^A(q_i, a)$ representa la probabilidad de que se genere a partir del estado q_i un símbolo a, produciéndose una transición posterior a $\delta^A(q_i, a)$. Se define además la función $p^A(q_i, \lambda)$ como

$$p^A(q_i, \lambda) = 1 - \sum_{a \in \mathcal{A}} p^A(q_i, a) \tag{5.5}$$

que representa la probabilidad de que la cadena termine en el nodo q_i. La restricción de que $p^A(q_i, \lambda) \geq 0$ debe cumplirse para todos los autómatas que han sido definidos correctamente.

Cada SDFA define un lenguaje regular determinista estocástico por medio de las probabilidades $p(w|L_A) = \pi(q_I^A, w)$, que a su vez se definen recursivamente como

$$\begin{aligned} \pi^A(q_i, \lambda) &= p^A(q_i, \lambda) \\ \pi^A(q_i, aw) &= p^A(q_i, a)\pi^A(\delta^A(q_i, a), w) \end{aligned} \tag{5.6}$$

Si $\delta^A(q_i, a)$ no está definida, entonces $\pi^A(\delta^A(q_i, a), w) = 0$.

Si comparamos las ecuaciones (5.4) y (5.6), observaremos la equivalencia formal entre los autómatas finitos estocásticos y las gramáticas regulares estocásticas. En caso de que la SDRG no contenga símbolos inútiles (Hopcroft & Ullman 1979), las probabilidades de todas las cadenas del lenguaje suman uno:

$$p(\mathcal{A}^*|L_G) = \sum_{w \in \mathcal{A}^*} p(w|L_G) = 1 \tag{5.7}$$

Un concepto que utilizaremos con frecuencia en lo que sigue es de lenguaje *cociente* $x^{-1}L$, que es un lenguaje estocástico cuyas probabilidades vienen dadas por:

$$p(w|x^{-1}L) = \frac{p(xw|L)}{p(x\mathcal{A}^*|L)} \tag{5.8}$$

expresión que puede entenderse como las probabilidades relativas de las subcadenas que admiten a x como prefijo. Por convención, cuando $p(x\mathcal{A}^*|L) = 0$, escribiremos que $x^{-1}L = \emptyset$ y entonces, $p(w|x^{-1}L) = 0$. Nótese que $\lambda^{-1}L = L$.

Si L es un lenguaje regular estocástico determinista, se define el *generador canónico* $M = (Q^M, \mathcal{A}, \delta^M, q_I^M, p^M)$ como:

$$\begin{aligned} Q^M &= \{x^{-1}L \neq \emptyset : x \in \mathcal{A}^*\} \\ \delta^M(x^{-1}L, a) &= (xa)^{-1}L \\ q_I^M &= \lambda^{-1}L \\ p^M(x^{-1}L, a) &= p(a\mathcal{A}^*|x^{-1}L) \end{aligned} \tag{5.9}$$

El autómata M es el SDFA mínimo que genera L y su construcción esta justificada por los siguientes hechos, que permiten generalizar el teorema de Myhill y Nerode (Hopcroft & Ullman 1979) al caso de los autómatas estocásticos:

1. El autómata M es finito y además es más pequeño que cualquier otro autómata $A = (Q^A, \mathcal{A}, \delta^A, q_I^A, p^A)$ que también genere L. Si

escribimos $q_x = \delta^A(q_I, x)$, y aplicamos repetidamente (5.6) a la definición (5.8), se llega a que

$$p(w|x^{-1}L) = \frac{\pi^A(q_I, xw)}{\pi^A(q_I, x\mathcal{A}^*)} = \frac{\pi^A(q_x, w)}{\pi^A(q_x, \mathcal{A}^*)} = \pi^A(q_x, w). \quad (5.10)$$

Como el número de valores diferentes de q_x está limitado por $|Q^A|$, esto es, por el tamaño del autómata A, la expresión anterior demuestra que también está acotado el número de lenguajes distintos $x^{-1}L$ y, en consecuencia, $|Q^M| \leq |Q^A|$.

2. La función de transición δ^M está bien definida, es decir,

$$x^{-1}L = y^{-1}L \Rightarrow \delta^M(x^{-1}L, a) = \delta^M(y^{-1}L, a). \quad (5.11)$$

De hecho, para cualquier cadena $w \in \mathcal{A}^*$ y cualquier lenguaje L

$$p(w|a^{-1}x^{-1}L) = \frac{p(aw|x^{-1}L)}{p(a\mathcal{A}^*|x^{-1}L)} = \frac{p(xaw|L)}{p(xa\mathcal{A}^*|L)} = p(w|(xa)^{-1}L). \quad (5.12)$$

y por tanto $(xa)^{-1}L = a^{-1}(x^{-1}L)$. Con esto, la relación (5.11) es inmediata. Además, la expresión (5.12) junto a la definición (5.9) nos permiten escribir $\delta^M(q_I, w) = w^{-1}L$.

3. El autómata M genera $L_M = L$. En realidad, es más sencillo probar que

$$\pi^M(x^{-1}L, w\mathcal{A}^*) = p(w\mathcal{A}^*|x^{-1}L) \quad (5.13)$$

para todas las cadenas $x, w \in \mathcal{A}^*$, propiedad que incluye el estado inicial $x = \lambda$ como caso particular: $\pi^M(q_I^M, w\mathcal{A}^*) = p(w\mathcal{A}^*|L)$, por lo que entonces $L_A = L$. La ecuación (5.13) se satisface trivialmente para cualquier x si $w = \lambda$. Siguiendo (5.6), se obtiene

$$\pi^M(x^{-1}L, aw\mathcal{A}^*) = p^M(x^{-1}L, a)\pi^M((xa)^{-1}L, w\mathcal{A}^*) \quad (5.14)$$

Finalmente, mediante un proceso de inducción en w utilizando (5.9), llegamos a que

$$\pi^M(x^{-1}L, aw\mathcal{A}^*) = p(a\mathcal{A}^*|x^{-1}L)p(w\mathcal{A}^*|(xa)^{-1}L) = p(aw\mathcal{A}^*|L). \quad (5.15)$$

Para identificar el generador canónico utilizaremos algunos conjuntos no estocásticos, como el *conjunto de prefijos* del lenguaje L que se define como

$$\Pr(L) = \{x \in \mathcal{A}^* : x^{-1}L \neq \emptyset\} \tag{5.16}$$

y el *conjunto de prefijos cortos* de L, que se define como:

$$\mathrm{Sp}(L) = \{x \in \Pr(L) : x^{-1}L = y^{-1}L \Rightarrow x \leq y\} \tag{5.17}$$

Debe observarse que $x^{-1}L \neq y^{-1}L$ para cualquier par de cadenas $x, y \in \mathrm{Sp}(L)$ distintas ($x \neq y$), y por tanto, cada cadena de $\mathrm{Sp}(L)$ puede entenderse como un representante de uno de los estados del generador canónico M. Por ello, utilizaremos estos elementos para construir el generador canónico M, añadiendo las transiciones adecuadas. Estas transiciones serán del tipo $\delta(x, a) = xa$, siempre y cuando xa sea un prefijo corto. En caso contrario, necesitaremos definir la transición de otro modo. Por este motivo, definimos el *kernel* de L

$$K(L) = \{\lambda\} \cup \{xa \in \Pr(L) : x \in \mathrm{Sp}(L) \wedge a \in \mathcal{A}\} \tag{5.18}$$

que está formado por los prefijos cortos más los prefijos que se obtienen de éstos por adición de un símbolo. Además, definimos la *frontera* de L como los elementos del kernel que no son prefijos cortos:

$$F(L) = K(L) - \mathrm{Sp}(L) \tag{5.19}$$

Nótese que el kernel $K(L)$ contiene como máximo $1 + |M||\mathcal{A}|$ cadenas e incluye a $\mathrm{Sp}(L)$ como parte propia.

Pretendemos identificar el generador canónico a partir de muestras aleatorias. Por ello, definimos una *muestra estocástica* S del lenguaje L como una secuencia infinita de cadenas generadas según la distribución de probabilidad $p(w|L)$. Representamos mediante S_n la subsecuencia de S formada por sus n primeras cadenas, que en general, no serán todas diferentes. El número de veces que aparece la cadena x en S_n se escribirá $c_n(x)$, y para subconjuntos de cadenas $X \subset \mathcal{A}^*$,

$$c_n(X) = \sum_{x \in X} c_n(x). \tag{5.20}$$

La subsecuencia S_n define un lenguaje estocástico L_n cuyas probabilidades son

$$p(x|L_n) = \frac{1}{n} c_n(x). \tag{5.21}$$

Por último, el *'arbol generador de prefijos* de S_n es un SDFA que genera L_n, es decir, asigna a cada cadena la probabilidad observada experimentalmente de la siguiente forma: $T_n = (Q^T, \mathcal{A}, \delta^T, q_I^T, p^T)$, con

$$\begin{aligned} Q^T &= \Pr(L_n) \\ \delta^T(x,a) &= \begin{cases} xa & \text{si } xa \in \Pr(L_n) \\ \emptyset & \text{en caso contrario} \end{cases} \\ q_I^T &= \lambda \\ p^T(x,a) &= \frac{c_n(xa\mathcal{A}^*)}{c_n(x\mathcal{A}^*)} \end{aligned} \tag{5.22}$$

Las probabilidades del tipo $p^T(x,\lambda)$ pueden calcularse usando la ecuación (5.5).

5.3 Algoritmo de inferencia

En este punto, es conveniente destacar algunas diferencias entre el proceso de identificación de lenguajes estocásticos y el de no estocásticos. Las muestras de lenguajes estocásticos contienen ejemplos repetidos que aparecen siguiendo una distribución de probabilidad $p(w|L)$, y la regularidad estadística es capaz de compensar la falta de datos negativos. Tal y como demostró Angluin (1988), muchas clases de distribuciones —y en particular los lenguajes regulares estocásticos— pueden ser identificados en el límite, en el sentido de que todas las cadenas de probabilidad no nula son aprendidas, con probabilidad uno. Para ello, es suficiente con seguir un procedimiento enumerativo como el descrito en la sección 1.3. Sin embargo, los métodos enumerativos son irrealizables en la práctica y es necesario buscar algoritmos que funcionen en tiempo polinómico.

Es importante recordar que no se trata tan sólo de aproximarse a la distribución de probabilidad correcta. Esto se puede conseguir sencillamente utilizando para las probabilidades $p(w|L)$ los valores $c_n(w)/n$ observados de la frecuencia relativa de aparición de la cadena. En la sección 3.6, señalamos que la aproximación de las probabilidades puede conseguir rápidamente valores pequeños de la entropía relativa con respecto al modelo correcto. De hecho, es sencillo comprobar que de esta forma la distancia entre la distribución correcta y la propuesta disminuye a medida que aumenta n. Sin embargo, por muy grande que sea n, si el lenguaje contiene un número infinito de cadenas con probabilidad no nula de aparición, siempre existirán cadenas que no están en S_n, lo que provoca un valor grande de la entropía relativa entre la distribución $c_n(w)/n$ y la correcta.

Por ello, lo que buscamos es un procedimiento que permita reducir el número de probabilidades a estimar a un número finito: si identificamos la estructura del generador canónico M, sólo es necesario evaluar las probabilidades de transición entre sus estados. Como el número de transiciones es finito (como mucho $|M||\mathcal{A}|$), podremos entonces estimar estas probabilidades a partir de la muestra y obtener mejores resultados. Esta estimación puede realizarse mediante el procedimiento descrito en Wetherell (1980) y justificado en Chaudury & Rao (1986). Un algoritmo que permite realizar esta tarea de identificación de la estructura probabilística es presentado en esta sección.

Para ello, vamos a definir la función lógica $\texttt{equiv}_L : K(L) \times K(L) \rightarrow \{\text{TRUE}, \text{FALSE}\}$ de la siguiente forma:

$$\texttt{equiv}_L(x, y) = \text{TRUE} \Leftrightarrow x^{-1}L = y^{-1}L. \tag{5.23}$$

La función $\texttt{equiv}_L$ sólo está definida en $K(L)$ y nos permite enunciar el siguiente lema.

Lema 1 *Dados* $\text{Sp}(L)$, $F(L)$ *y* $\texttt{equiv}_L$, *la estructura del generador*

canónico es isomorfa a:

$$\begin{aligned} Q &= \mathrm{Sp}(L) \\ q_I &= \lambda \\ \delta(x,a) &= y \end{aligned} \tag{5.24}$$

donde para cada $(x,a) \in \mathrm{Sp}(L) \times \mathcal{A}$*, y es la única cadena en* $Sp(L)$ *tal que* $\mathtt{equiv}_L(xa,y)$.

Demostraci'on. Sea $\Phi : Q \to Q^M$ la función definida como $\Phi(x) = x^{-1}L$. La función Φ es un isomorfismo si $\delta^M(\Phi(x),a) = \Phi(\delta(x,a))$, es decir, si $(xa)^{-1}L = \delta(x,a)^{-1}L$. Por tanto, Φ es un isomorfismo si y sólo si $\delta(x,a)$ es una cadena $y \in \mathrm{Sp}(L)$ que satisface $\mathtt{equiv}_L(xa,y)$. Además, de acuerdo con la definición (5.17), y es única. Nótese además, que $x \in \mathrm{Sp}(L) \Rightarrow xa \in K(L)$, por lo que el resultado de $\mathtt{equiv}_L(xa,y)$ está bien definido. ■

El siguiente lema demuestra que el problema de la inferencia se puede reducir al de aprender $\mathrm{Pr}(L)$ junto a la función $\mathtt{equiv}_L$.

Lema 2 *La estructura del generador canónico de L puede ser obtenida a partir de* $\mathtt{equiv}_L$ *y de cualquier conjunto* $A \subset \mathrm{Pr}(L)$ *tal que* $K(L) \subset A$ *con el algoritmo de la figura 5.1, que además proporciona como resultado adicional los conjuntos* $\mathrm{Sp}(L)$ *y* $F(L)$.

Demostraci'on. Por inducción en el número i de iteraciones, se obtiene que $\mathrm{Sp}^{[i]} \subset \mathrm{Sp}(L)$, $F^{[i]} \subset F(L)$ y que $W^{[i]} \subset K(L)$, donde el superíndice denota el resultado después de i iteraciones. Además, si xa pertenece a $K(L)$, se puede comprobar por inducción en la longitud de la cadena que xa siempre entra a formar parte de $W^{[i]}$ para algún valor de i. Siguiendo el lema 1, para cada cadena $x \in \mathrm{Sp}(L)$, si xa está también en $\mathrm{Sp}(L)$, entonces $\delta(x,a) = xa$. Por contra, si $xa \notin \mathrm{Sp}(L)$, es decir, $xa \in F(L)$, entonces existe una única $y \in \mathrm{Sp}(L)$ tal que $\mathtt{equiv}_L(xa,y)$ y $\delta(x,a) = y$. ■

En el algoritmo anterior, podemos reemplazar A por $\mathrm{Pr}(L_n) \subset \mathrm{Pr}(L)$, dado que este conjunto contiene a $K(L)$ con probabilidad creciente al aumentar n. Por otro lado, conviene destacar que $\mathtt{equiv}_L$

permanece siempre bien definida, porque todas las llamadas a esta función se producen dentro de su dominio $K(L)$. Además, y como consecuencia de ello, el algoritmo realiza como mucho $|K(L)| \times |\mathrm{Sp}(L)|$ llamadas a $\texttt{equiv}_L$. Por tanto, la complejidad global del algoritmo será $\mathcal{O}(|\mathcal{A}||M|^2|)$ multiplicado por la complejidad de $\texttt{equiv}_L$.

5.4 Convergencia del algoritmo

Para evaluar la relación de equivalencia $x^{-1}L = y^{-1}L$, es posible utilizar una variación de (6.12) que mejora la convergencia:

$$L_1 = L_2 \Leftrightarrow p(a\mathcal{A}^*|z^{-1}L_1) = p(a\mathcal{A}^*|z^{-1}L_2)\ \forall a \in \mathcal{A}, z \in \mathcal{A}^* \qquad (5.25)$$

Teniendo en cuenta (5.9), la relación anterior significa que para todas las cadenas $z \in \mathcal{A}^*$ y para todos los símbolos $a \in \mathcal{A} \cup \{\lambda\}$ se satisface

$$p^M((xz)^{-1}L, a) = p^M((yz)^{-1}L, a) \qquad (5.26)$$

En la práctica, no se conoce el lenguaje L y la función $\texttt{equiv}_L(x, y)$, definida como $x^{-1}L = y^{-1}L$, debe ser reemplazada por una función experimental $\texttt{comp}_\texttt{n}(x, y)$, que comprueba si $x^{-1}L_n$ coincide con $y^{-1}L_n$. Esto es, utilizamos los valores p^T del 'arbol generador de prefijos en vez de los desconocidos p^M en (5.26). La figura 5.2 muestra una implementación de la función $\texttt{comp}_\texttt{n}(x, y)$.

Como S_n es un muestra aleatoria, es preciso dar un margen de confianza a la diferencia entre las probabilidades de las cadenas en $x^{-1}L_n$ y $y^{-1}L_n$, tal y como lo hace la función `different` en $\texttt{comp}_\texttt{n}$, que permite una cierta holgura en los resultados. Existen numerosos criterios estadísticos (Hoeffding 1963; Feller 1950; Anthony & Biggs 1992) que conducen a diferentes variaciones del algoritmo básico. En este trabajo, hemos elegido la cota de Hoeffding (1963) descrita en la sección 3.5. Recordemos, que para una variable de Bernoulli con probabilidad p y frecuencia observada f/n, dado $\alpha > 0$ y

$$\epsilon_\alpha(m) = \sqrt{\frac{1}{2m}\log\frac{2}{\alpha}} \qquad (5.27)$$

entonces, con probabilidad mayor que $1-\alpha$,

$$\left| p - \frac{f}{m} \right| < \epsilon_\alpha(m) \tag{5.28}$$

De esta relación se deduce inmediatamente que, dadas dos variables de Bernoulli con probabilidades p y p' respectivamente, con probabilidad mayor que $(1-\alpha)^2$,

$$\begin{aligned} &\left| \tfrac{f}{m} - \tfrac{f'}{m'} \right| < \epsilon_\alpha(m) + \epsilon_\alpha(m') \quad \text{si} \qquad p = p' \\ &\left| \tfrac{f}{m} - \tfrac{f'}{m'} \right| > \epsilon_\alpha(m) + \epsilon_\alpha(m') \quad \text{si} \quad |p - p'| > 2(\epsilon_\alpha(m) + \epsilon_\alpha(m')) \end{aligned} \tag{5.29}$$

Debido a que $\lim_{m\to\infty} \epsilon_\alpha(m) = 0$, sólo una de las dos condiciones anteriores será cierta cuando m y m' son lo suficientemente grandes.

Esta comprobación ha sido implementada a través de la función `different` (representada en la figura 5.3), que comprueba si $p = p'$ con un nivel de confianza $(1-\alpha)^2$ para valores suficientemente grandes de m y m'.

Debido a que el número de llamadas a `different` crece si el tamaño t del 'arbol generador de prefijos aumenta, permitiremos que el parámetro α dependa de t. Incluso en ese caso, ϵ_α presenta el límite correcto para valores grandes de n, pues ϵ_α muestra una dependencia logarítmica respecto a α, y el comportamiento final depende moderadamente del valor exacto de α.

De acuerdo con (5.26), la compatibilidad de dos estados x e y de Q^T debe ser rechazada si existe alguna cadena $z \in \mathcal{A}^*$ tal que las probabilidades de transición desde xz e yz que han sido estimadas a partir de la muestra son diferentes según `different`. Demostraremos a continuación que $\texttt{comp}_\texttt{n}(x, y)$, representado en la figura 5.2, devuelve en el límite de $n \to \infty$ el mismo valor que $\texttt{equiv}_L(x, y)$ para todos los pares de cadenas x, y de $K(L)$. Por consiguiente, de acuerdo con el lema 2, la estructura correcta del generador canónico es inferida en el límite y las probabilidades de transición $p^M(x, a)$ definidas en (5.9) pueden

ser evaluadas a partir de S_n por medio de los valores experimentales $p^T(x,a)$, definidos por medio de (5.22).

Teorema 3 *Sea $t = |T_n|$ el tamaño del 'arbol generador de prefijos para S_n y $\alpha(t)$ un nivel de significación tal que $\lim_{t\to\infty}(1-\alpha(t))^t \to 1$. Entonces, con probabilidad uno, ambas funciones $\texttt{equiv}_L(x,y)$ y $\texttt{comp}_\texttt{n}(x,y)$ devuelven, excepto para un número finito de valores de n, el mismo valor para todos las cadenas $x, y \in K(L)$.*

Demostraci'on. Siguiendo (5.26), el bucle sobre z en la función $\texttt{comp}_\texttt{n}$ comprueba, para todos los subárboles con raíz en x e y, si las probabilidades de transición $p^T(xz,a)$ y $p^T(yz,a)$ son semejantes. También compara $p^T(xz,\lambda)$ con $p^T(xz,\lambda)$ en cada nodo. Hay como mucho $t-1$ arcos más t nodos en un subárbol y, por tanto, el número máximo de llamadas a `different` desde $\texttt{comp}_\texttt{n}$ es $2t$. Como `different` trabaja con un nivel de confianza por encima de $(1-\alpha)^2$, $\texttt{comp}_\texttt{n}(x,y)$ devolverá el mismo resultado que $\texttt{equiv}_L(x,y)$ con probabilidad mayor que $(1-\alpha(t))^{4t}$. ■

Por ejemplo, la condición $(1-\alpha(t))^t \to 1$ se satisface si $\alpha(t)$ decrece más rápidamente que $1/t$. Una consecuencia inmediata de la demostración anterior es que la complejidad de $\texttt{comp}_\texttt{n}$ está acotada por $|T_n|$ y por tanto, de acuerdo con la discusión del final de la sección anterior, el algoritmo `rlips` trabaja con complejidad temporal $\mathcal{O}(|T_n||\mathcal{A}||M|^2)$. Como $|T_n|$ no puede crecer más rápidamente que n, el algoritmo es, en el límite de muestras grandes, lineal con respecto al tamaño de la muestra.

Por otro lado, dado que `rlips` solamente necesita que $\texttt{comp}_\texttt{n}(x,y)$ sea correcta dentro del conjunto finito $K(L)\times \mathrm{Sp}(L)$, existe un natural n_0 tal que si $n > n_0$:

- $\Pr(L_n) \subset K(L)$;
- todas las llamadas a $\texttt{comp}_\texttt{n}$ devuelven el valor correcto.

En ese punto, `rlips` encuentra siempre la estructura correcta del generador canónico.

5.5 Número de ejemplos necesarios para la convergencia

Una cuestión interesante es el número de ejemplos que el algoritmo necesita para inferir correctamente el SDFA. Este número depende extraordinariamente de la estructura detallada del autómata. Sin embargo, es posible establecer una cota válida para cualquier algoritmo de la clase descrita en este trabajo.

Para cada par de cadenas $x_1, x_2 \in \mathrm{Sp}(L)$ tales que $x_1 \neq x_2$, el número aproximado de ejemplos requeridos para distinguir $x_1^{-1}L$ y $x_2^{-1}L$ será una función $\gamma(x_1, x_2)$. Dado que $x_1^{-1}L \neq x_2^{-1}L$, existe una cadena $z \in \mathcal{A}^*$ y un símbolo $a \in \mathcal{A}$, tales que, si denotamos $x_1' = x_1 z$ y $x_2' = x_2 z$, entonces:

$$|p^M(x_1', a) - p^M(x_2', a)| \neq 0. \tag{5.30}$$

No cabe esperar que la convergencia se alcance antes de que el error estadístico devenga menor que la diferencia anterior. Una estimación del error cometido, que es independiente del algoritmo particular utilizado, puede obtenerse tomando la suma $\sigma_1 + \sigma_2$, siendo,

$$\sigma_i \simeq \frac{1}{\sqrt{4np(x_i'\mathcal{A}^*|L)}} \tag{5.31}$$

donde n es el número de ejemplos en la muestra S_n.

Por tanto, sólo podemos esperar que la comparación de x_1 y x_2 sea correcta una vez que $n > N(x_1, x_2, z, a)$, siendo

$$\begin{aligned} N(x_1, x_2, z, a) = & \qquad (5.32) \\ = & \left(\frac{1}{p^M(x_1', a) - p^M(x_2', a)}\right)^2 \left(\frac{1}{2\sqrt{p(x_1'\mathcal{A}^*|L)}} + \frac{1}{2\sqrt{p(x_2'\mathcal{A}^*|L)}}\right)^2 \end{aligned}$$

Podemos tomar $\gamma(x_1, x_2) = \min_{(z,a)}\{N(x_1, x_2, z, a)\}$, dado que cualquier cadena z y cualquier símbolo a son válidas para establecer que x_1 y x_2 son incompatibles. La comparación más difícil establece una cota inferior para la dificultad de identificar el generador canónico:

$$\Gamma = \max_{x_1, x_2 \in \mathrm{Sp}(L)} \{\gamma(x_1, x_2) : x_1 < x_2\} \tag{5.33}$$

La misma cota Γ se obtiene si $x_1 \in \mathrm{Sp}(L)$ y $x_2 \in F(L)$, pero en este caso es suficiente con comparar x_2 con todos los prefijos x_1 que preceden a z_2, es decir $x_1 < z_2$, siendo z_2 la única cadena en $\mathrm{Sp}(L)$ equivalente a x_2. Como ejemplo, para la gramática de Reber de la figura 5.4, se obtiene como cota inferior $\Gamma \simeq 330$ (correspondiente al caso $x = BT, y = BTX$ y $z = \lambda$), un valor acorde con el comportamiento observado en la figura 5.5.

5.6 Resultados y discusión

Se ha estudiado el comportamiento del algoritmo para una serie de gramáticas distintas. Para cada gramática, se generaron diferentes muestras usando el autómata canónico y éstas fueron utilizadas posteriormente como entrada para `rlips`. Por ejemplo, se utilizó la gramática de Reber (1967) de la figura 5.4 con el objetivo de comparar los resultados con otros previos en los que redes neurales eran entrenadas con este lenguaje (Castaño, Casacuberta & Vidal 1993).

En la figura 5.5 se representa el promedio después de 10 experimentos del número de nodos del autómata propuesto por `rlips` en función del tamaño del conjunto de muestra generado por la gramática de Reber. Tal y como se aprecia en la figura, el número de estados converge al valor correcto cuando la muestra es lo suficientemente grande. Para comprobar que no sólo el número de estados, sino también la estructura había sido inferida correctamente, se calculó la entropía relativa $H(L, A)$ entre el lenguaje correcto L y la hipótesis generada A. Este resultado aparece representado en la figura 5.6. Como referencia, también se ha dibujado $H(L, L_n)$, la entropía relativa de la muestra o, lo que es equivalente, del 'arbol generador de prefijos con respecto a la gramática objetivo. Esta última converge mucho más lentamente, remarcando la importancia de haber identificado la estructura del generador canónico a la hora de estimar las probabilidades de las cadenas del lenguaje. La diferencia no estriba sólo en que el 'arbol generador de prefijos asigna una probabilidad cero a muchas cadenas

con probabilidad de aparición no nula. Además, una vez identificada la estructura, el número de probabilidades a estimar se reduce significativamente al pasar de infinito a finito ($|M||\mathcal{A}|$).

Tal y como sugiere la figura (5.5), cuando el número de ejemplos es pequeño, el algoritmo tiende a proponer hipótesis demasiado simples (que, de alguna forma, generalizan excesivamente). Sin embargo, en cuanto el número de ejemplos disponibles es suficiente, el algoritmo identifica la estructura correcta del generador canónico. El número de ejemplos que requiere esta identificación es relativamente pequeño (aproximadamente quinientos) y consistente con la cota que se obtuvo en la sección anterior. Comparativamente, este resultado es mucho mejor que el rendimiento obtenido utilizando redes neurales (Castaño, Casacuberta & Vidal 1993) que no garantizan la convergencia en las pruebas realizadas con esta gramática, incluso utilizando decenas de miles de ejemplos aleatorios.

En la figura 5.7, se representa el tiempo medio que consume el algoritmo en función del número de ejemplos de la muestra (las dispersiones observadas son inapreciables en la figura). Aquí se comprueba que la complejidad temporal es lineal y que además el algoritmo es muy rápido incluso para muestras enormes.

El funcionamiento de `rlips` para la gramática de Reber fue comparado con el uso de modelos de Markov ocultos entrenados por el procedimiento de Baum-Welch. Cuando el número de estados del modelo es similar al de la gramática, el procedimiento BW encuentra el modelo correcto. Sin embargo los tiempos de ejecución son enormes comparados con los de `rlips`. Por ejemplo, usando 8 estados y 400 iteraciones (número que se encontró idóneo experimentalmente para la identificación) en el método BW, el algoritmo `rlips` resultó del orden de 1000 veces más rápido. Nótese que la complejidad de `rlips` depende sólo de la dificultad intrínseca del problema (tamaño de la muestra y dificultad del autómata que se debe identificar), mientras que en el método de Baum-Welch depende del conocimiento *a priori* que se tenga del problema, por ejemplo, el número máximo de estados

en el modelo. Por otro lado, si el número de estados en el modelo de Markov es mucho más grande que el del generador canónico, sus predicciones tienden a ser las de L_n, pues el entrenamiento BW no contiene ninguna preferencia por los modelos más sencillos. Como en la práctica resulta prohibitivo utilizar un gran número de estados, la situación es más bien la contraria: con frecuencia el modelo de Markov no tendrá el número de estados suficiente como para identificar el modelo correcto. En cambio, en el algoritmo `rlips` el número de estados se elige automáticamente.

La gramática de Reber 5.4 no revela toda la riqueza del algoritmo, pues los estados de la gramática pueden ser identificados directamente por sus probabilidades de transición. En cambio, en un caso como el de la figura 5.8, la separación de los estados ha de realizarse de forma indirecta, pues algunos de ellos presentan las mismas probabilidades de transición. Se trata, por tanto, de un caso en el que la identificación de la estructura es más difícil. El número de transiciones obtenido por `rlips` para esta gramática aparece representado en la figura 5.9, donde se observa que el comportamiento del algoritmo es el adecuado. También se ha representado la entropía relativa del modelo a la hipótesis y a la muestra aleatoria respectivamente en la figura 5.10. De nuevo, el algoritmo construye una hipótesis que converge rápidamente al lenguaje correcto.

Finalmente, en la figura 5.11 se representa el número de ejemplos con los que se alcanza experimentalmente la convergencia, en función de la cota Γ obtenida en la sección 5.5 Para ello se generaron gramáticas al azar de hasta ocho estados. Tal y como se observa en la figura, el número de ejemplos resulta superior en todos los casos a la cota Γ.

```
algorithm rlips
input: A ⊂ Pr(L) tal que K(L) ⊂ A
output: Q^M = Sp (prefijos cortos)
        F (frontera)
        δ^M (función de transición)
begin algorithm
  F = ∅
  Sp = {λ}
  W = A* ∩ A
  do ( while W ≠ ∅ )
     x = min W
     W = W − {x}
     wa = x : w ∈ A* ∧ a ∈ A
     if ∃y ∈ Sp : equiv_L(x, y) then
        F = F ∪ {x}
        δ^M(w, a) = y
     else
        Sp = Sp ∪ {x}
        W = W ∪ {xa ∈ A : a ∈ A}
        δ^M(w, a) = x
     endif
  end do
end algorithm
```

Figura 5.1: Algoritmo `rlips`.

```
algorithm compatible
input: x, y (cadenas)
       T_n ( 'arbol generador de prefijos )
output: boolean
begin algorithm
  do ( ∀z ∈ A*: xz ∈ Pr(L_n) ∨ yz ∈ Pr(L_n ) )
     if different (c_n(xz), c_n(xzA*), c_n(yz), c_n(yzA*), α) then
        return FALSE
     endif
     do ( ∀a ∈ A )
        if different (c_n(xzaA*), c_n(xzA*), c_n(yzaA*), c_n(yzA*), α) then
           return FALSE
        endif
     end do
  end do
  return TRUE
end algorithm
```

Figura 5.2: Algoritmo `compatible`

```
algorithm different
input: f, n, f', n', α
output: boolean
begin algorithm
  if n = 0 or n' = 0 then
     return FALSE
  endif
  return |f/n − f'/n'| > ε_α(n) + ε_α(n')
end algorithm
```

Figura 5.3: Algoritmo `different`

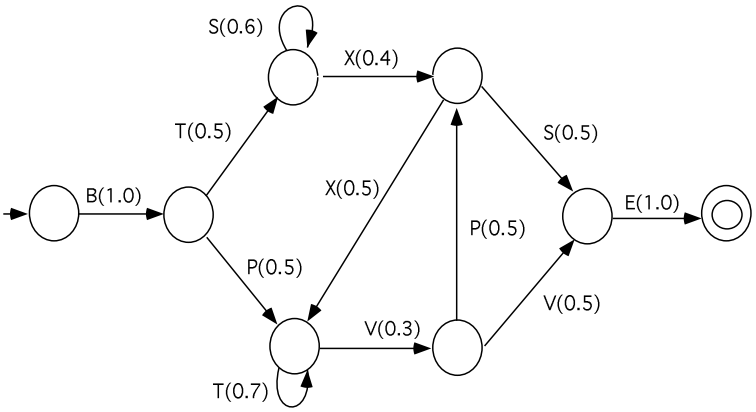

Figura 5.4: Autómata correspondiente a la gramática de Reber

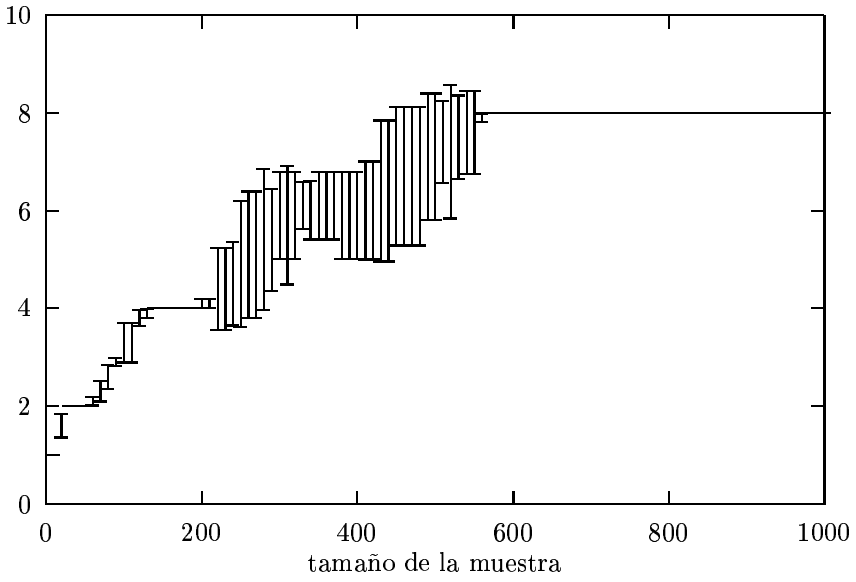

Figura 5.5: Número de nodos en la hipótesis para la gramática de Reber en función del tamaño de la muestra.

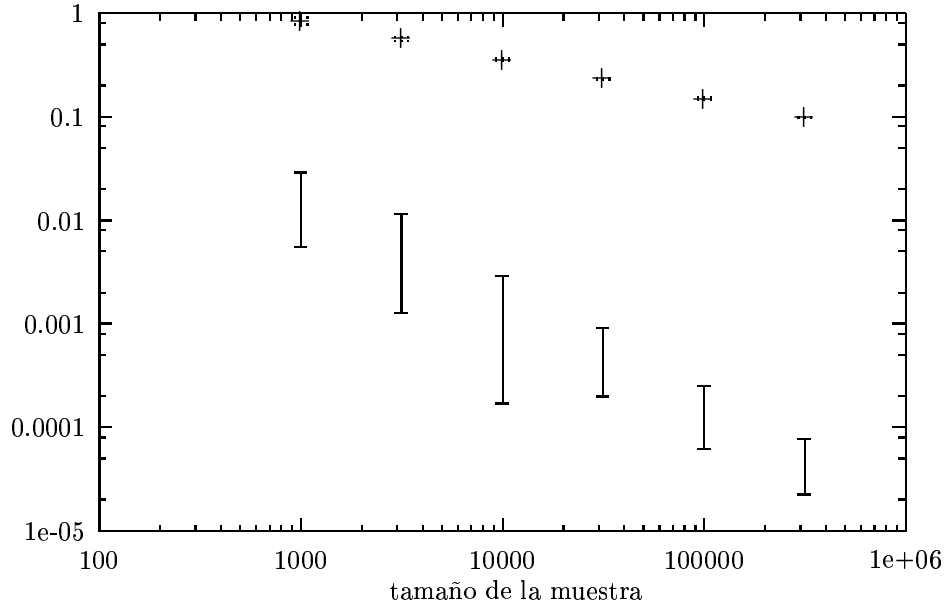

Figura 5.6: Entropía relativa (en bits) entre la gramática de Reber y: 1) la hipótesis propuesta por `rlips` (curva inferior); 2) el conjunto de muestra (curva superior).

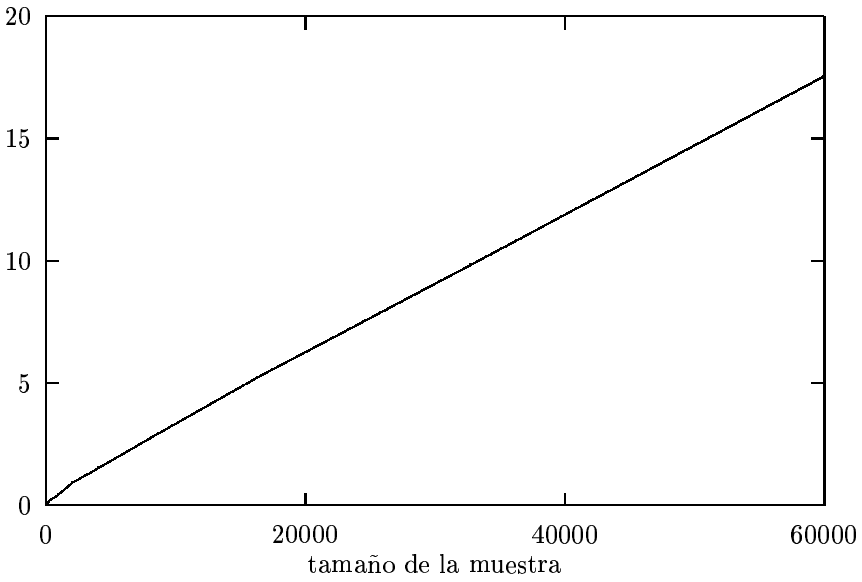

Figura 5.7: Tiempo (en segundos) que requiere la implementación del algoritmo al ser ejecutado en un Hewlett-Packard 715 (40 MIPS) en función del tamaño de la muestra.

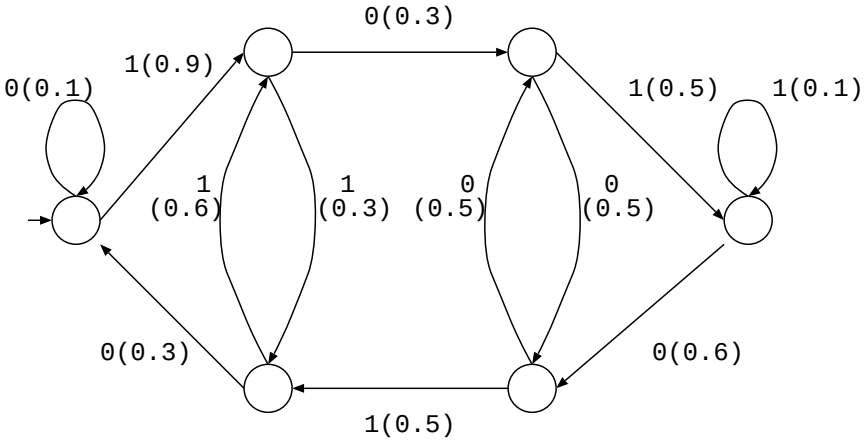

Figura 5.8: Autómata estocástico con estados cuyas probabilidades de transición son idénticas.

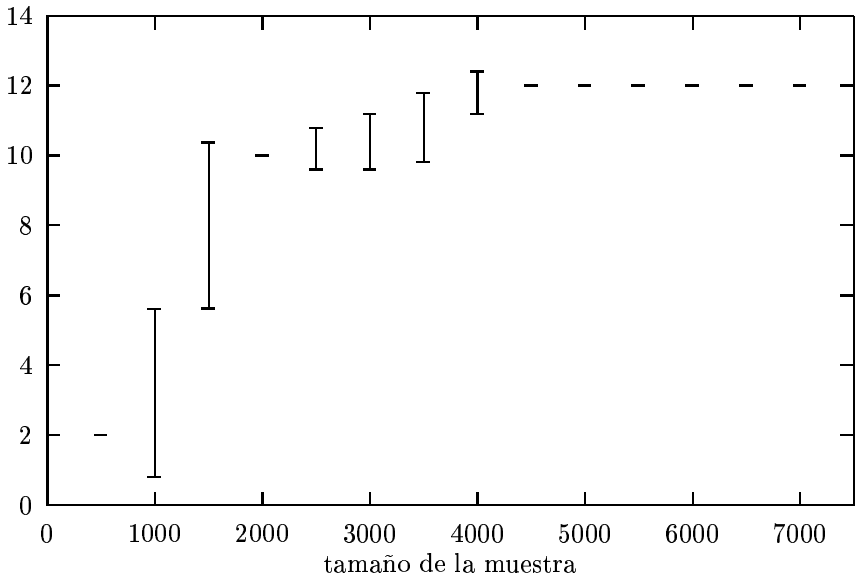

Figura 5.9: Número de nodos obtenido por `rlips` para la segunda gramática de prueba.

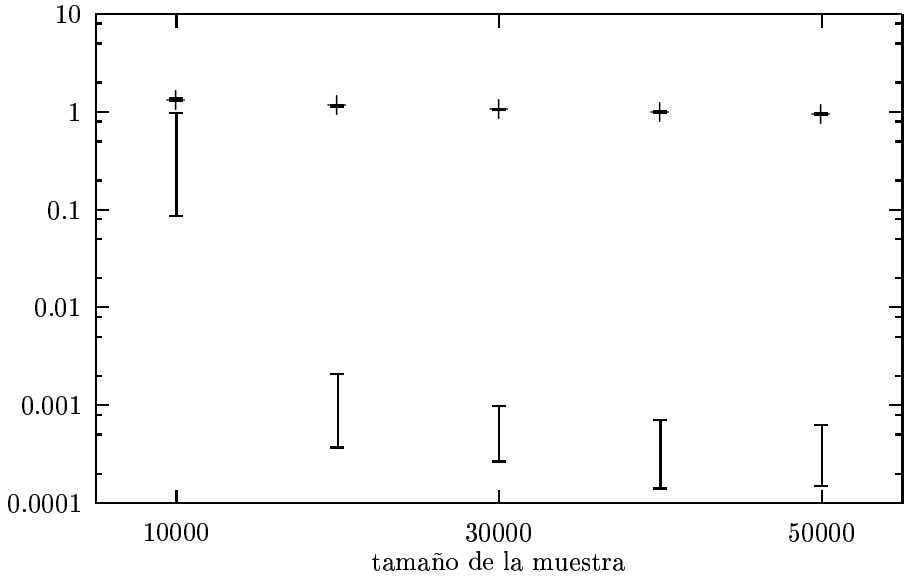

Figura 5.10: Entropía relativa entre la segunda gramática y la hipótesis propuesta por `rlips` comparada con la entropía relativa entre la gramática y el conjunto de muestra.

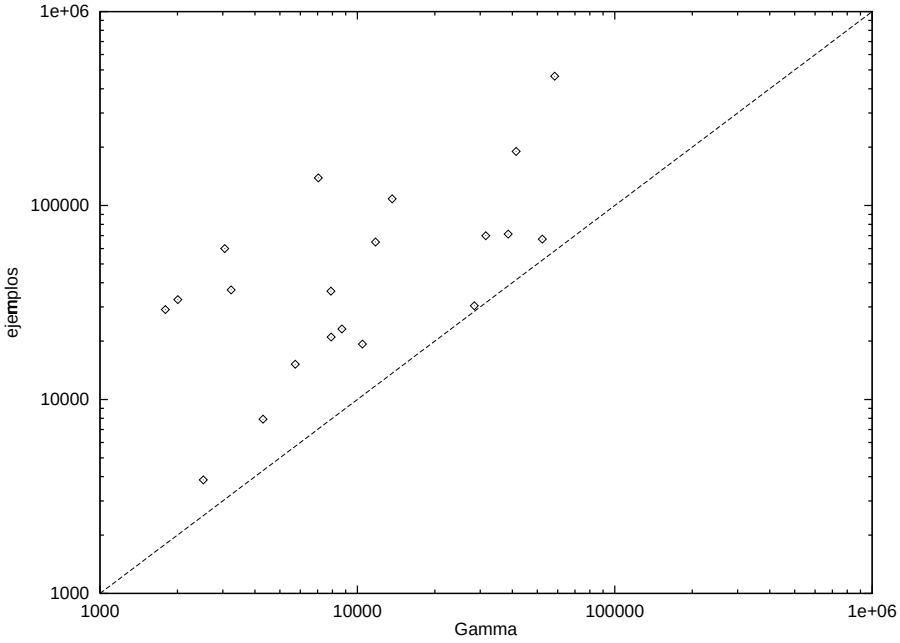

Figura 5.11: Número de ejemplos para los que se alcanza la convergencia en función de la cota teórica establecida.