USO DEL ALGORITMO INSIDE-OUTSIDE EN LA ESTIMACIÓN DE PARÁMETROS DE UNA GRAMÁTICA INCONTEXTUAL PROBABILÍSTICA

ASTRID YINNET ÁLVAREZ CASTRO
JHONY FERNEY IBARRA VÁSQUEZ

UNIVERSIDAD DEL CAUCA
FACULTAD DE CIENCIAS NATURALES, EXACTAS Y DE LA EDUCACIÓN
DEPARTAMENTO DE MATEMÁTICAS
POPAYÁN
2009

USO DEL ALGORITMO INSIDE-OUTSIDE EN LA ESTIMACIÓN DE PARÁMETROS DE UNA GRAMÁTICA INCONTEXTUAL PROBABILÍSTICA

ASTRID YINNET ÁLVAREZ CASTRO
JHONY FERNEY IBARRA VÁSQUEZ

Presentado como requisito parcial
para optar al título de
MATEMÁTICO

Director
Doctor FREDY ÁNGEL AMAYA

UNIVERSIDAD DEL CAUCA
FACULTAD DE CIENCIAS NATURALES, EXACTAS Y DE LA EDUCACIÓN
DEPARTAMENTO DE MATEMÁTICAS
POPAYÁN
2009

Nota de aceptación

_____

_____

_____

Director

_____

Dr. Fredy Ángel Amaya

Jurados

_____

Mg. Mauricio Maca

_____

Esp. Diego Correa

Fecha de sustentación: Popayán, Julio 2 de 2009.

Este triunfo a Dios, a la Universidad nuestros mas sinceros agradecimientos, a nuestro querido director Freddy Amaya, quien con su dedicación y paciencia hizo posible la culminación de este trabajo, a los profesores Mauricio Maca, Diego Correa, Luis Eduardo Montoya y Hevert Vivas por sus oportunos consejos, a nuestras familias porque nos dieron la oportunidad de dar este gran paso y a todos los profesores del departamento de matemáticas por sus enseñanzas y voces de aliento.

A Dios por que todo lo que soy y espero ser está en El, a mi familia porque sin su ayuda y paciencia no hubiera salido adelante, a mis amigos de siempre, a Danilo por su apoyo y compañia incondicional y al monito Jhony por que me enseñó que las cosas siempre pueden ser mejores.

Astrid

A Dios por iluminar mi camino, a mi familia por su apoyo y esfuerzo a Pillyta que me apoyó todo este tiempo, por su amor, comprensión y empeño para que consiguiera este logro y a Astrid mi amiga y compañera por su entusiasmo y perseverancia para lograr nuestro objetivo.

Jhony

Índice general

Notación

$\mathbb{R}$	Conjunto de los números reales.
$x \in$ A	x es elemento de A.
$x \notin$ A	x no es elemento de A.
ϕ	Conjunto vacío.
$\mathbb{N}$	Conjunto de los números naturales (cardinales).
div	División entera.
mod	Módulo.
$Dec(x)$	Decrementa x en 1.
$Inc(x)$	Incrementa x en 1.
$Length(M)$	Cantidad de elementos de un arreglo M.
$Setlength(M, n)$	Dimensiona un arreglo M con longitud n.
Ln	Logarítmo natural
e^x	Función exponencial

Introducción

En cumplimiento parcial de los requisitos para la obtención del título de Matemático presentamos el Proyecto titulado **"USO DEL ALGORITMO INSIDE-OUTSIDE EN LA ESTIMACIÓN DE PARÁMETROS DE UNA GRAMÁTICA INCONTEXTUAL PROBABILÍSTICA"** que se realiza al interior del grupo de Matemática Computacional, en la línea de Reconocimiento de Formas.

Este trabajo se enmarca dentro del proyecto de investigación del grupo de Matemática computacional (proyecto 1579 de la Vicerrectoria de Investigación de la Universidad del Cauca) titulado "Nuevas Alternativas para la Estimación de los Parámetros en una Gramática Incontextual Probabilística", al cual se le aporta la implementación computacional, el análisis de costes y las estructuras de datos de los algoritmos Inside, Outside e Inside-Outside.

Las Gramáticas Incontextuales Probabilísticas **(GIP)** son modelos, ideales para las aplicaciones computacionales que involucran elementos del lenguaje natural, tales como el reconocimiento automático del habla, la traducción automática, el modelado del lenguaje, así como en otras aplicaciones de reconocimiento sintáctico de patrones [SJ99]. Las gramáticas capturan la información sintáctica y pragmática de manera muy eficaz, pero lo que hace difícil su aplicación es el coste temporal del entrenamiento de sus parámetros.

El problema de la estimación de los parámetros de una GIP consiste en estimar las probabilidades de sus reglas a partir de una muestra. Para abordar este problema se utiliza alguna función objetivo dependiente tanto de la muestra como de las probabilidades de las reglas, y un marco para optimizarla. Una de las funciones objetivo que habitualmente se usa es la verosimilitud de la muestra [Bak79, LY90, Ney92, Cas96], la cual puede ser maximizada mediante el Teorema de Transformaciones Crecientes [Be67] en el cual se basa el desarrollo del algoritmo Inside-Outside (IO) [Bak79, LY90, Ney92,

Cas96]. En esta transformación se procede iterativamente, incrementando el valor de la función hasta alcanzar un óptimo local. En este proceso de estimación, se consideran todas las posibles derivaciones de cada cadena de la muestra según la gramática y, aunque necesita un elevado número de iteraciones para converger, los modelos obtenidos por el algoritmo ofrecen, en general, buenos resultados.

Las anteriores consideraciones nos llevaron a estudiar muy detalladamente este algoritmo, la función que él optimiza y el método utilizado para la estimación, trabajo que se viene desarrollando desde el segundo período académico de 2005 enmarcado dentro del proyecto de investigación mencionado anteriormente.

Capítulo 1

Preliminares

El presente proyecto tiene como tema central los lenguajes formales generados por una gramática libre de contexto; en este capítulo, se presentan las definiciones necesarias que serán utilizadas en el documento [SM04].

1.1. Alfabetos, cadenas y lenguajes

Definición 1.1. Un alfabeto o vocabulario, denotado como Σ, es un conjunto finito de símbolos. Una cadena (o palabra) x es una secuencia finita de símbolos de Σ.

Definición 1.2. La longitud de una cadena x es el número de símbolos que tiene, y se escribirá como $|x|$.

Definición 1.3. Se define la cadena vacía como aquella que no posee ningún elemento y se denota ϵ, $|\epsilon| = 0$.

Definición 1.4. Se define Σ^* como el conjunto de cadenas de longitud mayor o igual que 0 que se pueden formar con símbolos de Σ . En forma similar se denota Σ^+ al conjunto de todas las cadenas de longitud mayor o igual que 1 que se puede formar con elementos de Σ , es decir, $\Sigma^+ = \Sigma^* - \{\epsilon\}$.

Definición 1.5. Un lenguaje (formal) L sobre Σ es un subconjunto de Σ^*.

1.2. Gramáticas formales e incontextuales

Definición 1.6. Una *gramática formal* es un modelo matemático que permite especificar un lenguaje, es decir, un conjunto de reglas que generan todas las posibles cadenas de elementos de este, y es representada como una 4-tupla $G = (N, \Sigma, P, S)$ donde:

Σ: Es un conjunto finito denominado conjunto de terminales, (alfabeto).

N: Es un conjunto finito denominado conjunto de no terminales, $\Sigma \cap N = \phi$.

P: Es un conjunto finito de reglas llamado conjunto de reglas de producción.

S: Es el símbolo inicial de la gramática. $S \in N$.

En adelante utilizaremos las siguientes convenciones con respecto a las gramáticas.

1. Las letras mayúsculas del alfabeto castellano (A, B, C,...,Z) representan no terminales; S es el símbolo inicial.
2. Las letras minúsculas del alfabeto castellano (a,b,c,...,t) y los dígitos (0, 1, 2,...) serán los terminales.
3. Las últimas letras minúsculas del alfabeto castellano (u, v, w, x, y, z) denotan cadenas de terminales.
4. Las letras griegas minúsculas $(\alpha, \beta, \delta, \gamma, \sigma, \omega)$ denotan cadenas de no terminales y terminales.

Las reglas de la gramática (elementos de P) se escriben en la forma $\alpha \rightarrow \beta$, donde α se denomina antecedente y β se denomina consecuente.

El símbolo $\alpha \rightarrow \beta$ se lee "α deriva en β". Por ejemplo, si $\alpha \rightarrow \beta$ es una regla de producción que se aplica a la cadena $\gamma\alpha\delta$ para derivar la cadena $\gamma\beta\delta$ esto se denota $\gamma\alpha\delta \Rightarrow \gamma\beta\delta$.

Supóngase que $\alpha_1, \alpha_2, ..., \alpha_m$ son cadenas de $(N \cup \Sigma)^*$, $m \geq 1$ y $\alpha_1 \Rightarrow \alpha_2 \Rightarrow ... \Rightarrow \alpha_{m-1} \Rightarrow \alpha_m$, entonces se escribe $\alpha_1 \overset{*}{\Longrightarrow} \alpha_m$ o "α_1 deriva en α_m.$^{\text{en}}$ la gramática G. Si α_1 deriva en α_m exactamente en i pasos, se escribe $\alpha_1 \overset{i}{\Longrightarrow} \alpha_m$.

Si $A \rightarrow \alpha_1, A \rightarrow \alpha_2, ..., A \rightarrow \alpha_m$ son las reglas de producción que tienen como antecedente la variable A de alguna gramática, entonces podemos expresarlas mediante

la notación $A \rightarrow \alpha_1|\alpha_1|\alpha_2|...|\alpha_m$, en donde la línea vertical se lee "o".

Una derivación a izquierda de una cadena $x \in \Sigma^*$ en G, denotada d_x es una sucesión finita de reglas de producción $q_1, q_2, ..., q_m$ denotada como: $d_x = (q_1, q_2, ..., q_m)$, $m \geq 1$ tal que: $(S \overset{q_1}{\Longrightarrow} \alpha_1 \overset{q_2}{\Longrightarrow} \alpha_2 \overset{q_3}{\Longrightarrow} ... \overset{q_m}{\Longrightarrow} x)$, $\alpha \overset{q}{\Longrightarrow} \beta$ significa α deriva en β mediante la regla q, $\alpha_i \in (N \cup \Sigma)^*$, $1 \leq i \leq m-1$ y q_i reescribe el no terminal más a la izquierda de α_{i-1}.

De esta forma la derivación a izquierda d_x queda definida por la secuencia de reglas utilizadas, de forma análoga puede definirse la derivación a derecha. En este trabajo únicamente se considerarán derivaciones a izquierda, por lo que la referencia a ellas será simplemente como derivación d_x.

Ejemplo 1.1. *Dada la GI $G = (\{S, A, B\}, \{a, b\}, S, \{S \rightarrow AS, S \rightarrow AB, A \rightarrow BB, A \rightarrow a, B \rightarrow b\})$ y la cadena bbab, se tiene como derivación asociada de izquierda a derecha $d_x = (S \rightarrow AS, A \rightarrow BB, S \rightarrow AB, B \rightarrow b, B \rightarrow b, A \rightarrow a, B \rightarrow b)$.*

Definición 1.7. Los símbolos alcanzables de una gramática son aquellos que aparecen en cualquier derivación que se produce a partir de su símbolo inicial.

Definición 1.8. Se define una producción inútil como aquella que contiene símbolos no terminales desde los que no se puede llegar a una cadena de terminales y las producciones en las que aparezcan, además dicha producción con símbolos que no sean alcanzables desde el símbolo inicial, S, y las producciones en las que aparecen.

Definición 1.9. El lenguaje generado por G es el conjunto $L(G)$, definido como:

$$L(G) = \left\{ x \in \Sigma^* | S \overset{*}{\Rightarrow} x \right\}$$

Definición 1.10. Una *gramática incontextual* (GI) G es una gramática en la cual el conjunto P está constituido por reglas de la forma $A \rightarrow \alpha$ donde $A \in N$ y

$\alpha \in (N \cup \Sigma)^*$. Se dice que un lenguaje es incontextual si es generado por una gramática incontextual.

Definición 1.11. Una gramática incontextual está en *Forma Normal de Chomsky (FNC)* si no tiene producciones inútiles y si todas sus reglas son de la forma $A \rightarrow BC$ o $A \rightarrow a$, donde $A, B, C \in N$ y $a \in \Sigma$.

Definición 1.12. Un árbol es una colección de elementos llamados *nodos*, uno de los cuales se distingue como *raíz*, junto con una relación de («paternidad») que impone una estructura jerárgica sobre los nodos. Un nodo, como un elemento de una lista, puede ser del tipo que desee. A menudo se representa un nodo por medio de una letra, una cadena de caracteres o un círculo con un número en su interior [AHU88].

Definición 1.13. Un árbol de derivación o de análisis permite mostrar de manera gráfica las posibles formas de derivar cualquier cadena de un lenguaje a partir del símbolo inicial de la gramática, es así como se pueden conocer e interpretar las relaciones que existen entre las distintas partes de una cadena de terminales y las reglas de antecedentes asociadas a estos.

Definición 1.14. Un árbol etiquetado y ordenado t es un *árbol de derivación* o *de análisis* de una cadena en una GI G si [Fu82]:

- Cada nodo del árbol tiene una etiqueta que es un símbolo de $(N \cup \Sigma)$, esto es de terminales y no terminales.
- La raíz del árbol tiene como etiqueta al símbolo inicial de la gramática S.
- Si un nodo cuya etiqueta es A tiene un descendiente directo diferente de él mismo, entonces $A \in N$.
- Si los nodos $n_1, n_2, ..., n_m$ son descendientes directos del nodo n (cuya etiqueta es A) en orden de izquierda a derecha, con etiquetas $A_1, A_2, ..., A_m$ respectivamente, entonces $A \rightarrow A_1 A_2 ... A_m$ es una regla del conjunto de reglas de producción P.

Véase la figura 1.1 que representa un árbol de derivación de una cadena $n_1 n_2 n_3 n_4$ con no terminales $A_1, A_2, A_3, ..., A_6$ y terminales n_1, n_2, n_3 y n_4.

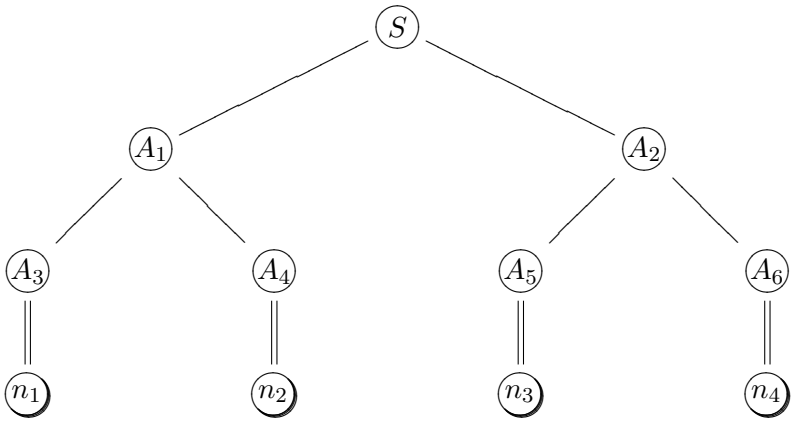

Figura 1.1: Árbol de derivación.

Nótese el uso de las reglas $S \rightarrow A_1A_2, A_1 \rightarrow A_3A_4, A_2 \rightarrow A_5A_6, A_3 \rightarrow n_1, A_4 \rightarrow n_2, A_5 \rightarrow n_3, A_6 \rightarrow n_4$.

Dada una cadena x y una GI G tal que $x \in$ L(G), es posible que exista más de un árbol de análisis que permita derivar x a partir del símbolo inicial S.

Ejemplo 1.2. *Dada la GI* $G = (\{S, A\}, \{a, b\}, S, \{S \rightarrow SS, S \rightarrow AS, S \rightarrow b, S \rightarrow a, A \rightarrow b\})$ *y la cadena baba, tenemos como derivación asociada de izquierda a derecha* $d = (S \rightarrow SS, S \rightarrow SS, S \rightarrow b, S \rightarrow a, S \rightarrow AS, A \rightarrow b, S \rightarrow a)$.

Es así como para la GI definida en el ejemplo 1.1, la cadena *baba* puede ser generada con la derivación mencionada o con la derivación ($S \rightarrow SS$, $S \rightarrow AS$, $A \rightarrow b$, $S \rightarrow a$, $S \rightarrow AS$, $A \rightarrow b$, $S \rightarrow a$). Ver figuras 1.2 y 1.3.

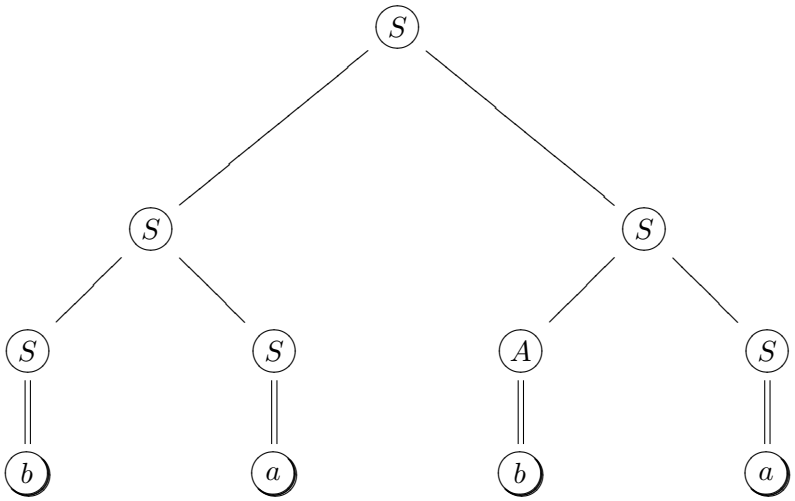

Figura 1.2: Árbol de derivación 1.

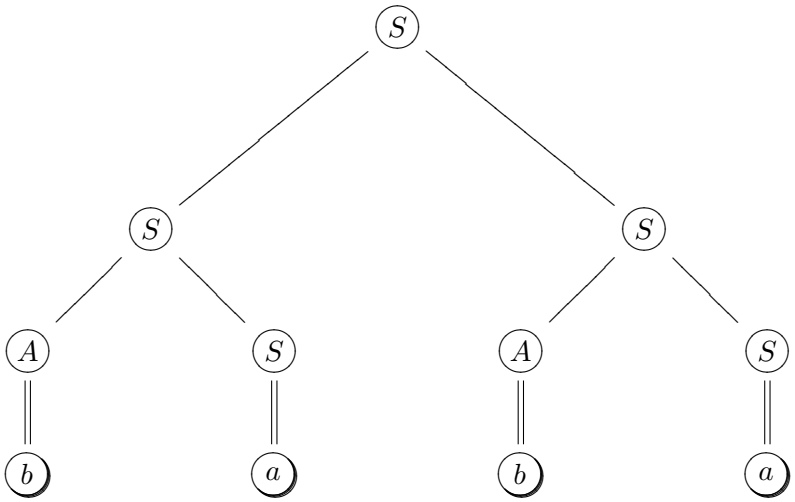

Figura 1.3: Árbol de derivación 2.

1.3. Análisis sintáctico de una cadena

Dada la cadena x y una GI G, el análisis sintáctico consiste en determinar si $x \in L(G)$.

Una solución eficiente para abordar el análisis sintáctico consiste en utilizar la técnica de diseño de algoritmos llamada programación dinámica [HU79]. Uno de los métodos más conocidos es el algoritmo denominado Cocke - Younger - Kasami [HU79], en el cual, dada una cadena y una gramática independiente de contexto en *Forma Normal de Chomsky (FNC)*, determina si la cadena pertenece o no al lenguaje generado por dicha gramática.

El algoritmo se enuncia de la siguiente manera:

Sea G una GI, en FNC. Sea $x = x_1, x_2, x_3, ...x_n$ una cadena de terminales de longitud n. Sea $w_{ij} = x_i, ..., x_j$ la subcadena de x que comienza en la posición i y tiene longitud $j - i + 1$.

- Para cada $j = 1, 2, 3, ..., n$; hacer $N_{1j} = \{A | A \rightarrow w_{ij}\}$.
  Donde, N_{1j} es el conjunto de los no terminales que producen directamente el j-ésimo símbolo de x.

- Para $i = 2, 3, ..., n$, hacer lo siguiente:

  - Para $j = 1, 2, 3, ..., n - i + 1$, hacer lo siguiente:
    - Inicializar N_{ij} con el conjunto vacío. (i,j) representa la fila i y la columna j.
    - Para $k = 1, 2, ..., i - 1$, añadir a N_{ij} todos los símbolos no terminales A para los cuales $A \rightarrow BC$ es una regla de P, con B perteneciente a N_{ik} y C perteneciente a $N_{i-k,j+k}$,

- Si S pertenece a N_{n1}, entonces x pertenece a $L(G)$.

Ejemplo 1.3. *Dada la gramática incontextual GI* $G = (\{S, A, B, C\}, \{a, b\}, S, \{S \rightarrow AB, S \rightarrow BC, A \rightarrow BA, A \rightarrow a, B \rightarrow CC, B \rightarrow b, C \rightarrow AB, C \rightarrow a\})$*, y la cadena bbab.*

Para la cadena x = bbab, se obtiene el cuadro 1.1, donde cada casilla representa al conjunto N_{ij}.

	b	b	a	b
	j = 1	j = 2	j = 3	j = 4
i = 1	B	B	A,C	B
i = 2	ϕ	A,S	S,C	
i = 3	A	S,C		
i = 4	**S**,C			

Cuadro 1.1: Ejemplo algoritmo CYK

Ahora bien, dado que S está en N_{41}, x puede ser generada a partir del símbolo inicial S. Por tanto, bbab pertenece al lenguaje generado por esta gramática.

1.4. Lenguajes y gramáticas formales probabilísticos

Definición 1.15. Un lenguaje probabilístico sobre un alfabeto Σ es un par (L, Φ), donde L es un lenguaje formal sobre Σ y $\Phi : \Sigma^* \to \mathbb{R}$ es una función de probabilidad sobre las cadenas de Σ^*. La función de probabilidad Φ satisface las siguientes condiciones:

1. $x \notin L \Rightarrow \Phi(x) = 0$, Para todo $x \in \Sigma^*$.
2. $x \in L \Rightarrow\ 0 < \Phi(x) \leq 1$, Para todo $x \in \Sigma^*$.
3. $\sum_{x \in L} \Phi(x) = 1$.

Definición 1.16. Se define una Gramática Incontextual Probabilística (GIP) G_p como un par (G, p) tal que G es una GI, denominada en este caso gramática característica y p es una función $p : P \to (0, 1]$ que posee la siguiente propiedad:

$$\forall A \in N, \sum_{(A \to \alpha) \in \Gamma_A} p(A \to \alpha) = 1$$

Donde Γ_A representa el conjunto de reglas de la gramática cuyo antecedente es A.

Las GIP tienen ventajas sobre las GI, por ejemplo las GI a diferencia de las GIP no tienen mecanismos para clasificar las ambigüedades de las frases, los fenómenos de ruido, los aspectos de incertidumbre o cualquiera de los diferentes aspectos aleatorios en el Reconocimiento Sintáctico de Formas (RSF), los cuales requieren un trato apropiado en el análisis de una frase. Para ilustrarlo, se presenta el siguiente ejemplo.

Ejemplo 1.4. *"El hombre vió el sapo con un telescopio"*

Tenemos que la frase preposicional "con un telescopio"podría tener dos significados, podría referirse al proceso de ver un sapo con un telescopio (en tal caso, la frase preposicional es asociada con el verbo de la frase) o podría hacer referencia al proceso de ver un sapo que posee un telescopio (en donde la frase preposicional se asociaría con la frase del nombre). De alguna manera, la primera opción es más creíble que la segunda. Ahora, nótese como se puede representar esto formalmente.
Primero, se presenta una GI capaz de producir la frase anteriormente mencionada:

$S \rightarrow NP\|VP$	$VP \rightarrow V\|NP\|PP$	$N \rightarrow telescopio$
$NP \rightarrow Det\|N$	$Det \rightarrow un$	$Prep \rightarrow con$
$NP \rightarrow Det\|N\|PP$	$Det \rightarrow el$	$V \rightarrow vió$
$PP \rightarrow Prep\|NP$	$N \rightarrow hombre$	
$VP \rightarrow V\|NP$	$N \rightarrow sapo$	

Dada la gramática incontextual anterior, se llega al siguiente árbol de derivación:

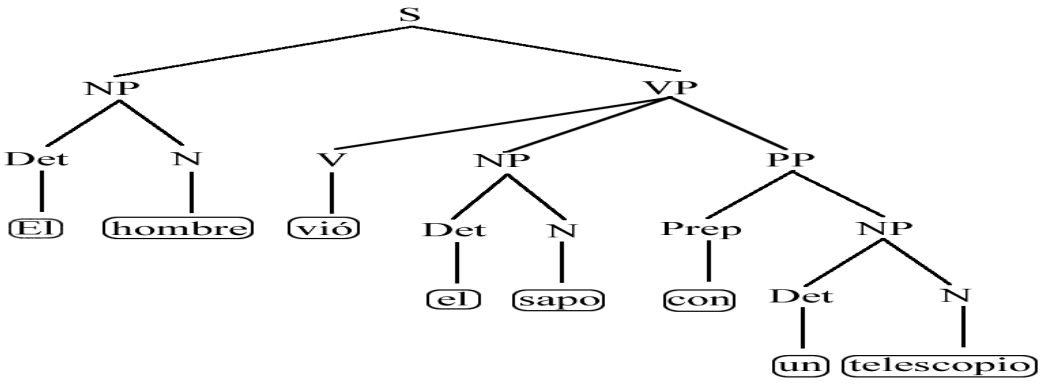

Figura 1.4: Árbol de derivación para la cadena "El hombre vió el sapo con un telescopio" con GI.

Ahora, se presenta una GIP que igualmente produce la frase del ejemplo.

1.0 $S \rightarrow NP\|VP$	*0.3* $VP \rightarrow V\|NP\|PP$	*0.2* $N \rightarrow$ *telescopio*
0.6 $NP \rightarrow Det\|N$	*0.5* $Det \rightarrow$ *un*	*1.0* $Prep \rightarrow$ *con*
0.4 $NP \rightarrow Det\|N\|PP$	*0.5* $Det \rightarrow$ *el*	*1.0* $V \rightarrow$ *vió*
1.0 $PP \rightarrow Prep\|NP$	*0.4* $N \rightarrow$ *hombre*	
0.7 $VP \rightarrow V\|NP$	*0.4* $N \rightarrow$ *sapo*	

Dada la gramática incontextual probabilística anterior, se llega al siguiente árbol de derivación:

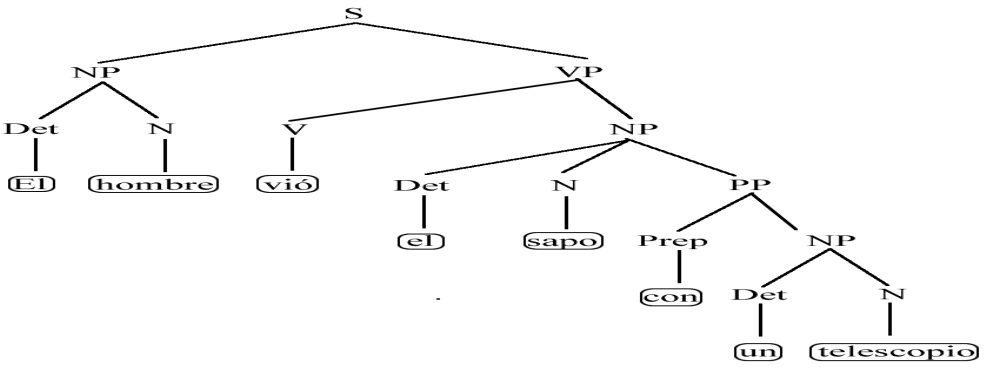

Figura 1.5: Árbol de derivación para la cadena "El hombre vió el sapo con un telescopio" con GIP.

Ahora bien, en este punto no hay forma de saber cómo favorecer una interpretación por encima de la otra. Las dos son consideradas igualmente creíbles. Lo que se podría hacer, es asignar algún peso a un análisis con un sentido semántico válido de la gramática. Una manera de lograrlo es integrar la gramática dentro de un cuerpo de probabilidades. Para convertir una GI en una GIP, se asignan las probabilidades a cada regla en una escena realista, estas probabilidades se computan por medio de un proceso de estimación.

Definición 1.17. Se define la probabilidad de la derivación d_x de la cadena x como:

$$\Pr(x, d_x|G_p) = \prod_{(A\rightarrow\alpha)\in P} p(A \rightarrow \alpha)^{N(A\rightarrow\alpha, d_x)}$$

Donde N(A$\rightarrow \alpha$,d_x) representa el número de veces que la regla A $\rightarrow \alpha$ ha aparecido en la derivación d_x.

Definición 1.18. Se define la probabilidad de la cadena x como:

$$\Pr(x|G_p) = \sum_{d_x \in D_x} \Pr(x, d_x|G_p)$$

Donde D_x denota el conjunto de todas las derivaciones de la cadena x.

Definición 1.19. Se define probabilidad de la mejor derivación de la cadena x como:

$$\hat{\Pr}(x|G_p) = \max_{d_x \in D_x} \Pr(x, d_x|G_p)$$

Definición 1.20. Se define derivación más probable o mejor derivación como:

$$\hat{d_x} = arg \max_{d_x \in D_x} \Pr(x, d_x|G_p)$$

Esto es, la derivación cuya probabilidad es el valor máximo de las probabilidades de las derivaciones de D_x.

Definición 1.21. Se define el *lenguaje generado* por una GIP G_p como:

$$L(G_p) = \{x \in L(G) | Pr(x|G_p) > 0\}$$

Definición 1.22. Una GIP G_p es consistente si y sólo sí:

$$\sum_{x \in L(G)} \Pr(x|G_p) = 1$$

En cualquier otro caso la gramática no es consistente.

Definición 1.23. Dada una GIP G_p consistente, el par $(L(G), \rho)$ es un *lenguaje incontextual probabilístico*, donde ρ es una función de probabilidad computada en términos de la expresión de la definición 1.18.

1.5. Estimación de las GIP

Como se mencionó antes, el problema de la estimación de las GIP consiste en estimar las probabilidades de esta a partir de una muestra compuesta por cadenas del lenguaje que se desea representar, estableciendo cierta función objetivo a optimizar y eligiendo

además un marco para optimizarla.

Para el análisis de la estimación de las GIP se encuentran dos problemas: La elección de la función objetivo a optimizar y el método de optimización. A continuación se describen los términos en los que son enunciados: Sea (L, Φ) un lenguaje probabilístico sobre un alfabeto Σ, tal que $L \subseteq L(G)$ para alguna GI G dada y sea Φ una función de probabilidad, generalmente desconocida. Dada una muestra Ω del lenguaje, que asumimos refleja la distribución (desconocida) Φ, el problema consiste en determinar las probabilidades de las reglas de la GIP a partir de dicha muestra para que representen la función Φ. Para ello es necesario asumir que la función Φ puede representarse por medio de una GIP, es decir que está determinada por los elementos que la representan, esto es, reglas y probabilidades. Por tanto, dada $G_p = (G, p)$ y Ω, se desea encontrar el conjunto de parámetros (probabilidad de las reglas) de la GIP que hacen que la distribución de probabilidad que esta define sobre L se ajuste lo máximo posible a la distribución desconocida Φ de manera que se plantea el problema de determinar ρ' tal que:

$$\rho' := \max_{p} f_p(\Omega) \tag{1.5.1}$$

Donde p es el conjunto de probabilidades de las reglas de la GIP y $f_p()$ es una función a ser optimizada. Por tanto para poder estimar los parámetros de una GIP, es necesario determinar la función a optimizar y el método de optimización que se ha de utilizar.

En este caso la función objetivo a optimizar será la función de verosimilitud que permite, en el caso de un parámetro o un vector de parámetros poblacionales desconocidos, determinar el estimador o vector de estimadores que maximizan la función de probabilidad conjunta de una muestra Ω, definida como: $\prod_{x\in\Omega} Pr(x|G_p)$ y según 1.17 y 1.18 se tiene:

$$\Pr(x|G_p) = \sum_{d_x \in D_x} \left(\prod_{(A\to\alpha)\in P} p(A \to \alpha)^{N(A\to\alpha, d_x)} \right) \tag{1.5.2}$$

Se puede notar que $f_p()$ está representada por un polinomio formado por las sumas del producto de las probabilidades de las derivaciones de la cadena x.

Ahora, para hacer más clara la representación de la función de verosimilitud, se muestra el siguiente ejemplo:

Ejemplo 1.5. *Dada la GI* $G = (\{S, A, B, C\}, \{a, b\}, S, \{S \to AB, S \to BC, A \to BA, A \to a, B \to CC, B \to b, C \to AB, C \to a\})$ *y las cadenas: baaba y bbab, se tiene como derivaciones asociadas para la cadena baaba a:*

$d_1 = S \to BC, B \to b, C \to AB, A \to a, B \to CC, C \to AB, A \to a, B \to b, C \to a;$
$d_2 = S \to AB, A \to BA, B \to CC, B \to b, A \to a, C \to AB, A \to a, B \to b, C \to a$

y para la cadena bbab se tiene:

$d_1 = S \to BC, B \to b, C \to AB, A \to BA, B \to b, A \to a, B \to b;$
$d_2 = S \to AB, A \to BA, B \to b, A \to BA, B \to b, A \to a, B \to b.$

Considerando a R_i *como la probabilidad de la i-ésima regla de la gramática, es decir* $R_1 = p(S \to AB)$, $R_2 = p(S \to BC)$, *etc. Se puede ver según la ecuación 1.5.2 la formación del polinomio correspondiente a las cadenas baaba y bbab:*

$Pr(baaba, dx|G_p) = R_2R_4^2R_5R_6^2R_7^2R_8 + R_1R_3R_4^2R_5R_6^2R_7R_8.$
$Pr(bbab, dx|G_p) = R_2R_3R_4R_6^3R_7 + R_1R_3^2R_4R_6^3.$

Ahora, respecto al método de optimización el de uso más generalizado está basado en el Teorema de Transformaciones Crecientes que se enuncia a continuación.

Teorema De Transformaciones Crecientes (TTC) ([BE67])

Sea $P(\Theta)$ un polinomio homogéneo con coeficientes no negativos de grado d en sus variables $\Theta = \{\Theta_{ij}\}$. Sea $\theta = \theta_{ij}$ un punto del dominio D, donde:

$$D = \{\theta_{ij} | \theta_{ij} \geq 0, \sum_{j=1}^{q_i} \theta_{ij} = 1, i = 1, \ldots, p, j = 1, \ldots, q_i\}$$

y sea $Q(\theta)$ un punto de D tal que:

$$Q(\theta)_{ij} = \frac{\theta_{ij}(\partial P/\partial \Theta_{ij})_\theta}{\sum_{k=1}^{q_i} \theta_{ik}(\partial P/\partial \Theta_{ik})_\theta}$$

donde $\forall i,\ \sum_{k=1}^{q_i} \theta_{ik}(\partial P/\partial \Theta_{ik})_\theta \neq 0$. Entonces, $P(Q(\theta)) > P(\theta)$ excepto si $Q(\theta) = \theta$.

La aplicación del teorema anterior permite obtener un máximo local del polinomio P en el espacio de búsqueda definido por D haciendo uso del algoritmo de descenso por gradiente, que tiene el siguiente esquema:

1. **Entrada** P(Θ)
2. **Método**
3. θ=Valores iniciales
4. **Repetir**
5. Calcular Q(θ) utilizando P(Θ)
6. θ = Q(Θ)
7. **Hasta** Converger
8. **Salida** θ

Dado un polinomio P que cumple las condiciones del Teorema de Transformaciones Crecientes y un punto inicial θ del dominio definido en el teorema, el problema de estimación se plantea como la aplicación repetida de la transformación dependiente de la muestra y definida en términos del conjunto de probabilidades, cuyo objetivo es alcanzar un máximo local de un polinomio definido; en ocasiones lo que suele hacerse es estimar hasta que la diferencia entre dos iteraciones consecutivas no supere a un valor establecido previamente, denominado umbral.

Ahora bien, para el trabajo con este algoritmo se debe tener en cuenta la elección del valor inicial de θ, es decir de las probabilidades iniciales de la GIP G_p:

$$p(A_i \to \alpha_j) = \theta_{ij} \qquad i = 1, ..., |N|; \quad j = 1, ..., |\Gamma_{A_i}|,$$

donde Γ_{A_i} representa el conjunto de reglas de la gramática cuyo antecedente es A_i.

De esta manera, el trabajo a desarrollar consiste en obtener el valor ρ' de la expresión 1.5.1 y donde la función de verosimilitud de la muestra $f_p()$ está dada por:

$$\rho' := \max_{p} \prod_{x \in \Omega} Pr(x|G_p) \tag{1.5.3}$$

Además, dadas las definiciones 1.17 y 1.18 tenemos que:

$$\rho' := \max_{p} \prod_{x \in \Omega} \left[\sum_{d_x \in D_x} \left(\prod_{(A \to \alpha) \in P} p(A \to \alpha)^{N(A \to \alpha, d_x)} \right) \right] \tag{1.5.4}$$

Un método para obtener ρ' consiste en utilizar el algoritmo Inside-Outside(IO) que se evaluará mas adelante, que procede iterativamente maximizando en cada etapa la verosimilitud de la muestra hasta alcanzar un máximo local y su convergencia puede ser convenientemente descrita en el marco del Teorema de Transformaciones Crecientes enunciado anteriormente.

Capítulo 2

Elementos de análisis de algoritmos y algoritmos básicos

En este capítulo se presentan los elementos básicos para el análisis de los algoritmos, su descripción en términos de funcionamiento, datos de entrada y salida, además del orden en el que se encuentre su función de complejidad.

2.1. Análisis de algoritmos

El análisis de algoritmos determina, desde el punto de vista teórico, la cantidad de recursos computacionales que emplea la ejecución de un programa de ordenador. Estos recursos pueden ser por ejemplo: Tiempo de ejecución, cantidad de memoria utilizada o código, etc, donde los más utilizados son el tiempo y el espacio.

2.1.1. Instancia

Una instancia de un problema es un conjunto de datos de entrada del problema, y se obtiene especificando valores para los parámetros. Un algoritmo soluciona un problema A, si el algoritmo aplicado a cualquier instancia I de A garantiza que se obtiene una solución para I, (es decir siempre se obtiene una solución).

2.1.2. Tamaño de una instancia

El tamaño de una instancia formalmente corresponde al número de bits necesarios para representar dicha instancia en un computador, utilizando un esquema de codificación

definido y razonablemente compacto. En adelante, la palabra tamaño se utilizará para indicar cualquier entero que mida de alguna forma el número de componentes de una instancia y se denotará por $|X|$ el tamaño de la instancia X. Por ejemplo, si se hace referencia a las reglas pertenecientes a una gramática, el tamaño de una instancia será el número de reglas de producción y para una cadena, el tamaño de la instancia estará representada por el número de elementos que la componen.

2.1.3. Operación elemental

Una operación elemental es aquella cuyo tiempo de ejecución se puede acotar superiormente por una constante que solamente dependerá de la implementación particular usada: de la máquina, del lenguaje de progaramación, etc. Dado que ineresan los tiempos de ejecución de algoritmos definidos salvo una constante multiplicativa, sólo el número de operaciones elementales importará en el análisis, y no el tiempo exacto requerido por cada una de ellas [GP97].

Se consideran como operaciones elementales:

- Operaciones Aritméticas: $+, -, \times, /$, ***div***, ***mod***.
- Operaciones Booleanas: conjunción, disjunción, negación.
- Operaciones de Relación: $=, \neq, <, \leq, >, \geq$.
- Operación de Asignación.

2.1.4. Eficiencia

La eficiencia está determinada por la cantidad de recursos (tiempo y memoria, principalmente) que consume un programa durante su ejecución. A menor consumo de recursos, mayor eficiencia. Se dice entonces que un algoritmo A es más eficiente que otro algoritmo B si la cantidad de recursos utilizados por A, como por ejemplo la cantidad de espacio o tiempo de ejecución para resolver el mismo problema, es menor que la cantidad utilizada por B. Es así como el conocer la eficiencia de un algoritmo permitirá:

- Estimar si un programa resolverá un problema usando un tiempo y memoria razonables.
- Dadas distintas formas de resolver un problema, escoger la más eficiente.

- Rediseñar algoritmos utilizando otras técnicas que mejoren su rendimiento.

Durante su ejecución, un programa consume básicamente dos recursos:

- Tiempo del procesador - Complejidad temporal.
- Espacio en memoria - Complejidad espacial.

Complejidad temporal[GV99]

La complejidad temporal de un algoritmo se define como el tiempo que emplea dicho algoritmo en ejecutarse dados unos datos de entrada. El tiempo de ejecución de un algoritmo va a depender de diversos factores como son: Los datos de entrada, la calidad del código generado por el compilador para crear el programa objeto, la naturaleza y rapidez de las instrucciones máquina del procesador concreto que ejecute el programa, y la complejidad intrínseca del algoritmo. Hay dos estudios posibles sobre el tiempo:

- Uno que proporciona una medida teórica (a priori), que consiste en obtener una función que acote (por arriba o por abajo) el tiempo de ejecución del algoritmo para unos valores de entrada dados.

- Y otro que ofrece una medida real (a posteriori), consistente en medir el tiempo de ejecución del algoritmo para unos valores de entrada dados y en un ordenador concreto.

Complejidad espacial[MQ03]

La complejidad espacial representa el gasto en memoria que realiza un algoritmo. Este gasto en memoria viene dado por las variables (estáticas o dinámicas) definidas y usadas en el algoritmo, así como en el uso de llamadas a funciones, especialmente las recursivas. Se pueden distingir dos tipos de gasto de memoria por parte del algoritmo:

- El gasto estático, que viene definido por las variables globales declaradas en el programa y que permanecen desde el principio hasta el final de la ejecución. Esta gasto es constante y definido desde el principio.

- El gasto dinámico producido por la llamada a cualquier función (ya que en ese momento se reserva la memoria para las varibles locales de la función más un espacio adicional para el retorno al programa inicial), las llamadas de funciones recursivas y la reserva de memoria dinámica en el montículo.

Para este trabajo se tiene en cuenta el tiempo de ejecución del algoritmo, aunque también se podría considerar el espacio de almacenamiento.

2.1.5. Función de complejidad

Dado un algoritmo A se trata de encontrar una función $f(n)$, llamada función de complejidad del algoritmo, que mida en términos del "tamaño" de la entrada n la cantidad de recursos utilizados, posteriormente se puede determinar el orden al que pertenece $f(n)$ utilizando la notación asintótica y sus reglas, las cuales serán explicadas posteriormente.

Definición 2.1. Para un algoritmo A que resuelve un problema, se denota por t_X el número de operaciones elementales que utiliza el algoritmo A para resolver la instancia X del problema; se define la **función de complejidad** $f : \mathbb{N} \to \mathbb{R}^+$ del algoritmo A como $f(n) = \text{máx}\{t_X : |X| = n\}$.

En la presentación de los algoritmos se utilizan los siguientes esquemas, donde n representa el tamaño de la entrada, t_i el número de operaciones elementales realizadas en la i-ésima instrucción. Particularmente también se denota con t_c el tiempo necesario para evaluar la condición en las instrucciones condicionales.

Con el ánimo de facilitar el cálculo de la función de complejidad de los algoritmos presentados en este documento, se establecen algunas reglas útiles para el análisis de las estructuras de control que con mayor frecuencia se encuentran en ellos.

2.1.6. Instrucciones

1. **Instrucciones Secuenciales**

	(O.E)
Procedimiento 1	t_1
Procedimiento 2	t_2
Procedimiento 3	t_3
$\vdots$	$\vdots$
procedimiento k	t_k

Si se denota por t_j el número de operaciones elementales del procedimiento j, $j = 1, ..., k$ la función de complejidad de esta secuencia estará dada por:

$$T(n) = t_1 + t_2 + t_3 + \ldots + t_k$$

2. **Instrucciones condicionales**

Este tipo de instrucciones permiten decidir entre una o varias alternativas que dependen de una expresión lógica.

- **Si _ Entonces** (Si simple)

	(O.E)
Si (Condición) **Entonces**	t_c
Procedimiento	t
Fin _Si	

La función de complejidad está dada por:

$$\mathrm{T(n)} = t_c + t$$

- **Si _ Entonces Si _ No** (Si compuesto)

	(O.E)
Si (Condición) **Entonces**	t_c
Procedimiento 1	t_1
Si_no	
Procedimiento 2	t_2
Fin _Si	

La función de complejidad de la instrucción condicional está dada por:

$$T(n) = t_c + \text{máx}\{t_1, t_2\}$$

3. **Instrucciones repetitivas**

- **Ciclo mientras** (Caso 1)

Si el número de operaciones elementales de las instrucciones dentro del ciclo no depende de la variable de control del ciclo:

	(O.E)
$i \leftarrow 1$	1
Mientras $i \leq n$ **Hacer**	1
Procedimiento	t_1
$i \leftarrow i + 1$	2
Fin _Mientras	

La función de complejidad de este ciclo está dada por:

$$T(n) = (t_1 + 3)n + 2$$

Una forma más general es la siguiente:

	(O.E)
Mientras (Condición) **Hacer**	t_c
Procedimiento	t_1
Fin _Mientras	

La función de complejidad del ciclo ahora es:

$$T(n) = (t_c + t_1)N_r + t_c$$

donde N_r es el número de veces que se ejecuta el procedimiento.

- **Ciclo mientras** (caso 2)

Si el número de operaciones elementales de las instrucciones dentro del ciclo depende de la variable de control del ciclo

	(O.E)
$i \leftarrow 1$	1
Mientras $i \leq n$ **(Operador lógico)**(Condición)**Hacer**	t_c
Procedimiento(i)	$t(i)$
$i \leftarrow i + 1$	2
Fin _Mientras	

Para este caso, la función de complejidad del ciclo está dada por:

$$T(n) = t_c(n + 1) + \sum_{i=1}^{n}(t(i) + 2) + 1$$

- **Para _ Hacer**

Se usa cuando se conoce el número de veces que se debe repetir un conjunto de instrucciones.

	(O.E)
Para (Variable de control = Valor inicial) **Hasta** Valor final **Hacer**	
Procedimiento	t
Fin _ Para	

La función de complejidad de la instrucción está dada por:

$$T(n)= (V_f - V_i)t + 1.$$

Donde V_f y V_i representan valor final y valor inicial respectivamente.

2.1.7. Notación asintótica

Estudia el comportamiento de un algoritmo cuando el tamaño de las entradas n, es lo suficientemente grande, sin tener en cuenta lo que ocurre para entradas pequeñas y obviando factores constantes.

El objetivo es conocer sobre qué curva de coste se encuentra dicho algoritmo. También es necesario comparar estas curvas con el fin de poder decidir si un determinado algoritmo es mejor, peor o equivalente a otro. Es por esto que lo que interesa es medir el índice de crecimiento de las funciones de coste y expresarlo en notación asintótica. Tres razones apoyan esta decisión:

- Para valores de n suficientemente grandes el valor de la función está completamente determinado por el término dominante.
- El valor exacto del coeficiente del término dominante no se conserva al cambiar de entorno de programación.
- El uso de la notación asintótica nos permite establecer un orden relativo entre funciones comparando términos dominantes [1].

Notación "O grande" (O)

Dada una función $g(n) : \mathbb{N} \longrightarrow \mathbb{R}^+ \cup \{0\}$, se define el orden O grande de g, notado $O(g(n))$, como el conjunto de las funciones $f : \mathbb{N} \to \mathbb{R}^+$ para constantes positivas $c \in \mathbb{R}$ y $n_0 \in \mathbb{N}$ que cumplen $0 \leq f(n) \leq cg(n), \quad \text{para todo} \quad n \geq n_0$.

Este conjunto contiene todas las funciones no negativas que pueden ser acotadas por un múltiplo real positivo de $g(n)$; ahora bien, si $h(n)$ es una función del conjunto $O(g(n))$ se escribe $h(n) = O(g(n))$ para indicar que $h(n) \in O(g(n))$.

Si $h(n) \in O(g(n))$ se lee: "$h(n)$ es del orden O grande de g de n."

[1] La eficiencia de los programas Jordi Linares Pellicer EPSA-DSIC

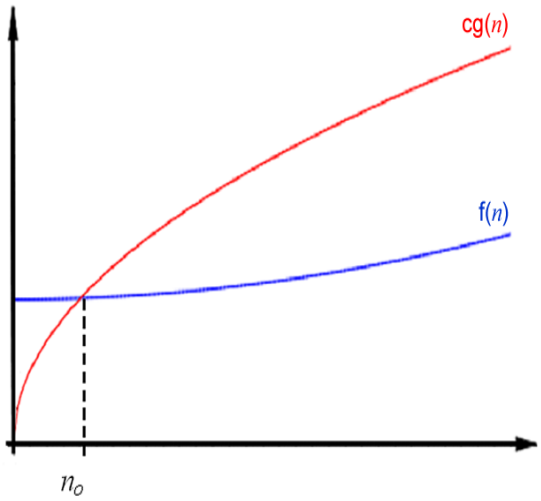

Figura 2.1: Notación "O grande".

Notación "Omega" (Ω)

Dada una función $g(n) : \mathbb{N} \longrightarrow \mathbb{R}^+ \cup \{0\}$, se define el orden omega de g(n), notado $\Omega(g(n))$ como el conjunto de las funciones $f : \mathbb{N} \to \mathbb{R}^+$ para constantes positivas $c \in \mathbb{R}$ y $n_0 \in \mathbb{N}$ que cumplen $cg(n) \leq f(n)$, para todo $n \geq n_0$.

Si $h(n) \in \Omega(g(n))$ se lee: "$h(n)$ es del orden omega de g de n."

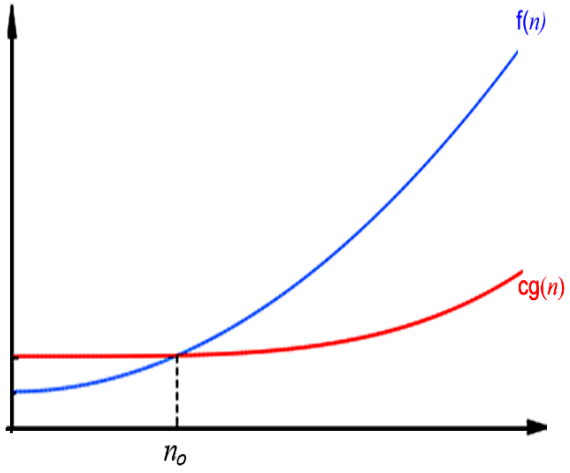

Figura 2.2: Notación "Omega".

Notación "Theta"(Θ)

Dada una función $g(n) : \mathbb{N} \longrightarrow \mathbb{R}^+ \cup \{0\}$, se define el orden theta de g(n), notado $\Theta(g(n))$ como el conjunto de las funciones $f : \mathbb{N} \to \mathbb{R}^+$ para constantes positivas $c \in \mathbb{R}$ y $n_0 \in \mathbb{N}$ que cumplen $c_1 g(n) \leq f(n) \leq c_2 g(n)$, para todo $n \geq n_0$.

Si $h(n) \in \Theta(g(n))$ se lee: "$h(n)$ es del orden theta de g de n."

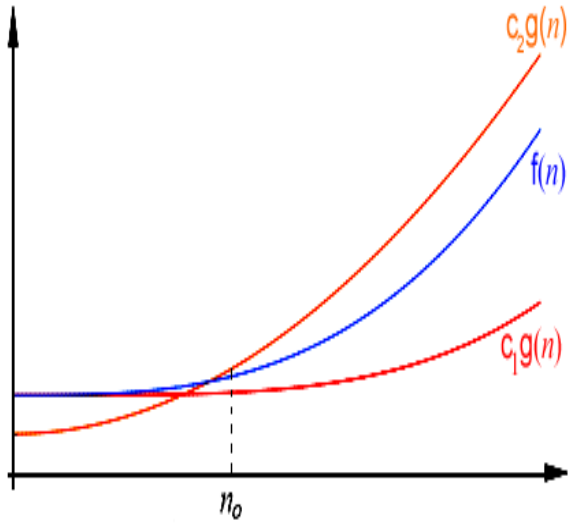

Figura 2.3: Notación "Theta".

2.2. Algoritmos previos

Antes de estudiar y presentar los algoritmos principales de este trabajo se describen algunos otros necesarios para presentarlos, como son los algoritmos indicadores de formas de las reglas, algoritmos de distribución de probabilidades y algoritmos de inicialización de arreglos en diferentes dimensiones, además de sus respectivos órdenes de complejidad [2].

[2]Para mayor detalle de los algoritmos remitirse al Apéndice B.

2.2.1. Algoritmos para la entrada de datos

- **Procedimiento OrdGraMm**

  Descripción

  Este agloritmo permite ordenar con respecto a los antecedentes, las reglas de la gramática tomadas de un conjunto de líneas de texto.

  Entrada: Lista de reglas de la gramática.
  Salida: Lista de reglas de la gramática ordenadas con respecto a los antecedentes.

  El orden de la función de complejidad del procedimiento OrdGraMm es $O(n^2)$.

- **Función IndiceSim**

  Descripción

  Determina la ubicación o índice de un símbolo de la gramática (terminales y no terminales), en el registro TRDatos.

  Entrada: Un registro de datos, una cadena y un indicador que es 0 si el símbolo que se busca es un terminal, y 1 si el símbolo que se busca es un no terminal.
  Salida: Valor que varía desde 1 hasta el número de símbolos de la gramática y que representa al número asociado al símbolo que se busca.

  El orden de la función de complejidad del procedimiento IndiceSim es $O(n)$.

- **Función IdForm**

  Descripción

  Identifica la forma de una regla, forma 1 ($A \rightarrow BC$) o forma 2 ($A \rightarrow a$) y guarda la ubicación de los espacios E1, E2 y E3 de la regla.

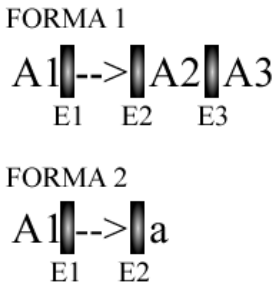

Figura 2.4: Espacios de las reglas.

  Entrada: Una regla del conjunto de reglas de producción.
  Salida: Un indicador (entero) que representa a las reglas de la forma 1 o 2.

  El orden de la función de complejidad del procedimiento IdForm es $\in O(1)$, constante.

- **IdFormPr**

  Descripción

  Identifica la forma de una regla, forma 1 ($A \rightarrow BC$) o forma 2 ($A \rightarrow a$) y guarda la ubicación de los espacios E1, E2, E3, E4 Y E5 de la regla, además de las probabilidades de cada una de ellas.

  Entrada: Una regla del conjunto de reglas de producción.
  Salida: Un indicador (entero) que representa a las reglas de la forma 1 o 2 con su respectiva probabilidad.

Figura 2.5: Espacios y probabilidades de las reglas.

El orden de la función de complejidad del procedimiento IdFormPr es $O(1)$, constante.

2.2.2. Algoritmos referentes a los árboles de derivación de una cadena

- **Procedimiento Unión**

Descripción

Une dos subárboles con la regla que lo hace posible.

Entrada: Regla que ha de ser la opción de unión de dos subárboles, posiciones de los símbolos en los arreglos de antecedentes comunes.
Salida: Nuevo árbol formado por los dos subárboles y el antecedente que lo hace posible.

El orden de la función de complejidad del procedimiento Unión es $O(n)$.

- **Función CYK2**

  Descripción

  Llena el arreglo con los posibles árboles de derivación de una cadena.

  Entrada: Cadena para determinar su pertenencia al lenguaje, la gramática y el arreglo de consecuentes.
  Salida: Conjunto de reglas que derivan en la respectiva cadena.

  El orden de la función de complejidad de la función CYK2 es $O(n^3)$.

- **Función MejorDer**

  Descripción

  Determina la mejor derivación de una cadena.

  Entrada: Árbol o árboles de derivación asociados a la cadena que es objeto de evaluación y la gramática que contiene las probabilidades de las reglas que pertenecen a los árboles.
  Salida: Árbol de derivación cuya probabilidad es el mayor valor de las probabilidades de las derivaciones.

  El orden de la función de complejidad de la función MejorDer es $O(n)$.

- **Función CadInd**

  Descripción

  Determina los índices en el registro TRDatos de los símbolos de una cadena.

  Entrada: La cadena a la que pertenece el símbolo al que se le busca su índice y el registro de datos que guarda los terminales y no terminales de la gramática.

Salida: Índice del terminal correspondiente.

El orden de la función de complejidad de la función CadInd es $O(n^2)$.

- **Procedimiento Monom**

Descripción

Asigna los exponentes del monomio que se genera a partir de cada árbol de derivación de una cadena; cada exponente representa el número de veces que se repite la regla correspondiente en el árbol de derivación.

Entrada: Arreglo para guardar todos los árboles de derivación y los monomios que se generan cuando se introduce un corpus, la gramática y el número de reglas de la gramática.
Salida: Monomios con sus respectivos exponentes.

El orden de la función de complejidad de la función Monom es $O(n)$.

2.2.3. Algoritmos para asignar y determinar las probabilidades de las reglas de una gramática y de las cadenas generadas

- **Procedimiento Dis_Unif**

Descripción

Asigna probabilidades a las reglas de la gramática usando una distribución uniforme.

Entrada: Reglas organizadas por antecedente común.
Salida: Reglas con su respectiva distribución de probabilidad de manera uniforme.

El orden de la función de complejidad del procedimiento Dis_Unif es $O(n^2)$.

- **Procedimiento Dis_Ale**

  Descripción

  Asigna probabilidades a las reglas de la gramática de manera aleatoria.

  Entrada: Reglas organizadas por antecedente común.
  Salida: Reglas con su respectiva distribución de probabilidad de manera aleatoria.

  El orden de la función de complejidad del procedimiento Dis_Ale es $O(n)$.

- **Procedimiento Dis_Frec**

  Descripción

  Asigna probabilidades a las reglas de una gramática. Dada una muestra del corpus generado por la gramática, se asigna la probabilidad según el número de veces que aparecen las reglas en los árboles de derivación.

  Entrada: El corpus o cadenas de la muestra, la gramática y el arreglo de consecuentes comunes para reglas de la forma 1 y 2.
  Salida: Reglas con su respectiva distribución de probabilidad dependiente de la frecuencia con que aparecen en los árboles de derivación.

  El orden de la función de complejidad del procedimiento Dis_Ale es $O(n^3)$.

- **Función ProCad**

  Descripción

  Determina la probabilidad de generar una cadena del corpus con una gramática determinada.

  Entrada: La gramática con las reglas y su respectiva probabilidad, la cadena y el arreglo de registros que guarda la información de los árboles de derivación de la cadena con el polinomio que se genera.
  Salida: La probabilidad de la cadena.

  El orden de la función de complejidad de la función ProCad es $O(n)$.

2.2.4. Algoritmos referentes a las matrices (arreglos)

- **Dim_Matriz**

  Descripción

  Determina el tamaño de una matriz tridimensional, sus dimensiones dependen del número de elementos de la cadena y del número de antecedentes de la gramática.

  Entrada: Número de elementos de la cadena y número de antecedentes.
  Salida: Matriz tridimensional.

  El orden de la función de complejidad del procedimiento Dim_Matriz es $O(n)$.

- **Ini_Matriz**

  Descripción

  Inicializa la matriz tridimensional con las reglas de la gramática, haciendo uso del procedimiento anterior para asignar las dimensiones.

  Entrada: La matriz a inicializar, la cadena y la gramática que contiene los antecedentes, las reglas y probabilidades.
  Salida: La matriz con las probabilidades de las reglas.

  El orden de la función de complejidad del procedimiento Ini_Matriz es $O(n^2)$.

- **Buscar_An**

  Descripción

  Busca un antecedente en el arreglo de antecedentes comunes.

  Entrada: La gramática y el antecedente común que se está buscando.
  Salida: El índice del antecedente buscado.

  El orden de la función de complejidad de la funcón Buscar_An es $O(n)$.

Estos algoritmos han sido de vital importancia para el desarrollo de los algoritmos Inside, Outside e IO cuyo análisis es el objetivo de este proyecto ya que hacen búsquedas de antecedentes, consecuentes según las distintas formas de las reglas y, por supuesto, hacen uso de sus probabilidades. Para experimentos posteriores las probabilidades se asignan de manera uniforme, aleatoria y por frecuencia.

Capítulo 3

Algoritmos de Estimación

En este capítulo se presentan los algoritmos que resuelven en un tiempo polinómico y por Programación Dinámica el problema de determinar si $Pr(x|G_p) > 0$; encontrando al menos una derivación de la cadena a partir del símbolo inicial de la gramática cuya probabilidad sea mayor que cero. Para ello se considera que la GIP G_p está en FNC, lo cual no supone ninguna pérdida de generalidad.

3.1. Algoritmo Inside

3.1.1. Descripción

El algoritmo Inside [Bak79, LY90, Ney92, Cas96] permite calcular $Pr(x|G_p)$ deduciendo la probabilidad de la cadena a partir de todas las posibles derivaciones. Calcula primero la probabilidad para símbolos sencillos y después expande, desde adentro, para incluir subárboles de análisis, como lo muestra la figura 3.1. El algoritmo Inside es un algoritmo basado en un esquema de programación dinámica, análogo al algoritmo de Cocke-Younger-Kasami. Se basa en la definición de $e(A\,\langle i,j\rangle) = Pr(A \overset{*}{\Rightarrow} x_i...x_j|G_p)$, como la probabilidad de que la subcadena $x_i...x_j$ sea generada a partir de A en un determinado número de pasos. La evaluación de esta probabilidad para todo no terminal A esta dada por:

$$\begin{cases} e(A\,\langle i,i\rangle) &= p(A \to x_i) \\ e(A\,\langle i,j\rangle) &= \sum_{B,C \in N} \left(p(A \to BC) \sum_{k=i}^{j-1} e(B\,\langle i,k\rangle) e(C\,\langle k+1,j\rangle) \right) \end{cases} \tag{3.1.1}$$

De esta manera:

$$Pr(x|G_p) = e(S\,\langle 1, |x|\rangle), \qquad 1 \leq i < j \leq |x|$$

El coste temporal del algoritmo es $O(|x|^3|N|^3)$ y el coste espacial es $O(|x|^2|N|)$.

En efecto [LY90]:

Las entradas del algoritmo son el número de símbolos no terminales N, el número de símbolos terminales Σ y la cadena x tal que $|x| = n$. El objetivo es determinar la complejidad del algoritmo en términos de los parámetros de N y $|x|$.

La probabilidad inside definida en la ecuación 3.1.1 es computada en dos pasos. Primero, son consideradas todas las cadenas de longitud dos. Esto elimina la suma interior de 3.1.1 y simplifica la ecuación como sigue:

$$e(A\,\langle i, j\rangle) \;=\; \sum_{A\in N}\sum_{i=1}^{n-1}\sum_{B,C\in N} p(A \to BC)p(B \to x_1)e(C \to x_2) \tag{3.1.2}$$

El producto de los tres términos probabilísticos es $O(1)$. La suma interior con índices B,C que se extiende sobre todos los no terminales N se ejecuta N^2 veces, la sumatoria media que se extiende sobre la longitud de la cadena n se ejecuta $n-1$ veces y la sumatoria externa repite todo el proceso para cada N, empleando un tiempo de $O(N^3)$ en no terminales y $O(n)$ en la longitud de la secuencia observada.

El análisis para el resto de las subcadenas (longitud mayor que dos) es más complejo. Las probabilidades inside son ahora computadas para j = i+l así:

$$e(A\,\langle i, j\rangle) \;=\; \sum_{l=2}^{n-1}\sum_{A\in N}\sum_{i=l}^{n-1}\sum_{B,C\in N}\sum_{k=i+1}^{j-2} p(A \to BC)e(B\,\langle i, k\rangle)e(C\,\langle k+1, j\rangle), \tag{3.1.3}$$

La sumatoria interna es ejecutada $j - i - 2$ veces o simplemente $l - 2$ veces. Contando el número de veces de cada iteración (o sumatoria) el recorrido ejecutado podría ser expresado de la siguiente manera:

$$
\begin{aligned}
&= \sum_{l=2}^{n-1} N^3 \sum_{i=l}^{n-1} \sum_{k=i+1}^{j-2} = \sum_{l=2}^{n-1} N^3 (n-l)(l-2) = N^3 \sum_{l=2}^{n-1} \left\{ -l^2 + nl + 2l - 2n \right\} \\
&= N^3 \left\{ -\textstyle\sum_{l=2}^{n-1} l^2 + (n+2) \sum_{l=2}^{n-2} l - 2n(n-3) \right\} \\
&= N^3 \left\{ -\tfrac{1}{6} \left[n(n-1)(n-\tfrac{1}{2}) \right] + 1 + \tfrac{1}{2} \left[n(n-1)(n+2) \right] - (n+2) - 2n(n-3) \right\} \\
&= N^3 \left\{ \tfrac{1}{3} n^3 - \tfrac{27}{12} n^2 + \tfrac{85}{12} n - 1 \right\}
\end{aligned}
\tag{3.1.4}
$$

Es así como el coste temporal de la función Inside $\in O(|x|^3 |N|^3)$, es decir, el cómputo de probabilidades Inside es cúbico tanto en términos del número de no terminales como en términos de la longitud de la cadena observada.

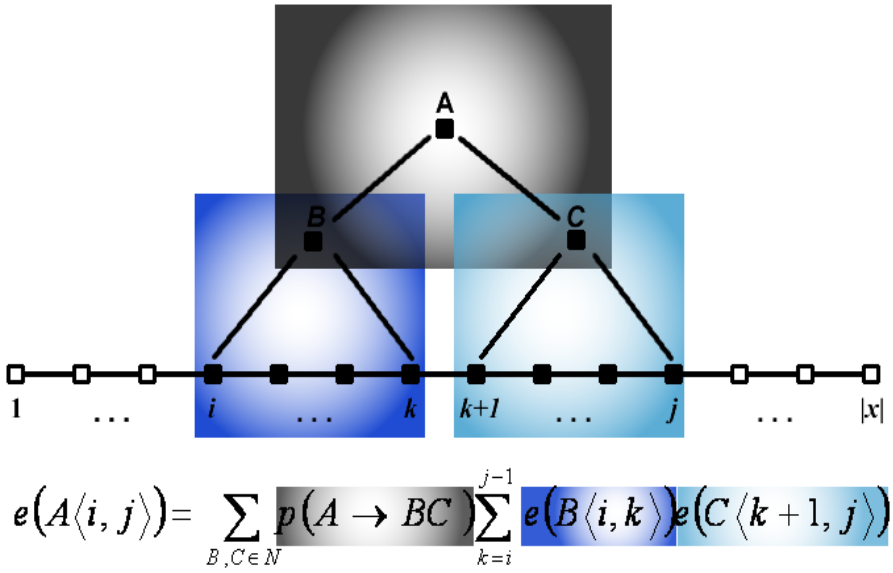

Figura 3.1: Función Inside

Ejemplo 3.1. *Dada la GI* $G = (\{T1, T2, T3\}, \{a, b, c\}, T1, \{T1 \to T2T3,\ T1 \to b,\ T2 \to T1T2,\ T2 \to c,\ T3 \to T2T1,\ T3 \to T1T2,\ T1 \to a\})$ *y la cadena de cinco terminales:* $x = bccac$*, se tiene como árbol de derivación asociado a esta cadena:*

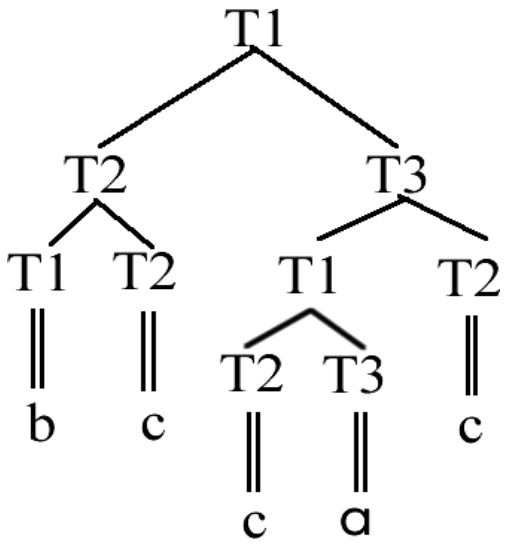

Figura 3.2: Ejemplo Algoritmo Inside.

Partiendo de la definición se tiene:

$$\begin{cases} e(T1\,\langle 1,1\rangle) &= p(T1 \to b) \\ e(T1\,\langle 1,5\rangle) &= \displaystyle\sum_{T2,T3\in N}\left(p(T1 \to T2T3)\sum_{k=1}^{4} e(T2\,\langle 1,k\rangle)e(T3\,\langle k+1,5\rangle)\right) \end{cases}$$

$$1 \leq i < j \leq 5$$

$$\begin{cases} e(T1\,\langle 1,1\rangle) &= p(T1 \to b) \\ e(T1\,\langle 1,5\rangle) &= \displaystyle\sum_{T2,T3\in N}(p(T1 \to T2T3) \\ & \qquad e(T2\,\langle 1,1\rangle)e(T3\,\langle 2,5\rangle) + e(T2\,\langle 1,2\rangle)e(T3\,\langle 3,5\rangle) + \\ & \qquad e(T2\,\langle 1,3\rangle)e(T3\,\langle 4,5\rangle) + e(T2\,\langle 1,4\rangle)e(T3\,\langle 5,5\rangle)) \end{cases}$$

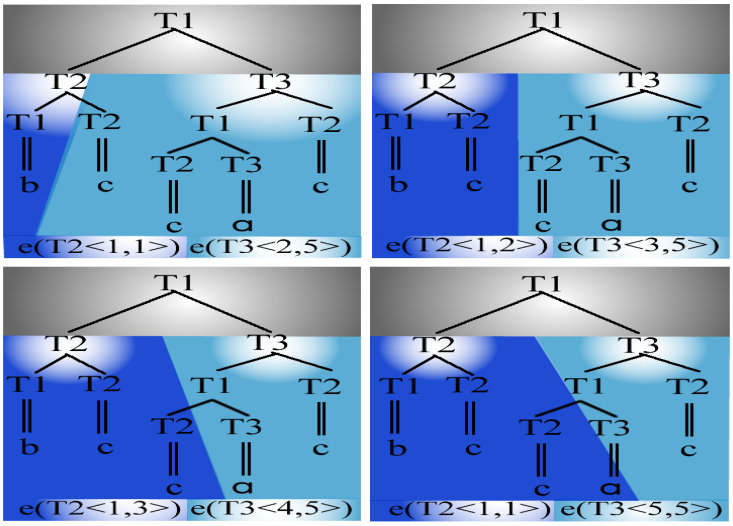

Figura 3.3: Ejemplo Algoritmo Inside.

3.1.2. Algoritmo de la función Inside

A continuación se presenta una tabla con las variables usadas por el algoritmo y el pseudocódigo del mismo; las funciones que se utlizan están documentadas en el apéndice B.

Tabla de variables		
Nombre	**Tipo**	**Uso**
G	GRA	Símbolos y reglas
Cad	TRCadena	Cadena en la que se está evaluando la función
AN	TIndSim	Antecedente de la regla
AuxM	Matriz	Auxiliar del arreglo tridimensional donde se guardan las probabilidades
AuxAN	TIndSim	Auxiliar para el antecedente común en la AuxM
NRAC	TIndReg	Contador para las reglas con antecedente común
AuxPi	TProb	Auxiliares para las probabilidades de las reglas
AuxCad	TRCadena	Auxiliar para la cadena
C_1	TIndSim	Consecuente en la primera posición de la regla
C_2	TindSim	Consecuente en la segunda posición de la regla

3.1.3. Pseudocódigo de la función Inside

		OE
	Inicio	**OE**
1.	AuxAn ← Buscar_An(G,An);	N
2.	**Si** AuxAN = 0 **Entonces**	1
3.	Inside ←0;	1
4.	**Si_No**	
5.	**Inicio1**	
6.	n = (j-i) + 1;	3
7.	**Si** n = NSC **Entonces**	1
8.	AuxCad ← Cad;	1
9.	**Si_No**	
10.	**Inicio2**	
11.	NSC ← n;	1
12.	**Para** k ← 0 **Hasta** n−1 **Hacer**	2
13.	AuxCad (k)← AuxCad (i+k);	2
14.	**Fin2**	
15.	**Si** AuxM[j,i,AuxAn-1] <> 2 **Entonces**	1
16.	Inside ← AuxM[j,i,AuxAn-1];	1
17.	**Sino**	
18.	**Inicio3**	
19.	AuxP3 ← 0;	1
20.	**Para** NR ← 1 **Hasta** NRaAC1 **Hacer**	1
21.	**Inicio4**	
22.	AuxP ←Probabilidad de la regla cuyo antecedente es AuxAN;	1
23.	C_1 ←Consecuente en la primera posición;	1
24.	C_2 ←Consecuente en la segunda posición;	1
25.	AuxP2 ←0;	1
26.	**Para** s ← i **Hasta** j−1 **Hacer**	2
27.	**Inicio5**	
28.	**Si** s ≤ (i+j) ÷ 2 **Entonces**	
29.	**Inicio6**	
30.	AuxP1 ← Inside(G,Cad,C1,i,s,AuxM)	x^2
31.	**Si** AuxP1 <> 0 **Entonces**	
32.	**Inicio7**	
33.	AuxP4 ← Inside(G,Cad,C2,s+1,j,AuxM)	x^2
34.	AuxP2 ← AuxP2 + (AuxP1 × AuxP4);	2
35.	**Fin7**	
36.	**Fin6**	
37.	**Sino**	
38.	**Inicio8**	
39.	AuxP1 ← Inside(G,Cad,C2,s +1,j,AuxM)	x^2

```
40.                 Si AuxP1 <> 0 Entonces
41.                 Inicio9
42.                   AuxP4 ← Inside(G,Cad,C1,i,s,AuxM)                    x^2
43.               AuxP2 ← AuxP2 + (AuxP1 × AuxP4);                          2
44.                 Fin9
45.               Fin8
46.             Fin5
47.           AuxP3 ← (AuxP × AuxP2) + AuxP3;                             3
48.         Fin4
49.       AuxM[j,i,AuxAN-1]← AuxP3;                                       1
50.       Inside ← AuxP3;                                                 1
51.     Fin3
52.   Fin1
```

Fin

Nota:

- $Buscar\_AN(G, AN)$: Es una función que busca un antecedente en la gramática.

3.1.4. Análisis del Algoritmo Inside

Líneas	Número de operaciones
28-45	$2x^2 + 7$
26-46	$2x^3 + 7x + 2$
20-48	$n_r(2x^3 + 7x + 2) + 19$
15-48	$n_r(2x^3 + 7x + 2) + 22$
6-48	$n_r(2x^3 + 7x + 2) + 35$
1-50	$n_r(2x^3 + 7x + 2) + N + 38$
$f(n)=$	$n_r(2x^3 + 7x + 2) + N + 38$

Cuadro 3.1: Operaciones realizadas por la función Inside

El cuadro 3.1 muestra que la función de complejidad de esta función es $f(n) \in O(|n_r||x^3|)$, donde n_r representa el número de reglas y x la longitud de la cadena observada. En el peor de los casos, el número de reglas, se opera de manera análoga a la cantidad de no terminales por regla, es decir, para A $\rightarrow$ BC sobre todos los no terminales N se ejecuta N^2 veces dados los consecuentes B y C, repitiendo además la misma operación para todos los $A \in N$; es así como se puede notar que este proceso tiene un orden de $O(N^3)$

y por tanto $f(n) \in O(|N^3||x^3|)$.

3.2. Algoritmo Outside

3.2.1. Descripción

El algoritmo Outside permite calcular $\Pr(x|G_P)$ deduciendo la probabilidad de la cadena a partir de todas las posibles derivaciones. Comienza desde afuera con la cadena completa x y excluye subárboles de análisis como lo muestra la Figura 3.2. La evaluación de esta probabilidad puede ser calculada para todo $A \in N$ de la siguiente manera:

Sea

$$f(A\,\langle i,j\rangle) \quad = \quad \Pr\left(S \overset{*}{\Rightarrow} x_1,\ldots,x_{i-1}Ax_{j+1},\ldots,x_{|x|}|G_p\right)$$

$$f(A\,\langle 1,|x|\rangle) \quad = \quad \begin{cases} 1, & \text{si} \quad A = S \\ 0, & \text{si} \quad A \neq S \end{cases}$$

Y para 1≤ i≤ j ≤ $|x|$

$$f(A\,\langle i,j\rangle) \quad = \quad \sum_{B,C\in N}\left[p(B\to CA)\sum_{k=1}^{i-1} f(B\,\langle k,j\rangle)e(C\,\langle k,i-1\rangle) + p(B\to AC)\sum_{k=j+1}^{|x|} f(B\,\langle i,k\rangle)e(C\,\langle j+1,k\rangle)\right]$$

Es así como:

Para cualquier i, $1 \leq i \leq |x|$

$$\Pr(x|G_p) = \sum_{A\in N} f(A\,\langle i,i\rangle)p(A\to x_i),$$

El coste temporal del algoritmo es $O(|x|^3|N^3|)$ y el coste espacial es $O(|x|^2|N|)$.

El análisis de la complejidad del algoritmo Outside puede ser calculado de manera idéntica al análisis realizado para el orden de complejidad del algoritmo Inside (llegando a la misma conclusión).

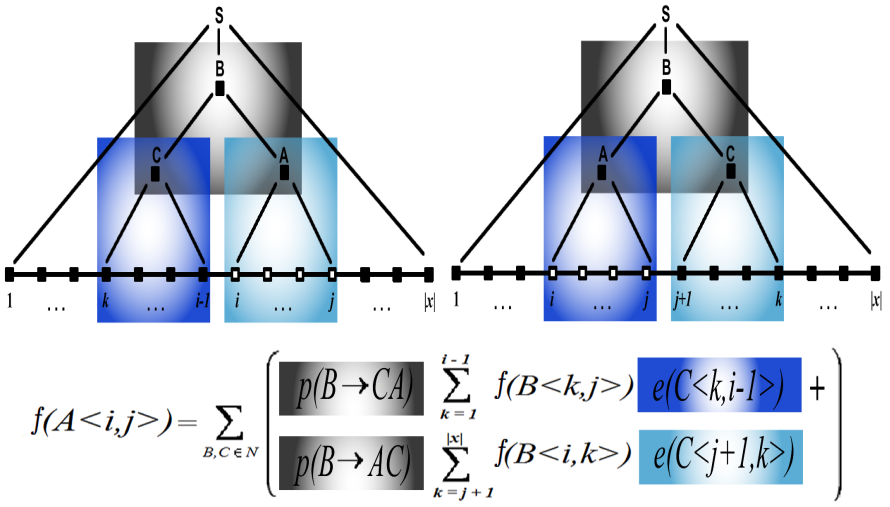

Figura 3.4: Función Outside

Ejemplo 3.2. *Dada la GI* $G = (\{S,A,B,C\}, \{a,b,c\}, S, \{S \rightarrow BB, B \rightarrow CA, B \rightarrow AC, B \overset{*}{\Longrightarrow} aabc, A \overset{*}{\Longrightarrow} abacc, C \overset{*}{\Longrightarrow} babc\})$ *y la cadena de trece terminales:* $x =$ *aabcabaccbabc, el árbol de derivación asociado a* x *es:*

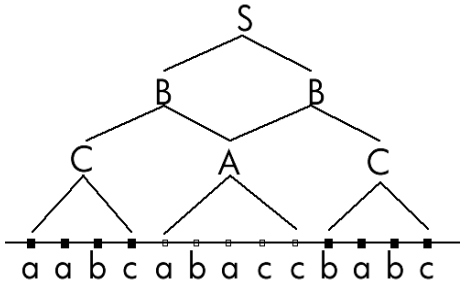

Figura 3.5: Ejemplo Algoritmo Outside.

Ahora, partiendo de la definición:

$$f(A\,\langle 5,9\rangle) = \sum_{B,C\in N}\Bigg[p(B\to CA)\sum_{k=1}^{5-1} f(B\,\langle k,9\rangle)e(C\,\langle k,5-1\rangle)+ p(B\to AC)\sum_{k=9+1}^{13} f(B\,\langle 5,k\rangle)e(C\,\langle 9+1,k\rangle)\Bigg]$$

$$\begin{aligned} f(A\,\langle 5,9\rangle) = \sum_{B,C\in N} [&p(B\to CA)(f(B\,\langle 2,9\rangle)e(C\,\langle 2,5-1\rangle)+ \\ &f(B\,\langle 3,9\rangle)e(C\,\langle 3,5-1\rangle)+ \\ &f(B\,\langle 4,9\rangle)e(C\,\langle 4,5-1\rangle)+ \\ &f(B\,\langle 4,9\rangle)e(C\,\langle 4,5-1\rangle)) \\ &p(B\to AC)(f(B\,\langle 5,9+1\rangle)e(C\,\langle 9+1,9+1\rangle)+ \\ &f(B\,\langle 5,9+2\rangle)e(C\,\langle 9+1,9+2\rangle)+ \\ &f(B\,\langle 5,9+3\rangle)e(C\,\langle 9+1,9+3\rangle)+ \\ &f(B\,\langle 5,9+4\rangle)e(C\,\langle 9+1,9+4\rangle))] \end{aligned}$$

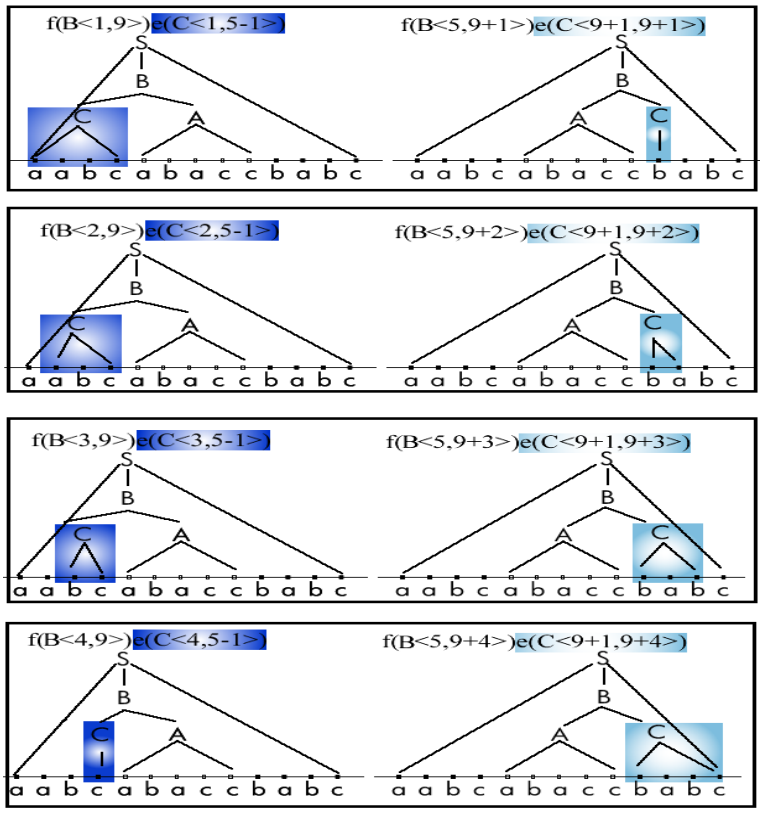

Figura 3.6: Ejemplo Algoritmo Outside.

3.2.2. Algoritmo de la función Outside

A continuación se presenta una tabla con las variables usadas por el algoritmo y el pseudocódigo del mismo; las funciones que se utlizan están documentadas en el apéndice B.

Tabla de variables		
Nombre	**Tipo**	**Uso**
G	GRA	Gramática para conocer sus reglas y probabilidades
ArrCC	RTARRaCC	Acceso a la información del consecuente común
Cad	TRCadena	Cadena en la que se está evaluando la función
CC	TIndSim	Consecuente común
AuxMA	Matriz	Auxiliar para la matriz de antecedentes
i	Byte	Contador para la longitud de la cadena
j	Byte	Contador para la longitud de la cadena
NumSC	Byte	Número de símbolos de la cadena
AuxCC	TIndSim	Auxiliar para el consecuente común
k	TIndSim	Auxiliar que varía en la longitud de la cadena
AuxNRaCC	TIndReg	Auxiliara para las reglas con consecuente común
AuxPi	Tprob	Auxiliares para las probabilidades de las reglas

3.2.3. Pseudocódigo de la función Outside

```
Inicio                                                          OE
 1. NumSC ← (j − i) + 1;                                         3
 2.   Si CC ← NT + 1 Entonces                                    2
 3.     Outside ← 1;                                             1
 4.   Sino
 5.     Outside ← 0;                                             1
 6. Sino
 7.   Inicio1
 8.     AuxCC ← CC − NT;                                         2
 9.       Si AuxMA[j,i,CC−1] <> 2 Entonces                       2
10.         Outside ← AuxMA[j,i,CC−1];                           2
11.       Sino
12.       Inicio2
13.         Si NRaCC = 0 Entonces                                1
14.         Inicio3
15.           AuxMA[j,i,CC−1] ← 0;                               2
16.           Outside ← 0;                                       1
17.         Fin3
18.         Sino
19.         Inicio4
20.           AuxNRaCC ← 1;                                      1
21.           AuxP ← 0;                                          1
```

```
22.          Repetir
23.            AuxP3 ← 0                                                    1
24.            AuxP5← 0                                                     1
25.            Si El CC está en la primera posición Entonces                1
26.            Inicio5
27.              Si j ≤ NSC−2 Entonces                                      2
28.              Inicio6
29.                Para k ← j+1 Hasta NSC − 1 Hacer                         3
30.                Inicio7
31.                  AuxP2 ←Outside(G,ArrCC,Cad,P,AuxMC,AuxMA,i,k);         x^2
32.                  SI (k − i) + 1 <> Cad.NSC Entonces                     4
33.                  Inicio8
34.                  AuxP2←Inside(G, Cad,S, j + 1, k, AuxMC);               x^2
35.                  Si AuxP2 <> 0 Entonces                                 1
36.                  AuxP3 → AuxP3 + (AuxP2×Inside(G,Cad,S,j+1,k,AuxMC));   x^2
37.                  Fin8
38.                  Sino
39.                   Inicio9
40.                     AuxP2 → Outside(G,ArrCC,Cad,P,AuxMC,AuxMA,k,j);     x^2
41.                     Si AuxP2 <> 0 Entonces                              1
42.                       AuxP3→AuxP3 + (AuxP2 × Inside(G,Cad,S,k,i-1,AuxMC));x^2
43.                   Fin9
44.                Fin7
45.              AuxP4← (Probabilidad de la regla) × AuxP3;                 2
46.              AuxP ← AuxP4 + AuxP;                                       2
47.              Fin6
48.              Inc(AuxNRaCC);                                             1
49.            Fin5
50.            Sino
51.            Inicio10
52.              Si i − 1 ≥ 0 Entonces                                      2
53.              Inicio11
54.                Para k ← 0 Hasta i − 1 Hacer                             2
55.                Inicio12
56.                Si (j − k) + 1 <> Cad.NSC Entonces                       4
57.                 Inicio13
58.                   AuxP2 ← Inside(G,Cad,S,k,i-1,AuxMC);                  x^2
59.                   Si AuxP2 <> 0 Entonces                                1
60.                     AuxP3←AuxP3 + (AuxP2 ×
                              Outside(G,ArrCC,Cad,P,AuxMC,AuxMA,k,j));       x^2
61.                 Fin13
62.                Sino
63.                  Inicio14
64.                    AuxP2 ←Outside(G,ArrCC,Cad,P,AuxMC,AuxMA,k,j);      x^2
65.                   SiAuxP2<>0Entonces                                     1
66.                     AuxP3 → AuxP3 +
                              (AuxP2 × Inside(G,Cad,S,k,i−1,AuxMC));         x^2
67.                  Fin14
68.                Fin12
69.                AuxP4→ Probabilidad de la regla × AuxP3;                 2
70.                AuxP ← AuxP4 + AuxP;                                     2
71.              Fin11
72.              Inc(AuxNRaCC);                                             1
73.            Fin10
74.            Hasta que AuxNRaCC > ArrCC.TARRaCC[AuxCC].NRaCC;             3
75.            AuxMA[j,i,AuxCC − 1]→AuxP;                                   2
76.            Outside → AuxP;                                              1
77.          Fin4;
78.        Fin2;
79.      Fin1;
Fin
```

Líneas	Número de operaciones
32-37	$2x^2 + 5$
30-44	$3x^2 + 5$
29-47	$3x^3 + 5x + 3$
26-49	$3x^3 + 5x + 5$
57-61	$2x^2 + 1$
53-72	$2x^3 + 5x$
51-74	$2x^3 + 5x + 3$
25-74	$3x^3 + 5x + 6$
23-74	$3x^3 + 5x + 8$
22-75	$n_r(3x^3 + 5x + 11) + 3$
19-78	$n_r(3x^3 + 5x + 11) + 6$
9-79	$n_r(3x^3 + 5x + 11) + 9$
1-80	$n_r(3x^3 + 5x + 11) + 13$
$f(n)=$	$n_r(3x^3 + 5x + 11) + 13$

Cuadro 3.2: Cantidad de operaciones que realiza la función Outside

3.2.4. Análisis del Algoritmo Outside

El cuadro 3.2 muestra que la función de complejidad de esta función es $f(n) \in O(|n_r||x^3|)$, donde n_r representa el número de reglas y x la longitud de la cadena observada. En el peor de los casos, el número de reglas, se opera de manera análoga a la cantidad de no terminales por regla, es decir, para A $\rightarrow$ BC sobre todos los no terminales N se ejecuta N^2 veces dados los consecuentes B y C, repitiendo además la misma operación para todos los $A \in N$, es así como se puede notar que este proceso tiene un orden de $O(N^3)$, y por tanto $f(n) \in O(|N^3||x^3|)$.

Capítulo 4

Algoritmo Inside-Outside (IO)

4.1. Descripción

El algoritmo IO está basado en la optimización de la función de verosimilitud de la muestra formada por cadenas pertenecientes al lenguaje. Como la función de verosimilitud de la muestra está representada por un polinomio definido en los términos del Teorema de Transformaciones Crecientes (TTC), enunciado en el capítulo 1, se aprovechan las características del polinomio obtenido mediante la gramática para desarrollar el algoritmo IO.

4.2. Algoritmo del procedimiento IO

Sea $G_p = (G, p)$ una GIP y sea Ω una muestra de $L(G)$, es decir, un conjunto de cadenas de $L(G)$ en el cual puede haber cadenas repetidas.

La función de verosimilitud de la muestra Ω dada la GIP se define como:

$$\Pr(\Omega|G_p)=\prod_{x\in\Omega}\Pr(x|G_p) \tag{4.1}$$

ya que esta función es un polinomio que cumple las condiciones del TTC y considerando una regla de la forma $A \to BC$ con $A, B, C \in N$. Sea d_x una derivación de la cadena x, y t_x el árbol de derivación correspondiente, de manera que la regla $A \to BC$ aparece en t_x delimitada por las posiciones i, j, k de la cadena y sumando la probabilidad de todas las derivaciones de la regla mencionada, se tiene:

$$\begin{aligned}\Pr(S \overset{*}{\Rightarrow} x_1 \dots x_{|x|}|G_p) = &\quad \Pr(S \overset{*}{\Rightarrow} x_1 \dots x_{i-1} A x_{j+1} \dots x_{|x|}|G_p) p(A \to BC) \\ &\quad \Pr(B \overset{*}{\Rightarrow} x_i \dots x_k|G_p) \Pr(C \overset{*}{\Rightarrow} x_{k+1} \dots x_j|G_p) \\ \Pr(S \overset{*}{\Rightarrow} x_1 \dots x_{|x|}|G_p) = &\quad f(A\,\langle i,j\rangle) p(A \to BC) e(B\,\langle i,k\rangle) e(C\,\langle k+1,j\rangle)\end{aligned}$$

Al considerar todas las posibles derivaciones y haciendo uso del TTC, se tiene:

$\forall (A \to BC) \in P$:

$$\bar{p}(A \to BC) = \frac{\sum_{x\in\Omega} \frac{p(A\to BC)}{\Pr(x|G_p)} \sum_{1\le i\le k<j\le|x|} f(A\,\langle i,j\rangle) e(B\,\langle i,k\rangle) e(C\,\langle k+1,j\rangle)}{\sum_{x\in\Omega} \frac{1}{\Pr(x|G_p)} \sum_{i=1}^{|x|} \sum_{j=i}^{|x|} f(A\,\langle i,j\rangle) e(A\,\langle i,j\rangle)}$$

$\forall (A \to \alpha) \in P$:

$$\bar{p}(A \to \alpha) = \frac{\sum_{x\in\Omega} \frac{1}{\Pr(x|G_p)} \sum_{i=1}^{|x|} f(A\,\langle i,i\rangle) p(A \to x_i)}{\sum_{x\in\Omega} \frac{1}{\Pr(x|G_p)} \sum_{i=1}^{|x|} \sum_{j=i}^{|x|} f(A\,\langle i,j\rangle) e(A\,\langle i,j\rangle)}$$

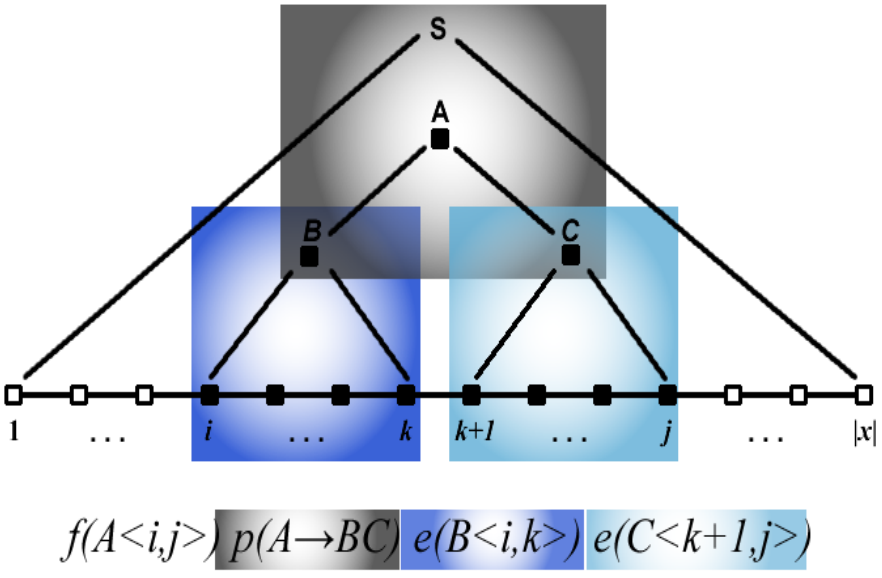

Figura 4.1: Inside-Outside

El algoritmo IO consiste en la aplicación iterativa de esta transformación, siguiendo el esquema del algoritmo basado en el Teorema de Transformaciones Crecientes, hasta maximizar localmente la función de verosimilitud de la muestra. Es importante destacar en este algoritmo cómo las probabilidades iniciales de la gramática condicionan completamente el óptimo alcanzado. La convergencia del algoritmo está garantizada por el TTC y se produce cuando las probabilidades de las reglas de la gramática no varían de una iteración a la siguiente, significativamente.

Cada iteración del algoritmo requiere aplicar el algoritmo Inside, a continuación el algoritmo Outside y después la transformación, por lo que el coste temporal asintótico en cada iteración es $O(3|\Omega|)l_m^3|P|$, con $l_m = \text{máx}_{x\in\Omega}|x|$, es decir, $O(|\Omega|)l_m^3|P|$. En el peor de los casos $|P| \in O(|N|^3)$. El coste espacial del algoritmo es $O(l_m^2|N|)$.

Tabla de variables		
Nombre	**Tipo**	**Uso**
G	GRA	Símbolos y reglas
RC	RTARRaCC	Registro para acceder a la información del consecuentecomún
Cads	TRCads	Cadenas en las que se evalúa la función
AN	TIndSim	Antecedente de la regla
AuxNR	Word	Auxiliar para el número de reglas
AuxMC2	Matriz	Auxiliar del arreglo de consecuentes de reglas de la forma 2
AuxMA2	Matriz	Auxiliar del arreglo de antecedentes de reglas de la forma 2
MatCor	MatCorA	Arreglo con los árboles de derivación
AuxSumNR	Word	Auxiliar para alcanzar el número de reglas
AuxReg1	TProb	Probabilidad estimada para reglas de la forma 1
AuxReg2	TProb	Probabilidad estimada para reglas de la forma 2
a	TIndSim	Número de antecedentes comunes
b	TIndSim	Reglas de la forma uno con el a-ésimo antecedente
c	TIndSim	Número de cadenas
d	TIndSim	Reglas de la forma dos con el a-ésimo antecedente
e	TIndSim	Número de cadenas
i	TIndSim	Elementos de la cadena (límite inferior)
j	TIndSim	Elementos de la cadena (límite superior)
k	TIndSim	Contador interno de la cadena (entre i y j-1)
l	TIndSim	Contador para los elementos de la cadena desde el símbolo inicial
m	TIndSim	Contador para los elementos de la cadena desde l
n	TIndSim	Contador para el número de elementos de la cadena e-ésima
o	TIndSim	Contador para los elementos de la cadena desde el símbolo inicial
p	TIndSim	Contador para los elementos de la cadena desde o

4.3. Pseudocódigo del procedimiento IO

```
Inicio                                                                          OE
 1.  AuxNR ← NReg;                                                               1
 2.  AuxSumNR ← 0;                                                               1
 3.  Mientras AuxSumNR < AuxNR Hacer                                             1
 4.   Inicio1
 5.    Para a←1 Hasta NAC Hacer                                                  1
 6.     Inicio2
 7.      Para b←1 HastaReglas de la forma 1 con el a-ésimo antec Hacer           1
 8.      Inicio3
 9.        Si No hay convergencia Entonces                                       1
10.        Inicio4
11.          Inc(Número de iteraciones);                                        1
12.          Aux4 ← 0;                                                          1
13.          Aux8 ← 0;                                                          1
14.          Para c←1 Hasta NCads Hacer                                         1
15.          Inicio5
16.            Aux2 ← 0;                                                        1
17.            Ini_Matriz(AuxMC2,G,Cads,ACads[c].RC2);                          x
18.            Ini_Matriz2(AuxMA2,G,Cads,ACads[c]);                             x
19.            Para i ← 0 Hasta Cads.ACads[c].NSC-2 Hacer                       4
20.              Para j ← i+1 Hasta Cads.ACads[c].NSC-1 Hacer                   5
21.                Para k ← i Hasta j-1 Hacer                                   1
22.                Inicio6
23.                  Aux1 ← Outside(G,RC,Cads,AC,AUXMC2,AUXMA2,i,j)           x^2
24.                  Si Aux1 <> 0 Entonces                                      1
25.                  Inicio7
26.                    Aux101 ← Inside(G,Cads,C1,i,k,AUXMC2);                   x^2
27.                    Si Aux101 <> 0 Entonces                                  1
28.                    Inicio8
29.                      Aux102 ← Inside(G,Cads,C1,k+1,j,AUXMC2)                x^2
30.                      Si Aux102 <> 0 Entonces                                1
31.                        Aux2 ← (Aux1×Aux101×Aux102)+Aux2;                    4
32.                    Fin8
33.                  Fin7
34.                Fin6
35.            Aux3 ← Aux2×P(AN → C1C2)×(1/Procad(G,MatCor,c));                 x
36.            Aux4 ← Aux3+Aux4;                                                2
37.          Fin5
38.          Para c←1 Hasta NCads Hacer                                         1
39.          Inicio9
40.            Aux6 ← 0;                                                        1
41.            Ini_Matriz(AuxMC2,G,Cads,ACads[c],RC2);                          x
42.            Ini_Matriz2(AuxMA2,G,Cads,ACads[c]);                             x
43.            Para l ← 0 Hasta Cads.ACads[c].NSC-1 Hacer                       4
44.              Para m ← l Hasta Cads.ACads[c].NSC-1 Hacer                     5
45.              Inicio10
46.                Aux1 ← Outside(G,RC,Cads,AC,AUXMC2,AUXMA2,l,m)               x^2
47.                Si Aux5 <> 0 Entonces                                        1
48.                Inicio11
49.                  Aux501 ← Inside(G,Cads,C1,l,m,AUXMC2)                      x^2
50.                  Si Aux501 <> 0 Entonces                                    1
51.                    Aux6 ← (Aux5×Aux501)+Aux6;                               3
52.                Fin11
53.              Fin10
```

```
54.              Aux7 ← Aux6×(1/Procad(G,MatCor,c−1));                                  x
55.              Aux8 ← Aux7+Aux8;                                                      2
56.            Fin9
57.            AuxReg1 ← Aux4/Aux8;                                                   2
58.            AuxES1 ← P(AN → C1C2)−AuxReg1;                                         2
59.            Si ((AuxES1 > (-0.0000000001)) Y (AuxES1 < (0.0000000001))Entonces     3
60.            Inicio12
61.              Inc(AuxSumNR);                                                       1
62.              Marcar la regla como estimada;                                       1
63.              P(AN → C1C2) ← AuxReg1;                                              1
64.            Fin12
65.            Sino
66.              P(AN → C1C2) ← AuxReg1;                                              1
67.          Fin4
68.        Fin3
69.        Para d←1 Hasta Reglas de la forma 2 con el a-ésimo antecedente Hacer       1
70.          Inicio13
71.            Si No hay convergencia Entonces                                        1
72.            Inicio14
73.              Inc(Número de iteraciones);                                          1
74.              Aux12 ← 0;                                                           1
75.              Aux16 ← 0;                                                           1
76.              Para e ← 1 Hasta NCads Hacer;                                        1
77.              Inicio15
78.                Aux10 ← 0;                                                         1
79.                Ini_Matriz(AuxMC2,G,Cads,ACads[e],RC2);                            x
80.                Ini_Matriz2(AuxMA2,G,Cads,ACads[e]);                               x
81.                Para n ← 1 Hasta Cads.ACads[e].NSC-1 Hacer                         4
82.                Inicio16
83.                  Aux9 ← Outside(G,RC,Cads,AC,AUXMC2,AUXMA2,n,n)
                           × P(AN → n );                                           x^2
84.                  Aux10 ← Aux9 + Aux10;                                            2
85.                Fin16
86.              Aux11 ← ×(1/Procad(G,MatCor,e−1));                                 x
87.              Aux12 ← + Aux12;                                                   1
88.              Fin16
89.              Para e←1 Hasta NCads Hacer                                         1
90.              Inicio17
91.                Aux14 ← 0;                                                       1
92.                Ini_Matriz(AuxMC2,G,Cads,ACads[e],RC2);                          x
93.                Ini_Matriz2(AuxMA2,G,Cads,ACads[e]);                             x
94.                Para o ← 0 Hasta Cads.ACads[e].NSC-1 Hacer                       4
95.                  Para p ← i+1 Hasta Cads.ACads[e].NSC-1 Hacer                   5
96.                  Inicio18
97.                    Aux13 ← Outside(G,RC,Cads,AC,AUXMC2,AUXMA2,o,p)            x^2
98.                    Si Aux13 <> 0 Entonces                                       1
99.                    Inicio19
100.                     Aux131 ← Inside(G,Cads,C1,o,p,AUXMC2)                     x^2
101.                     Si Aux131 <> 0 Entonces                                    1
102.                       Aux14 ← (Aux13×Aux131)+Aux14;                            3
103.                   Fin19
104.                 Fin18
105.               Aux15 ← Aux14×(1/Procad(G,MatCor,e−1));                          x
106.               Aux16 ← Aux15+Aux16;                                             2
107.             Fin17
108.           AuxReg2 ← Aux12/Aux16;                                               2
109.           AuxES2 ← P(AN → n) − AuxReg2;                                        2
```

```
110.            Si ((AuxES2 > (-0.0000000001)) Y (AuxES2 < (0.0000000001)) Entonces 3
111.              Inicio20
112.                Inc(AuxSumNR);                                              1
113.                Marcar la regla como estimada;                              1
114.                P(AN → n) ← AuxReg2;                                        1
115.              Fin20
116.            Sino
117.              P(AN → n) ← AuxReg2;                                          1
118.            Fin14
119.          Fin13
120.      Fin2
121.  Fin1
Fin
```

4.4. Análisis del Algoritmo IO

La función de complejidad del Procedimiento IO es $f(n) \in O(k|\Omega|n^3|n_r|)$, donde n_r es el número de reglas, $|\Omega|$ es el número de elementos del corpus, x es la longitud de la cadena observada y k representa el número de iteraciones del procedimiento.

Líneas	Número de operaciones
96-104	$2x^2 + 5$
94-104	$2x^2 + 5x + 9$
89-109	$\|\Omega\|(2x^3 + 7x + 14) + 1$
81-85	$x^3 + 2x + 4$
77-88	$x^3 + 5x + 6$
76-88	$\|\Omega\|(x^3 + 5x + 6) + 4$
72-109	$\|\Omega\|(3x^3 + 12x + 10) + 5$
72-119	$\|\Omega\|(3x^3 + 12x + 10) + 12$
69-119	$n_r[\|\Omega\|(3x^3 + 12x + 10) + 12]$
45-55	$2x^2 + x + 7$
40-56	$2x^3 + x^2 + 9x + 10$
38-56	$\|\Omega\|(2x^3 + x^2 + 9x + 10)$
25-33	$2x^2 + 6$
19-34	$3x^3 + 6x$
15-37	$3x^3 + 9x + 3$
14-37	$\|\Omega\|(3x^3 + 9x + 3)$
10-37	$\|\Omega\|(3x^3 + 9x + 3) + 3$
10-56	$\|\Omega\|(5x^3 + x^2 + 18x + 13) + 4$
8-68	$\|\Omega\|(5x^3 + x^2 + 18x + 13) + 13$
7-119	$n_r[\|\Omega\|(8x^3 + x^2 + 30x + 23) + 13]$
5-119	$kn_r[\|\Omega\|(8x^3 + x^2 + 30x + 23) + 13]$
1-121	$kn_r[\|\Omega\|(8x^3 + x^2 + 30x + 23) + 13] + 3$
$f(n) =$	$kn_r[\|\Omega\|(8x^3 + x^2 + 30x + 23) + 13] + 3$

Cuadro 4.1: Cantidad de operaciones del procedimiento IO

Capítulo 5

Experimentos

En capítulos anteriores se presentaron los Algoritmos de Estimación Inside, Outside e IO, así como el orden de complejidad. En adelante, se trata su implementación en un lenguaje de programación determinado, se analiza experimentalmente el coste temporal además de otros factores que se deben tener en cuenta como las características de la máquina con la que se realizaron las pruebas y el tamaño de los datos de entrada de los algoritmos. De la misma manera, se estudiarán las implementaciones de las funciones y procedimientos útiles para el desarrollo de los Algoritmos Inside, Outside y de Estimación IO, así como la calidad del modelo resultante.

5.1. Marco Experimental

En esta sección se exponen ciertos factores que insiden en la estimación de los parámteros de las GIP. Los experimentos se realizaron con las diferentes asignaciones de probabilidad inicial para observar los efectos de su implementación en el coste temporal de los algoritmos Inside, Outside y el algoritmo IO.

5.1.1. Rango de datos de entrada

Reglas de producción: $2 \leq |P| \leq 47$. El caso $|P| = 1$ carece de relevancia pues quiere decir que se tiene una sola regla de producción, y como se trabaja con gramáticas en Forma Normal de Chomsky hay dos opciones para esa regla, $A \rightarrow BC$ y $A \rightarrow a$, descartando la primera opción, pues esta regla no generaría ninguna derivación.

Símbolos: $2 \leq |N \cup \Sigma| \leq 27$. El conjunto de terminales y no terminales debe estar formado por al menos dos elementos suficientes para formar una regla de producción, esto es, un no terminal como el símbolo inicial de la gramática y un terminal.

Cadena: $2 \leq |x| \leq 33$. La longitud de la cadena varía desde dos por ser el caso básico en el proceso de estimación.

Corpus: El número de elementos del conjunto de muestras o corpus se determina en partes representativas para las distintas pruebas, es decir, para el proceso de estimación, las gramática G1 trabaja con un corpus de 3.000 cadenas, mientras que las gramátcias G2, G2 sentences, y G6 realizan su proceso de estimación con corpus de 4.000, ahora, para la asignación de probabilidad por frecuencia se hacen experimentos con un máximo de 50.000 cadenas.

Recursos de máquina y software

Las características básicas de los computadores donde se realizaron los experimentos son las siguientes:

- Procesador Intel Pentium Dual Core T2390
  Memoria RAM 3072MB.
  Disco Duro 250GB.
  Sistema operativo Windows Vista Home Premium 32bits

- Procesador AMD Athlon(tm) Dual Core 64.
  Memoria 2048MB.
  Disco Duro 120GB.
  Sistema operativo Windows Vista Home Basic 32bits

- Delphi versión 5.0 (Build 5,62) como el entorno de desarrollo de software.

- Software para generar y transformar Gramáticas Libres de Contexto a Forma Normal de Chormsky (FNC) y aplicación para generar muestras dada una gramática en FNC desarrollado en el trabajo de grado titulado Reducción de Gramáticas

5.1.2. Parámetros de una GIP a tener en cuenta

Para realizar el estudio de la estimación de los parámetros de una GIP, desde el punto de vista del coste temporal de la estimación de sus parámetros se consideran los siguientes elementos, que se supone afectan el rendimiento de los algoritmos previos y de estimación: Número de símbolos terminales Σ, número de símbolos no terminales N, número de reglas de producción $|P|$, asignación de probabilidades Pb, longitud de la cadena $|x|$ y tamaño del corpus o número de cadenas.

5.1.3. Metodología experimental

Tiempo empleado en la asignación de las probabilidades de las reglas de las gramáticas

La asignación de las probabilidades ha sido determinada de la siguiente manera, inicialmente se ha trabajado con distribución de probabilidad uniforme (DU), es decir un valor distribuido de manera equitativa para las reglas con un antecedente común, distribución aleatoria (DA) es una distribución realizada al azar y la distribución asociada a los elementos del corpus o distribución por frecuencia (DF), determinada por el nivel de ocurrencia de las reglas asociadas a las cadenas del corpus. En esta sección se va a presentar un cuadro comparativo de las asignaciones de probabilidad con relación al tiempo empleado en realizar dicho procedimiento y las características de la gramática G(NT,NNT,P), donde $|NT|$ es el número de terminales, $|NNT|$ es el número de no terminales y $|P|$ el número de reglas de producción[1].

Obsérvese como, la tendencia del tiempo empleado por cada una de las asignaciones de probabilidad tiende a crecer a medida que el número de reglas y símbolos aumenta, además, con respecto a la cantidad de tiempo empleado en realizar la respectiva asignación se tiene en órden ascendente a la Distribución Uniforme, Distribución Aleatoria y Distribución por Frecuencia, representada en la figura 5.1 para un corpus de 100 cadenas.

[1]Las gramáticas evaluadas y sus características están referenciadas en el apéndice C.

ASIGNACIÓN DE LA PROBABILIDAD				
GRAMÁTICA	**TIEMPO (Seg)**			
	DU	**DA**	**DF**	
			C100	**C50.000**
G1(2,3,5)	0,015	0,047	0,062	9,641
G2(5,12,25)	0,031	0,031	0,109	68,236
G3(14,13,47)	0,031	0,078	0,093	5,354
G4(14,13,47)sentences	0,047	0,094	0,11	6,635

Cuadro 5.1: Tiempo empleado por las diferentes asignaciones de probabilidad, para distribución uniforme, aleatoria y por frecuencia dadas las gramáticas G1, G2, G3 Y G4, donde C100 y C50.000 representan corpus de 100 y 50.000 cadenas respectivamente.

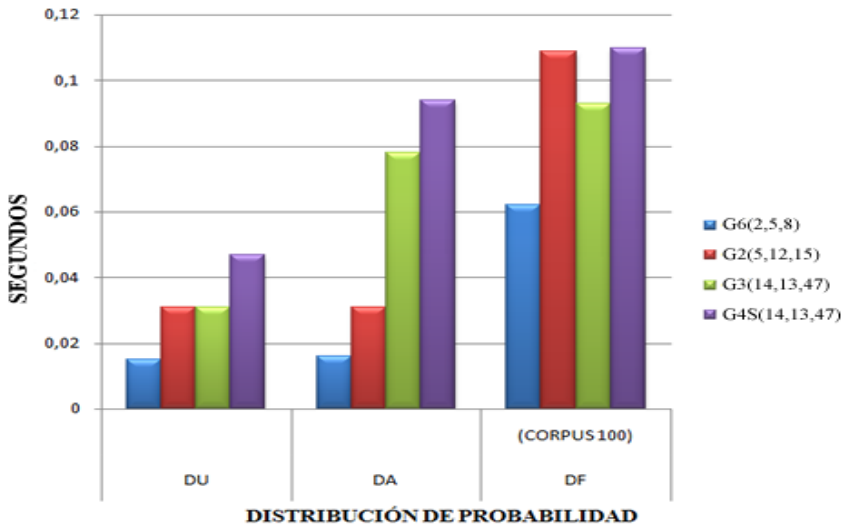

Figura 5.1: Relación de la asignación de probabilidades y el tiempo empleado.

Experimentos para la función CYK2

El algoritmo CYK se puede ampliar para determinar no sólo si una cadena o frase pertenece al lenguaje generado por una gramática, sino también puede utilizarse para obtener los árboles de derivación de una cadena. A continuación se presenta el tiempo empleado por esta función en generar esos árboles para algunas gramáticas y corpus en particular y los respectivos monomios, es decir el número de veces que aparece cada regla en la derivación[2] d_x, además se muestra el tiempo empleado en mostrar los árboles, con la respectiva cantidad de los mismos. En este experimento, para las cadenas de G2 se tiene una longitud mínima y máxima de 2 y 15 respectivamente, mientras que para G3, estas longitudes son 3 y 16.

A partir de este y otros experimentos se puede notar la eficiencia del algoritmo, debido al uso de programación dinámica y las estructuras por antecedente común y consecuente común que se diseñaron, permitiendo búsquedas más eficientes sobre las reglas de las gramáticas. Como puede verse en el cuadro 5.2, el tiempo necesario para mostrar los árboles es mayor que el tiempo empleado por el algoritmo para generarlos, lo cual no afecta su uso en algoritmos posteriores ya que en éstos procedimientos solo se requiere generar los árboles y no mostrarlos.

[2]En términos de la definición 1.17

GRAMÁTICA	N° DE CADENAS	LONG. PROMEDIO DE LAS CADENAS	TIEMPO EN GENERAR ÁRBOLES Y MONOMIOS	N° DE ÁRBOLES	TIEMPO EN MOSTRAR ÁRBOLES	TIEMPO EN MOSTRAR MONOMIOS
G2(5,12,25)	500	8,878	0,156	1138	6,13	0,265
	1000	9,477	0,281	2944	16,692	0,608
	1500	9,801	0,39	4553	26,286	1,061
	2000	10,055	0,593	6557	39,998	1,699
	2500	10,24	0,733	8196	47,86	2,262
	3000	10,376	0,905	10669	68,406	3,962
	3500	10,494	1,076	12641	81,307	5,335
	4000	10,604	1,264	14778	97,968	6,911
	4500	10,687	1,42	16879	111,431	8,751
	5000	10,767	1,591	19127	129,464	11,013
	5500	10,596	1,763	20265	136,11	12,215
	6000	10,452	1,81	21403	143,847	13,057
	6500	10,331	1,919	22541	154,596	14,18
	7000	10,227	2,044	23679	157,934	15,163
	7500	10,137	2,199	24817	169,576	15,865
	8000	10,059	2,231	25995	180,897	17,191
	8500	9,989	2,355	27093	189,56	18,03
	9000	9,398	2,496	28231	193,362	19,515
	9500	9,450	2,59	29369	197,87	21,029
	10000	10,767	3,182	38254	278,397	35,474

Cuadro 5.2: Experimento CYK2.

Evaluación de la función Outside dependiendo de la posición inicial del parámetro i para determinar $Pr(x|G_p)$.

Del capítulo 3 se tiene que $\Pr(x|G_p) = \sum_{A \in N} f(A\langle i, i\rangle)p(A \to x_i)$, para cualquier i fijo, $1 \leq i \leq |x|$, es la probabilidad de generar una cadena mediante el algoritmo Outside, ahora, se va a determinar cómo varía el tiempo de evaluación respecto a la posición i fijada en la cadena usando para los experimentos la gramática G2.

- Para una cadena de longitud diez y con una distribución de probabilidad uniforme.

CADENA DE LONGITUD 10		
Posición(i)	**Probabilidad**	**Tiempo**
1	4,17E-05	0,008
2	4,17E-05	0,023
3	4,17E-05	0,014
4	4,17E-05	0,008
5	4,17E-05	0,008
6	4,17E-05	0,006
7	4,17E-05	0,008
8	4,17E-05	0,009
9	4,17E-05	0,011
10	4,17E-05	0,013

Cuadro 5.3: Tiempo empleado por la función Outside para determinar $Pr(x|G_p)$ en cada una de las posiciones (i) de una cadena de longitud 10.

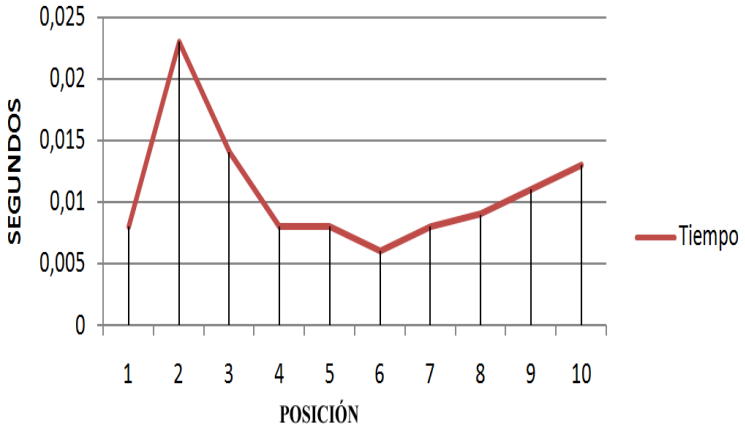

Figura 5.2: Tiempo empleado por la función Outside en calcular la probabilidad para cada posición en una cadena de longitud 10.

- Para una cadena de longitud quince y con una distribución de probabilidad uniforme.

CADENA DE LONGITUD 15		
Posición(i)	**Probabilidad**	**Tiempo**
1	4,62E-06	0,171
2	4,62E-06	0,14
3	4,62E-06	0,141
4	4,62E-06	0,094
5	4,62E-06	0,078
6	4,62E-06	0,062
7	4,62E-06	0,078
8	4,62E-06	0,062
9	4,62E-06	0,063
10	4,62E-06	0,047
11	4,62E-06	0,062
12	4,62E-06	0,078
13	4,62E-06	0,093
14	4,62E-06	0,125
15	4,62E-06	0,094

Cuadro 5.4: Tiempo empleado por la función Outside para determinar $Pr(x|G_p)$ en cada una de las posiciones (i) de una cadena de longitud 15.

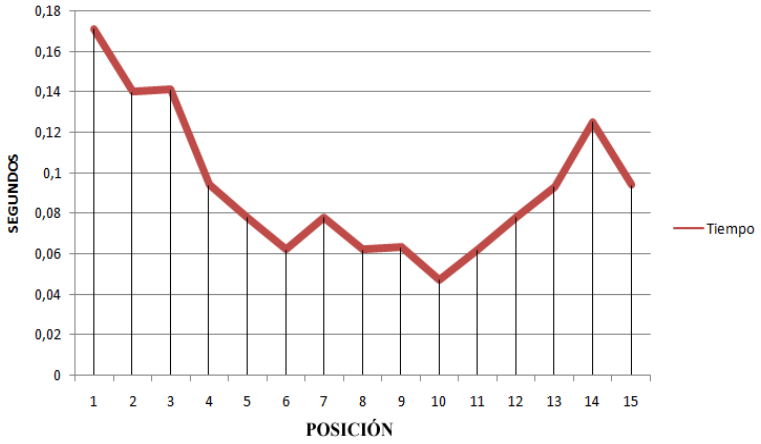

Figura 5.3: Tiempo empleado por la función Outside en calcular la probabilidad para cada posición en una cadena de longitud 15.

- Para una cadena de longitud veinte y con una distribución de probabilidad uniforme.

CADENA DE LONGITUD 20		
Posición(i)	Probabilidad	Tiempo
1	2,67E-09	0,246
2	2,67E-09	0,265
3	2,67E-09	0,237
4	2,67E-09	0,225
5	2,67E-09	0,208
6	2,67E-09	0,19
7	2,67E-09	0,178
8	2,67E-09	0,168
9	2,67E-09	0,17
10	2,67E-09	0,159
11	2,67E-09	0,148
12	2,67E-09	0,145
13	2,67E-09	0,154
14	2,67E-09	0,16
15	2,67E-09	0,178
16	2,67E-09	0,191
17	2,67E-09	0,203
18	2,67E-09	0,209
19	2,67E-09	0,218
20	2,67E-09	0,225

Cuadro 5.5: Tiempo determinado por la función Outside en cada una de las posiciones de una cadena de longitud 20.

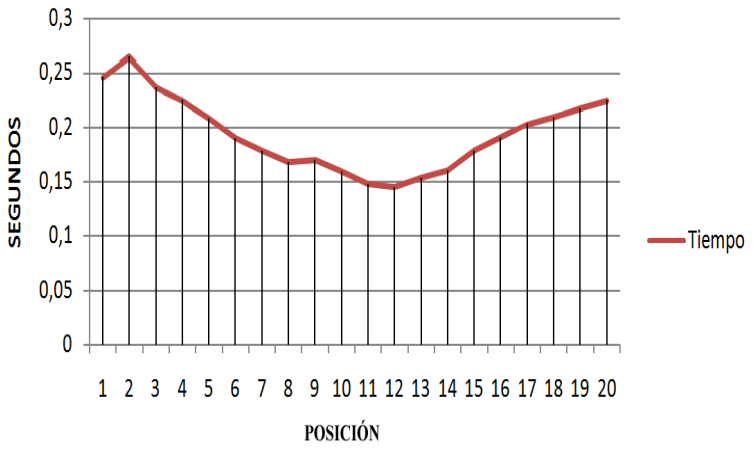

Figura 5.4: Tiempo empleado por la función Outside en calcular la probabilidad para cada posición en una cadena de longitud 20.

- Para una cadena de longitud veinticinco y con una distribución de probabilidad uniforme.

CADENA DE LONGITUD 25		
Posición(i)	Probabilidad	Tiempo
1	1,90E-09	0,952
2	1,90E-09	0,764
3	1,90E-09	0,718
4	1,90E-09	0,686
5	1,90E-09	0,905
6	1,90E-09	0,608
7	1,90E-09	0,562
8	1,90E-09	0,499
9	1,90E-09	0,468
10	1,90E-09	0,452
11	1,90E-09	0,452
12	1,90E-09	0,422
13	1,90E-09	0,453
14	1,90E-09	0,436
15	1,90E-09	0,484
16	1,90E-09	0,53
17	1,90E-09	0,484
18	1,90E-09	0,468

19	1,90E-09	0,484
20	1,90E-09	0,639
21	1,90E-09	1,061
22	1,90E-09	0,858
23	1,90E-09	0,842
24	1,90E-09	0,718
25	1,90E-09	0,936

Cuadro 5.6: Tiempo determinado por la función Outside en cada una de las posiciones de una cadena de longitud 25.

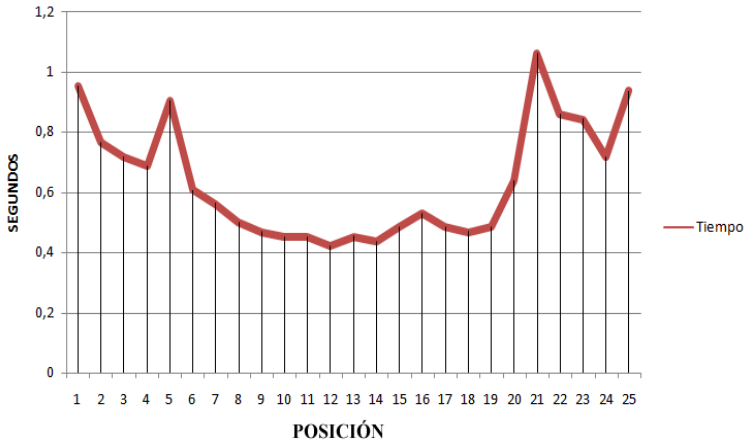

Figura 5.5: Tiempo empleado por la función Outside en calcular la probabilidad para cada posición en una cadena de longitud 25.

- Para una cadena de longitud treinta y con una distribución de probabilidad uniforme.

CADENA DE LONGITUD 30		
Posición(i)	Probabilidad	Tiempo
1	2,41E-15	0,854
2	2,41E-15	0,975
3	2,41E-15	0,914
4	2,41E-15	0,845

5	2,41E-15	0,794
6	2,41E-15	0,726
7	2,41E-15	0,663
8	2,41E-15	0,636
9	2,41E-15	0,585
10	2,41E-15	0,518
11	2,41E-15	0,466
12	2,41E-15	0,434
13	2,41E-15	0,395
14	2,41E-15	0,4
15	2,41E-15	0,402
16	2,41E-15	0,428
17	2,41E-15	0,447
18	2,41E-15	0,459
19	2,41E-15	0,484
20	2,41E-15	0,49
21	2,41E-15	0,502
22	2,41E-15	0,502
23	2,41E-15	0,497
24	2,41E-15	0,488
25	2,41E-15	0,565
26	2,41E-15	0,561
27	2,41E-15	0,629
28	2,41E-15	0,71
29	2,41E-15	0,734
30	2,41E-15	0,753

Cuadro 5.7: Tiempo determinado por la función Outside en cada una de las posiciones de una cadena de longitud 30.

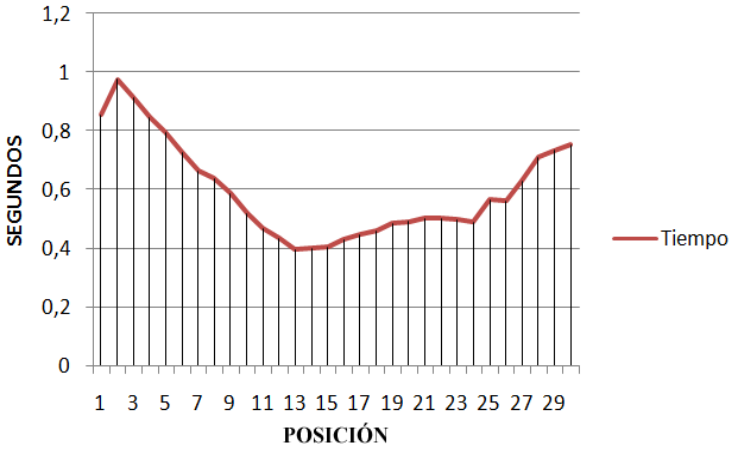

Figura 5.6: Tiempo empleado por la función Outside en calcular la probabilidad para cada posición en una cadena de longitud 30.

De estos y otros experimentos (los que pueden verse en el Apéndice C) se concluye que para posiciones próximas al punto medio de la longitud de la cadena, el valor de la probabilidad se estima en un tiempo relativamente menor al comparado con el valor de los cercanos a los puntos extremos, es importante anotar que para cadenas de longitud menor que diez, los tiempos no determinan en general el mismo comportamiento.

Comparación de los tiempos empleados al calcular las probabilidades de una cadena mediante los algoritmos Inside y Outside y su relación con la longitud de la cadena.

En estos experimentos se hace la comparación entre el tiempo usado por los algoritmos Inside y Outside en encontrar el valor de probabilidad de la cadena dada la misma gramática que la genera. Se procede entonces, variando la longitud de la cadena desde 5 hasta 30.

Observando el cuadro 5.8 para la gramática dos, constituida por cinco terminales, doce no terminales y quince reglas de producción. 2DU, 2DA y 2DF, hacen referencia a la gramática dos y a sus distribuciones de probabilidad uniforme, aleatoria y por frecuencia respectivamente, se tiene:

TIEMPO DE ESTIMACIÓN CON RESPECTO A LA LONGITUD DE LA CADENA										
GRAMÁTICA	LONG.CAD	DU			DA			DF		
			TIEMPO			TIEMPO			TIEMPO	
		VALORPROB.	INSIDE	OUTSIDE	VALOR PROB	INSIDE	OUTSIDE	VALOR PROB	INSIDE	OUTSIDE
G2(5,12,15)	5	1,95E-03	0	0,015	1,90E-07	0	0	1,37E-03	0	0,016
	10	6,77E-05	0,015	0,016	3,65E-09	0,016	0,016	3,56E-05	0,016	0,015
	15	4,62E-06	0,063	0,062	5,58E-09	0,046	0,062	1,46E-06	0,047	0,078
	20	1,44E-06	0,265	0,234	2,89E-09	0,25	0,234	5,58E-07	0,25	0,265
	25	1,90E-09	0,624	0,468	3,01E-15	0,593	0,468	3,01E-09	0,639	0,453
	30	1,81E-08	0,936	0,89	5,61E-14	0,92	0,874	7,21E-09	0,952	0,873
G3(14,13,47)	5	4,76E-06	0	0,016	7,90E-13	0	0	4,76E-06	0	0
	10	9,93E-11	0,016	0,015	1,47E-17	0	0,031	9,93E-11	0,015	0,031
	15	1,31E-17	0,031	0,078	2,81E-41	0,031	0,078	1,31E-17	0,031	0,078
	20	6,97E-27	0,094	0,093	4,43E-60	0,078	0,093	6,97E-27	0,078	0,093
	25	4,14E-36	0,172	0,172	6,96E-79	0,172	0,187	4,14E-36	0,187	0,187
	30	2,64E-45	0,281	0,453	1,09E-97	0,296	0,468	2,64E-45	0,297	0,453
G4(14,13,47) sentences	5	6,98E-07	0	0,015	6,69E-10	0	0	2,26E-07	0	0
	10	3,93E-11	0,015	0	1,23E-24	0,015	0,016	9,42E-11	0,016	0,016
	15	1,12E-18	0,047	0,031	2,33E-30	0,047	0,046	1,45E-16	0,046	0,046
	20	5,16E-27	0,109	0,265	2,99E-37	0,093	0,281	4,10E-23	0,094	0,312
	25	2,67E-36	0,187	0,483	9,05E-49	0,219	0,499	2,29E-30	0,187	0,53
	30	1,38E-45	0,312	0,842	2,74E-60	0,343	0,827	1,27E-37	0,296	0,858

Cuadro 5.8: Tiempo empleado por las funciones Inside y Outside en calcular la probabilidad de cadenas de longitud 5, 10, 15, 20, 25 y 30 dadas las gramáticas G2, G3 y G4.

1. Tiempo empleado por las funciones Inside y Outside en calcular la probabilidad de las cadenas para una distribución de probabilidad uniforme.

2DU		
LONG.CAD	**TIEMPO**	
	INSIDE	**OUTSIDE**
5	0	0,015
10	0,015	0,016
15	0,063	0,062
20	0,265	0,234
25	0,624	0,468
30	0,936	0,89

Cuadro 5.9: Tiempo empleado por las funciones Inside y Outside en calcular la probabilidad de las cadenas, con una distribución de probabilidad uniforme dada la gramátia G2.

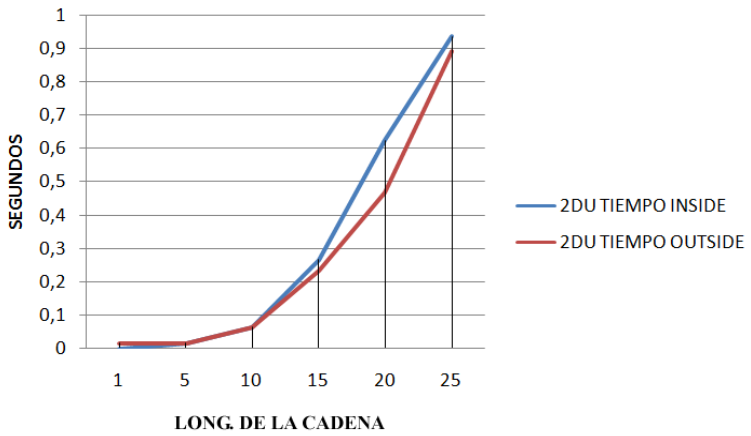

Figura 5.7: Tiempo empleado por las funciones Inside y Outside en calcular la probabilidad de las cadenas, con una distribución uniforme.

Tiempo empleado por las funciones Inside y Outside en calcular la probabilidad de las cadenas para una distribución de probabilidad aleatoria.

2DA		
LONG.CAD	**TIEMPO)**	
	INSIDE	**OUTSIDE**
5	0	0
10	0,016	0,016
15	0,046	0,062
20	0,25	0,234
25	0,593	0,468
30	0,92	0,874

Cuadro 5.10: Tiempo empleado por las funciones Inside y Outside en calcular la probabilidad de las cadenas, con una distribución de probabilidad aleatoria dada la gramátia G2.

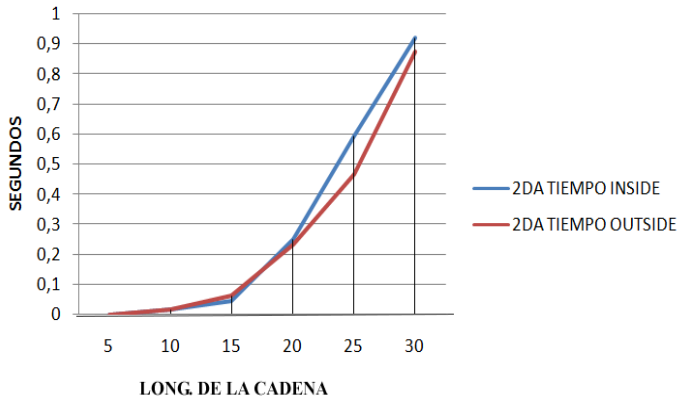

Figura 5.8: Tiempo empleado por las funciones Inside y Outside en calcular la probabilidad de las cadenas, con una distribución aleatoria.

Tiempo empleado por las funciones Inside y Outside en calcular la probabilidad de las cadenas para una distribución de probabilidad por frecuencia.

2DF		
LONG.CAD	**TIEMPO**	
	INSIDE	**OUTSIDE**
5	0	0,016
10	0,016	0,015
15	0,047	0,078
20	0,25	0,265
25	0,639	0,453
30	0,962	0,873

Cuadro 5.11: Tiempo empleado por las funciones Inside y Outside en calcular la probabilidad de las cadenas, con una distribución de probabilidad por frecuencia dada la gramátia G2.

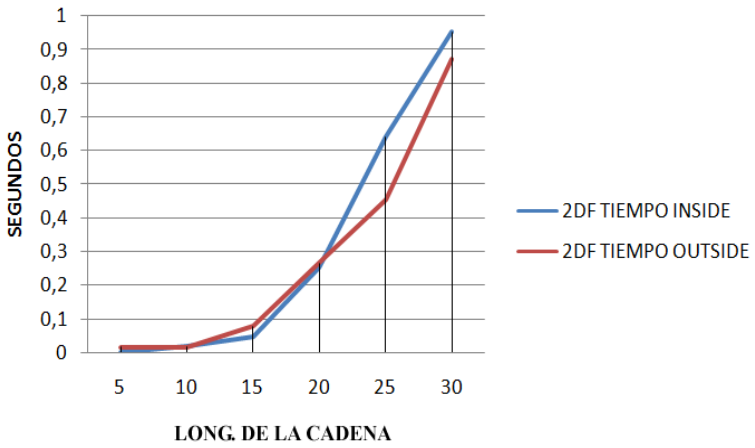

Figura 5.9: Tiempo empleado por las funciones Inside y Outside en calcular la probabilidad de las cadenas, con una distribución por frecuencia.

Los cuadros 5.9, 5.10 y 5.11 muestran que el tiempo empleado por el algoritmo Inside para asignar la probabilidad a una cadena de la gramática 2 es mayor que el empleado por el algoritmo Outside, y además esta diferencia se incrementa con la longitud de la cadena, debido a que el algoritmo Outside toma la posición i en el punto medio de la cadena y procede a hacer los cálculos en cada una de las dos mitades. Los tiempos mínimo y máximo son 0,015s y 0,962s, este último determinado con la distribución

aleatoria de esta gramática. A continuación se presenta un gráfico representativo para un corpus formado por treinta y tres cadenas, cada una de ellas con esa respectiva longitud.

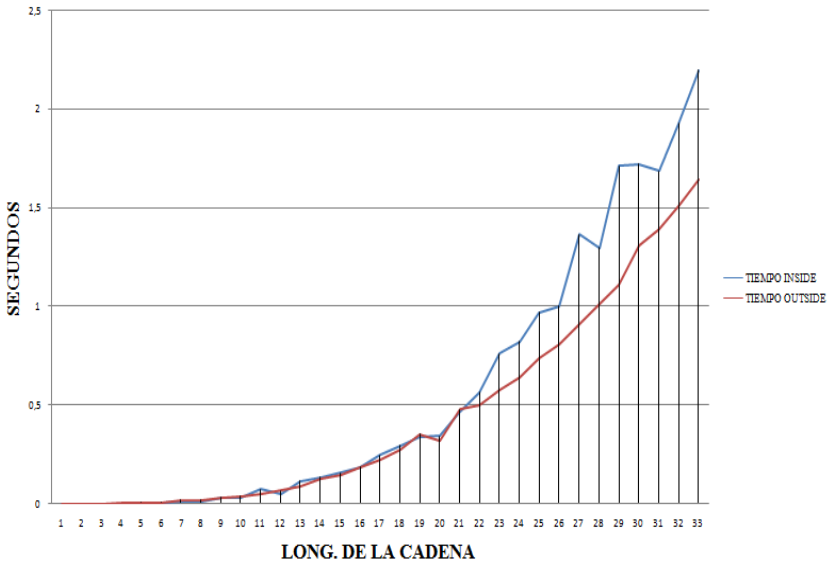

Figura 5.10:Tiempo empleado por las funciones Inside y Outside en calcular la probabilidad de las cadenas con distribución uniforme

Experimentos con el algoritmo de estimación IO.

En esta sección, se exponen los experimentos realizados con el algoritmo IO, como tiempo en el proceso de estimación de las distintas gramáticas con respecto a cada distribución de probabilidad inicial, también el promedio de iteraciones empleados en este procedimiento, además de los modelos resultantes de la estimación.

Se estima el valor de la función de probabilidad de una GIP usando el algoritmo IO, con las gramáticas G1, G2, G2 sentences y G6, partiendo de dos casos, con distribución inicial de probabilidad uniforme y otro por frecuencia, empleando un mismo corpus para cada opción.

Las características de los corpus son las siguientes:

GRAMÁTICA	ELEMENTOS DEL CORPUS	LONG. MÍNIMA	LONG. MÁXIMA	LONG. MEDIA
G1(2,3,5)	3.000	2	14	11,671
G2(5,12,25)	4.000	2	15	10,604
G2(5,12,25)sentences	4.000	2	14	9,909
G6(2,5,8)	4.000	2	16	12,869

Cuadro 5.12: Características de los corpus utilizados según las gramáticas G1, G2, G2 (sentences) y G6.

Proceso de estimación para G1, G2, G2 sentences y G6:

Proceso de estimación para G1:

N°	Regla	Probabilidad Asignada inicialmente (Por Frecuencia)	Probabilidad Estimada	Probabilidad Asignada inicialmente (Uniforme)	Probabilidad Estimada
1	A1 → A2 A3	1,000E+00	1,000E+00	1,000E+00	1,000E+00
2	A2 → A3 A1	2,941E-01	8,571E-01	5,000E-01	1,168E-01
3	A2 → b	7,059E-01	1,429E-01	5,000E-01	1,156E+00
4	A3 → A1 A2	2,963E-01	7,237E-12	5,000E-01	5,539E-01
5	A3 → a	7,037E-01	1,051E+00	5,000E-01	4,476E-01

Cuadro 5.13: Características de la estimación para las gramáticas G1.

Proceso de estimación para G2:

N°	Regla	Probabilidad Asignada inicialmente (Por Frecuencia)	Probabilidad Estimada	Probabilidad Asignada inicialmente (Uniforme)	Probabilidad Estimada
1	A0 → A3 A7	1,567E-01	1,636E-10	1,250E-01	1,311E-10
2	A0 → A2 A11	1,779E-01	2,903E-01	1,250E-01	2,903E-01
3	A0 → A4 A10	1,397E-01	7,097E-01	1,250E-01	7,097E-01
4	A0 → A4 A4	9,747E-02	2,454E-13	1,250E-01	9,673E-16
5	A0 → A5 A0	1,792E-01	1,518E-20	1,250E-01	4,061E-12
6	A0 → A5 A4	6,417E-02	3,181E-13	1,250E-01	1,063E-13
7	A0 → A6 A6	5,375E-02	1,928E-12	1,250E-01	2,267E-15
8	A0 → t2	1,312E-01	4,026E-11	1,250E-01	6,675E-11
9	A1 → A2 A11	1,455E-01	1,087E-01	2,000E-01	1,087E-01
10	A1 → A4 A4	1,717E-01	1,098E-20	2,000E-01	2,003E-14
11	A1 → A5 A0	2,634E-01	8,985E-12	2,000E-01	9,055E-19
12	A1 → A6 A6	1,220E-01	1,467E-26	2,000E-01	1,862E-13
13	A1 → t2	2,975E-01	8,913E-01	2,000E-01	8,913E-01
14	A2 → A1 A9	3,391E-01	1,483E-17	3,333E-01	7,866E-14
15	A2 → A4 A8	2,139E-01	1,000E+00	3,333E-01	1,000E+00
16	A2 → t2	4,470E-01	3,906E-16	3,333E-01	2,777E-10
17	A3 → t3	1,000E+00	1,000E+00	1,000E+00	1,000E+00
18	A4 → t1	1,000E+00	1,000E+00	1,000E+00	1,000E+00
19	A5 → t2	1,000E+00	1,000E+00	1,000E+00	1,000E+00
20	A6 → t4	1,000E+00	1,000E+00	1,000E+00	1,000E+00
21	A7 → A1 A0	1,000E+00	1,000E+00	1,000E+00	1,000E+00
22	A8 → A1 A4	1,000E+00	1,000E+00	1,000E+00	1,000E+00
23	A9 → A2 A0	1,000E+00	1,000E+00	1,000E+00	1,000E+00
24	A10 → A4 A0	1,000E+00	1,000E+00	1,000E+00	1,000E+00
25	A11 → A4 A1	1,000E+00	1,000E+00	1,000E+00	1,000E+00

Cuadro 5.14: Características de la estimación para la gramática G2.

Proceso de estimación para G2 sentences:

N°	Regla	Probabilidad Asignada inicialmente (Por Frecuencia)	Probabilidad Estimada	Probabilidad Asignada inicialmente (Uniforme)	Probabilidad Estimada
1	Sentence → Subject PrePhrase	1,594E-01	6,745E-15	1,250E-01	3,318E-12
2	Sentence → Object Conjunction	1,815E-01	6,667E-01	1,250E-01	1,000E+00
3	Sentence → NounPhrase Adverb	1,432E-01	2,517E-15	1,250E-01	4,753E-12
4	Sentence → NounPhrase NounPhrase	9,347E-02	3,123E-12	1,250E-01	7,041E-13
5	Sentence → NounGroup Sentence	1,783E-01	7,500E-01	1,250E-01	2,407E-10
6	Sentence → NounGroup NounPhrase	6,551E-02	4,570E-12	1,250E-01	3,349E-13
7	Sentence → Noun Noun	5,692E-02	1,348E-12	1,250E-01	6,350E-14
8	Sentence → like	1,218E-01	1,574E-11	1,250E-01	2,373E-13
9	Verb → Object Conjunction	1,396E-01	1,617E-11	2,000E-01	2,500E-0
1 10	Verb → NounPhrase NounPhrase	1,816E-01	3,938E-19	2,000E-01	5,000E-01
11	Verb → NounGroup Sentence	2,499E-01	1,142E-51	2,000E-01	9,751E-11
12	Verb → Noun Noun	1,321E-01	6,514E-11	2,000E-01	9,788E-11
13	Verb → like	2,969E-01	1,000E+00	2,000E-01	2,500E-01
14	Object → Verb Article	3,243E-01	7,740E-17	3,333E-01	2,555E-17
15	Object → NounPhrase Preposition	2,128E-01	1,000E+00	3,333E-01	1,000E+00
16	Object → like	4,630E-01	3,591E-14	3,333E-01	7,092E-12
17	Subject→ time	1,000E+00	1,000E+00	1,000E+00	1,000E+00
18	NounPhrase → fly	1,000E+00	1,000E+00	1,000E+00	1,000E+00
19	NounGroup → like	1,000E+00	1,000E+00	1,000E+00	1,000E+00
20	Noun → arrow	1,000E+00	1,000E+00	1,000E+00	1,000E+00
21	PrePhrase → Verb Sentence	1,000E+00	1,000E+00	1,000E+00	1,000E+00
22	Preposition → Verb NounPhrase	1,000E+00	1,000E+00	1,000E+00	1,000E+00
23	Article → Object Sentence	1,000E+00	1,000E+00	1,000E+00	1,000E+00
24	Adverb → NounPhrase Sentence	1,000E+00	1,000E+00	1,000E+00	1,000E+00
25	Conjunction → NounPhrase Verb	1,000E+00	1,000E+00	1,000E+00	1,000E+00

Cuadro 5.15: Características de la estimación para la gramática G2 sentences.

Proceso de estimación para G6:

N°	Regla	Probabilidad Asignada inicialmente (Por Frecuencia)	Probabilidad Estimada	Probabilidad Asignada inicialmente (Uniforme)	Probabilidad Estimada
1	S → A C	4,227E-01	4,002E-01	2,500E-01	4,002E-01
2	S → B D	4,219E-01	4,400E-01	2,500E-01	4,400E-01
3	S → A A	7,459E-02	1,724E-02	2,500E-01	1,724E-02
4	S → B B	8,081E-02	1,426E-01	2,500E-01	1,426E-01
5	A → a	1,000E+00	1,000E+00	1,000E+00	1,000E+00
6	B → b	1,000E+00	1,000E+00	1,000E+00	1,000E+00
7	C → S A	1,000E+00	1,000E+00	1,000E+00	1,000E+00
8	D → S B	1,000E+00	1,000E+00	1,000E+00	1,000E+00

Cuadro 5.16: Características de la estimación para la gramática G6.

5.1.4. Calidad del modelo estimado

Ahora, conociendo los valores de probabilidad, asignados por el algoritmo IO a las reglas de la gramática, es lógico preguntarse respecto a la calidad del modelo del lenguaje así obtenido. Es así como en el procesamiento del lenguaje natural, la perplejidad por palabra (PP) [BJM83, Jel98] es usada frecuentemente a la hora de evaluar la calidad de los modelos del lenguaje, o esta misma medida, se usa para evaluar la bondad de las gramáticas, en el sentido de la calidad del lenguaje que produce. Una interpretación intuitiva de esta medida es considerarla como la capacidad del modelo para tratar los eventos que se van produciendo en el proceso de análisis, valores próximos a cero indican que el modelo tiene mayor capacidad expresiva y por tanto es mejor. Esta medida se evalúa sobre un conjunto de datos que no han sido utilizados en el proceso de entrenamiento denominado conjunto de test (Ts). Cuando el modelo es una GIP esta medida se define como [Dup96]:

$$PP(T_s, G_p) = e^{-\frac{\Sigma_{x \in T_s} LnPr(x|G_p)}{\Sigma_{x \in T_s} |x|}} \tag{5.1.1}$$

Cuando Ts es exactamente el conjunto de entrenamiento o corpus, la maximización de la verosimilitud de la muestra hace decrecer esta medida. Por tanto, si el conjunto Ts sigue una distribución similar a la del conjunto de entrenamiento, entonces es de esperar que esta medida tienda a decrecer.

Conjunto de entrenamiento, conjunto de test y gramática inicial para los experimentos

El conjunto de datos descrito inicialmente se dividió en conjuntos de entrenamiento o corpus y para este corpus, un conjunto de test (Ts). Cada corpus se utilizó para estimar una GIP a partir de una GIP inicial con las gramáticas G1, G2, G2 sentences y G6.

Características del conjunto de test (Ts) utilizados en el experimento:

GRAMÁTICA	ELEMENTOS DEL Ts	LONG. MÍNIMA	LONG. MÁXIMA	LONG. MEDIA
G1(2,3,5)	2.000	10	14	13,019
G2(5,12,25)	2.000	2	15	10,604
G2(5,12,25)sentences	2.000	6	14	10,991
G6(2,5,8)	2.000	12	16	15,573

Cuadro 5.17: Características de los conjuntos de Test utilizados según las gramáticas G1, G2, G2 (sentences) y G6.

En la siguiente tabla se presenta el valor de la perplejidad para los modelos descritos anteriormente.

GRAMÁTICA	ASIGNACIÓN DE PROBABILIDAD	ELEMENTOS DEL CORPUS	ELEMENTOS DEL CORPUS SIN REPETIR	TAMAÑO Ts	TIEMPO DE ESTIMACIÓN	PROMEDIO DE ITERACIONES	PP
G1(2,3,5)	Por frecuencia	5.000	400	2.000	3036,192	52,6	2,220
	Uniforme	3.000	3.000	2.000	7006,412	104,25	1,115
G2(5,12,25)	Por frecuencia	4.000	4.000	2.000	14122,315	22,8	27,791
	Uniforme	4.000	4.000	2.000	15642,416	24,52	25,424
G2(5,12,25) sentences	Por frecuencia	4.000	3.018	2.000	9104,463	15,92	10,341
	Uniforme	4.000	3.018	2.000	11833,391	21	8,048
G6(2,5,8)	Por frecuencia	4.000	300	2.000	143941,188	1477,5	1,244
	Uniforme	4.000	300	2.000	148030,398	1462	1,244

Cuadro 5.18: Resultados de los análisis con corpus y conjuntos de test (Ts) para determinar la perplejidad por palabra.

5.1.5. Conclusiones

- En primer lugar, es importante notar la sensibilidad del algoritmo IO a las asignaciones de probabilidad inicial, uniforme, aleatoria o por frecuencia, dependiendo de éstas distribuciones, el algoritmo IO requiere un mayor o menor tiempo de estimación. Además, influye también en el tiempo de estimación, las características de las cadenas del corpus como longitud mínima y máxima y número de elementos.

- Para realizar un aprendizaje satisfactorio, además de los mencionados, sería importante que las reglas de las gramáticas no generen símbolos redundantes, es decir, los símbolos que tienen un excesivo número de parejas no terminales, por ejemplo, para las reglas:
  $\mathbf{A} \rightarrow$ BC, $\mathbf{A} \rightarrow$ B$\mathbf{A}$, B $\rightarrow$ a y C $\rightarrow$ b, el no terminal **A** está en las reglas como antecedente y como consecuente, lo cual eleva el número de iteraciones de manera repetitiva sobre este no terminal.

Capítulo 6

Conclusiones y Trabajos Futuros

Conclusiones

Después de hacer un estudio detallado de la estructura de los algoritmos Inside y Outside y de realizar varias pruebas con distintas gramáticas y corpus, se hizo necesario diseñar estructuras de datos que entre otras cosas clasificara las reglas en dos tipos distintos, uno de ellos es por antecedente común y otro por consecuente común. En la implementación esto hace más eficiente la búsqueda sobre los arreglos que contienen la información sobre símbolos y probabilidades, ya que en el caso del algoritmo Inside se requiere básicamente la probabilidad de las reglas que tienen a un no terminal en particular como antecedente y para el algoritmo Outside es necesario conocer la probabilidad de las reglas que tengan el mismo antecedente común pero que además tenga los mismos consecuentes (haciendo referencia a reglas de la forma uno) pero en posiciones distintas.

Se consideró el número de iteraciones necesarias para la convergencia del algoritmo IO, notando que requiere un número elevado de ellas comparado con otros algoritmos como el denominado Viterbi Score (VS), el cual utiliza la mejor derivación de la muestra y que al igual que IO trabaja con el Teorema de Transformaciones Crecientes.[1].

La asignación inicial de probabilidades es un factor importante, ya que parte de la suposición que una buena gramática es la que asigna probabilidades según la frecuencia de ocurrencia de las cadenas del corpus, es por eso que se obtuvieron mejores resultados con respecto al tiempo necesario para hacer la estimación, con la asignación de probabilidad por frecuencia, mejorando el tiempo necesario para alcancar los máximos locales con respecto a las otras asignaciones de probabilidad.

[1]Para mayores detalles acerca de el algoritmo VS y su comparación con IO, remitirse a [SJ99]

En contraste, aunque se requiere mayor tiempo para hacer la estimación con una distribución inicial uniforme, los resultados muestran una perplejidad menor que en los otros casos.

Se diseñó un software con entornos que facilitaran el ingreso de datos a los usuarios y que permitieran el estudio detallado de las gramáticas, las cadenas y los árboles de derivacíón, además como resultado adicional el software permite tomar información para la entrada de datos a otros trabajos que se estan realizando actualmente.

Trabajos futuros

- Sería interesante extender los experimentos con corpus mas grandes. Esto plantea el problema de la escasez de recursos de máquina y el tiempo necesario para hacer la estimación, por lo que la agrupación de los símbolos terminales que puedan ser etiquetados con un mismo símbolo sería esencial.

- Analizar e implementar el proceso de minimización de gramáticas propuesto en [LY90] que determine y relocalize símbolos no terminales redundantes.

- Estudiar y aplicar otros métodos que optimizen la función de verosimilitud, por ejemplo los métodos cuasi-Newton para la obtención de los valores de los parámetros de la gramática incontextual probabilística que maximizan esta función de verosimilitud, generando así modelos más eficientes para la estimación de las probabilidades[2].

[2]Trabajo que se está desarrollando actualmente en el marco del proyecto de investigación «Nuevas Alternativas para la Estimación de los Parámetros de una Gramática Incontextual Probabilística»

Bibliografía

[AHU88] A. Aho, J. Hopcroft y J. Ullman. Estructuras datos y análisis de algoritmos. 1988.

[BAK79] J.K. Baker. Trainable grammars for speech recognition. In Klatt and Wolf, editors, Speech Communications for the 97th Meeting of the Acoustical Society of America, pages 31-35. Acoustical Society of America, June 1979.

[BE67] L.E. Baum and J.A. Eagon. An inequeality with applications to statistical prediction for functions of markov chains. *Bull. Amer. Math.* Soc., 73:360-363, 1967.

[BJM83] L.R. Bahl, F. Jelinek, and R.L. Mercer. A maximum likelihood approach to continuous speech recognition. IEEE Trans. Pattern Analysis and Machine Intelligence, PAMI-5(2):179-190, 1983.

[BN04] B. Nelson. Grammar Induction Using Co-Training. School of Informatics University of Edinburgh, 2004.

[CAS96] F. Casacuberta Growth transformations for probabilistic functions of stochastic grammars. IJPRAI, 10(3):183-201, 1996.

[CH02] C. Manning and H. Schütze. Foundations of Statistical Natural Language Processing. MIT Press. 2002.

[DD91] D. Sleator and D. Temperley. Parsing English with a link grammar. Technical Report CMU-CS-91-96, School of computer sciences, Carnegie Mellon

University, 1991.

[Dup93] P. Dupont. Efficient integration of context-free grammars based language models in continuous speech recognition. In New Advances and Trends in Speech Recognition and Coding, pages 179182. 1993.

[Fu82] K.S. Fu. Syntactic Pattern Recognition and Applications. Prentice-Hall, 1982.

[GP97] G. Brassard y P. Bratley. Fundamentos de algoritmia. Universidad de Montreal, 1997.

[GV99] R. Guerequeta y A. Vallecillo. Técnicas de diseño de algoritmos. 1999.

[Ney92] H. Ney. Stochastic Grammars and Pattern Recognition. In P. La face and R. De Mori, Editors, Speech Recognition and Understanding. Recent. Advances, pages 319-344 Springer Verlag, 1992.

[HU79] J. Hopcroft y J. Ullman. Introduction to automata theory languages and computation. Editorial Addison Wesley Publishing Company 1979. Pág. 77-106.

[Jel98] F. Jelinek. Statistical Methods for Speech Recognition. MIT Press,1998.

[KS75] KS. Fu and T. L. Booth. Grammatical Inference: Introduction and Survey part I. IEEE Trans on System, man and Cybernetics, Smc-5(1): 95-111, 1975.

[LY90] K. Lari and S.J. Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. Computer, Speech and Language, 4:35-56, 1990.

[MF07] E. Murillo Fernández. Reducción de Gramáticas Libres de Contexto, Aproximación Computacional. 2007

[MQ03] F. Martinez y G. Quetglas. Introducción a la programación estructurada, 2003.

[SJ99] J.A Sánchez Peiró. Estimación de Gramáticas Incontextuales Probabilísticas y su Aplicación en Modelación del Lenguaje. Tesis Doctoral. Universidad Politécnica de Valencia. Departamento de Sistemas Informáticos y computación. 1999.

[SM04] Seminario de Investigación, Notas de exposiciones en el seminario, 2005.

[VS90] V. Sakakibara. Learning context-free grammar from structural data in polynomial time. T heretical computer Science 76:233-242, 1990.

Apéndice A

Estructuras de datos utilizadas

Para la implementación computacional de los algoritmos de este proyecto fue necesario diseñar unas estructuras de datos específicas que permitieron manejar de forma eficiente los algoritmos de estimación.

Ahora bien, para hacer más claro el diseño de las estructuras se parte del siguiente ejemplo:

Ejemplo 6.1. *Nótese que las reglas están en Forma Normal de Chomsky, es decir, como ya se mencionó anteriormente las reglas de la forma uno y dos que son aquellas representadas respectivamente como:* $A \rightarrow BC$ *y* $A \rightarrow \alpha$.

Terminales	**No Terminales**	**Reglas**	**Probabilidad**	**Cadena**
0. α	2. S	0. S $\rightarrow$ AB	0.3	$\beta\alpha\alpha\beta\alpha$
1. β	3. A	1. S $\rightarrow$ BC	0.7	
	4. B	2. A $\rightarrow$ BA	0.5	
	5. C	3. A $\rightarrow \alpha$	0.5	
		4. B $\rightarrow$ CC	0.8	
		5. B $\rightarrow \beta$	0.2	
		6. C $\rightarrow$ AB	0.3	
		7. C $\rightarrow \alpha$	0.5	
		8. C $\rightarrow$ CB	0.2	

Constantes

- MNReg: Entero que indica el máximo número de reglas. En general: 135.

Ejemplo 6.2. *9.*

- NMRaAC : Entero que indica el número máximo de reglas con antecedente común. En general: 135.

  Ejemplo 6.3. *2, si se refiere al antecedente común B.*

- NMRaCC : Entero que indica el número máximo de reglas con consecuente común. En general: 135.

  Ejemplo 6.4. *3, si se refiere al consecuente común A.*

- MNT: Entero que indica el máximo número de terminales. En general: 50.

  Ejemplo 6.5. *2.*

- MNNT: Entero que indica el máximo número de no terminales. En general: 50.

  Ejemplo 6.6. *4.*

- MNSC: Entero que indica el máximo número de símbolos de la cadena. En general: 35.

  Ejemplo 6.7. *5.*

- MNCads: Entero que indica el máximo número de cadenas. En general: 50.000.

  Ejemplo 6.8. *1.*

Tipos

- TProb: Real que indica el valor de la probabilidad de la regla.

  Ejemplo 6.9. *Para la regla 4: 0.8.*

- TIndReg: Entero que representa el índice de las reglas. Se refiere a el número correspondiente a cada regla.

  Ejemplo 6.10. *Desde 0 hasta 8.*

- TIndSim: Entero que representa el número correspondiente de cada símbolo (terminales y no terminales).

  Ejemplo 6.11. *En este caso particular, desde 0 hasta 5.*

- TIndSc: Entero que representa el número correspondiente de cada símbolo de la cadena. Hace referencia a la posición en la que se encuentra cada terminal en una cadena determinada.

  Ejemplo 6.12. *En este caso, en la posición tres se tendría al terminal* α.

Registros y arreglos

Datos

En este registro se van a guardar los símbolos pertenecientes a la gramática, es decir los terminales y no terminales, esto es:

- TRDatos = Registro
  - TAS: Arreglo [1..MNT + MNNT] de Cadena[10];
  - NT: TIndSim;
  - NNT:TIndSim;

  Fin_Registro

TAS representa al arreglo de símbolos, constituidos, como ya se mencionó anteriormente por los terminales y no terminales, si un elemento de este arreglo son terminales, entonces se refiere a una cadena de la gramática cuya longitud máxima es 10, ahora bien, el elemento de la posición nt+1 será el símbolo inicial de la gramática, Esto es $\Sigma = 1, 2, ..., MNT$ y $N = MNT + 1, MNT + 2, ..., NS$.

Ejemplo 6.13.

TRDatos					
TAS					
α	β	S	A	B	C
0	*1*	*2*	*3*	*4*	*5*
NT: 2 **NNT: 4**					

- NForm = Registro
  - Form:TIndSim;
  - PosEsp:Array [1..5] of TIndSim;

  Fin_Registro

NForm es un registro para guardar la información de una regla, si es de la forma 1 $A \rightarrow BC$ o de la forma 2 $A \rightarrow \alpha$ y además las posiciones en las que están ubicados los espacios entre símbolos en las reglas; Form es un campo para guardar la información del tipo de reglas, 1 para la forma $A \rightarrow BC$ y 2 para la forma $A \rightarrow \alpha$ y PosEsp es un arreglo para guardar la información de la ubicación de los espacios, para una regla de la forma $A \rightarrow BC$ se tendría en su orden, $A1 \rightarrow 2B3C$ y para una regla de la forma 2 $A1 \rightarrow 2\alpha$, en el caso que la regla ya tenga asignada la probabilidad, se tendría: $A1 \rightarrow 2B3C4 \rightarrow 5Pb$ y para una regla de la forma 2 $A1 \rightarrow 2a3 \rightarrow 4Pb$ (los números representan los espacios).

Por otro lado, la representación que se hará para las reglas va a estar clasificada con respecto a los antecedentes y consecuentes comunes, además de la forma a la cual pertenecen, pues será de vital importancia para el acceso desde los algoritmos cuando se necesita solamente trabajar con cierta parte de las mismas.

Registros y arreglos referentes a antecedentes comunes

- TRCPb = Registro
  - TAC: Arreglo [0..1] de TIndSim
  - Pb: TProb
  - EC:Byte
  - Ord: TIndReg

  Fin_Registro

- TRCPb2 = Registro
  - TAC2: TIndSim
  - Pb2: TProb
  - EC2:Byte
  - Ord: TIndReg

  Fin_Registro

TRCPb y TRCPb2 son registros que sirven para guardar los consecuentes, la probabilidad y el indicador de convergencia de reglas de la forma uno y dos respectivamente, los arreglos de consecuentes TAC y TAC2 guardan los índices en dónde está el símbolo en el registro TRDatos; Pb y Pb2 se refieren a la probabilidad que tiene asignada la regla a la que se hace referencia, EC junto a EC2 serán utiles para el caso en que se estime la probabilidad de la gramática, en donde el valor de 1 indicará la convergencia y el valor cero indica que el valor no se alcanza todavía y Ord es útil para conocer el índice de las reglas.

Ejemplo 6.14. *Si se está trabajando con el consecuente B y la regla 2, es decir:* $B \rightarrow BA$.

TRCPb	
TAC	
5	4
Pb: 0.5	
EC: 0	

Ejemplo 6.15. *Si se está trabajando con el consecuente* β *y la regla 5, es decir:* $B \rightarrow \beta$.

TRCPb2
TAC2:1
Pb2: 0.2
EC2: 0

Registros para guardar reglas

- TARaAC1: Arreglo [0..NMRaAC] de TRCPb.
- TARaAC2: Arreglo [0..NMRaAC] de TRCPb2.

TARaAC1 y TARaAC2, son arreglos que representan respectivamente a las reglas con antecedente común de la forma uno y dos del tipo TRCPb y TRCPb2.

- TRRaAC = Registro
  - AC: TIndSim
  - NRaAC1, NRaAC2: TIndReg
  - AAC1:TARaAC1
  - AAC2:TARaAC2

  Fin_Registro

TRRaAC es un registro útil para guardar las reglas con un mismo antecedente, donde AC representa el índice del antecedente común en TRDatos, NRaAC1 y NRaAC2 son respectivamente el número de reglas con antecedente común de la forma 1 y el número de reglas con antecedente común de la forma 2, además de AAC1 y AAC2 que representan arreglos de registros cuyo antecedente es común de la forma uno y dos.

Ejemplo 6.16. *En este caso haciendo referencia al antecedente común A.*

TRRaAC		
AC	**NRaAC1/ NRaAC2**	**AAC1/ AAC2**
4	1/1	(5,4) / (1,) 0.5/0.5 0/0

- RTARRaAC = Registro
  - TARRaAC: Arreglo [0..MNNT] de TRRaAC
  - NAC: Byte

  Fin_Registro

RTARRaAC es un registro de reglas cuyo antecedente es común, TARRaAC es un arreglo de registros de reglas con antecedente común y NAC caracteriza al número de antecedentes comunes; es decir un registro con las mismas características que el anterior donde además se representan todos los antecedentes comunes.

Ejemplo 6.17. *En la primera columna estan representados por sus símbolos todos los antecedentes comunes del ejemplo principal.*

RTaRRaAC		
AC	**NRaAC1/ NRaAC2**	**AAC1/ AAC2**
3	2/0	(4,5) / (5,6) ⋮
4	1/1	(5,4) / (1,) ⋮
5	1/1	(6,6) / (2,) ⋮
6	2/1	(4,5) / (6,5) ⋮

Registro para guardar la gramática

- GRA = Registro
  - Sim: TRDatos
  - Reglas: RTARRaAC

  Fin_Registro

GRA es un registro para guardar la gramática, Sim es útil para conocer los símbolos guardados en TRDatos que como ya se sabe está constituido por los terminales y no terminales que forman la gramática y Reglas es un registro de arreglos de reglas con antecedente común de RTARRaAC.

Registros y arreglos referentes a consecuentes comunes

- Info = Array[1..2] de TIndReg.

Info es un arreglo útil para guardar la información de la ubicación de la regla en el arreglo de antecedentes comunes.

- TRGICC = Registro
  - P,S: TIndSim
  - Q: Info
  - R: Byte

  Fin_Registro

TRGICC es un registro para guardar la información referente a un consecuente común de la forma uno, P es el índice del antecedente correspondiente a la regla con ese consecuente común, S representa el índice del consecuente que lo acompaña, además, Q es un arreglo donde se guarda la información en la que está ubicada la regla a la que se hace referencia en el registro de reglas con antecedente común (RTARRAaC), en Q[1] se guarda la posición en la que está ubicada el antecedente de la regla (TARRaAC), en Q[2] se guarda la posición de la regla en el arreglo de registros cuyo antecedente es común (AAC1) y R guarda la posición en la que se encuentra el consecuente (primera o segunda).

Ejemplo 6.18. *En este caso se hace referencia al consecuente común B en la regla 2:* $A \rightarrow BA$.

TRGICC	
Q	
2	1
P: 4	
R: 2	
S: 5	

- TRGICC2 = Registro
  - P2: TIndSim
  - Q2: Info

  Fin_Registro

TRGICC2 es un registro para guardar la información referente al consecuente común de la forma dos, P2 es el índice del antecedente correspondiente a la regla con el consecuente común (CC), Q es un arreglo donde se guarda la información en la que está ubiccada la regla a la que se hace referencia en el registro de reglas con antecedente común

(RTARRAaC), en Q[1] se guarda la posición en la que está ubicada el antecedente de la regla (TARRaAC), en Q[2] se guarda la posición de la regla en el arreglo de registros cuyo antecedente es común (AAC2).

Ejemplo 6.19. *En este caso se hace referencia al consecuente común α en la regla 3.*

TRGICC	
Q2	
2	1
P2: 4	

- TARaCC: Arreglo [0..NMRaCC] de TRGICC.

- TARaCC2: Arreglo [0..NMRaCC] de TRGICC2.

TARaCC y TARaCC2, son arreglos que representan respectivamente a las reglas con consecuente común de la forma uno y dos.

- TRRaCC = Registro
  - CC: TIndSim
  - NRaCC: TIndReg
  - ArR:TARaCC

  Fin_Registro
- TRRaCC2 = Registro
  - CC2: TIndSim
  - NRaCC2: TIndReg
  - ArR2:TARaCC2

  Fin_Registro

TRRaCC y TRRaCC2 son registros útiles para guardar las reglas con un mismo consecuente, para reglas de la forma uno y dos respectivamente, donde CC y CC2 representan el índice del consecuente común, NRaCC y NRaCC2 son respectivamente el número

de reglas con consecuente común, además de ArR y ArR2 que representan arreglos de registros cuyo consecuente es común, del tipo TARaCC y TARaCC2, es decir, representan los campos ya mencionados anteriormente en su órden: P,Q,R y S.

Ejemplo 6.20. *En este caso se hace referencia al consecuente común A y sus reglas asociadas: 0, 2 y 6.*

TARRaCC		
CC	**NRaCC**	**ArR**
4	3	(3,[1,1],1,6) (4,[2,1],2,5) (6,[4,1],1,5)

Ejemplo 6.21. *Sea α el consecuente común y sus reglas asociadas: 3 y 7.*

TARRaCC2		
CC2	**NRaCC2**	**ArR2**
2	1	(5,[3,1])

- RTARRaCC = Registro
  - TARRaCC: Arreglo[1..MNNT] de TRRaCC
  - TARRaCC2: Arreglo[1..MNT] de TRRaCC2
  - NCC: Byte

  Fin_Registro

RTARRaCC es un registro de reglas con consecuente común, donde TARRaCC y TARRaCC2 son arreglos de registros de reglas con consecuente común de la forma uno y dos respectivamente y NCC representa el número de consecuentes comunes.

Ejemplo 6.22. *Se hace referencia a B como consecuente común.*

RTARRaCC				
TARRaCC				
1	2	3	...	MNNT
NCC: 5				

Registros y arreglos referentes a los símbolos de la cadena

- TRCadena = Registro
  - TACad: Arreglo[0..MNSC] de TIndSim
  - NSC:TIndSC
  - Prob: TProb

  Fin_Registro

TRCadena es un registro para guardar la información de una cadena, TACad es un arreglo útil para guardar los índices de los símbolos de la cadena, NSC guarda el número de símbolos de la cadena y Prob guarda la probabilidad de generar la cadena.

Ejemplo 6.23. *Refiréndose a la cadena del ejemplo principal:* $\beta\alpha\alpha\beta\alpha$.

TRCadena				
TACad				
β	α	α	β	α
NSC: 5				

- TRCads = Registro
  - ACads: Arreglo[0..MNCads] de TRCadena
  - NCad: Entero

  Fin_Registro

TRCads es un registro útil para guardar un conjunto de cadenas, para lo cual se usa ACads como un arreglo desde uno hasta el máximo número de cadenas y donde NCad representa el número de cadenas.

Ejemplo 6.24. *Refiriéndose a la cadena del ejemplo principal: $\beta\alpha\alpha\beta\alpha$.*

TRCadS				
ACadS				
β	α	α	β	α
NCads: 1				

- Matriz = Arreglo de Arreglo de Arreglo de Tprob

Matriz es un arreglo tridimensional para guardar las probabilidades de que una subcadena o cadena sea generada a partir de los antecedentes.

Ejemplo 6.25. *Cada "lámina"de la matriz representa a un antecedente, en este caso, al símbolo inicial de la gramática S, y a los símbolos A,B y C, ahora bien, en cada posisión de la matriz M(i,j,k) se encuentran las probabilidades de que cada no terminal genere la subcadena asociada a esas posiciones.*

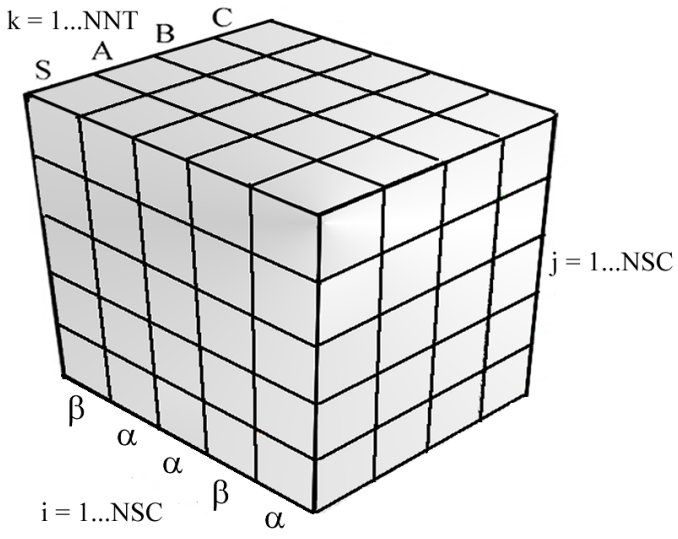

Figura 3.1.4 Matriz Tridimensional

Arreglos auxiliares para el algoritmo CYK2

- TRInf = Registro
  - AInf: Info
  - NArr: Byte

  Fin_Registro

TRInf es un registro útil para guardar la información correspondiente a la regla en la que se está trabajando.

Ejemplo 6.26. *Para la regla 2 se tiene:*

TRInf	
AInf	
2	1
NArr: 2	

- ASubArb = Arreglo de Arreglo de TRInf

AsubArb es un arreglo bidimensional para guardar la información de todas las posibles formas de generar un mismo subárbol.

- Pol = Arreglo de Arreglo de TIndReg

Campo para guardar los exponentes de los monomios que se generan.

- TRCorPol = Record
  - Arb: ASubArb
  - Mono: Pol

  Fin_ Registro

Registro para guardar la información de los árboles de derivación de una cadena y el polinomio que se genera.

- MatCorA = Arreglo de TRCorPol

Arreglo para guardar todos los árboles de derivación y los monomios que se generan cuando se introduce un corpus.

- AMatrizCYK2 = Arreglo de Arreglo de ASubArb

AMatrizCYK2 es una matriz que representa los subárboles de derivación que caracterizan a las reglas. Cada posición de la matriz CYK2(i,j) representa un arreglo en dos dimensiones de subárboles en los que se manejan diferentes combinaciones para cada una de ellas, dependiendo del subárbol que se quiera generar; el de esta matriz es tomar los árboles de derivación de la última casilla y eliminar aquellos que no provengan del símbolo inicial de la gramática, para el ejemplo sería el símbolo S, lo cual es útil para calcular $Pr(x|G_p)$.

Ejemplo 6.27. *Las posiciones (1,1) y (1,2) de la matriz CYK2 están formadas por un subárbol de la misma, los números que se ubican en estas posiciones son los que corresponden a las reglas que derivan en una subcadena, en este caso formada por los dos primeros elementos β y α de la cadena cadena $x = \beta\alpha\alpha\beta\alpha$; es decir, para la primera fila siempre se tendrán reglas de la forma uno $A \to \alpha$, esto es, para la posición (1,1) está ubicado el número 5 que correponde a la regla 5: $B \to \beta$, en la posición (1,2) se encuentran los números 3 y 7 que corresponden a las reglas: $A \to \alpha$ y $C \to \alpha$ respectivamente, ahora bien, para las segunda fila se tendrán reglas de la forma dos: $A \to BC$, en la posición (2,1) se ubica la regla o reglas correspondientes a las posibles combinaciones entre los elementos de (1,1) y (1,2), es decir, aquellas reglas (si existen) cuyas derivaciones sean estos elementos en su órden, es por eso que se hallan los números 2 y 1 pues la regla 2: $A \to BA$ donde B es la raíz de la regla 5: $B \to \beta$ y A la raíz de la regla 3: $A \to \alpha$, de la misma manera la regla 1: $S \to BC$ donde B es la raíz de la regla 5: $B \to \beta$ y C es la raíz de la regla 7: $A \to \alpha$; y de la misma manera para cada una de las posiciones de la matriz CYK2. Ahora, como se puede notar en la posición (5,1) se tienen cinco árboles de derivación para la cadena completa x, donde 4 de ellos inician con las regla uno, es decir con $S \to BC$, esto quiere decir que el símbolo inicial S genera la cadena $\beta\alpha\alpha\beta\alpha$.*

β	α	α	β	α
5	3 7	3 7	5	3 7
1(5,7) 2(5,3)	4(7,7)	0(3,5) 6(3,5) 8(7,5)	1(5,7) 2(5,3)	
ϕ	4(7,6(3,5)) 4(7,8(7,5))	4(6,3(5,7)) 4(8,7(5,7))		
ϕ	2(4(7,6(3,5),3) 2(4(7,8(7,5),3) 1(4(7,6(3,5),7) 1(4(7,8(7,5),3) 8(7(4,6(3,5),7) 8(7(4,8(7,5),7) 0(3(4,6(3,5),7) 0(3(4,8(7,5),7) 6(3(4,6(3,5),7) 6(3(4,8(7,5),7) 2(4(7,7(2,5),3)			
1(5,6(3,4,6(3,5),7)) 1(5,6(3,4,8(7,5),7)) 2(5,2(4,7,7(2,5),3)) 1(5,8(7,4,6(3,5),7)) 1(5,8(7,4,8(7,5),7))				

Figura 6.1: Ejemplo para la matriz CYK2.

Apéndice B

Análisis de algoritmos previos

En este apéndice se presenta el análisis de los distintos órdenes de complejidad de los algoritmos necesarios para el desarrollo de los algoritmos básicos de este trabajo de grado, para ello se presentan las siguientes convenciones:

Símbolo	**Representa**
c	Longitud de la cadena
M	Longitud de la matriz
m_i	Posiciones en la matriz
o	Número de símbolos
p	Número de terminales
q	Número de no terminales
n	Número de reglas
r	Longitud de la regla
r_p	Longitud de la regla con probabilidad
s	Low(M[i,j])
t	Low(M[i])
u	Low(M)
n_1	NRaAC1
n_2	NRaAC2
S_1	Subárbol
N_r	$n_1 + n_2$
A	Longitud del árbol
Ai	Longitud del subárbol

Algoritmos para la entrada de datos

- **Procedimiento OrdGraMm**

Tabla de variables		
Nombre	**Tipo**	**Uso**
Mem	TMemo	Conjunto de líneas de donde se toman las reglas a ordenar.
AuxNT	Entero	Auxiliar para el número de terminales.
AuxNNT	Entero	Auxiliar para el número de no terminales.
AuxNR	Entero	Auxiliar para el número de reglas.
AuxNSim	Entero	Auxiliar para el número de símbolos.
NRegs	Entero	Primera regla de la gramática.
AuxAnt	Cadena	Auxiliar para el antecedente correpondiente.
Ant	Cadena	Antecedente de la j-ésima posición.
Ant2	Cadena	Antecedente de la (j-1)-ésima posición.
i	Entero	Contador para los terminales.
j	Entero	Contador para buscar las reglas con el antecedente correspondiente al no terminal ubicado en la i-ésima posición.
k	Entero	Contador para las reglas.

Pseudocódigo del Procedimiento OrdGraMm

Inicio	**OE**
AuxNT ← Auxiliar correpondiente al número de terminales;	1
AuxNNT ← Auxiliar correspondiente al número de no terminales;	1
AuxNSim ← NT + NNT;	2
AuxNR ← Auxiliar correspondiente al número de reglas;	1
NRegs ← AuxNSim + 3;	2
k ← NRegs;	1
Para i ← AuxNT + 2 **Hasta** AuxNSim **Hacer**	2
Inicio1	
AuxAnt ← Antecedente correspondiente + ‘ ’;	2
Para j ← NRegs **Hasta** AuxNSim + AuxNR + 2 **Hacer**	3
Inicio2	
Ant ← Antecedente de la regla en la j-ésima posición;	7
Si AuxAnt = Ant **Y** j = NRegs **Entonces**	3
Inc(k)	1
Sino	

```
16.        Si AuxAnt = Ant Y j > NRegs Entonces                          3
17.        Inicio3
18.        Ant2 ← Antecedente de la regla en la (j-1)-ésima posición;    7
19.              Si Ant = Ant2 Entonces                                  1
20.                Inc(k);                                               1
21.              Si No
22.                Si Ant <> Ant2 Entonces                               1
23.                  Inicio4
24.                    Intercambia las filas de las posiciones k y j;    1
25.                    Inc(k);                                           1
26.                  Fin4
27.            Fin3
28.      Fin2
29.      NRegs ← k;                                                      1
30.   Fin1
Fin
```

Nota: En las líneas 12 y 18 el algoritmo hace uso de las funciones copiar y longitud, además de cinco operaciones elementales mas, para un total de siete operaciones elementales en cada una de estas líneas mencionadas.

Líneas	Número de operaciones
18-26	11
11-28	24
10-28	$24(o+n+2)+3$
9-28	$24o+24n48+5$
7-28	$24o^2+24no+53o+3$
1-30	$24o^2+24no+53o+12$
$f(n)=$	$24o^2+24no+53o+12$

Cuadro 6.2: Cantidad de operaciones del procedimiento OrdGraMm

La función de complejidad del procedimiento OrdGraMm es $O(n^2)$.

- **Función IndiceSim**

Tabla de variables		
Nombre	**Tipo**	**Uso**
RDat	TRDatos	Acceso a los símbolos guardados.
Cad	Cadena	Cadena en la que se está evaluando la función.
ok	Byte	Indicador que determina si el símbolo buscado es terminal (0) o no terminal (1).
k	Entero	Contador para los símbolos.

Pseudocódigo de la Función IndiceSim

		OE
Inicio		
1.	**Si** ok = 0 **Entonces**	1
2.	**Inicio1**	
3.	k ← 0;	1
4.	**Repetir**	
5.	Inc(k);	1
6.	**Hasta que** Se encuentre el símbolo;	1
7.	IndiceSim ← k;	1
8.	**Fin1**	
9.	**Sino**	
10.	**Inicio2**	
11.	k ← NT;	1
12.	**Repetir**	
13.	Inc(k);	1
14.	**Hasta que** Se encuentre el símbolo;	1
15.	IndiceSim ← k;	1
16.	**Fin2**	1
Fin		

La función de complejidad de la función IndiceSim es $O(n)$.

Líneas	Número de operaciones
4-6	$3q + 2$
1-8	$3q + 5$
1-16	$3q + 5$
$f(n) =$	$3q + 5$

Cuadro 6.3: Cantidad de operaciones de la función IndiceSim

- **Función IdForm**

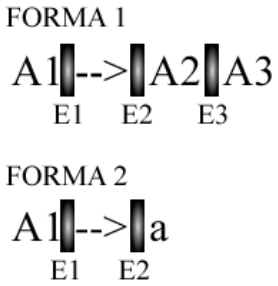

Figura 6.2: Espacios de las reglas.

Tabla de variables		
Nombre	**Tipo**	**Uso**
CadReg	Cadena	Regla a la que se quiere determinar la forma.
AuxNForm	NForm	Indicador para el tipo de forma.
Pos1	Entero	Posición del primer elemento de la regla.
AuxPos	Entero	Posición del último elemento de la regla.

Pseudocódigo de la Función IdForm

Inicio **OE**

1. Pos1 ← 1; 1
2. AuxPos ← Posición del último elemento de la regla; 1
3. **Repetir** 1
4. Inc(Pos1); 1
5. **Hasta que** Se encuentre un espacio vacío; 1
6. **Repetir** 1

```
7.    Dec(Pos1);                                                    1
8.  Hasta que Se encuentre un espacio vacío;                        1
9.  Si (Pos1 + 4) = AuxPos Entonces                                 2
10.   Inicio1                                                       1
11.     Form ← 2;                                                   1
12.     PosEsp[1] ← Pos1;                                           1
13.     PosEsp[2] ← AuxPos;                                         1
14.   Fin1                                                          1
15. Sino
16.   Inicio2                                                       1
17.     Form ← 1;                                                   1
18.     PosEsp[1] ← Pos1;                                           1
19.     PosEsp[2] ← Pos1 + 4;                                       2
20.     PosEsp[3] ← AuxPos;                                         1
21.   Fin2                                                          1
22. IdForm ← AuxNForm;                                              1
Fin
```

Líneas	Número de operaciones
8-19	5
3-7	$4r + 2$
1-21	$4r + 10$
$f(n) =$	$4r + 10$

Cuadro 6.4: Cantidad de operaciones de la función IdForm

De acuerdo con la tabla anterior $f(n) = 4r + 10$. Ahora, $r \leq 9$ pues representa la longitud de la regla, luego $4r + 10 \leq 4(9) + 10 = 46$, por lo tanto la función de complejidad del procedimiento IdForm es $O(1)$.

- **IdFormPr**

Figura 6.3: Espacios y probabilidades de las reglas.

Tabla de variables		
Nombre	**Tipo**	**Uso**
CadReg	Cadena	Regla a la que se quiere determinar la forma.
AuxNForm	NForm	Indicador para el tipo de forma.
Pos1	Entero	Posición del primer elemento de la regla.
AuxPos	Entero	Posición del último elemento de la regla.

Pseudocódigo de la función IdFormPr

Inicio	**OE**
1. Pos1 ← 1;	1
2. AuxPos ← Posición del último elemento de la regla;	1
3. **Repetir**	
4. Inc(Pos1);	1
5. **Hasta que** Se encuentre un espacio vacío;	1
6. PosEsp[1] ← Pos1;	1
7. Pos1 ← Pos1 + 4;	2
8. PosEsp[2] ← Pos1;	1
9. **Repetir**	
10. Inc(Pos1);	1
11. **Hasta que** Se encuentre un espacio vacío;	1
12. PosEsp[3] ← Pos1;	1
13. **Repetir**	
14. Dec(AuxPos);	1

15. **Hasta que** Se encuentre un espacio vacío; 1
16. **Si** (AuxPos - 4) = Pos1 **Entonces** 2
17. **Inicio1**
18. Form ← 2; 1
19. PosEsp[4] ← AuxPos; 1
20. **Fin1**
21. **Sino**
22. **Inicio2** 1
23. Form ← 1; 1
24. PosEsp[4] ← AuxPos - 4; 2
25. PosEsp[5] ← AuxPos; 1
26. **Fin2**
27. IdFormPr ← AuxNForm; 1

Fin

Líneas	Número de operaciones
16-26	4
9-15	$4r_p + 3$
1-8	$2r_p + 6$
1-27	$6r_p + 14$
$f(n) =$	$6r_p + 14$

Cuadro 6.5: Cantidad de operaciones de la función IdFormPr

De acuerdo con la tabla anterior $f(n) = 6r_p + 14$. Ahora, $r \leq 15$ pues representa la longitud de la regla, luego $6r_p + 14 \leq 6(15) + 14 = 94$, por lo tanto la función de complejidad del procedimiento IdFormPr es $O(1)$.

Algoritmos referentes a los árboles de derivación de una cadena

- **Procedimiento unión**

Tabla de variables		
Nombre	**Tipo**	**Uso**
SubArb1, SubArb2	ASubArb	Subárboles que van a ser unidos
AuxSub	ASubArb	Auxiliar de los subárboles
Sb1, Sb2	TIndReg	Guardar longitud de elementos de cada areglo
AuxLenght	TIndReg	Auxiliar de la longitud del nuevo subárbol
i,j	Entero	Contadores para la longitud de los elementos

Pseudocódigo del procedimiento unión

Inicio

1. Sb1 ← Número de elementos que hay en el arreglo SubArb1; 1
2. Sb2 ← Número de elementos que hay en el arreglo SubArb2; 1
3. Auxlength ← Sb1 + Sb2 + 1; 3
4. Se dimensiona la fila del nuevo subárbol con Auxlength; 1
5. En la primera posición del nuevo subárbol se adiciona la regla que une
6. a Sb1 y Sb2; 1
7. **Para** i ← 1 **Hasta** Sb1 **Hacer** 1
8. AuxSub ← Sb1; 1
9. **Para** j ← Sb1 + 1 **Hasta** Auxlength − 1 **Hacer** 3
10. AuxSub ←Sb2; 1

Fin

Líneas	Número de operaciones
7-8	$s1$
1-8	$s1 + 8$
1-10	$s1 + s2 + 11$
$f(n)$=	$s1 + s2 + 11$

Cuadro 6.6: Cantidad de operaciones del procedimiento Unión

La función de complejidad del procedimiento Unión es $O(n)$.

- **Función CYK2**

Tabla de variables		
Nombre	**Tipo**	**Uso**
Cad	TRCad	Cadena a la que se le va a determinar su pertenencia a la gramática
G	GRA	Gramática para conocer sus reglas y probabilidades
ACC	RTARRaAC	Conocer símbolos y consecuentes comunes de las reglas de la forma 1
ACC2	RTARRaAC2	Conocer símbolos y consecuentes comunes de las reglas dela forma 2
M	AMatrizCYK	Arreglo bidimensional donde se guardan las reglas que derivan en la cadena
c	TIndSim	Cantidad de símbolos de la cadena
AuxC1	TIndSim	Auxiliar para guardar la raíz del subárbol más a la izquierda
AuxC2	TIndSim	Auxiliar para guardar la raíz del subárbol más a la derecha
AuxPosCC	TIndSim	Auxiliar para guardar la posición del primer consecuente en el arreglo de consecuentes comunes
DimR2	TIndReg	Asignar el número de reglas con el respectivo consecuente
AuxOp	TIndReg	Contador para el número de posibilidades de generar la subcadena
AuxNRaCC	TIndReg	Contador para el número de reglas con el respectivo consecuente común
TamSubA	TIndReg	Determina el tamaño de cada subárbol que se genera en la matriz
AuxFin	ASubArb	Guardar los árboles que derivan en la cadena

Pseudocódigo de la Función CYK2

Inicio **OE**

1. c ← Número de símbolos de la cadena; 1
2. Setlength(Matriz,c); 1

```
3.  Para i ← 0 Hasta c−1 Hacer                                          2
4.    Setlength(Matriz[i],c-i);                                         2
5.  Para j ← 0 Hasta c−1 Hacer                                          2
6.  Inicio1
7.    DimR2 ← Número de reglas asociadas al j-ésimo consecuente;        3
8.    Dimensionar el arreglo interno con una columna y tantas
      filas como consecuentes hayan;                                    1
9.    Para k←0 Hasta DimR2 -1 Hacer                                     2
10.   Inicio2
11.     M.Ainf ← Posición en el arreglo de antecedentes comunes;        5
12.     M.NArr ← Arreglo para antecedentes de la forma 2;               2
13.   Fin2
14. Fin1
15.   Para l ← 1 Hasta c − 1 Hacer                                      2
16.   Inicio3
17.     Para m ← 0 Hasta c−(l+1) Hacer                                  3
18.     Inicio4
19.       q ← l+1;                                                      2
20.       r ← m+1;                                                      2
21.       AuxOp ← 1;                                                    1
22.       Para n ← 0 Hasta l−1 Hacer                                    2
23.       Inicio5
24.         Si (Length(M[n,m])>0) Y (Length(M[q,r])>0) Entonces         5
25.         Inicio6
26.           Para p ← 0 Hasta Length(M[n,m])−1 Hacer                   3
27.           Inicio7
28.             Para s ← 0 Hasta Length(M[q,r])−1 Hacer                 3
29.               Inicio8
30.                 AuxC1 ← Raíz del subárbol más a la izquierda;       5
31.                 AuxC2 ← Raíz del subárbol más a la derecha;         5
32.                 AuxPosCC ← Posición del primer consecuente en el ACC; 4
33.                 AuxNRaCC ← Número de reglas con el CC;              3
34.                 Si AuxNRaCC > 0 Entonces;                           1
35.                 Para t ← 1 Hasta AuxNRaCC Hacer                     1
36.                 Inicio9
```

```
37.                    Si La regla puede formar el subárbol Entonces          9
38.                    Inicio10
39.                      SetLength(Matriz[l,m],AuxOp);                        1
40.                      Unión(Sb1,Sb2,AC);                                   n
41.                      Inc(AuxOp);                                          1
42.                    Fin10
43.                  Fin9
44.                Fin8
45.              Fin7
46.            Fin6
47.            Dec(q);                                                        1
48.            Inc(r);                                                        1
49.          Fin5
50.        Fin4
51.      Fin3
52.      TamSubA ← 0;                                                         1
53.      Para v ← 0 Hasta Length(M[c−1,0]) − 1 Hacer                          3
54.      Si AC = NT + 1 Entonces                                              2
55.      Inicio11
56.        Inc(TamSubA);                                                      1
57.        SetLength(AuxFin,TamSubA);                                         1
58.        AuxFin[TamSubA−1]← M[c−1,0,v];                                     1
59.      Fin11
60.    CYK2 ← AuxFin;                                                         1
Fin
```

Notese que para las líneas 15-51 se parte de $[c-(l+1)]l[(S1S2nr+11S1S2r+19S1S2+3S1+8)+9]$ luego, denotando con la letra D la exprexión [(S1S2nr +11S1S2r + 19S1S2 + 3S1 + 8) + 9], se tiene $D[c-(l+1)]l$, ahora, la expresión inicial incluyendo al ciclo ***Para*** se puede ver como sigue : $D\sum_{c-1}^{l=1}(cl-l^2-l) = D[c\sum_{c-1}^{l=1}(l)-\sum_{c-1}^{l=1}(l^2)-\sum_{c-1}^{l=1}(l)] = D[c(\frac{(c-1)c}{2})-(\frac{(c-1)c(2(c-1)+1)}{6})-(\frac{(c-1)c}{2})] = D[\frac{1}{2}c^3-\frac{1}{2}c^2-\frac{(c^2-c)(2c-1)}{6}-\frac{1}{2}c^2+\frac{1}{2}c] = D[\frac{1}{2}c^3-\frac{1}{2}c^2-(\frac{1}{3}c^3-\frac{1}{6}c^2-\frac{1}{3}c^2+\frac{1}{6}c)-\frac{1}{2}c^2+\frac{1}{2}c] = D[\frac{1}{2}c^3-\frac{1}{2}c^2-(\frac{1}{3}c^3-\frac{1}{3}c^2+\frac{1}{6}c)-\frac{1}{2}c^2+\frac{1}{2}c] = D[\frac{1}{6}c^3-\frac{2}{3}c^2+\frac{2}{3}c]$.

Líneas	Número de operaciones
1-2	2
3-4	$c+2$
5-8	$4c+2$
9-14	$7r+2$
37-42	$n+11$
35-43	$r(n+11)+1$
30-43	$r(n+11)+19$
28-44	$S2(nr+11r+19)+3$
26-46	$S1(S2nr+11S2r+19S2+3)+3$
24-46	$(S1S2nr+11S1S2r+19S1S2+3S1+3)+5$
19-49	$l(S1S2nr+11S1S2r+19S1S2+3S1+8)+9$
17-50	$[c-(l+1)]l[(S1S2nr+11S1S2r+19S1S2+3S1+8)+9]$
15-51	$[\frac{1}{6}c^3-\frac{2}{3}c^2+\frac{2}{3}c][S1S2nr+11S1S2r+19S1S2+3S1+17]$
15-52	$[\frac{1}{6}c^3-\frac{2}{3}c^2+\frac{2}{3}c][S1S2nr+11S1S2r+19S1S2+3S1+17]+1$
1-60	$[\frac{1}{6}c^3-\frac{2}{3}c^2+\frac{2}{3}c][S1S2nr+11S1S2r+19S1S2+3S1+17]+5c+7r+5S+13$
$f(n)=$	$[\frac{1}{6}c^3-\frac{2}{3}c^2+\frac{2}{3}c][S1S2nr+11S1S2r+19S1S2+3S1+17]+5c+7r+5S+13$

Cuadro 6.7: Cantidad de operaciones de la función CYK2

La función de complejidad de la función CYK2 es n^3.

- **Función MejorDer**

Tabla de variables		
Nombre	**Tipo**	**Uso**
Arb	ASubArb	Arbol de derivación de la cadena
G	GRA	Gramática para conocer sus reglas y probabilidades
Aux1	TProb	Auxiliar para guardar la probabilidad de la regla en la forma 2
Aux2	TProb	Auxiliar para guardar la probabilidad de la regla en la forma 1
Mejor	TIndReg	Conocer el índice de cada regla
Mj	ASubArb	Guardar la información de todas las posibles formas de generar un mismo subárbol
i	Entero	Contador para la longitud del árbol.
j	Entero	Contador para moverse al interior del árbol.

Pseudocódigo de la Función MejorDer

Inicio	**OE**
1. Aux2 = 0;	1
2. **Para** $i = 0$ **Hasta** Length(Arb)-1 **Hacer**	3
3. **Inicio2**	
4. Aux1 = 1;	1
5. **Para** $j = 0$ **Hasta** Length(Arb[i])-1 **Hacer**	4
6. **Inicio3**	
7. **Si** Arboles[i,j].NArr = 2 **Entonces**	1
8. Aux1 ← Aux1 × Pb2;	1
9. **Sino**	
10. Aux1 ← Aux1 × Pb;	1
11. **Fin3**	
12. **Si** Aux1 > Aux2 **Entonces**	1

```
13.     Inicio4
14.       Aux2 ← Aux1;                                  1
15.       Mejor ← i;                                    1
16.     Fin4
17.   Fin2
18.   Setlength(Mj,1,Length(Arb[Mejor]));               2
19.       Mj[0]← Arb[Mejor];                            3
20.       MejorDer ← Mj;                                1
Fin
```

Líneas	Número de operaciones
12-16	3
6-11	4
6-16	7
4-16	$7Ai + 5$
2-17	$A(7Ai + 5) + 3$
1-20	$A(7Ai + 5) + 10$
$f(n) =$	$7AAi + 5A + 10$

Cuadro 6.8: Cantidad de operaciones de la función MejorDer

La función de complejidad de la función MejorDer es $O(n)$.

- **Función CadInd**

Tabla de variables		
Nombre	**Tipo**	**Uso**
Cad	Cadena	Cadena a la que se le van a determinar sus índices
Dat	TRDatos	Acceso a los símbolos guardados
AuxC	Cadena	Auxiliar para copiar el símbolo de la cadena
NCar1	TIndSim	Posición inicial del símbolo en la cadena
NCar2	TIndSim	Posición final del símbolo en la cadena
P	TIndSim	Símbolo de la cadena
NSim	TIndSim	Número de símbolos
AuxCad	TRCad	Auxiliar para la cadena

Pseudocódigo de la Función CadInd

Inicio	**OE**
1. NCar2 = 0;	1
2. P = 0;	1
3. NSim = 0;	1
4. **Repetir**	
5. Inc(NCar2);	1
6. NCar1 ← NCar2;	1
7. Inc(P);	1
8. **Mientras** (Cad[P] ≠ ' ') **Y** (P ≤ c) **Hacer**	3
9. **Inicio2**	
10. Inc(NCar2);	1
11. Inc(P);	1
12. **Fin2**	
13. AuxC ← Copiar el símbolo al que se busca el índice;	1
14. AuxCad ← Indice del AuxC;	1
15. Inc(NSim);	1
16. **Hasta que** P > c;	1
17. CadInd.TaCad ← AuxCad.TACad;	3
18. CadInd.NSC ← NSim;	2
Fin	

Líneas	Número de operaciones
8-12	$5c + 3$
8-15	$5c + 6$
4-16	$5c^2 + 7c + 1$
1-18	$5c^2 + 7c + 8$
$f(n) =$	$5c^2 + 7c + 8$

Cuadro 6.9: Cantidad de operaciones de la función CadInd

La función de complejidad de la función CadInd es $O(n^2)$.

- **Procedimiento Monom**

Tabla de variables		
Nombre	**Tipo**	**Uso**
ArbyPol	TRCorPol	Registro para guardar la información de los árboles de derivación
G	GRA	Acceso a los símbolos guardados
i	Cadena	Contador para los árboles del registro
j	TIndSim	Contador para el interior de cada árbol del registro

Pseudocódigo del procedimiento Monom

Inicio **OE**

1. Setlength(ArbyPol,NReg); 1
2. **Para** i $\leftarrow$ 0 **Hasta** Length(ArbyPol.Arb) $-$ 1 **Hacer** 1
3. **Para** j $\leftarrow$ 0 **Hasta** Length(ArbyPol.Arb[i]) $-$ 1 **Hacer** 1
4. **Inicio1**
5. **Si** Es una regla de la forma 2 **Entonces** 3
6. **Inicio2**
7. Buscar el índice de la regla en AAC2 **E** Incrementar
8. el arreglo que guarda los exponentes de los monomios generados; 10
9. **Fin2**
10. **Sino**
11. **Inicio3**
12. Buscar el índice de la regla en AAC1 **E** Incrementar
13. el arreglo que guarda los exponentes de los monomios generados; 10
14. **Fin3**
15. **Inicio1**

Fin

La función de complejidad del procedimiento Monom es $O(n)$.

Líneas	Número de operaciones
5-14	13
3-14	$13Ai + 5$
2-14	$A(13Ai + 5) + 4$
1-15	$13AAi + 5A + 5$
$f(n) =$	$13AAi + 5A + 5$

Cuadro 6.10: Cantidad de operaciones de la función Monom

Algoritmos para asignar y determinar las probabilidades de las reglas de una gramática y de las cadenas generadas

- **Procedimiento Dis_Unif**

Tabla de variables		
Nombre	**Tipo**	**Uso**
DU	RTARRaAC	Registro de las reglas con antecedente común.
AuxS	TProb	Auxiliar para la probabilidad de la regla.
i	Entero	Contador para el número de antecedentes comunes.
j	Entero	Contador para el número de reglas de la forma 1.
k	Entero	Contador para el número de reglas de la forma 2.

Pseudocódigo del procedimiento Dis_Unif

```
Inicio                                                    OE
1.  Para i ← 1 Hasta DU.NAC Hacer                          2
2.    Inicio1
3.      AuxS ← 1 / (NraAC1 + NraAC2);                      3
4.        Para j ← 1 Hasta NraAC1 Hacer                    1
5.          Pb ← AuxS;                                     1
6.        Para k ← 1 Hasta NraAC2 Hacer                    1
7.          Pb2 ← AuxS;                                    1
8.    Fin1
Fin
```

La función de complejidad del procedimiento Dis_Unif es $O(n^2)$.

Líneas	Número de operaciones
6-7	$n_2 + 1$
4-7	$n_1(n_2 + 1) + 1$
2-7	$n_1 n_2 + n_1 + 4$
1-8	$n_1 n_2 q + n_1 q + 4q + 1$
$f(n)$=	$n_1 n_2 q + n_1 q + 4q + 1$

Cuadro 6.11: Cantidad de operaciones del procedimiento Dis_Unif

- **Procedimiento Dis_Ale**

Tabla de variables		
Nombre	**Tipo**	**Uso**
DA	RTARRaAC	Registro de las reglas con antecedente común.
AuxS	TProb	Auxiliar para la probabilidad de la regla.
x	TProb	Auxiliar para guardar el valor que genera la función random.
i	Entero	Contador para el número de antecedentes comunes.
j	Entero	Contador para el número de reglas de la forma 1.
k	Entero	Contador para el número de reglas de la forma 2.

Pseudocódigo del Procedimiento Dis_Ale

Inicio	**OE**
Para i← 1 **Hasta** DA.NAC **Hacer**	2
Inicio1	
Aux1 ← (NraAC1 + NraAC2);	2
AuxS ← 0;	1
Randomize;	1
Inicio2	
Para j ← 1 **Hasta** Aux1 -1 **Hacer**	2
Inicio3	
Repetir	
x ← Random;	1
Hasta que 0 < x **Y** (x < 1-AuxS);	3
AuxS ← AuxS + x	2

```
13.            Si j <= NraAC1 Entonces                    1
14.              Pb ← x;                                  1
15.            Sino
16.              Pb2 ← x;                                 1
17.          Fin3
18.          Si Aux1 > NraAC1 Entonces                    1
19.            Pb2 ← 1 - AuxS;                            2
20.          Sino
21.            Pb ← 1 - AuxS;                             2
22.      Fin2
23.  Fin1
Fin
```

Líneas	Número de operaciones
18-21	3
13-17	2
13-21	5
9-21	$6P_b + 5$
6-21	$N_R(6P_b + 5) + 2$
1-8	$AC[N_R(6P_b + 5) + 2] + 6$
$f(n)$=	$AC[N_R(6P_b + 5) + 2] + 6$

Cuadro 6.12: Cantidad de operaciones del procedimiento Dis_Ale

La función de complejidad del procedimiento Dis_Ale es $O(n)$.

- **Procedimiento Dis_Frec**

Tabla de variables		
Nombre	**Tipo**	**Uso**
Arbol	ASubArb	Registro del subárbol de derivación por cada antecedente.
RegFre	Arreglo de enteros	Número de veces que se repite la regla por cada antecedente.
AuxPb	TProb	Auxiliar para guardar el valor de la probabilidad.
AuxPbSum	TProb	Auxiliar para guardar el valor de la suma de las probabilidades.
Aux2,Aux3, Aux4	TIndReg	Contador para el número de reglas por antecedente común.
i	Entero	Contador para el número de cadenas del corpus.
j	Entero	Contador para los árboles de derivación.
k	Entero	Contador para los campos de las reglas usadas en el árbol de derivación.
m	Entero	Contador para cada antecedente común.
n	Entero	Contador para los antecedentes comunes.
p	Entero	Contador para las reglas de la forma 1.
q	Entero	Contador para las reglas de la forma 2.

Pseudocódigo del Procedimiento Dis_Frec

Inicio	**OE**
1. SetLength(RegFre,G.Reglas.NReg);	1
2. **Para** i← 1 **Hasta** NCads **Hacer**	1
3. **Inicio1**	1
4. Arbol ← CYK2(Cads.ACads[i],G,ACC,ACC2)	c^3
5. **Para** j← 0 **Hasta** (Long(Arbol) − 1) **Hacer**	2
6. **Para** k← 0 **Hasta** (Long(Arbol[j]) − 1) **Hacer**	3
7. **Inicio2**	
8. **Si** Arbol[j,k].NArr = 2 **Entonces**	2
9. Incrementa en el arreglo RegFre el número de veces que aparece la regla correspondiente;	6

```
10.        Sino
11.          Incrementa en el arreglo RegFre el número de veces
             que aparece la regla correspondiente;                    6
12.      Fin2                                                         1
13.  Fin1                                                             1
14.  Aux3 ← 0;                                                        1
15.  Para m← 1 Hasta NAC Hacer                                        1
16.  Inicio3
17.    Aux2 ← NRaAC1 + NRaAC2;                                        2
18.    Si Aux2 = 1 Entonces                                           1
19.    Inicio4
20.      Inc(Aux3);                                                   1
21.      Si G.Reglas.TARRaAC[m].NRaAC1 = 1 Entonces                   5
22.      G.Reglas.TARRaAC[m].AAC1[1].Pb; ← 1                          7
23.      Sino                                                         1
24.      G.Reglas.TARRaAC[m].AAC2[1].Pb2; ← 1                         7
25.    Fin4                                                           1
26.    Sino                                                           1
27.    Inicio5                                                        1
28.      Sum←0;                                                       1
29.      Para n←0 Hasta Aux2 − 1 Hacer                                2
30.        Sum← Sum + RegFre[n + Aux3];                               3
31.  Aux3← Aux3 + Aux2;                                               2
32.  Aux4← 1;                                                         1
33.  AuxPbSum← 0;                                                     1
34.  Para p←1 Hasta G.Reglas.TARRaAC[m].NRaAC1 Hacer                  5
35.  Inicio6
36.    Si Aux4 < Aux2 Entonces                                        1
37.    Inicio7
38.      AuxPb← RegFre[G.Reglas.TARRaAC[m].AAC1[p].Ord]/Sum;          9
39.      G.Reglas.TARRaAC[m].AAC1[p].Pb ← AuxPb;                      7
40.      AuxPbSum← AuxPb + AuxPbSum;                                  2
41.       Inc(Aux4);                                                  1
42.    Fin7
43.    Sino
```

44. G.Reglas.TARRaAC[m].AAC1[p].Pb ← 1 − AuxPbSum; 8
45. **Fin6**
46. **Para** q ← 1 To G.Reglas.TARRaAC[m].NRaAC2 **Hacer** 5
47. **Inicio8** 1
48. **Si** Aux4 < Aux2 **Entonces** 1
49. **Inicio9**
50. AuxPb ← RegFre[G.Reglas.TARRaAC[m].AAC2[q].Ord]/Sum; 9
51. G.Reglas.TARRaAC[m].AAC2[q].Pb2 ← AuxPb; 7
52. AuxPbSum ← AuxPb + AuxPbSum; 2
53. Inc(Aux4); 1
54. **Fin9**
55. **Sino**
56. G.Reglas.TARRaAC[m].AAC2[q].Pb2 ← 1 − AuxPbSum; 8
57. **Fin8** 1
58. **Fin5** 1
59. **Fin3** 1

Fin

Líneas	Número de operaciones
46-57	$20n_2 + 5$
34-45	$20n_1 + 5$
29-33	$7N_r + 2$
16-28	16
16-57	$27N_r + 28$
15-57	$q(27N_r + 28) + 1$
5-14	$A(8A_i + 3) + 2$
4-14	$A(8A_i + 3) + c^3 + 3$
1-59	$NCads[q(27N_r + 28) + A(8A_i + 3) + c^3 + 4]$
$f(n)$=	$NCads[q(27N_r + 28) + A(8A_i + 3) + c^3 + 4]$

Cuadro 6.13: Cantidad de operaciones procedimiento Dis_Fre

La función de complejidad del procedimiento Dis_Fre es $O(n^3)$.

- **Función ProCad**

Tabla de variables		
Nombre	**Tipo**	**Uso**
G	GRA	Gramática para conocer sus reglas y probabilidades
M	MatCorA	Arreglo para guardar todos los árboles de derivación
k	Word	Contador para los elementos de la matriz
i	Integer	Contador para los elementos del árbol
j	Integer	Contador para los elementos del árbol
Prob	TProb	Probabilidad de la cadena
AuxP	TProb	auxiliar para la probabilidad de la cadena

Pseudocódigo de la Función ProCad

Inicio	**OE**
1. AuxP ← 0;	1
2. **Para** i←0 **Hasta** Length(M[k].Arb)-1 **Hacer**	5
3. **Inicio2**	
4. Prob ← 1;	1
5. **Para** j←0 **Hasta** Length(M[k].Arb[i])-1 **Hacer**	6
6. **Inicio3**	
7. **Si** M[k].Arb[i,j].NArr <> 2 **Entonces**	5
8. Prob ← Pb;	1
9. **Sino**	
10. Prob ← Pb2;	1
11. **Fin3**	
12. AuxP ← AuxP + Prob;	2
13. **Fin2**	
14. ProCad ← AuxP;	1
Fin	

La función de complejidad del procedimiento ProCad es $O(n)$.

Líneas	Número de operaciones
6-11	6
5-11	$6n + 6$
3-13	$6n + 7$
2-13	$c(6n + 7) + 5$
1-13	$6cn + 7c + 6$
$f(n) =$	$6cn + 7c + 6$

Cuadro 6.14: Cantidad de operaciones de la función ProCad

Algoritmos referentes a las matrices(Arreglos)

- **Procedimiento Dim_Matriz**

Tabla de variables		
Nombre	**Tipo**	**Uso**
x	Entero	número de elementos de la cadena.
NA	Entero	Número de antecedentes.
i	Entero	Contador para las longitudes de la matriz.
j	Entero	Contador para las longitudes de la matriz.

Pseudocódigo del Procedimiento Dim_Matriz

Inicio	**OE**
1. Setlength (M,x);	1
2. **Para** i ← Low(M) **Hasta** High(M) **Hacer**	3
3. **Inicio1**	
4. Setlength(M[i],i+1);	2
5. **Para** j ← Low(M[i]) **Hasta** High(M[i])**Hacer**	3
6. Setlength(M[i,j],NA);	1
7. **Fin1**	

Fin

La función de complejidad del procedimiento Dim_Matriz es $O(n)$.

Líneas	Número de operaciones
5-6	$m_i + 3$
3-6	$m_i + 5$
2-6	$M(m_i + 5) + 3$
1-7	$M(m_i + 5) + 4$
$f(n) =$	$M(m_i + 5) + 4$

Cuadro 6.15: Cantidad de operaciones del procedimiento Dim_Matriz

- **Procemiento Ini_Matriz**

Tabla de variables		
Nombre	**Tipo**	**Uso**
M	Matriz	Matriz para inicializar.
G	GRA	Tomar las reglas que inicializan la matriz.
Cad	TRCadena	Cadena para dimensioinar la matriz.
AConC2	TARRaCC2	Arreglo de consecuentes comunes.
i	Entero	Contador para las longitudes de la matriz.
j	Entero	Contador para las longitudes de la matriz.
k	Entero	Contador para los antecedentes.

Pseudocódigo del Procemiento Ini_Matriz

Inicio	**OE**
1. Dim_Matriz(Cad.NSC, G.Reglas.NAC, M);	M
2. **Para** i ← Low(M) **Hasta** High(M) **Hacer**	1
3. **Para** j ← Low(M[i]) **Hasta** High(M[i])**Hacer**	1
4. **Si** i <> j **Entonces**	1
5. **Para** k ← Low(M[i,j]) **Hasta** High(M[i,j])**Hacer**	1
6. M[i,j,k] ← 2	1
7. **Para** p ← 0 **Hasta** Cad.NSC -1**Hacer**	2
8. **Para** r ← 1 **Hasta** AConC2[Cad.TACad[p]].NRaCC2 **Hacer**	3
9. M[p,p,AN -1] ← Probabilidad de la regla;	1
Fin	

Líneas	Número de operaciones
8-9	$r+3$
7-9	$c(r+3)+2$
6-9	$cr+3c+3$
5-9	$s(cr+3c+3)+1$
3-9	$t(crs+3cs+3s+2)$
2-9	$u(crst+3cst+3st+2t)$
1-10	$u(crst+3cst+3st+2t)+M$
$f(n)=$	$u(crst+3cst+3st+2t)+M$

Cuadro 6.16: Cantidad de operaciones del procedimiento Ini_Matriz

La función de complejidad del procedimiento Ini_Matriz es $O(n)$.

- **Función Buscar_An**

Tabla de variables		
Nombre	**Tipo**	**Uso**
G	GRA	Conocer los elementos de la regla
An	TIndSim	Antecedente que se desea buscar
AuxPos	TIndSim	Auxiliar para la posición del antecedente buscado

Pseudocódigo del Procemiento Buscar_An

Inicio	**OE**
1. **Si** (An - NT) > NAC **Entonces**	2
2. AuxPos ← NAC	1
3. **Sino**	
4. AuxPos ← (An - NT);	2
5. **Mientras** AC[AuxPos] > An **Y** (AuxPos <> 0) **Hacer**	3
6. Dec(Auxpos);	1
7. **Si** AC[AuxPos] = An **Entonces**	1
8. Buscar_An ← AuxPos;	1
9. **Sino**	
10. Buscar_An ← 0;	1

Líneas	Número de operaciones
1-4	5
5-6	$4NT + 3$
7-10	2
1-10	$4NT + 10$
$f(n) =$	$4NT + 10$

Cuadro 6.17: Cantidad de operaciones procedimiento Buscar_An

Fin

La función de complejidad del procedimiento Buscar_An es $O(n)$.

- **Función OutsideOK**

Tabla de variables		
Nombre	**Tipo**	**Uso**
G	GRA	Acceso a los símbolos guardados
ArrCC	RTARRaCC	Acceso a los consecuentes comunes y su información
Cad	TRCadena	Cadena en la que se evalúa la función
CC	TIndSim	Consecuente común
AuxMCok	TIndSim	Auxiliar de la matriz de consecuentes en el arreglo tridimensional
i	TIndSim	Contador para los índices de la cadena
j	TIndSim	Contador para los no terminales
P	TProb	Guardar la información de la probabilidad outside
PTotal	TProb	Probabilidad outside final

Pseudocódigo de la Función OutsideOK

Inicio	**OE**
1. **Si** CC <> (G.Sim.NT) + 1**Entonces**	4
2. OutsideOk ← 0;	1
3. **Sino**	
4. **Inicio1**	
5. Ini_Matriz(AuxMCok,G,Cad);	c
6. Ini_Matriz2(AuxMAok,G,Cad);	q

```
7.      PTotal ← 0;                                                    1
8.      Para j ← 1 Hasta NNT Hacer                                     1
9.        Si AuxMCok[i,i,j -1] <> 0 Entonces                           2
10.       Inicio2
11.         P ← Outside(G,ArrCC,Cad,NT,AuxMCok,AuxMAok,i,i)
12.               × AuxMCok[i,i,j - 1];                                x^2
13.         PTotal ← P + PTotal;                                       2
14.       Fin2
15.       OutsideOk ← PTotal;                                          1
16.   Fin1
Fin
```

Líneas	Número de operaciones
9-15	$x^2 + 5$
8-15	$q(x^2 + 5) + 1$
4-15	$qx^2 + 6q + c + 2$
1-15	$qx^2 + 6q + c + 6$
f(n)=	$qx^2 + 6q + c + 6$

Cuadro 6.18: Cantidad de operaciones de la función OutsideOk

La función de complejidad del Función OutsideOk es $O(x^2)$

Apéndice C

Experimentos

Gramáticas

Las gramáticas que son objeto de evaluación tienen el formato G(NT,NNT,P), donde NT, NNT Y P representan respectivamente al número de terminales, número de no terminales y número de reglas de producción:

Gramática 1:
La gramática uno tiene dos terminales a y b, además de tres no terminales $A1$, $A2$ y $A3$ y cinco reglas de producción:

A1 → A2 A3
A2 → A3 A1
A2 → b
A3 → A1 A2
A3 → a

Gramática 2:
La gramática dos tiene cinco terminales $t0$, $t1$, $t2$, $t3$ y $t4$, además de doce no terminales $A0$, $A1$, $A2$, $A3$, $A4$, $A5$, $A6$, $A7$, $A8$, $A9$, $A10$ y $A11$ y veinticinco reglas de producción:

A0 → A3 A7
A0 → A2 A11
A0 → A4 A10
A0 → A4 A4
A0 → A5 A0
A0 → A5 A4
A0 → A6 A6
A0 → t2
A1 → A2 A11
A1 → A4 A4
A1 → A5 A0
A1 → A6 A6
A1 → t2
A2 → A1 A9
A2 → A4 A8
A2 → t2
A3 → t3
A4 → t1
A5 → t2
A6 → t4
A7 → A1 A0
A8 → A1 A4
A9 → A2 A0
A10 → A4 A0
A11 → A4 A1

Gramática 2 (Sentences):

La gramática dos (Sentences) tiene cinco terminales *null*, *fly*, *like*, *time* y *arrow* además de doce no terminales *Sentence*, *Verb*, *Object*, *Subject*, *NounPhrase*, *NounGroup*, *Noun*, *PrePhrase*, *Preposition*, *Article*, *Adverb*, y *Conjunction* y veinticinco reglas de producción:

Sentence → Subject PrePhrase
Sentence → Object Conjunction
Sentence → NounPhrase Adverb
Sentence → NounPhrase NounPhrase
Sentence → NounGroup Sentence
Sentence → NounGroup NounPhrase
Sentence → Noun Noun
Sentence → like
Verb → Object Conjunction
Verb → NounPhrase NounPhrase
Verb → NounGroup Sentence
Verb → Noun Noun
Verb → like
Object → Verb Article
Object → NounPhrase Preposition
Object → like
Subject → time
NounPhrase → fly
NounGroup → like
Noun → arrow
PrePhrase → Verb Sentence
Preposition → Verb NounPhras
Article → Object Sentence
Adverb → NounPhrase Sentence
Conjunction → NounPhrase Verb

Gramática 3:

La gramática tres tiene catorce terminales $t0$, $t1$, $t2$, $t3$, $t4$, $t5$, $t6$, $t7$, $t8$, $t9$, $t10$, $t11$, $t12$, y $t13$, además de trece no terminales $A0$, $A1$, $A2$, $A3$, $A4$, $A5$, $A6$, $A7$, $A8$, $A9$, $A10$, $A11$ y $A12$ y cuarentaisiete reglas de producción:

A0 → A2 A14	A7 → t4
A0 → A2 A4	A7 → t5
A0 → A5 A12	A7 → t6
A0 → A5 A13	A7 → t7
A2 → t1	A7 → t8
A2 → t2	A8 → t1
A2 → t3	A8 → t2
A2 → t4	A8 → t3
A2 → t5	A8 → t4
A2 → t6	A8 → t5
A2 → t7	A8 → t6
A2 → t8	A8 → t7
A4 → A11 A7	A8 → t8
A4 → A6 A9	A9 → A10 A6
A4 → A8 A7	A10 → t10
A5 → A11 A7	A10 → t3
A5 → A6 A9	A10 → t9
A5 → A8 A7	A11 → t11
A6 → A11 A7	A11 → t12
A6 → A8 A7	A11 → t13
A7 → A8 A7	A12 → A2 A4
A7 → t1	A13 → A2 A9
A7 → t2	A14 → A4 A9
A7 → t3	

Gramática 4 (sentences):

La gramática cuatro tiene catorce terminales *null*, fly, $flies$, *like*, *likes*, *time*, *times*, *arrow*, *arrows*, of, *with*, *the*, *a*, y *an*, además de trece no terminales $Sentence$, $Verb$, $Object$, $Subject$, $NounPhrase$, $NounGroup$, $Noun$, $PrePhrase$, $Preposition$, $Article$, $D[Verb\_Object]$, $D[Verb\_PrePhrase]$ y $D[Object\_PrePhrase]$ y veinticinco reglas de producción:

Sentence → Subject D[Verb_Object]	NounGroup → time
Sentence → Subject D[Verb_PrePhrase]	NounGroup → times
Sentence → Verb D[Object_PrePhrase]	NounPhrase → Article NounGroup

Article → a
Article → an
Article → the
D[Object_PrePhrase]→ Object PrePhrase
D[Verb_Object] → Verb Object
D[Verb_PrePhrase] → Verb PrePhrase
Noun → arrow
Noun → arrows
Noun → flies
Noun → fly
Noun → like
Noun → likes
Noun → time
Noun → times
NounGroup → arrow
NounGroup → arrows
NounGroup → flies
NounGroup → fly
NounGroup → like
NounGroup → likes
NounGroup → Noun NounGroup
NounPhrase → Noun NounGroup
Object → Article NounGroup
Object → Noun NounGroup
Object → NounPhrase PrePhrase
PrePhrase → Preposition NounPhrase
Preposition → like
Preposition → of
Preposition → with
Sentence → Verb Object
Subject → Article NounGroup
Subject → Noun NounGroup
Subject → NounPhrase PrePhrase
Verb → arrow
Verb → arrows
Verb → flies
Verb → fly
Verb → like
Verb → likes
Verb → time
Verb → times

Gramática 6:

La gramática uno tiene dos terminales a y b, además de cinco no terminales S, A, B y C y ocho reglas de producción:

S → A C
S → B D
S → A A
S → B B
A → a
B → b
C → S A
D → S B

Experimentos para la función CYK2

GRAMÁTICA	N° DE CADENAS	LONG. PROMEDIO DE LAS CADENAS	TIEMPO EN GENERAR ÁRBOLES Y MONOMIOS	N° DE ÁRBOLES	TIEMPO EN MOSTRAR ÁRBOLES	TIEMPO EN MOSTRAR MONOMIOS
G3(14,13,47)	500	6,536	0,094	1172	4,493	0,312
	1000	6,573	0,172	2256	9,08	0,515
	1500	6,648	0,234	3515	14,336	0,811
	2000	6,6775	0,359	4709	19,329	1,107
	2500	6,7152	0,422	5943	24,164	1,544
	3000	6,717	0,468	7126	29,172	2,012
	3500	6,746	0,562	8354	34,663	2,45
	4000	6,753	0,656	9591	39,53	2,996
	4500	6,747	0,733	10772	44,912	3,416
	5000	6,759	0,796	12046	49,795	3,994
	5500	6,774	0,889	13205	54,819	4,524
	6000	6,792	0,983	14531	60,262	5,288
	6500	6,802	1,076	15828	66,238	6,053
	7000	6,827	1,154	17136	72,26	6,802
	7500	6,843	1,233	18331	77,579	7,426
	8000	6,858	1,311	19563	84,271	8,362
	8500	6,865	1,404	20832	88,28	9,11
	9000	6,862	1,513	22110	95,16	9,859
	9500	6,869	1,545	23347	102,991	10,951
	10000	6,874	1,638	24624	108,576	11,746

Cuadro 6.23: Experimento CYK2.

Evaluación de la función Outside dependiendo del i elegido que varía en los elementos de la cadena.

Se muestran los resultados de los experimentos, en este caso con la gramática 3.

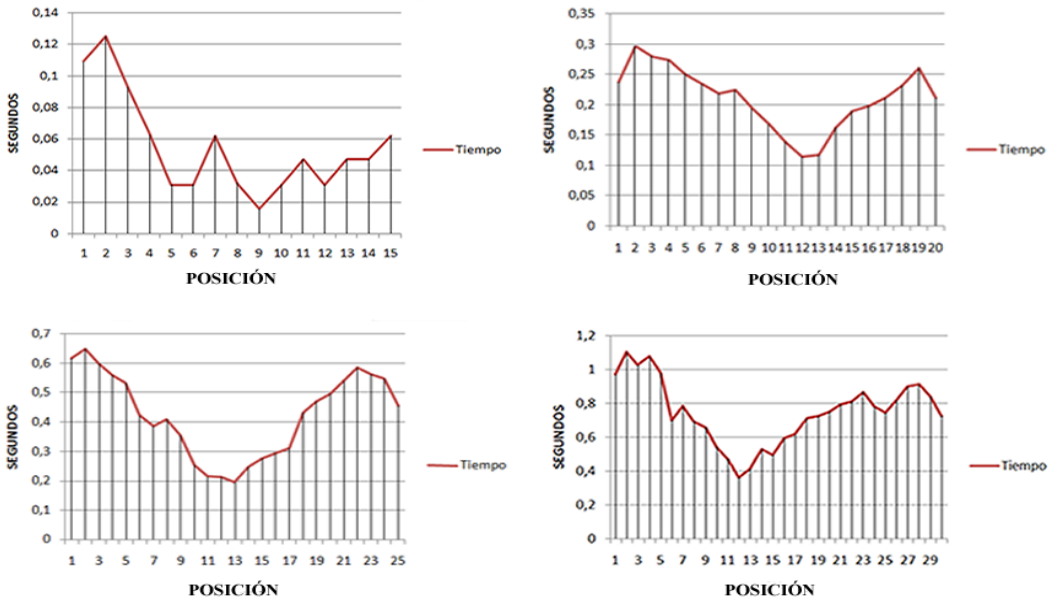

Figura 6.4: Evolución del tiempo empleado por la función Outside en calcular la probabilidad para cada elemento de las cadenas de longitud 15, 20, 25 y 30 con una distribución de probabilidad uniforme dada la gramática 3.

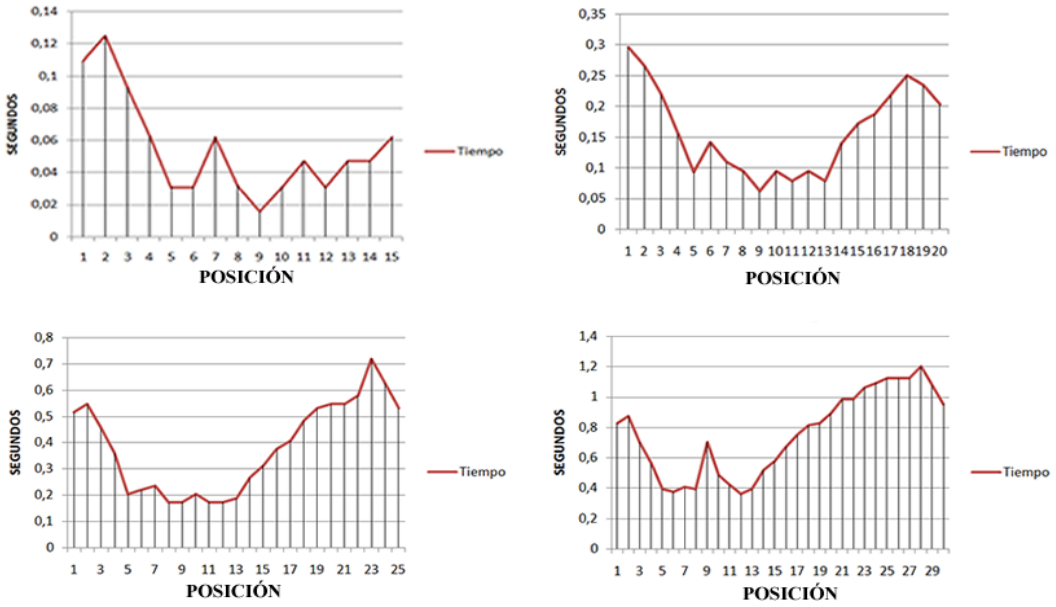

Figura 6.5: Evolución del tiempo empleado por la función Outside en calcular la probabilidad para cada elemento de las cadenas de longitud 15, 20, 25 y 30 con una distribución de probabilidad aleatoria dada la gramática 3.

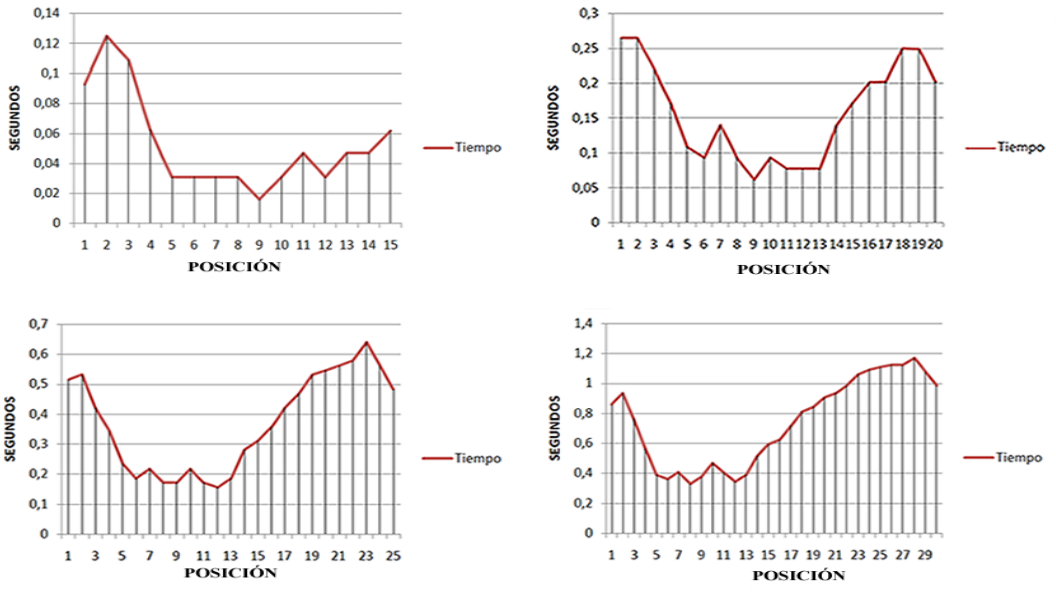

Figura 6.6: Evolución del tiempo empleado por la función Outside en calcular la probabilidad para cada elemento de las cadenas de longitud 15, 20, 25 y 30 con una distribución de probabilidad por frecuencia dada la gramática 3.

Para esta gramática, también es notorio, que para posiciones próximas al punto medio de la longitud de la cadena, en comparación con los puntos extremos es significativamente menor.

Comparación de los algoritmos Inside y Outside asociada a la longitud de la cadena.

Anteriormente se hizo esta representación con la gramática tres, con una distribución de probabilidad uniforme, ahora, se presenta el mismo análisis ampliado a distribución aleatoria y por frecuencia.

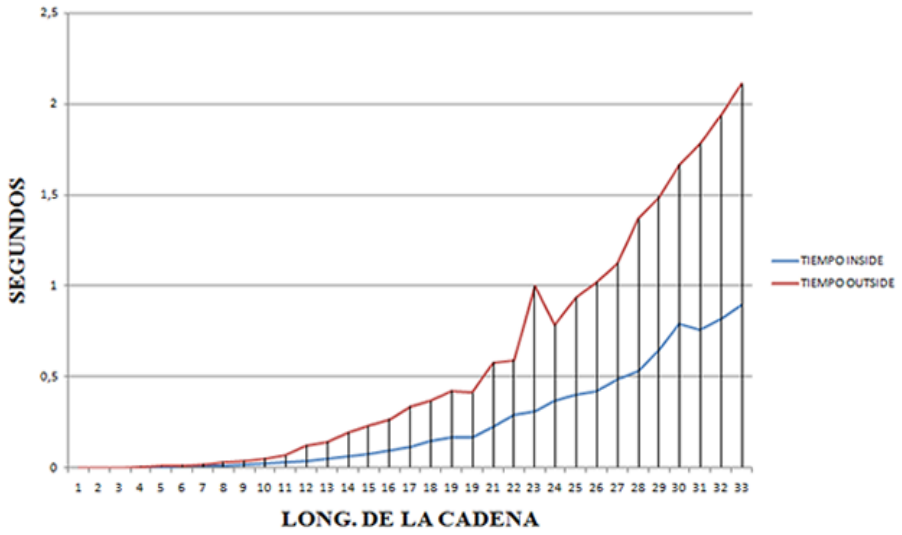

Figura 6.7: Tiempo empleado por las funciones Inside y Outside en calcular la probabilidad de treintaitres cadenas con distribución de probabilidad aleatoria.

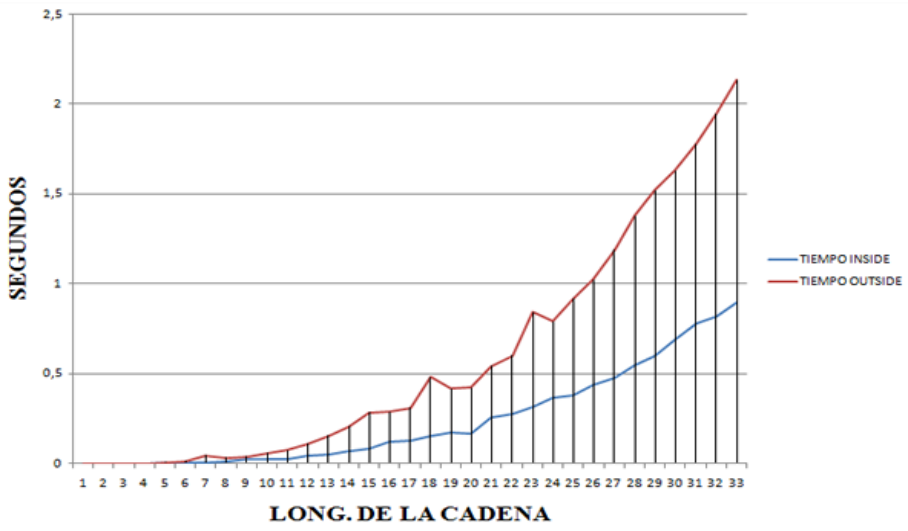

Figura 6.8: Tiempo empleado por las funciones Inside y Outside en calcular la probabilidad de treintaitres cadenas con una distribución de probabilidad por frecuencia.