Abstract

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

Facultad de Ciencias de la Computación

Maestría en Ciencias de la Computación

Un Lenguaje de Apoyo para el Preprocesamiento de Textos

TESIS

QUE PARA OBTENER EL GRADO DE

Maestro en Ciencias de la Computación

PRESENTA

I.S.C Erika Olivia Hernández Beristain

ASESORES

M.C David Eduardo Pinto Avendaño

M.C José de Jesús Lavalle Martínez

Puebla, Pue. Otoño de 2005

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

Facultad de Ciencias de la Computación

Maestría en Ci a Computación

"Un Lenguaje de Apoyo para el Preprocesamiento de Textos"

Tesis profesional presentada por

I.S.C Erika Olivia Hernández Beristain

como requisito parcial
para obtener el título de Maestro
en Maestro en Ciencias de la Computación

Jurado Calificador

Dr. Nombre del Presidente
Presidente

Dr. Nombre del Director y Vocal
Vocal y Director

Dr. Nombre del Secretario
Secretario

Puebla, Pue. Agosto de 2005

Tesis profesional sustentada por

I.S.C Erika Olivia Hernández Beristain

como requisito parcial para obtener el título de

Maestro en Maestro en Ciencias de la Computación.

Aceptada por el Maestría en Ciencias de la Computación

Dr. Nombre del Director y Vocal Vocal y Director	Dr. Nombre del Presidente Presidente
Dr. Nombre del Secretario Secretario	Dr. Nombre del Jefe de Departamento Jefe del Departamento

19 de Agosto de 2005

D e d i c a t o r i a.

A Dios por concederme el privilegio de la vida. Gracias Señor por ser la luz que ilumina mi camino.

A mi madre por ser lo más valioso que tengo y por enseñarme a luchar para alcanzar mis metas. Lo logramos mami ..!

A mi novio Franco, por estar conmigo en aquellos momentos en que el estudio ocupo mi tiempo. Gracias Frank, Te amo.

A mis asesores M.C. David Pinto y M.C. Jesús Lavalle por su interés y dedicación en la dirección de este trabajo. Gracias...

Índice general

Índice de Tablas

Índice de figuras

Introducción

El tratamiento de textos en Internet para su uso posterior en Sistemas de Recuperación de Información (SRI), es de vital importancia. Es un hecho que los resultados de precisión y evocación en los SRI dependen en gran medida de la fase de preprocesamiento.

Es por ello que con el fin de desarrollar sistemas robustos para el Procesamiento de Lenguaje Natural (PLN), es necesario emplear algoritmos que comúnmente requieren que los documentos fuente sean preprocesados antes de utilizarlos en alguna técnica de análisis de información. La revisión de las publicaciones de los últimos años no deja lugar a dudas acerca de la importancia que tiene el preprocesamiento en el momento de aplicar técnicas de Procesamiento de Lenguaje Natural(PLN).

Este trabajo de tesis de maestría expone un lenguaje con la capacidad de preprocesar una gran variedad de *corpora*, con la finalidad de convertirlos en textos planos preprocesados y listos para tareas específicas de PLN. La gramática brinda una sintaxis casi en Lenguaje Natural(LN), lo que hace de éste un lenguaje de fácil aprendizaje.

Este documento tiene la siguiente organización, el capítulo 1 presenta los antecedentes de mayor importancia para la construcción del trabajo. Así también, se expone la problemática de manera detallada. El capítulo 2 describe el lenguaje desarrollado, exponiendo la gramática como parte fundamental en el diseño del lenguaje. El capítulo

3 presenta la implementación de la gramática. El capítulo 4 muestra un conjunto de pruebas aplicadas sobre diferentes *corpora*. Finalmente, en el capítulo 5 se establecen las conclusiones y los detalles que hemos considerado relevantes y suficientes para la defensa de esta Tesis.

Capítulo 1

Marco Teórico

Actualmente, el avance tecnológico en los medios de comunicación impresos y electrónicos permite obtener grandes volúmenes de información en forma digital. La mayoría de esta información se presenta en forma de textos en lenguajes naturales. Toda esa información contenida en los textos es muy importante ya que permite analizar, comparar y entender el entorno en el que vive el ser humano. Sin embargo, se presentan difícultades por la imposibilidad humana de manejar esa enorme cantidad de textos.

Entre las herramientas que ayudan en las tareas diarias, la computadora es, hoy en día, una herramienta indispensable para el procesamiento de grandes volúmenes de datos.

Para convertir a la computadora en nuestro verdadero ayudante en el procesamiento de textos, se necesita pasar un largo camino de aprendizaje de la estructura de textos y de su formalización.

1.1. Planteamiento de Problema

El Procesamiento de Lenguaje Natural es un área de investigación que, sin ser de reciente creación, ha ido adquiriendo un peso importante por las necesidades de infor-

mación de la sociedad. La explosiva necesidad de manejar grandes cantidades de información textual, trae consigo que repentinamente, todo el mundo tenga a su disposición cantidades enormes de información difíciles de procesar y de gestionar adecuadamente.

El preprocesamiento universal de *corpora* es una tarea difícil y aun por resolver; de hecho, investigadores en el área de PLN requieren programar métodos de preprocesamiento diferentes, cada vez que necesitan aplicar alguna técnica de análisis de información a nuevos *corpora*. Adicionalmente, es posible verificar que los conjuntos de información existentes carecen de una estructura y simbología homogénea que permita crear un programa *ad-hoc* para esta tarea. Así el tiempo empleado en programar métodos de preprocesamiento para *corpora* heterogéneos crece de manera sustancial, produciendo una inversión de tiempo considerable en la fase de preprocesamiento, el cual podría reducirse para incrementar el periodo de pruebas de los sistemas de PLN.

1.2. El preprocesamiento

Procesar datos es el equivalente a pensar para la computadora, calculando comparando y tomando decisiones. Concretamente, la función del preproceso es la de convertir el texto de entrada (secuencia de caracteres) en un formato adecuado para su tratamiento.

De manera general, el preprocesamiento puede ser planteado de la manera siguiente: dado un texto D, denotamos con D' al que se obtiene por eliminar las palabras cerradas o *stopwords*, símbolos y etiquetas[6].

En este proceso eliminamos o cambiamos aquellos elementos que pueden presentar ambigüedad dentro del texto o que puedan incomodar para los propósitos de análisis de información, distorsionando los resultados.

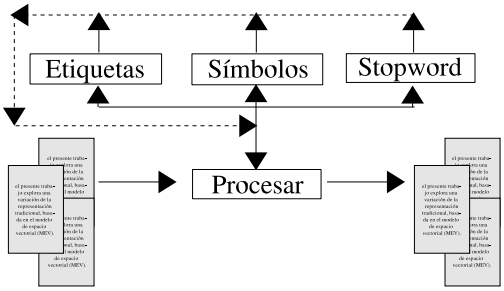


Figura 1.1: El preprocesamiento

1.2.1. Las palabras cerradas o *stopwords*

Una *stopword* es una palabra que se repite mucho en una colección y que no es relevante para dicha colección en los procesos de recuperación de información, por lo cual sería mejor que la palabra no sea considerada. Por ejemplo, las preposiciones y los artículos, son ejemplos clásicos de *stopwords* ya que por lo general nadie quiere hacer una búsqueda de palabras como "de" o "el". Además que son irrelevantes, por lo general estas palabras aparecen en todos los documentos, lo que hace que el tamaño del vocabulario de la colección sea más grande.

1.2.2. Etiquetas de formato

Son aquellas que algunas veces sirven como delimitadoras de documentos pertenecientes a un *corpora*. Para el caso de documentos de Internet, se pueden tener por ejemplo, las siguientes:

$< TEXT >$

$< /TEXT >$

$< HEADLINE >$

$< DOC >$

$< /DOC >$, etc.

1.2.3. Símbolos

Cuando se hace mención de los símbolos, se contemplan no solamente los signos de puntuación, sino también los símbolos especiales que muchas veces forman parte de algún tipo de *corpora*.

1.3. *Corpora*

Según el Diccionario de la Real Academia de la lengua Española[1] *corpora* es un "conjunto lo más extenso y ordenado posible de datos o textos científicos, literarios, etc., que pueden servir de base a una investigación". En nuestro caso, estamos trabajando no solamente con *corpora* lingüisticos usados a nivel mundial con licencia, sino también con *corpora* construidos por alumnos de maestría de la Facultad de Ciencias de la Computación (FCC). La función del *corpora* es la de almacenar textos de tal forma que puedan ser analizados automáticamente. A continuación se muestra una descripción detallada de las características que tienen los *corpora* utilizados como recursos de prueba en la Facultad de Ciencias de la Computación de la BUAP.

1.3.1. *Corpus* TREC-5

TREC (Text Retrieval Conference) constituye uno de los esfuerzos más significativos de investigación experimental en recuperación de información. El patrocinio de elaboración de estas conferencias se encuentra a cargo de la Nacional Institute of Standards and Technology (NIST) y de la Defense Advanced Research Projects Agency (DARPA).

[1] El Diccionario de la Real Academia de la Lengua Española (DRAE) se encuentra disponible en http://www.rae.es

Dichas conferencias comenzarón en 1992 (TREC-1) y vienen celebrándose con periodicidad anual hasta la fecha. De manera particular, en 1996 se celebró TREC-5 el cual es una colección de la agencia de noticias del periódico "El Norte" de Monterrey, Nuevo León conformada por un total de 58,062 documentos, con un tamaño aproximado de 198 MB; contiene un total de 50 tópicos o consultas supervisadas; para cada tópico se tiene asociado un archivo con los documentos relevantes (denominado *qrels*), éste no es un indicativo del grado de importancia, ya que solamente contiene una indicación binaria de (1) relevante o (0) no-relevante[5].

La figura 1.2 muestra el formato típico de los textos almacenados en el *corpus* TREC-5.

<DOC>
<DOCNO> SP94-0000001 </DOCNO>
<ARTNUM> 0000001 </ARTNUM>
<HEADLINE>
José Luis Parga
</HEADLINE>
<TEXT>
Derroche de Civismo

NOS gustó asistir a la ceremonia que inauguró el XXI Torneo Universitario que se juega en el Campestre porque los jóvenes estadounidenses que participan se emocionaron con la organización y respeto a su bandera y a la nuestra...
</TEXT>
</DOC>

Figura 1.2: Formato TREC-5

Las etiquetas "$< DOC >$" "$< /DOC >$" se usan para delimitar documentos, las etiquetas "$< DOCNO >$" "$< /DOCNO >$" se usan para identificar a los documentos, las etiquetas "$< HEADLINE >$" "$< /HEADLINE >$" enmarcan el título del documento, y por último las etiquetas "$< TEXT >$" "$< /TEXT >$" contienen el cuerpo del documento.

1.3.2. *Corpus* EuroGOV

Está compuesto de documentos Web obtenidos de sitios gubernamentales Europeos. El *corpus* tiene cerca de 3 millones y medio de documentos en diferentes idiomas, provenientes de diversos paises o dominios de Internet (fr, es, uk, pt, be, sk, lv, se, etc). El tamaño aproximado es de 70 Gigabytes.

La colección EuroGOV se divide en directorios, uno para cada dominio, cada directorio contiene uno o más archivos comprimidos, cada archivo contiene hasta 25,000 documentos.

Como se puede observar en la figura 1.3, se utilizan una serie de etiquetas *XML* con la finalidad de estructurar el *corpus*. A continuación se presentan las etiquetas usadas para delimitar cada parte del *corpus*.

EuroGOV:bin domain=“”Indica el dominio de nivel superior,(es, fr, uk, etc)

url=“”Contiene el *URL* de la página.

id=“” Identifica el documento siguiendo el formato (Exx-yyy-z) donde:

E es E y esta puesto para EuroGOV.

xx es el nivel superior del dominio.

yyy es el identificador del archivo.

fetchDate=“” Muestra la fecha en la cual fue descargada la página *web*.

contentType=“” Presenta el tipo de contenido (html, msword, pdf, text, unknown) según lo dado por el *server web*.

Por último entre la etiqueta <![**CDATA**[y]] > se encuentra el contenido de la página *web* (es decir la información más importante para el usuario). Una observación rápida a este *corpus* permite verificar al existencia de una gran cantidad de información que podría no ser útil en los procesos de recuperación de información. Tal es el caso de procedimientos de programación script (vbscript y javascript), procedimientos para plantillas de páginas *web*, y por supuesto, todas aquellas etiquetas de formato de texto ($< h1 >$, $< b >$, $< hr >$, etc.). De esta manera, este *corpus* presenta un gran reto para este trabajo de Tesis.

```
url="http://www.mma.es/eus/parques/centasoc/posidonia.htm"
id="Ees-002-2130916"
md5="96c5c5c34a8d5bc83adecb8b1ee0c017"
fetchDate="Sun Nov07 19:37:20 MET 2004"
contentType="">
<EuroGOV/content>
<![CDATA[
<html><!-- #BeginTemplate "/Templates/plantilla_parques_centasoc.dwt" -->
<head>
<META name="a_category" content="Parques">
<META name="keywords" content="">
<META name="r_modify_date" content="2004-03-23 10:12:06">
<META name="title" content="Caracter?sticas y h?bitat de la posidonia">
<!-- #BeginEditable "doctitle" -->
<title> </title>
<!-- #EndEditable -->
</head>
body leftmargin="0" background="/imagenes/plantillas/fondos_parques/f_parqgen.gif" >
<tr>
<a href="javascript:traduccion('')" ><img src="/imagenes/comun/banderas/castellano.gif"></a>
<a href="javascript:traduccion('cat')" ><img src="/imagenes/comun/banderas/catalan.gif"></a>
<p><font size="2">Forma praderas submarinas en fondos de la
                zona litoral en profundidades que oscilan desde la superficie
                hasta los 35-40 metros, llegando excepcionalmente, en zonas
                de aguas muy transparentes, a profundidades cercanas al
                centenar de metros.</font></p>
</body>
<!-- #EndTemplate --></html>
]]>
```

Figura 1.3: Formato EuroGov

HIPERHIDROSIS ***

Definición

La hiperhidrosis primaria o sudoración excesiva principalmente de las palmas de las manos, axilas,cara y planta de los pies.

DIABETES Y EJERCICIO ****

El ejercicio fisico regular diario forma parte del tratamiento de la diabetes, junto con la dieta y la insulina o las pastillas.

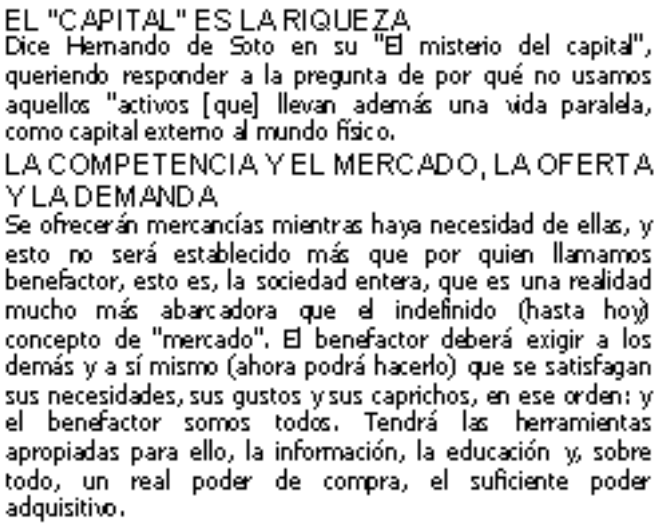
Figura 1.4: Formato *Corpus* Medicina

1.3.3. *Corpus* de Medicina

Este *corpus* está compuesto de textos del área de Medicina, compilados en un solo archivo de 5.17MB. Fue construido por un alumno de la FCC de la Benemérita Universidad Autónoma de Puebla, con la finalidad de realizar experimentos en PLN. La estructura es bastante simple, ya que utilizan un delimitador para indicar el comienzo de cada noticia. La figura 1.4 muestra un ejemplo de la estructura de dicho *corpus*.

1.3.4. *Corpus* Economía

Este *corpus* está compuesto de textos del área de Economía, compilados en un solo archivo de 3.18MB. Fue construido por un alumno de la FCC de la Benemérita Universidad Autónoma de Puebla con la finalidad de realizar experimentos en PLN. La estructura es muy simple ya que utilizan letras mayúsculas para indicar el comienzo de cada noticia. La figura 1.5 muestra una representación de dicho *corpus*.



EL "CAPITAL" ES LA RIQUEZA
Dice Hernando de Soto en su "El misterio del capital", queriendo responder a la pregunta de por qué no usamos aquellos "activos [que] llevan además una vida paralela, como capital externo al mundo físico.
LA COMPETENCIA Y EL MERCADO, LA OFERTA Y LA DEMANDA
Se ofrecerán mercancías mientras haya necesidad de ellas, y esto no será establecido más que por quien llamamos benefactor, esto es, la sociedad entera, que es una realidad mucho más abarcadora que el indefinido (hasta hoy) concepto de "mercado". El benefactor deberá exigir a los demás y a sí mismo (ahora podrá hacerlo) que se satisfagan sus necesidades, sus gustos y sus caprichos, en ese orden: y el benefactor somos todos. Tendrá las herramientas apropiadas para ello, la información, la educación y, sobre todo, un real poder de compra, el suficiente poder adquisitivo.

Figura 1.5: Formato *Corpus* Economía

1.4. La Gramática

Una gramática es una herramienta o notación que nos permite definir un lenguaje, por medio de una serie de reglas que indican como construir cadenas válidas (oraciones) para el lenguaje. Chomsky formalizó el concepto de una gramática, al hacer observaciones importantes en la complejidad de una gramática que a su vez establece la complejidad del lenguaje. Es por ello que se dice que una gramática es una forma de describir al lenguaje [7].

1.4.1. Clasificación de las Gramáticas

N. Chomsky en 1959 en su famoso artículo *"On certain formal properties of grammars"* definió cuatro familias de gramáticas (y lenguajes) que forman lo que se llama la jerarquía de Chomsky. Estas son: gramáticas sin restricciones, sensibles al contexto, libres de contexto y regulares (tipo 0, 1, 2 y 3).

La restricciones puestas en cada regla aumentan con el tipo de la gramática [7]. En

nuestro caso hemos utilizado las de tipo 2, por lo cual a continuación se introducen conceptos relacionados con el tema.

1.4.2. Gramáticas Libres de Contexto

Definición 1.1. *Una gramática Libre de Contexto (GLC) es una descripción estructural precisa de un lenguaje. Formalmente es una tupla* $G = (V_n, V_t, S, P)$*, donde :*

V_n = *Un conjunto de símbolos no terminales*

V_t = *Un conjunto de símbolos terminales*

S = *Símbolo inicial de la gramática*

P = *Un conjunto de producciones*

Las producciones tienen la forma $A \rightarrow \alpha$ donde $A \in V_n$ y α es una expresión ya sea compuesta tanto por símbolos no terminales como terminales ($\alpha \in (\ V_t \cup V_n\ )^*$) o también puede ser la expresión nula ϵ. Una producción puede verse como una regla de reescritura que indica cómo reemplazar símbolos no terminales por la expresión correspondiente en la parte derecha de la producción. Así, partiendo de algún $A \in V_n$, se puede aplicar las reglas de P hasta alcanzar eventualmente a una expresión compuesta únicamente por símbolos terminales. A esta expresión se le denomina *frase* o *lexema*, mientras que a las expresiones formadas por símbolos terminales y no terminales se les denomina *formas de frase*. Las gramáticas libres de Contexto (GLC) y los lenguajes que generan (Lenguajes Libres de Contexto (LLC)) permiten definir la sintaxis de los lenguajes de programación. La forma típica de las GLC es la notación BNF (Backus-Naur-Form), la cual consiste de un conjunto finito de reglas o cláusulas, descritas en dos partes y separadas por ::=. En la parte izquierda aparece una variable (símbolo no terminal) que presenta un lenguaje; las variables se definen recursivamente en términos

de otras variables y de átomos o símbolos primitivos denominados terminales. La parte derecha es justamente la definición del lenguaje, el cual contiene cadenas del lenguaje de otras variables posiblemente junto con algunos terminales [4].

1.5. Forma Normal de Chomsky

Una GLC está en forma normal de chomsky si y solo si todas las producciones tienen la forma: $A \rightarrow BC$ o $A \rightarrow$ a donde, A, B, C son variables y a es un símbolo terminal. Para conseguir esta forma, hay que hacer algunas simplificaciones preliminares que son útiles en varios contextos:

Eliminar los símbolos inútiles.

Eliminar las producciones ϵ.

Eliminar las producciones unitarias.

1.5.1. Eliminación de símbolos inútiles

Primero se identifican los símbolos inútiles en una GLC $G = (V_n, V_t, S, P)$ y se muestra cómo eliminarlos.

Definición 1.2. *Un símbolo no terminal X es útil si existe una derivación.*
$S \overset{*}{\Rightarrow} \alpha\ X\ \beta\ {}^{*}\Rightarrow \omega$ (para $\omega \in V_t^*$). De lo contrario se dice que X es inútil.

Esta definición indica dos condiciones que todo símbolo X no terminal debe cumplir para considerársele útil:

1. Una frase ω debe ser derivable de X:$X \overset{*}{\Rightarrow} \omega$

2. X debe ocurrir en alguna forma de frase derivable desde el símbolo inicial S:$S \overset{*}{\Rightarrow} \alpha X \beta$.

Tales condiciones no son suficientes, pues debe asegurarse que X no ocurra en alguna forma de frase en la cual contenga alguna variable Y a partir de la cual ninguna frase pueda derivarse. se eliminan los no terminales inútiles de una GLC como resultado de los lemas que se describen a continuación. El primero de ellos elimina los no terminales inútiles, mientras que el segundo refina el conjunto de símbolos terminales.

Lema 1.1. *Dada una GLC* $G = (V_n, V_t, S, P)$ *tal que L(G)*$\neq \emptyset$*, se puede encontrar una GLC* $G' = (V_n', V_t', S', P')$ *equivalente a G la cual, para cada* $X \in V_n'$ *existe alguna frase* $\omega \in V_t^*$ *que es deriva por* X:$X \overset{*}{\Rightarrow} \omega$.

A continuación se presenta el algoritmo 1.1 para calcular V_n'. P' es el conjunto de todas las producciones $X \rightarrow \alpha$ tales que $X \in V_n'$ y $\alpha \in (V_n' \cup V_t)$.

Algoritmo 1.1. Cálculo de V_n

Entrada: V_n,V_t,P,

Salida: V_n'

Old $\leftarrow \emptyset$;

NewV $\leftarrow$ { X | $X \rightarrow \omega$ para alguna frase $\omega \in V_t^*$;

while OldV $\neq$ NewV **do**

OldV $\leftarrow$ NewV;

NewV $\leftarrow$ OldV $\cup$ { A | A $\rightarrow \alpha$, para forma de frase $\alpha \in (V_t \cup \text{OldV})^*$;

end while

$V_n' \leftarrow$ NewV.

Aplicando primero el lema 1.1 y entonces con el lema siguiente podemos convertir una GLC en una equivalente sin símbolos inútiles.

Lema 1.2. *Dada una GLC* $G = (V_n, V_t, S, P)$ *es posible construir una GLC equivalente* $G' = (V_n', V_t', S', P')$ *en la cual, para cada símbolo* $X \in (V_n \cup V_t)$ *existen formas de frase* α, β *tales que* $S \overset{*}{\Rightarrow} \alpha\ X\ \beta$.

El conjunto de símbolos que aparecerán en las formas de frase G' se obtiene por el algoritmo 1.2 que a continuación se presenta.

Algoritmo 1.2. Cálculo de P'

$V_n' \leftarrow V_n' \cup \{S\}$;

if A $\in V_n'$ y A $\rightarrow \alpha_1 \mid \alpha_2 \mid ... \mid \alpha_n$ **then**

$V_n' \leftarrow V_n' \cup \{$ variables que ocurren en α_k $\}$ **para**

k=1,...,n;

$V_t' \leftarrow V_t' \cup \{$ terminales que ocurren en α_k $\}$

para $k = 1, ..., n$;

end if

1.5.2. Eliminación de Producciones ϵ y unitarias

Una producción ϵ es de la forma $A \rightarrow \epsilon$. Si $\epsilon \in$ L(G) no es posible eliminar todas las producciones ϵ, de lo contrario si es posible. Para ello se debe determinar para cada $A \in V_n$ si $A \overset{*}{\Rightarrow} \epsilon$. De ser el caso se dice que A es nulificable. se reemplaza cada producción $B \rightarrow X_1, X_2 ... X_n$ por las producciones en las cuales se eliminan subconjuntos de X_i son nulificables(lo cual significa que $B \rightarrow \epsilon$), se debe descartar la producción $B \rightarrow \epsilon$. Por su parte, una producción unitaria tienen la forma $A \rightarrow B$. Tanto las producciones

unitarias como las producciones ϵ son estorbosas pues dificultan determinar si al aplicar una producción se logra algún progreso para derivar una frase. Las producciones unitarias pueden causar ciclos en la derivación, y las producciones ϵ pueden generar formas de frase de no terminales muy grandes y al final eliminarlas todas. Sin producciones unitarias y ϵ, cada paso de la derivación logra progreso en el sentido que la forma de frase crece estrictamente o se incorpora un nuevo terminal en cada paso de la derivación[4].

Lema 1.3. *Sea $G = (V_n, V_t, S, P)$ una GLC cualquiera, existe una GLC G' sin producciones ϵ ó producciones unitaras tal que $L(G') = L(G)$- $\{\epsilon\}$.*

1.5.3. Forma Normal de Chomsky

Una vez que hemos prescindido de las producciones unitarias y ϵ, es tarea simple poner la GLC resultante en la Forma Normal de Chomsky(FNC)[2] :

1. Para cada terminal $a \in V_t$, se introduce un nuevo no terminal $A_a \rightarrow a$, y se reemplaza toda ocurrencia de a en la parte derecha de una producción que originalmente ya estaba P (excepto en producciones de la forma $B \rightarrow a$) por $A_a \rightarrow a$.

2. Todas las producciones quedan de la forma $A \rightarrow a$ ó $A \rightarrow B_1B_2...B_k$, con $k \geq 2$ donde los B_k, son no terminales. El conjunto de las cadenas derivadas con esta nueva gramática es exactamente el mismo, solo necesitan un paso más antes de generar un símbolo terminal.

[2] N.Chomsky es el primer lingüista que propuso las gramáticas libres de contexto como una forma de escribir los lenguajes naturales. Es interesante que la FNC no parece tener aplicaciones importantes en lingüistica natural, aunque tiene aplicación como una forma eficiente de comprobar si una cadena pertenece a un lenguaje libre de contexto[7].

3. Para cada producción $A \rightarrow B_1B_2...B_k$, con $k \geq 3$ se introduce un nuevo no terminal C y se reemplaza tal producción con las dos producciones $A \rightarrow B_1\ C\ y\ C \rightarrow B_2B_3...B_k$. Se repite este proceso hasta que todas las partes derechas de las producciones sean de longitud a lo más dos [4].

En este capítulo se han introducido conceptos relevantes para la comprensión del proyecto de Tesis tales como: la importancia que tiene el preprocesamiento para la obtención de buenos resultados en el momento de aplicar técnicas de PLN, y el uso de las Gramática Libres de Contexto como una forma de escribir los Lenguajes Naturales.
En el siguiente capítulo se presenta el diseño de la gramática como parte medular en la construcción del Lenguaje para el Preprocesamiento de Textos.

Capítulo 2

Diseño de la gramática

Entre las diversas herramientas adecuadas para aplicaciones de Procesamiento del Lenguaje Natural, las gramáticas formales revisten particular interés, pues permiten caracterizar con precisión las oraciones de un lenguaje, así como distinguir su estructura y componentes. Dada una gramática y una oración o sentencia, es posible determinar con precisión si la sentencia pertenece o no al lenguaje descrito por la gramática[2].
En este trabajo se propone una gramática para el preprocesamiento de textos. Ésta debe incluir una sintaxis bien definida para las operaciones a realizar. Tres elementos son fundamentales en el lenguaje, las palabras cerradas, los símbolos y las etiquetas de formato. Se han creado mecanismos para administrar estos elementos.
Los servicios que el lenguaje proporciona se han clasificado de la siguiente manera:

1. **Servicios a nivel de primitivas**
2. **Servicios a nivel de preprocesamiento**
    - **Operaciones sobre *corpora***
    - **Operaciones de preproceso**

2.1. Descripción del lenguaje

Lenguaje es el empleo de la palabra para expresar ideas, comunicarse, establecer relaciones entre los seres humanos. Un lenguaje es un conjunto de palabras, su pronunciación y los métodos para combinarlas en frases y oraciones, generalmente infinito y que se forma mediante combinaciones de palabras definidas en un diccionario terminológico previamente establecido. Las combinaciones posibles deben respetar un conjunto de reglas sintácticas establecidas, a ello se le conoce con el nombre de "Sintaxis". Además, las palabras deben tener determinado sentido, deben ser comprendidas por un grupo humano en un contexto dado, a ello se le denomina "Semántica".

A lo largo de la historia, el ser humano ha utilizado el lenguaje para trasmitir sus conocimientos, sentimientos, emociones, sensaciones; comunicarse con el resto de los humanos, y esta función del lenguaje la ha desarrollado de manera oral, gráfica, escrita o por señas. Cuando se habla de lenguajes se pueden diferenciar dos clases muy bien definidas los lenguajes naturales[1] y los lenguajes formales[2] .

El trabajo desarrollado caé dentro de los lenguajes formales y a continuación se describe cada uno de los elementos involucrados en el mismo.

En la herramienta diseñada, se han incluido tres archivos (*stopwords.txt, symbols.txt y labels.txt*), los cuales contienen elementos iniciales que, se espera, faciliten la labor de preprocesamiento de información. Estos cuentan con un tamaño inicial de 20KB, 9KB y 7KB, respectivamente. Se han diseñado diversos mecanismos para administrar dichos archivos. A continuación se puntualiza sobre cada uno de ellos, así como también se hace mención de los servicios soportados por el lenguaje.

Los servicios que el lenguaje proporciona se han clasificado de la siguiente manera:

[1] como el español, el ruso , el ingles, el francés, etc.

[2] como los lenguajes de programación, el lenguaje de la lógica matemática, etc.

2.1.1. Servicios a nivel de primitivas

Fueron denominados así por el hecho de permitir únicamente el manejo de las operaciønes básicas, efectuadas sobre las colecciones de palabras cerradas, símbolos y etiquetas de formato.
La herramienta proporciona tres archivos[3] o colecciones. A continuación se describen las operaciones brindadas por este servicio:

- **agregar** : permite añadir nuevos elementos[4] a las colecciones.
- **eliminar** : permite quitar elementos existentes de las colecciones.
- **consultar** : muestra los elementos que conforman las colecciones.

2.1.2. Servicios a nivel de preprocesamiento

Estos tipos de servicios permiten la preparación de *corpora* existentes, mediante la invocación de los servicios primitivos cada vez que sea necesario. Este tipo de servicios estan divididos en: operaciones sobre *corpora* y operaciones de preproceso.

Operaciones sobre *corpora*

El lenguaje permite la creación de un nuevo *corpus* como un contenedor de archivos. Acepta la eliminación de un *corpus* ya existente en caso que así se requiera. Proporciona un listado de archivos pertenecientes a un determinado *corpus*. Muestra el contenido de un archivo perteneciente a un *corpus*, así como las propiedades[5] del mismo. Permite la

[3] *stopwords.txt, symbols.txt y labels.txt*
[4] Tales como palabras cerradas, símbolos y etiquetas de formato
[5] Tales como: ruta, tamaño del archivo y fecha de la última modificación.

inserción de nuevos archivos a diferentes *corpus*.
En conclusión puede ser visto como un sistema de archivos (o mejor dicho como un sistema de *corpus*).

Operaciones de preproceso

El lenguaje permite preprocesar un determinado archivo perteneciente a un *corpus*, bajo diferentes niveles de preprocesamiento tales como: eliminación de palabras cerradas, eliminación de símbolos, eliminación de etiquetas de formato, así como cualquier combinación de las operaciones anteriores.

Extrae sólo partes de un determinado archivo perteneciente a un *corpus*, mediante la inclusión de cadenas de inicio y fin. Permite la sustitución de una determinada etiqueta por otra, mediante la inclusión de cadenas de inicio y fin.

Para hacer uso de los servicios, a nivel primitivo y a nivel de preproceso, es necesario contar con una gramática que verifique las sentencias introducidas por el usuario para aceptarlas o en su defecto rechazarlas, por lo que en la siguiente sección se introduce la estructura de la misma.

2.2. Estructura de la gramática

El establecimiento de las reglas gramaticales que el lenguaje debe de cumplir es una de las situaciones más complejas para su desarrollo.

Es por ello que se ha diseñado cuidadosamente una GLC que permita validar todas aquellas posibles formaciones aceptadas por el lenguaje, abarcando los servicios para el Preprocesamiento, descritos anteriormente.
A continuación se presenta la GLC del lenguaje desarrollado.

Propuesta de la Gramática Libre de Contexto

$$G = (V_n, V_t, S, P)$$
S=<SENTENCIA>

$$V_n = \begin{cases} < SENTENCIA > , < CREATE > , < DROP > , < ADD > , < DELETE > \\ < LIST > , < INSERT > , < DISPLAY > , < PREPARE > , < REMOVE > \\ < REPLACEALL > , < EXTRACT > , < INFOALL > , < ID > , < USING > \\ , < CICLOS > , < LETRA > , < DIGITO > , < NOMARCH > , < STRING > \\ , < SIMBOLO > , < DICCIONARIOS > \end{cases} \tag{2.1}$$

$$V_t = \begin{cases} \text{create, corpus, drop, list, display, add, from, delete, insert, put, prepare, and, extract, to, replaceall,} \\ \text{infoall, for, acutes, a, b, c, d, e, f, g, h, i, j, k, l, m, 0, p, q, r, s, t, u, v, w, x, y, z, A, B, C, D, E, F,} \\ \text{G, H, I, J, K, L, M, , O, P, Q, R, S, T, U, V, W, X, Y, Z,), (, /, *, ascii, \,} \end{cases} \tag{2.2}$$

$$P = \begin{cases} < SENTENCIA > ::= < CREATE > | < DROP > | < LIST > | < DISPLAY > | < ADD > | < DELETE > \\ \qquad | < INSERT > | < PREPARE > | < EXTRACT > | < REPLACEALL > \\ \qquad | < REMOVE > | < INFOALL > \\ < CREATE > ::= \text{create corpus } < ID >; \\ < DROP > ::= \text{drop corpus } < ID >; \\ < LIST > ::= \text{list * from } < ID >; \\ < DISPLAY > ::= \text{display } < ID > . < NOMARCH >; \\ < ADD > ::= \text{add } < DICCIONARIOS > \text{ from } < NOMARCH >; \\ \qquad | \text{ add } < DICCIONARIOS > (\{< STRING > \text{\,,}\}); \\ < DELETE > ::= \text{delete } < DICCIONARIOS > \text{ from } < NOMARCH >; \\ \qquad | \text{ delete } < DICCIONARIOS > (\{< STRING > \text{\,,}\}); \\ < INSERT > ::= \text{insert } < ID > . < NOMARCH >; \\ < PREPARE > ::= \text{prepare } < ID > . < NOMARCH >< USING > \text{ put } < NOMARCH >; \\ < EXTRACT > ::= \text{extract } < STRING > \text{ to } < STRING >< ID > . < NOMARCH > \text{ put } < NOMARCH >; \\ < REPLACEALL > ::= \text{replaceall } < STRING > \text{ for } < STRING >< ID > . < NOMARCH > \text{ put } < NOMARCH >; \\ < REMOVE > ::= \text{remove acutes } < ID > . < NOMARCH > \text{ put } < NOMARCH >; \\ < INFOALL > ::= \text{infoall } < ID > . < NOMARCH > \text{ put } < NOMARCH >; \\ < USING > ::= (< CICLOS >) | < DICCIONARIOS > \\ < CICLOS > ::= < DICCIONARIOS > \{< DICCIONARIOS > \text{ and}\} \\ < NOMARCH > ::= < ID > . < ID > \\ < ID > ::= < LETRA > \{< LETRA > | < DIGITO >\} \\ < STRING > ::= < LETRA > \{< LETRA >\} | < DIGITO > \{< DIGITO >\} \\ \qquad | < DIGITO > \{< LETRA >\} | < SIMBOLO > \{< LETRA > | < SIMBOLO >\} \\ < LETRA > ::= a \mid b \mid c...z \mid A \mid B \mid C...Z \\ < DIGITO > ::= 0 \mid 1 \mid 2..,9 \\ < DICCIONARIOS > ::= STOPWORDS \mid SYMBOLS \mid LABELS \\ < SIMBOLO > ::= \text{ascii} \end{cases} \tag{2.3}$$

2.3. Obtención de la Forma normal de Chomsky (FNC)

A continuación se presentan los pasos necesarios para convertir la GLC a FNC. Para llegar a obtener la FNC, se deben realizar algunas simplificaciones preliminares que son útiles en varios contextos:

PASO 1: Eliminación de símbolos inútiles[6]

Una vez aplicando los algoritmos 1.1 y 1.2 se concluye que todos los símbolos son generadores y son alcanzables a partir del símbolo inicial.

PASO 2: Eliminación de producciones ϵ[7]

La GLC no contiene producciones ϵ, por lo cual se evalúa este paso como obtenido, observando que la gramática sigue sin cambio alguno.

PASO 3: Eliminación de producciones unitarias[8]

Al eliminar las producciones unitarias de la GLC original, se obtiene el conjunto de producciones presentado en la ecuación 2.4.

PASO 4: Obtención de la Forma Normal de Chomsky[9]

Existen 93 símbolos terminales tales como: $\{create, corpus, list, *, (,)...etc\}$, cada uno de los cuales aparece en un cuerpo en el que no hay un único símbolo terminal. Posteriormente se introducen 93 variables no terminales nuevas, correspondientes a estos símbolos terminales, y 93 producciones en las que se reemplaza la variable no terminal nueva por su símbolo terminal. Usando las iniciales para las nuevas variables no terminales, se introducen:

V_1 ::= create

[6] Ver capítulo 1, sección 1.5.1
[7] Ver capítulo 1, sección 1.5.2
[8] Ver capítulo 1, sección 1.5.2
[9] Ver capítulo 1, sección 1.5.3

$\vdots$

$V_{93} ::= 0$

Se añaden estas producciones, y se reemplazan los símbolos terminales por las variables no terminales en todas las producciones en las que no aparece un único símbolo terminal, y se obtiene una nueva gramática.

Ahora, todas las producciones están en la Forma Normal de Chomsky, excepto aquellas que tienen cuerpos de longitud 3 o más. Algunos de estos cuerpos aparecen en más de una producción, se introducen variables no terminales adicionales para cada uno, cuando se presenta este caso.

El conjunto de pasos necesarios para la construcción de la Forma Normal de Chomsky es muy extenso y se muestra de manera detallada en el apéndice A de esta Tesis.

Las ecuaciones [2.5−2.10] muestran la FNC resultante de aplicar los pasos 1, 2, 3 y 4 expuestos anteriormente.

$$
P' = \begin{cases}
< SENTENCIA > ::= \text{create corpus} < ID >; \\
\quad | \text{ drop corpus} < ID >; \\
\quad | \text{ list * from} < ID >; \\
\quad | \text{ display} < ID > . < NOMARCH >; \\
\quad | \text{ add} < DICCIONARIOS > \text{from} < NOMARCH >; \\
\quad | \text{ add} < DICCIONARIOS > (\{< STRING > \backslash,\}); \\
\quad | \text{ delete} < DICCIONARIOS > \text{from} < NOMARCH >; \\
\quad | \text{ delete} < DICCIONARIOS > (\{< STRING > \backslash,\}); \\
\quad | \text{ insert} < ID > . < NOMARCH >; \\
\quad | \text{ prepare} < ID > . < NOMARCH >< USING > \text{put} < NOMARCH >; \\
\quad | \text{ extract} < STRING > \text{to} < STRING >< ID > . < NOMARCH > \text{put} < NOMARCH >; \\
\quad | \text{ replaceall} < STRING > \text{for} < STRING >< ID > . < NOMARCH > \text{put} < NOMARCH >; \\
\quad | \text{ remove acutes} < ID > . < NOMARCH > \text{put} < NOMARCH >; \\
\quad | \text{ infoall} < ID > . < NOMARCH > \text{put} < NOMARCH >; \\
< CREATE > ::= \text{create corpus} < ID >; \\
< DROP > ::= \text{drop corpus} < ID >; \\
< LIST > ::= \text{list * from} < ID >; \\
< DISPLAY > ::= \text{display} < ID > . < NOMARCH >; \\
< ADD > ::= \text{add} < DICCIONARIOS > \text{from} < NOMARCH >; \\
\quad | \text{ add} < DICCIONARIOS > (\{< STRING > \backslash,\}); \\
< DELETE > ::= \text{delete} < DICCIONARIOS > \text{from} < NOMARCH >; \\
\quad | \text{ delete} < DICCIONARIOS > (\{< STRING > \backslash,\}); \\
< INSERT > ::= \text{insert} < ID > . < NOMARCH >; \\
< PREPARE > ::= \text{prepare} < ID > . < NOMARCH >< USING > \text{put} < NOMARCH >; \\
< EXTRACT > ::= \text{extract} < STRING > \text{to} < STRING >< ID > . < NOMARCH > \text{put} < NOMARCH >; \\
< REPLACEALL > ::= \text{replaceall} < STRING > \text{for} < STRING >< ID > . < NOMARCH > \text{put} < NOMARCH >; \\
< REMOVE > ::= \text{remove acutes} < ID > . < NOMARCH > \text{put} < NOMARCH >; \\
< INFOALL > ::= \text{infoall} < ID > . < NOMARCH > \text{put} < NOMARCH >; \\
< USING > ::= (< CICLOS >) \mid STOPWORDS \mid SYMBOLS \mid LABELS \\
< CICLOS > ::=< DICCIONARIOS > \{< DICCIONARIOS > \text{and}\} \\
< NOMARCH > ::=< ID > . < ID > \\
< ID > ::=< LETRA > \{< LETRA > |< DIGITO >\} \\
< STRING > ::=< LETRA > \{< LETRA >\} |< DIGITO > \{< DIGITO >\} \\
\quad |< DIGITO > \{< LETRA >\} |< SIMBOLO > \{< LETRA > | < SIMBOLO >\} \\
< LETRA > ::= a \mid b \mid c...z \mid A \mid B \mid C...Z \\
< DIGITO > ::= 0 \mid 1 \mid 2..,9 \\
< DICCIONARIOS > ::= STOPWORDS \mid SYMBOLS \mid LABELS \\
< SIMBOLO > ::= \text{ascii}
\end{cases}
\tag{2.4}
$$

Gramática obtenida en Forma Normal de Chomsky

$$G^{9'} = (V_n, V_t, S, P)$$
S=<SENTENCIA>

$$V_n = \begin{cases} < SENTENCIA > , < CREATE > , < DROP > , < ADD > , < DELETE > , < LIST > \\ , < INSERT > , < DISPLAY > , < PREPARE > , < REMOVE > , < REPLACEALL > \\ , < EXTRACT > , < INFOALL > , < ID > , < USING > , < CICLOS >< LETRA > \\ , < DIGITO > , < NOMARCH > , < STRING > , < SIMBOLO > , < DICCIONARIOS > \\ ,V_1,V_2,V_3,V_4,V_5,V_6,V_7,V_8,V_9,V_{10},V_{11},V_{12},V_{13},V_{14},V_{15},V_{16},V_{17},V_{18},V_{19},V_{20},V_{21},V_{22},V_{23},V_{24},V_{25} \\ ,V_{26},V_{27},V_{28},V_{29},V_{30},V_{31},V_{32},V_{33},V_{34},V_{35},V_{36},V_{37},V_{38},V_{39},V_{40},V_{41},V_{42},V_{43},V_{44},V_{45},V_{46},V_{47},V_{48} \\ ,V_{49},V_{50},V_{51},V_{52},V_{53},V_{54},V_{55},V_{56},V_{57},V_{58},V_{59},V_{60},V_{61},V_{62},V_{63},V_{64},V_{65},V_{66},V_{67},V_{68},V_{69},V_{70},V_{71} \\ ,V_{72},V_{73},V_{74},V_{75},V_{76},V_{77},V_{78},V_{79},V_{80},V_{81},V_{82},V_{83},V_{84},V_{85},V_{86},V_{87},V_{88},V_{89},V_{90},V_{91},V_{92},V_{93},V_{94} \\ ,V_{95},V_{100},V_{101},V_{102},V_{103},V_{104},V_{105},V_{106},V_{107},V_{108},V_{109},V_{110},V_{111},V_{112},V_{113},V_{114},V_{115},V_{116},V_{117} \\ ,V_{118},V_{119},V_{120},V_{121},V_{122},V_{123},V_{124},V_{125},V_{126},V_{127},V_{128},V_{129},V_{130},V_{131},V_{132},V_{133},V_{134},V_{135} \end{cases} \quad (2.5)$$

$$V_t = \begin{cases} \text{create, corpus, drop, list, display, add, from, delete, insert, put, prepare, and, extract, to, replaceall,} \\ \text{infoall, for, acutes, a, b, c, d, e, f, g, h, i, j, k, l, m, 0, p, q, r, s, t, u, v, w, x, y, z, A, B, C, D, E, F,} \\ \text{G, H, I, J, K, L, M, , O, P, Q, R, S, T, U, V, W, X, Y, Z,), (, /, *, ascii, \textbackslash,} \end{cases} \quad (2.6)$$

$$P^{9'} = \begin{cases} < SENTENCIA > ::= V_1V_{112} \mid V_3V_{105} \mid V_4V_{113} \mid V_{13}V_{114} \mid V_5V_{114} \mid V_5V_{124} \mid V_7V_{118} \mid V_7V_{125} \\ \qquad \mid V_8V_{114} \mid V_{10}V_{128} \mid V_{12}V_{134} \mid V_{14}V_{135} \mid V_{11}V_{130} \mid V_{15}V_{131} \\ < CREATE > ::= V_1V_{112} \\ < DROP > ::= V_3V_{105} \\ < LIST > ::= V_4V_{113} \\ < DISPLAY > ::= V_{13}V_{114} \\ < ADD > ::= V_5V_{114} \mid V_5V_{124}) \\ < DELETE > ::= V_7V_{118} \mid V_7V_{125} \\ < INSERT > ::= V_8V_{114} \\ < PREPARE > ::= V_{10}V_{128} \\ < EXTRACT > ::= V_{12}V_{134} \\ < REPLACEALL > ::= V_{14}V_{135} \\ < REMOVE > ::= V_{11}V_{130} \\ < INFOALL > ::= V_{15}V_{131} \\ < USING > ::= V_{19}V_{101} \mid STOPWORDS \mid SYMBOLS \mid LABELS \\ < CICLOS > ::=< DICCIONARIOS > \{V_{103}\} \\ < NOMARCH > ::=< ID > V_{24} < ID > \\ < ID > ::=< LETRA > \{V_{102}\} \\ < STRING > ::=< LETRA > \{< LETRA >\} \mid< DIGITO > \{< DIGITO >\} \\ \qquad \mid< DIGITO > \{< LETRA >\} \mid< SIMBOLO > \{< LETRA > \mid < SIMBOLO >\} \\ < LETRA > ::= a\ midb \mid c...z \mid A \mid B \mid C...Z \\ < DIGITO > ::= 0 \mid 1 \mid 2..,9 \\ < DICCIONARIOS > ::= STOPWORDS \mid SYMBOLS \mid LABELS \end{cases} \quad (2.7)$$

$$
P^{9'} = \begin{cases}
< SIMBOLO > ::= \text{ascii} \\
V_1 ::= \text{create} \\
V_2 ::= \text{corpus} \\
V_3 ::= \text{drop} \\
V_4 ::= \text{list} \\
V_5 ::= \text{add} \\
V_6 ::= \text{from} \\
V_7 ::= \text{delete} \\
V_8 ::= \text{insert} \\
V_9 ::= \text{put} \\
V_{10} ::= \text{prepare} \\
V_{11} ::= \text{remove} \\
V_{12} ::= \text{extract} \\
V_{13} ::= \text{display} \\
V_{14} ::= \text{replaceall} \\
V_{15} ::= \text{infoall} \\
V_{16} ::= \text{for} \\
V_{17} ::= \text{acutes} \\
V_{18} ::=) \\
V_{19} ::= (\\
V_{20} ::= \\
V_{21} ::= * \\
V_{22} ::= \text{ascii} \\
V_{23} ::= ; \\
V_{24} ::= . \\
V_{25} ::= \text{stopwords} \\
V_{26} ::= \text{symbols} \\
V_{27} ::= \text{labels} \\
V_{28} ::= \text{and} \\
V_{29} ::= \text{to} \\
V_{30} ::= \text{from} \\
V_{31} ::= 1 \\
V_{32} ::= 2 \\
V_{33} ::= 3 \\
V_{34} ::= 4 \\
V_{35} ::= 5 \\
V_{36} ::= 6 \\
V_{37} ::= 7 \\
V_{39} ::= 8 \\
V_{40} ::= 9 \\
V_{41} ::= \text{a} \\
V_{42} ::= \text{b} \\
V_{43} ::= \text{c} \\
V_{44} ::= \text{d}
\end{cases} \tag{2.8}
$$

$$P^{9'} = \begin{cases} V_{45} ::= \text{e} \\ V_{46} ::= \text{f} \\ V_{47} ::= \text{g} \\ V_{48} ::= \text{h} \\ V_{49} ::= \text{i} \\ V_{50} ::= \text{j} \\ V_{51} ::= \text{k} \\ V_{52} ::= \text{l} \\ V_{53} ::= \text{m} \\ V_{54} ::= \text{n} \\ V_{55} ::= \text{o} \\ V_{56} ::= \text{p} \\ V_{57} ::= \text{q} \\ V_{58} ::= \text{r} \\ V_{59} ::= \text{s} \\ V_{60} ::= \text{t} \\ V_{61} ::= \text{u} \\ V_{62} ::= \text{v} \\ V_{63} ::= \text{w} \\ V_{64} ::= \text{x} \\ V_{65} ::= \text{y} \\ V_{66} ::= \text{z} \\ V_{67} ::= \text{A} \\ V_{68} ::= \text{B} \\ V_{69} ::= \text{C} \\ V_{70} ::= \text{D} \\ V_{71} ::= \text{E} \\ V_{72} ::= \text{F} \\ V_{73} ::= \text{G} \\ V_{74} ::= \text{H} \\ V_{75} ::= \text{I} \\ V_{76} ::= \text{J} \\ V_{77} ::= \text{K} \\ V_{78} ::= \text{L} \\ V_{79} ::= \text{M} \\ V_{80} ::= \text{N} \\ V_{81} ::= \text{O} \\ V_{82} ::= \text{P} \\ V_{83} ::= \text{Q} \\ V_{84} ::= \text{R} \\ V_{85} ::= \text{S} \\ V_{86} ::= \text{T} \\ V_{87} ::= \text{U} \\ V_{88} ::= \text{V} \\ V_{89} ::= \text{W} \\ V_{90} ::= \text{X} \end{cases} \tag{2.9}$$

$$
P^{9'} = \begin{cases}
V_{91} ::= \mathrm{Y} \\
V_{92} ::= \mathrm{Z} \\
V_{93} ::= 0 \\
V_{94} ::= < ID > V_{23} \\
V_{95} ::= < NOMARCH > V_{23} \\
V_{100} ::= V_{18} V_{23} \\
V_{101} ::= < CICLOS > V_{18} \\
V_{102} ::= V_{24} < ID > \\
V_{103} ::= < DICCIONARIOS > V_{28} \\
V_{104} ::= V24 V_{94} \\
V_{105} ::= V_{2} V_{94} \\
V_{106} ::= V_{30} V_{94} \\
V_{107} ::= V_{24} V_{95} \\
V_{108} ::= V_{20} V_{100} \\
V_{109} ::= V_{30} V_{95} \\
V_{110} ::= < STRING > V_{20} \\
V_{111} ::= V_{9} V_{95} \\
V_{112} ::= V_{2} V_{104} \\
V_{113} ::= V_{21} V_{106} \\
V_{114} ::= < ID > V_{107} \\
V_{115} ::= \{ < STRING > \} V_{108} \\
V_{116} ::= \{ V_{110} \} V_{100} \\
V_{117} ::= < NOMARCH > V_{111} \\
V_{118} ::= < DICCIONARIOS > V_{109} \\
V_{119} ::= < USING > V_{111} \\
V_{120} ::= V_{20} V_{115} \\
V_{121} ::= V_{19} V_{116} \\
V_{122} ::= < STRING > V_{119} \\
V_{123} ::= V_{20} V_{117} \\
V_{124} ::= < DICCIONARIOS > V_{120} \\
V_{125} ::= < DICCIONARIOS > V_{121} \\
V_{126} ::= V_{24} V_{122} \\
V_{127} ::= < ID > V_{123} \\
V_{128} ::= < ID > V_{126} \\
V_{129} ::= < STRING > V_{127} \\
V_{130} ::= V_{17} V_{127} \\
V_{131} ::= < ID > V_{127} \\
V_{132} ::= V_{29} V_{129} \\
V_{133} ::= V_{16} V_{129} \\
V_{134} ::= < STRING > V_{132} \\
V_{135} ::= < STRING > V_{133}
\end{cases} \tag{2.10}
$$

Capítulo 3

Implementación de la gramática

3.1. Implementación en Java

Se ha elegido a Java como lenguaje de programación para la implementación de la gramática y del lenguaje en general, por varias razones. Entre las más importantes se tienen las siguientes:

- **Java es orientado a objetos.**

- **Arquitectura neutral, portátil y robusta:**

  - es neutral: al adoptar un sistema de código binario que es independiente de aquitecturas *hardware*, sistemas operativos.

  - es portátil: al definir de forma precisa los tipos y tamaños de los datos.

  - es robusta: al poseer un chequeo del código tanto en tiempo de compilación como de ejecución.

- **Simple:** posee las estructuras mínimas de un lenguaje de programación tradicional, sin añadir ninguna estructura más.

- **Independiente de la plataforma:** Java se compila a un formato de código de byte que puede ser leído e interpretado por muchas plataformas, incluyendo *Windows*, *Solaris 2.3*, *Linux*, *Mac OS*, etc.

- **Seguro:** el código de Java puede ser ejecutado en un entorno que prohíbe la introducción de virus, borrar y modificar ficheros o la ejecución de operaciones que provoquen la caída del ordenador y la pérdida de datos.

- **Multithread:** un único programa Java puede procesar diferentes cosas de forma independiente y continua.

- **Cadenas de caracteres(*String*):** Java tiene como uno de sus objetos basicos el *String*. Un *String* no es un *array* de caracteres, como en C, Hay dos tipos de *Strings*. La clase *String*, que sirve para guardar cadenas de caracteres que no cambian, y la clase *StringBuffer*, que se utiliza para guardar cadenas de caracteres que más tarde serán modificadas.

- **Gestión de memoria:** cuando se necesite crear un objeto se utiliza el operador new (como en C++). Pero no existe la preocupación de borrarlo ni de liberar

memoria que está consuminedo ese objeto, ya que Java lo háce por si mismo, y de forma automática, cuando el objeto ya no sea necesario.

- **Uso de expresiones regulares:** tiene un amplio abanico de posibilidades, principalmente para hacer búsquedas, para sustituir ocurrencias y para comprobar formaciones de cadenas.

- **Arreglos:** en Java los arreglos son objetos. No son un simple puntero a la colección de tipos de datos. El acceso a los elementos del array es igual que en C, pero aquí aparece otra sorpresa: Java checa constantemente que estemos accediendo dentro del rango de valores del arreglo.

- ***Booleano:*** Java ha añadido el tipo de datos boolean. Los dos únicos valores que pueden admitir las variables *boolean* son *true* y *false.* Y a diferencia de C, un tipo *booleano* no puede ser convertido a un tipo de datos numérico[1] .

3.2. Construcción del lenguaje

El lenguaje para el preprocesamiento de Textos está compuesto por dos módulos: análisis y síntesis (ver figura 3.1). El módulo de análisis tiene tres fases: análisis léxico, sintáctico y semántico. El módulo de síntesis tiene una fase de generación de resultados.

[1] www.gui.uva.es/login/16/java.html

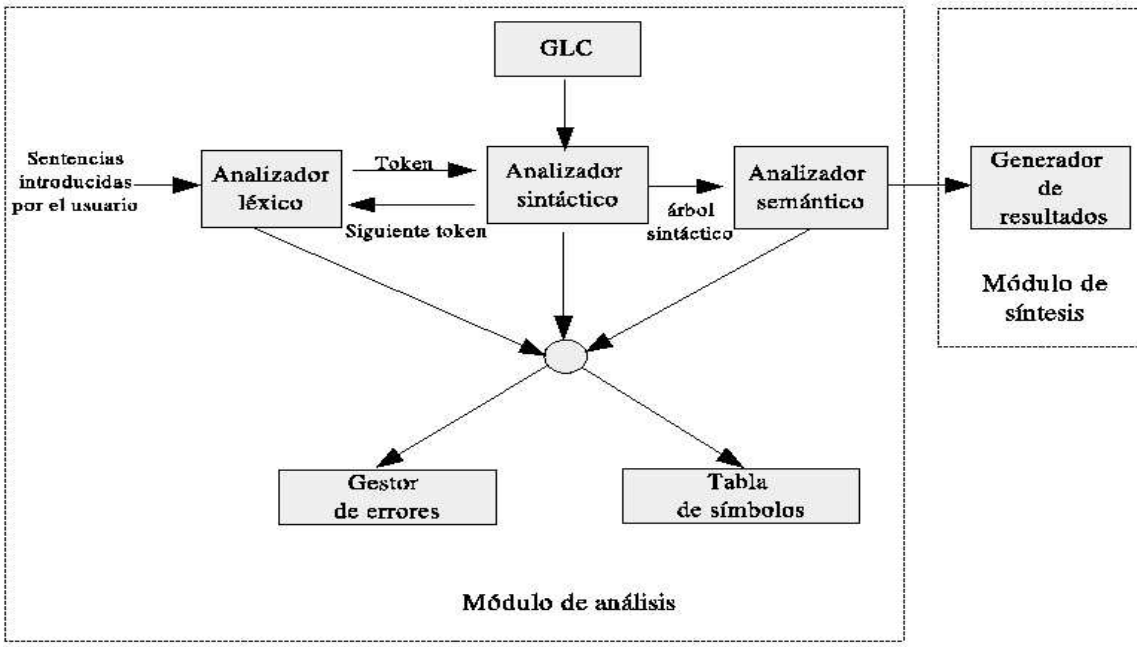


Figura 3.1: Módulos análisis/síntesis

3.2.1. Módulo de análisis

El módulo de análisis verifica si las sentencias introducidas por el usuario son correctas, y recoge la información necesaria de las tablas de símbolos[2] para el módulo de síntesis. En caso de que el módulo de análisis detecte la existencia de errores en las sentencias, se envían los mensajes correspondientes al usuario por medio del manejador de errores.

Analizador léxico

El analizador léxico o *scanner* lee los caracteres (de las sentencias introducidas por el usuario) y constituye una cadena de lexemas.
Los lexemas son agrupaciones de caracteres que constituyen los símbolos terminales de la gramática. Los lexemas se agrupan en componentes léxicos o *tokens* que son conjuntos de símbolos terminales de la gramática que se pueden tratar juntos como unidades sintácticas, con un determinado significado sintáctico y semántico.
Los *tokens* pueden ser específicos como operadores lógicos o matemáticos y palabras reservadas, o no específicos como constantes, variables.
A continuación se presentan algunos tokens válidos para el lenguaje:

- palabras reservadas: *prepare, drop, insert, from, put*
- sígnos de puntuación: .,∗,(,),;
- identificadores: *trec, eurogov, hola*

[2] Es una estructura de datos que almacena la información recogida en el módulo de análisis, para emplear dicha información en el módulo de síntesis.

Analizador sintáctico (*parsing*)

Frente a la simplicidad del reconocimiento de lenguajes regulares mediante autómatas, donde la única variación consiste en construir un autómata como la unión de muchos otros subautómatas, el "*parsing*" (reconocimiento de que una cadena perteneciente a un lenguaje) admite más posibilidades al usarse gramáticas libres de contexto.
La forma más inmediata consiste en convertir cada regla sintáctica[3] en un procedimiento o función, la parte izquierda de la regla da nombre a dicha función y la derecha indica qué procedimientos y en qué secuencia deben ser invocados para verificar la aplicabilidad de la regla. Como los símbolos terminales no tienen procedimiento asociado, al encontrarse uno, se comprueba si coincide con el símbolo siguiente y se prosigue el análisis hasta devolver un código de aceptación; en el caso de haber diferido, se intenta aplicar otra regla que, por tener el mismo símbolo no terminal a la izquierda, estará integrada en la misma función. Si no existiera más reglas o no fuesen aplicables, se devolvería un código de error a la función que invocó.

En conclusión, el analizador sintáctico toma como entrada los *tokens* que recibe del analizador léxico y comprueba si éstos llegan en orden correcto (orden con respecto a la sintaxis del lenguaje[4]).

A continuación se muestran algunos ejemplos de construcciones sintácticas válidas y aceptadas por el lenguaje de preprocesamiento:

[3] Ver capítulo 2, sección 2.2.
[4] Ver capítulo 2

1. `create corpus trec;`
2. `add stopwords from temp.txt;`
3. `insert trec . ism001.txt;`
4. `display trec . introduccion.txt;`
5. `list * from trec;`
6. `prepare trec . ism001.txt using(stopwords and symbols) put salida.txt;`
7. `drop corpus trec;`
8. `remove all acutes trec . uno.txt put temp.txt;`
9. `infoall eurogov . uno.txt put salida.txt;`

Analizador semántico

La función que realiza el analizador semántico es verificar que las sentencias tecleadas por el usuario tengan significado en el lenguaje y no sean un absurdo.
En general el analizador semántico depende en gran medida de los análizadores léxico y sintáctico.
El haber programado la gramática en el lenguaje Java, proporcionó un mayor control sobre el código generado, sin embargo se invirtió una mayor cantidad de tiempo en la programación del módulo de análisis.

3.2.2. Módulo de síntesis

El módulo de síntesis se encarga de la generación de los resultados, una vez que el módulo de análisis ha verificado que las sentencias introducidas por el usuario son correctas.

3.3. Implementación en Flex y Bison

Existe un par de herramientas que hacen más fácil el desarrollo de compiladores a partir de la especificación de los lenguajes en expresiones regulares y de reglas BNF. Estos programas, llamados *flex* y *bison*, fueron utilizados para comprobar si la gramática fue construida correctamente. Se tradujo la gramática diseñada en el formato requerido por *flex* y la salida obtenida de éste fue usada como archivo de entrada para *bison*.
El programa *bison* a su vez generó un archivo de salida con código fuente en lenguaje C que, después de ser compilado, generó un programa con la capacidad de verificar la sintaxis del lenguaje de preprocesamiento.
En el apéndice C se describe a detalle el uso de *Flex* y *Bison* para el proyecto.

Capítulo 4

Pruebas

4.1. Pruebas con TREC-5

A continuación se muestra el conjunto de sentencias introducidas por el usuario para realizar el preprocesamiento del *corpus* TREC-5, usando el programa generado en Java. La figura 4.1 muestra un ejemplo del *corpus* TREC-5 sin preprocesar, posteriormente con ayuda de las siguientes sentencias se puede observar como se realiza el preprocesamiento del mismo eliminando las etiquetas de formato, *stopwords* y símbolos. El resultado de ejecutar estas sentencias es mostrado en la figura 4.2[1] .

```
create corpus trec;
insert trec . ism001.txt;
list * from trec;
display trec . introduccion.txt;
prepare trec . ism001.txt using(stopwords and symbols and labels)
put salida.txt;
```

[1] Notese que no se han eliminado todos los símbolos como: \ el cual no se ha incluido como elemento del archivo *symbols.txt.*

```
<DOC>
<DOCNO> SP94-0202041 </DOCNO>
<ARTNUM> 0202041 </ARTNUM>
<HEADLINE>
 Para viajar en avión
</HEADLINE>
<TEXT>
hay que tener buen oído
El NORTE / Especial
        Como consecuencia de los cambios en la presión atmosférica que llevan
aparejados, los viajes en avión pueden producir serias alteraciones en el oído
medio, sobre todo si estaba previamente lesionado.
      En condiciones normales, gracias a la compensación que se produce
mediante la entrada de aire a través de la trompa de Eustaquio ésta comunica el
oído medio con la nasofaringe la presión que existe a ambos lados del tímpano es
expandirse.
</TEXT>
</DOC>
```

Figura 4.1: *Corpus* TREC-5 sin preprocesar

```
Para viajar  avión
tener buen oído
NORTE / Especial
        Como consecuencia  cambios presión atmosférica  llevan
aparejados viajes  avión pueden producir serias alteraciones  oído
medio sobre todo si estaba previamente lesionado
      condiciones normales, gracias  compensación  produce
mediante  entrada  aire  través trompa  Eustaquio comunica
oído medio nasofaringe  presión existe ambos lados  tímpano
expandirse
```

Figura 4.2: *Corpus* TREC-5 preprocesado

4.2. Pruebas con Eurogov

A continuación se muestra el conjunto de sentencias introducidas por el usuario para obtener información importante (en tareas de PLN). Se realizaron las pruebas tomando como entrada el archivo *muestra.txt*[2] el cual contiene información en formato *Euro-GOV*. Se obtuvo el URL de la página *web*[3] (para posteriorente buscarla en *Internet*), y así tener un punto de referencia para comparar la información real con la información arrojada por el lenguaje para el preprocesamiento de Textos.

La sentencia 3 obtiene toda la información importante del archivo *muestra.txt*, esta sentencia tan sencilla tiene inmersa una gran variedad de expresiones regulares que

[2] Ver figura 4.4
[3] Ver figura 4.3.

juntas hacen que esta función tenga vida, dando como resultado el archivo *datos.txt*[4] . La sentencia 4 termina de limpiar la información, quitando código basura o ruido (en este caso los llamados acutes de *html*). Dando como resultado el archivo *final.txt*[5] . Por último la sentencia 5 preprocesa el archivo *final.txt*, eliminado *stopwords* y símbolos. Dando como resultado el archivo *final2.txt*[6] .

1. `create corpus eurogov;`
2. `insert eurogov . muestra.txt;`
3. `infoall eurogov . muestra.txt put datos.txt;`
4. `remove acutes eurogov . datos.txt put final.txt;`
5. `prepare eurogov . final.txt using (stopwords and symbols) put final2.txt;`

[4] Ver figura 4.5
[5] Ver figura 4.6
[6] Ver figura 4.7

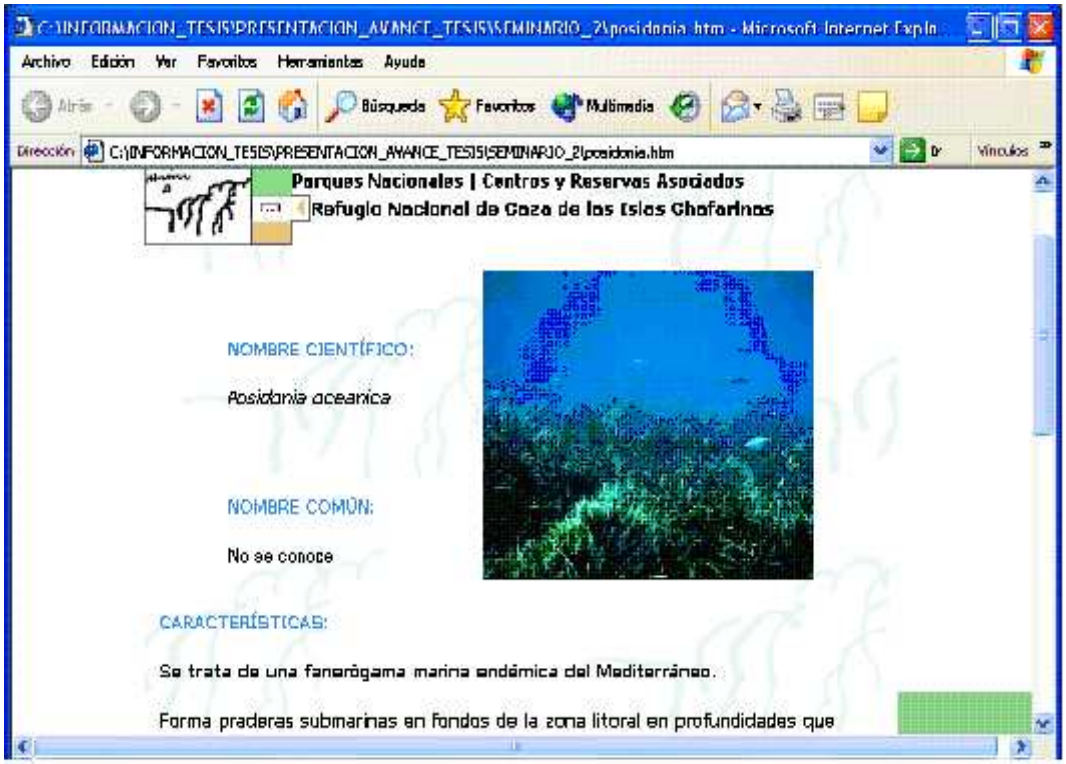


Figura 4.3: http://www.mma.es/eus/parques/centasoc/posidonia.htm

```
<EuroGOV>doc
url="http://www.mma.es/eus/parques/centasoc/posidonia.htm"
id="Ees-002-2130916"
md5="96c5c5c34a8d5bc83adecb8b1ee0c017"
fetchDate="Sun Nov07 19:37:20 MET 2004"
contentType="">
<EuroGOV>content>
<![CDATA[
<html> <!- #BeginTemplate "/Templates/plantilla_parques_centasoc.dwt" -->
<head>
<!- Content Generated by Documentum 4 -->
<META name="a_category" content="Parques">
<META name="keywords" content="">
<META name="r_modify_date" content="2004-03-23 10:12:06">
<META name="title" content="Caracter?sticas y h?bitat de la posidonia">
<!- #BeginEditable "doctitle" -->
<title> </title>
<!- #EndEditable -->
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<link rel="stylesheet" href="/library/hoja_estilos.css">
<script language="JavaScript1.2" src="/js/general.js"></script>
</head>
<body leftmargin="0" margin width="0" marginheight="0" topmargin="0" background="/imagenes/plantillas/fondos_parques/f_parqgen.gif"
onLoad="comprobarDireccion();inicio()">
<table width="780" border="0" cellspacing="0" cellpadding="0" align="center" vspace="0" hspace="0">
  <tr>
    <td>
      <table width="100%" border="0" cellspacing="0" cellpadding="0" vspace="0" hspace="0">
        <tr>          <td id="logotipo" rowspan="2" width="215" bgcolor="#FFD700"><a href="/index.htm"><img src="/imagenes/comun/logo.gif"
width="215" height="85" border="0" alt="Home"></a></td>
          <td height="18" bgcolor="D5D4D4" colspan="2">
            <table width="565" cellspacing="0" cellpadding="0" border="0"   bgcolor="#EBEBEB">
              <tr>
                <td height="18" width="355" bgcolor="D5D4D4">  </td>
                <td height="18" align="center" width="30"><a href="javascript:traduccion('');" ><img src="/imagenes/comun/banderas/castellano.gif"
width="22" height="13" alt="Espa&ntilde;ol" border="0"></a> </td>
                <td height="18" align="center" width="30"><a href="javascript:traduccion('cat');" ><img src="/imagenes/comun/banderas/catalan.gif"
width="22" height="13" alt="Catal&agrave;" border="0"></a></td>
              </tr>
            </table>
          </td>
        </tr>
        <tr>
<td height="64" width="565" bgcolor="D5D4D4" align="center" id="superior"> <object id="flashsuIE" value="/flash/actualidad.swf"
classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=4,0,2,0" width="391" height="64"
align="absmiddle" >
            <embed src="/flash/actualidad.swf" quality=high
pluginspage="http://www.macromedia.com/shockwave/download/index.cgi?P1_Prod_Version=ShockwaveFlash" type="application/x-
shockwave-flash" width="391" height="64" align="absmiddle">
            </embed>
          </object></td>
          <td height="64" width="114" bgcolor="D5D4D4" align="center" > </td>
</tr>
      </table>
      <table width="780" height="100%" border="0" cellspacing="0" cellpadding="0" align="center" vspace="0" hspace="0">
       <tr valign="top">  <td width="100" bgcolor="#FFD700" rowspan="2" id="izquierdo"><object id="flashizIE"
value="/flash/informacion_general.swf" classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=5,0,42,0" width="100" height="450"><param
name=movie id="flashizNS" value="/flash/informacion_general.swf">
            <param name=quality value=high> <param name="MENU" value="false">
            <embed src="/flash/informacion_general.swf" quality=high
pluginspage="http://www.macromedia.com/shockwave/download/index.cgi?P1_Prod_Version=ShockwaveFlash" type="application/x-
shockwave-flash" width="100" height="450">
            </embed>
          </object>  </td><td >  <table width="100%" border="0" cellspacing="0" cellpadding="0" vspace="0" hspace="0">
            <tr>
            </tr>
            <tr>
             <td rowspan="2" width="30" align="left" valign="top"> <img src="/imagenes/plantillas/t2/areas.gif" width="30" height="20"><img
src="/imagenes/plantillas/t3/naranja_c.gif" width="30" height="17"></td>
              <td  height="20" colspan="3" align="left">
                <table width="100%" border="0" cellspacing="0" cellpadding="0" vspace="0" hspace="0">
                 <tr>
<td  align="left" height="20" width="15" valign="top" background="/imagenes/plantillas/fondo_fareas.gif"><a href="#"
onClick="history.back()"><img src="/imagenes/plantillas/bf_naranja_c.gif" width="15" height="20" border="0"   alt="Atr&aacute;s"></a></td>
                  <td align="left" height="15" class="titulos" background="/imagenes/plantillas/fondo2/naranja_c.gif" width="444"><!-
#BeginEditable "titulo_pagina" -->Refugio
                    Nacional de Caza de las Islas Chafarinas<!- #EndEditable --></td>
                 </tr>
                </table>
              </td>
            </tr>
            <tr>
             <td align="left" colspan="2" valign="top" height="6"> </td>
             <td height="6"></td>
            </tr>
          </table>
          <table valign="top" width="96%" border="0" cellspacing="0" cellpadding="0" align="center">
            <tr>
             <td align="justify" span class="contenido">  <div><!- #BeginEditable "contenido" --><br>
                <table width="82%" border="0" align="center">
                 <tr>
                  <td width="54%" height="111">
                   <p><font size="2" color="#398AB6">NOMBRE CIENT&iacute;FICO:</font></p>
                   <p><font size="4"><i><font size="2"> Posidonia oceanica</font></i></font></p>
                  </td>
                 </tr>
                 <tr>
                  <td width="54%">
                   <p><font size="2" color="#398AB6">NOMBRE COM&Uacute;N:</font></p>
                  <p><font size="2" face="Arial">No se conoce</font></p>
                  </td>
                 </tr>
                </table>
                <p><font size="2" color="#398AB6">CARACTER&iacute;STICAS:</font><br>
                </p>
                <p><font size="2">Se trata de una fanerÓgama marina end&eacute;mica
                 del Mediterr&aacute;neo. </font> </p>
                <p><font size="2">Forma praderas submarinas en fondos de la
                zona litoral en profundidades que oscilan desde la superficie
                hasta los 35-40 metros, llegando excepcionalmente, en zonas
                de aguas muy transparentes, a profundidades cercanas al
                centenar de metros.</font></p>
                <p><font size="2">Las praderas de <i>Posidonia</i> constituyen
                un ecosistema sumergido de gran importancia ecol&oacute;gica, donde
                puede instalarse una gran variedad de fauna y flora. Dada
                su heterogeneidad es utilizada como zona de cr&iacute;a o protecci&oacute;n
                de j&oacute;venes de numerosas especies de peces litorales, y adem&aacute;s,
                estas praderas son excelentes indicadoras de la calidad
                ambiental de su litoral sumergido.</font> </p>
                <p><font size="2">Las praderas de <i>Posidonia</i> de Chafarinas
                se localizan principalmente al sur de la Isla del Rey y
                de Isabel II, en Playa Larga en la isla del Congreso y con
                menor grado de desarrollo en la zona de La Sangre y de las
                Cuevas de Lara, siempre en esta misma isla.</font></p>
<!- #EndEditable --></div>
                </td>
               </tr>
              </table>
             </td>
             <embed src="/flash/grandes_temas.swf" quality=high
pluginspage="http://www.macromedia.com/shockwave/download/index.cgi?P1_Prod_Version=ShockwaveFlash" type="application/x-
shockwave-flash" width="110" height="377">
             </embed>
            </object></td>
        </tr>
        <tr>
         <td align="center" class="pie_pagina" valign="bottom">  <br>
          <hr noshade size="1" width="250">
          <font face="Arial Narrow" size="1">El Ministerio de Medio Ambiente
          agradece sus comentarios.Copyright ? 2004 Ministerio de Medio Ambiente</font></td>
        </tr>
       </table>
      </td>
    </tr>
</table>
</body>
<!- #EndTemplate --></html>
]]>
</EuroGOV>content>
</EuroGOV>doc>
```

Figura 4.4: muestra.txt

<EuroGOV:doc
url="http://www.mma.es/eus/parques/centasoc/posidonia.htm"
id="Ees-002-2130916"
md5="96c5c5c34a8d5bc83adecb8b1ee0d017"
fetchDate="Sun Nov07 19:37:20 MET 2004"
contentType="">
<![CDATA[

Refugio
Nacional de Caza de las Islas Chafarinas

NOMBRE CIENTíFICO:
Posidonia oceanica
NOMBRE COMúN:
No se conoce
CARACTERíSTICAS:
Se trata de una fanerógama marina endémica
del Mediterráneo.
Forma praderas submarinas en fondos de la
zona litoral en profundidades que oscilan desde la superficie
hasta los 35-40 metros, llegando excepcionalmente, en zonas
de aguas muy transparentes, a profundidades cercanas al
centenar de metros.
Las praderas de Posidonia constituyen
un ecosistema sumergido de gran importancia ecológica, donde
puede instalarse una gran variedad de fauna y flora. Dada
su heterogeneidad es utilizada como zona de cría o protección
de jóvenes de numerosas especies de peces litorales, y además,
estas praderas son excelentes indicadoras de la calidad
ambiental de su litoral sumergido.
Las praderas de Posidonia de Chafarinas
se localizan principalmente al sur de la Isla del Rey y
de Isabel II, en Playa Larga en la isla del Congreso y con
menor grado de desarrollo en la zona de La Sangre y de las
Cuevas de Lara, siempre en esta misma isla.
Dos son los aspectos de importancia de la
pradera de Chafarinas, el primero, el hecho de que constituyan
las praderas más occidentales de las costas norteafricanas
del Mediterráneo y el segundo, el carácter reciente de su
instauración en el archipiélago, en contra de la opinión
de algún autor que las considera praderas relictas. En efecto,
no aparecen cartografiadas en el levantamiento de la carta
hidrográfica del archipiélago realizada por la Comisión
Hidrográfica en 1924 y se ha podido observar que crecen
por encima de un cable telegráfico que unía las islas con
la costa de Marruecos. Adestacar, igualmente, como una
prueba de su probable origen, la presencia del fenotipo
de hojas anchas hasta ahora sólo conocido de las costas
argelinas.
El Ministerio de Medio Ambiente
agradece sus comentarios.

Figura 4.5: datos.txt

<EuroGOVdoc
url="http://www.mma.es/eus/parques/centasoc/posidonia.htm"
id="Ees-002-2130916"
md5="96c5c5c34a8d5bc83adecb8b1ee0c017"
fetchDate="Sun Nov07 19:37:20 MET 2004"
contentType="">
<![CDATA[
Refugio
Nacional de Caza de las Islas Chafarinas
NOMBRE CIENTÍFICO:
Posidonia oceanica
NOMBRE COMÚN:
No se conoce
CARACTERÍSTICAS:
Se trata de una fanerógama marina endémica
del Mediterráneo.
Forma praderas submarinas en fondos de la
zona litoral en profundidades que oscilan desde la superficie
hasta los 35-40 metros, llegando excepcionalmente, en zonas
de aguas muy transparentes, a profundidades cercanas al
centenar de metros.
Las praderas de Posidonia constituyen
un ecosistema sumergido de gran importancia ecológica, donde
puede instalarse una gran variedad de fauna y flora. Dada
su heterogeneidad es utilizada como zona de cría o protección
de jóvenes de numerosas especies de peces litorales, y además,
estas praderas son excelentes indicadoras de la calidad
ambiental de su litoral sumergido.
Las praderas de Posidonia de Chafarinas
se localizan principalmente al sur de la Isla del Rey y
de Isabel II, en Playa Larga en la isla del Congreso y con
menor grado de desarrollo en la zona de La Sangre y de las
Cuevas de Lara, siempre en esta misma isla.
Dos son los aspectos de importancia de la
pradera de Chafarinas, el primero, el hecho de que constituyan
las praderas más occidentales de las costas norteafricanas
del Mediterráneo y el segundo, el carácter reciente de su
instauración en el archipiélago, en contra de la opinión
de algún autor que las considera praderas relictas. En efecto,
no aparecen cartografiadas en el levantamiento de la carta
hidrográfica del archipiélago realizada por la Comisión
Hidrográfica en 1924 y se ha podido observar que crecen
por encima de un cable telegráfico que unía las islas con
la costa de Marruecos. A destacar, igualmente, como una
prueba de su probable origen, la presencia del fenotipo
de hojas anchas hasta ahora sólo conocido de las costas
argelinas.
El Ministerio de Medio Ambiente
agradece sus comentarios.Copyright ?2004 Ministerio de Medio Ambiente

Figura 4.6: final.txt

EuroGOV doc
url http www mma es eus parques centasoc posidonia htm
id Ees -002 -2130916
md5 96c5c5c34a8d5bc83adecb8b1ee0c017
fetchDate Sun Nov 07 19 37 20 MET 2004
contentType
CDATA
Refugio
Nacional Caza Islas Chafarinas
NOMBRE CIENTÍFICO
Posidonia oceanica
NOMBRE COMÚN
No conoce
CARACTERÍSTICAS
Se trata fanerógama marina endémica
Mediterráneo
Forma praderas submarinas fondos
zona litoral profundidades oscilan superficie
35 -40 metros llegando excepcionalmente zonas
aguas transparentes profundidades cercanas
centenar metros
praderas Posidonia constituyen
ecosistema sumergido gran importancia ecológica
puede instalarse gran variedad fauna flora Dada
heterogeneidad utilizada zona cría protección
jóvenes numerosas especies peces litorales además
praderas excelentes indicadoras calidad
ambiental litoral sumergido
praderas Posidonia Chafarinas
localizan principalmente sur Isla Rey
Isabel II Playa Larga isla Congreso
menor grado desarrollo zona Sangre
Cuevas Lara siempre misma isla
Dos aspectos importancia
pradera Chafarinas primero hecho constituyan
praderas occidentales costas norteafricanas
Mediterráneo segundo carácter reciente
instauración archipiélago opinión
algún autor considera praderas relictas En
aparecen cartografiadas levantamiento carta
hidrográfica archipiélago realizada Comisión
Hidrográfica 1924 ha podido observar crecen
encima cable telegráfico unía islas
costa Marruecos destacar igualmente
prueba probable origen presencia fenotipo
hojas anchas sólo conocido costas
argelinas
Ministerio Medio Ambiente
agradece sus comentarios Copyright 2004 Ministerio Medio Ambiente

Figura 4.7: final2.txt

Capítulo 5

Conclusiones

Se ha presentado una nueva alternativa para el preprocesamiento de textos. La herramienta desarrollada permitirá al investigador de PLN evitar programar procedimientos de textos *ad-hoc* para sus *corpora*. Tal vez el mayor aporte de este trabajo sea el hecho de aplicar procedimientos estándar para la etapa de proprocesamiento, facilitando de esta manera el análisis comparativo de las técnicas de PLN, sin importar el investigador que lo realice.
Se diseño la gramática libre de contexto, se paso dicha gramática a la forma normal de chomsky y se implemento dicha gramática en el lenguaje Java.

Las pruebas realizadas sobre los corpora: EuroGOV, TREC-5, etc; mostraron su utilidad como herramienta de apoyo en tareas de preprocesamiento de lenguaje natural. Resultados preliminares de este trabajo fueron presentados en el *PRIMER FORO NACIONAL DE ESTUDIANTES DE LINGÜISTICA Y LITERATURA"*, celebrado en la ciudad de Hermosillo Sonora, México, los dás 14, 15, 16 de abril del 2005.
La gramática final, así como algunas pruebas se presentaron en el CORE ("6^{to} *CONGRESO ESTUDIANTIL DE COMPUTACION*"), celebrado en la ciudad de México, D.F., los días 12 y 13 de Mayo del 2005, en el Centro de Investigación en Computación

del Instituto Politécnico Nacional[3].

Bibliografía

[1] Rodríguez Galdo Alberto. Expresiones regulares en java. *http://bulmalug.net/body.phtml?nldNoticia=770*, 2005.

[2] Nandayapa Parada Arturo. *Uso de gramáticas de rasgos para reconocimiento de oraciones en español*. Reporte interno de investigación LANIA-RT-97-14, 1997.

[3] Erika Hernández, David Pinto, Héctor Jiménez, and Jesús Lavalle. Un lenguaje de apoyo para tareas de preprocesamiento de textos. *Avances en la ciencia de la Computación en México, Research on Computing Sciencie, 13, 2005, pp.91-98, ISSN: 1665-9899*, 2005.

[4] Palacios Pérez José Juan. *LENGUAJES FORMALES Y AUTÓMATAS*. Benemérita Universidad Autonoma de Puebla, 2002.

[5] Retrieval Conference TREC. *http://trec.nist.gov*, 13/sep/2004.

[6] Pinto David y Jiménez Héctor. Recuperación de información. *Notas de Academia*, 2003.

[7] Hopcroft John y Motwan Rajeev. *Introducción a la teoría de Automatas, Lenguajes y Computación*. Cambridge University Press, 2002.

Apéndice A

Transformando la GLC a su FNC

En este apéndice se muestra el desarrollo completo que sufre la Gramática Libre de Contexto, para llegar a su Forma Normal de Chomsky. El conjunto de producciones está denotado de la siguiente manera:
$P^{\#'} = \{$ donde # es el número asignado al nuevo conjunto de producciones (gramática equivalente).

El proceso es relativamente sencillo, consiste en ir creando una variable terminal hasta que se cumpla lo siguiente: si el lado derecho de cada una de las reglas está constituido por un único símbolo terminal o por dos no terminales.

$$
P^{2'} = \begin{cases}
< SENTENCIA > ::= V_1V_2V_{24} < ID > V_{23} \\
\quad | \; V_3V_2 < ID > V_{23} \\
\quad | \; V_4V_{21}V_{30} < ID > V_{23} \\
\quad | \; V_{13} < ID > V_{24} < NOMARCH > V_{23} \\
\quad | \; V_5 < DICCIONARIOS > V_{30} < NOMARCH > V_{23} \\
\quad | \; V_5 < DICCIONARIOS > V_{19}\{< STRING >\}V_{18}V_{23} \\
\quad | \; V_7 < DICCIONARIOS > V_{30} < NOMARCH > V_{23} \\
\quad | \; V_7 < DICCIONARIOS > V_{19}\{< STRING >\}V_{18}V_{23} \\
\quad | \; V_8 < ID > V_{24} < NOMARCH > V_{23} \\
\quad | \; V_{10} < ID > V_{24} < NOMARCH >< USING > V_9 < NOMARCH > V_{23} \\
\quad | \; V_{12} < STRING > V_{29} < STRING >< ID > V_{24} < NOMARCH > V_9 < NOMARCH > V_{23} \\
\quad | \; V_{14} < STRING > V_{16} < STRING >< ID > V_{24} < NOMARCH > V_9 < NOMARCH > V_{23} \\
\quad | \; V_{11}V_{17} < ID > V_{24} < NOMARCH > V_9 < NOMARCH > V_{23} \\
\quad | \; V_{15} < ID > V_{24} < NOMARCH > V_9 < NOMARCH > V_{23} \\
< CREATE > ::= V_1V_2V_{24} < ID > V_{23} \\
< DROP > ::= V_3V_2 < ID > V_{23} \\
< LIST > ::= V_4V_{21}V_{30} < ID > V_{23} \\
< DISPLAY > : V_{13} < ID > V_{24} < NOMARCH > V_{23} : \\
< ADD > ::= V_5 < DICCIONARIOS > V_{30} < NOMARCH > V_{23} \\
\quad | \; V_5 < DICCIONARIOS > V_{19}\{< STRING >\}V_{18}V_{23} \\
< DELETE > ::= V_7 < DICCIONARIOS > V_{30} < NOMARCH > V_{23} \\
\quad | \; V_7 < DICCIONARIOS > V_{19}\{< STRING >\}V_{18}V_{23} \\
< INSERT > ::= V_8 < ID > V_{24} < NOMARCH > V_{23} \\
< PREPARE > ::= V_{10} < ID > V_{24} < NOMARCH >< USING > V_9 < NOMARCH > V_{23} \\
< EXTRACT > ::= V_{12} < STRING > V_{29} < STRING >< ID > V_{24} < NOMARCH > V_9 < NOMARCH > V_{23} \\
< REPLACEALL > ::= V_{14} < STRING > V_{16} < STRING >< ID > V_{24} < NOMARCH > V_9 < NOMARCH > V_{23} \\
< REMOVE > ::= V_{11}V_{17} < ID > V_{24} < NOMARCH > V_9 < NOMARCH > V_{23} \\
< INFOALL > ::= V_{15} < ID > V_{24} < NOMARCH > V_9 < NOMARCH > V_{23} \\
< USING > ::= V_{19} < CICLOS > V_{18} \mid STOPWORDS \mid SYMBOLS \mid LABELS \\
< CICLOS > ::=< DICCIONARIOS > \{V_{103}\}
\end{cases} \tag{A.1}
$$

$$
P^{2'} = \begin{cases}
< NOMARCH > ::= < ID > V_{24} < ID > \\
< ID > ::= < LETRA > \{V_{102}\} \\
< STRING > ::= < LETRA > \{< LETRA >\} \mid < DIGITO > \{< DIGITO >\} \\
\qquad \mid < DIGITO > \{< LETRA >\} \mid < SIMBOLO > \{< LETRA > \mid < SIMBOLO >\} \\
< LETRA > ::= a \mid b \mid c...z \mid A \mid B \mid C...Z \\
< DIGITO > ::= 0 \mid 1 \mid 2..,9 \\
< DICCIONARIOS > ::= STOPWORDS \mid SYMBOLS \mid LABELS \\
< SIMBOLO > ::= \text{ascii} \\
V_1 ::= \text{create} \\
V_2 ::= \text{corpus} \\
V_3 ::= \text{drop} \\
V_4 ::= \text{list} \\
V_5 ::= \text{add} \\
V_6 ::= \text{from} \\
V_7 ::= \text{delete} \\
V_8 ::= \text{insert} \\
V_9 ::= \text{put} \\
V_{10} ::= \text{prepare} \\
V_{11} ::= \text{remove} \\
V_{12} ::= \text{extract} \\
V_{13} ::= \text{display} \\
V_{14} ::= \text{replaceall} \\
V_{15} ::= \text{infoall} \\
V_{16} ::= \text{for} \\
V_{17} ::= \text{acutes} \\
V_{18} ::=) \\
V_{19} ::= (\\
V_{20} ::= \\
V_{21} ::= * \\
V_{22} ::= \text{ascii} \\
V_{23} ::= ; \\
V_{24} ::= . \\
V_{25} ::= \text{stopwords} \\
V_{26} ::= \text{symbols} \\
V_{27} ::= \text{labels} \\
V_{28} ::= \text{and} \\
V_{29} ::= \text{to} \\
V_{30} ::= \text{from} \\
V_{31} ::= 1 \\
V_{32} ::= 2 \\
V_{33} ::= 3 \\
V_{34} ::= 4 \\
V_{35} ::= 5 \\
V_{36} ::= 6 \\
V_{37} ::= 7 \\
V_{39} ::= 8 \\
V_{40} ::= 9
\end{cases} \tag{A.2}
$$

$$P^{2'} = \begin{cases} V_{45} ::= \text{e} \\ V_{46} ::= \text{f} \\ V_{47} ::= \text{g} \\ V_{48} ::= \text{h} \\ V_{49} ::= \text{i} \\ V_{50} ::= \text{j} \\ V_{51} ::= \text{k} \\ V_{52} ::= \text{l} \\ V_{53} ::= \text{m} \\ V_{54} ::= \text{n} \\ V_{55} ::= \text{o} \\ V_{56} ::= \text{p} \\ V_{57} ::= \text{q} \\ V_{58} ::= \text{r} \\ V_{59} ::= \text{s} \\ V_{60} ::= \text{t} \\ V_{61} ::= \text{u} \\ V_{62} ::= \text{v} \\ V_{63} ::= \text{w} \\ V_{64} ::= \text{x} \\ V_{65} ::= \text{y} \\ V_{66} ::= \text{z} \\ V_{67} ::= \text{A} \\ V_{68} ::= \text{B} \\ V_{69} ::= \text{C} \\ V_{70} ::= \text{D} \\ V_{71} ::= \text{E} \\ V_{72} ::= \text{F} \\ V_{73} ::= \text{G} \\ V_{74} ::= \text{H} \\ V_{75} ::= \text{I} \\ V_{76} ::= \text{J} \\ V_{77} ::= \text{K} \\ V_{78} ::= \text{L} \\ V_{79} ::= \text{M} \\ V_{80} ::= \text{N} \\ V_{81} ::= \text{O} \\ V_{82} ::= \text{P} \\ V_{83} ::= \text{Q} \\ V_{84} ::= \text{R} \\ V_{85} ::= \text{S} \\ V_{86} ::= \text{T} \\ V_{87} ::= \text{U} \\ V_{88} ::= \text{V} \\ V_{89} ::= \text{W} \\ V_{90} ::= \text{X} \end{cases} \tag{A.3}$$

$$P^{2'} = \begin{cases} V_{91} ::= \text{Y} \\ V_{92} ::= \text{Z} \\ V_{93} ::= 0 \end{cases} \tag{A.4}$$

$$P^{3'} = \begin{cases} < SENTENCIA > ::= V_1V_2V_{24}V_{94} \\ \quad | \; V_3V_2V_{94} \\ \quad | \; V_4V_{21}V_{30}V_{94} \\ \quad | \; V_{13} < ID > V_{24}V_{95} \\ \quad | \; V_5 < DICCIONARIOS > V_{24}V_{95} \\ \quad | \; V_5 < DICCIONARIOS > V_{19}\{< STRING >\}V_{20}V_{100} \\ \quad | \; V_7 < DICCIONARIOS > V_{30}V_{95} \\ \quad | \; V_7 < DICCIONARIOS > V_{19}\{< STRING > V_{20}\}V_{100} \\ \quad | \; V_8 < ID > V_{24}V_{95} \\ \quad | \; V_{10} < ID > V_{24} < NOMARCH >< USING > V_9V_{95} \\ \quad | \; V_{12} < STRING > V_{29} < STRING >< ID > V_{24} < NOMARCH > V_9V_{95} \\ \quad | \; V_{14} < STRING > V_{16} < STRING >< ID > V_{24} < NOMARCH > V_9V_{95} \\ \quad | \; V_{11}V_{17} < ID > V_{24} < NOMARCH > V_9V_{95} \\ \quad | \; V_{15} < ID > V_{24} < NOMARCH > V_9V_{95} \\ < CREATE > ::= V_1V_2V_{24}V_{94} \\ < DROP > ::= V_3V_2V_{94} \\ < LIST > ::= V_4V_{21}V_{30}V_{94} \\ < DISPLAY > := V_{13} < ID > V_{24}V_{95} \\ < ADD > ::= V_5 < DICCIONARIOS > V_{24}V_{95} \\ \quad | \; V_5 < DICCIONARIOS > V_{19}\{< STRING >\}V_{20}V_{100} \\ < DELETE > ::= V_7 < DICCIONARIOS > V_{30}V_{95} \\ \quad | \; V_7 < DICCIONARIOS > V_{19}\{< STRING > V_{20}\}V_{100} \\ < INSERT > ::= V_8 < ID > V_{24}V_{95} \\ < PREPARE > ::= V_{10} < ID > V_{24} < NOMARCH >< USING > V_9V_{95} \\ < EXTRACT > ::= V_{12} < STRING > V_{29} < STRING >< ID > V_{24} < NOMARCH > V_9V_{95} \\ < REPLACEALL > ::= V_{14} < STRING > V_{16} < STRING >< ID > V_{24} < NOMARCH > V_9V_{95} \\ < REMOVE > ::= V_{11}V_{17} < ID > V_{24} < NOMARCH > V_9V_{95} \\ < INFOALL > ::= V_{15} < ID > V_{24} < NOMARCH > V_9V_{95} \\ < USING > ::= V_{19}V_{101} \mid STOPWORDS \mid SYMBOLS \mid LABELS \\ < CICLOS > ::=< DICCIONARIOS > \{V_{103}\} \\ < NOMARCH > ::=< ID > V_{102} \\ < ID > ::=< LETRA > \{V_{102}\} \\ < STRING > ::=< LETRA > \{< LETRA >\} \mid < DIGITO > \{< DIGITO >\} \\ \quad \mid < DIGITO > \{< LETRA >\} \mid < SIMBOLO > \{< LETRA > \mid < SIMBOLO >\} \\ < LETRA > ::= a \mid b \mid c...z \mid A \mid B \mid C...Z \\ < DIGITO > ::= 0 \mid 1 \mid 2..,9 \\ < DICCIONARIOS > ::= STOPWORDS \mid SYMBOLS \mid LABELS \\ < SIMBOLO > ::= \text{ascii} \end{cases} \tag{A.5}$$

$$P^{3'} = \begin{cases} V_1 ::= \text{create} \\ V_2 ::= \text{corpus} \\ V_3 ::= \text{drop} \\ V_4 ::= \text{list} \\ V_5 ::= \text{add} \\ V_6 ::= \text{from} \\ V_7 ::= \text{delete} \\ V_9 ::= \text{put} \\ V_{10} ::= \text{prepare} \\ V_{11} ::= \text{remove} \\ V_{12} ::= \text{extract} \\ V_{13} ::= \text{display} \\ V_{14} ::= \text{replaceall} \\ V_{15} ::= \text{infoall} \\ V_{16} ::= \text{for} \\ V_{17} ::= \text{acutes} \\ V_{18} ::=) \\ V_{19} ::= (\\ V_{20} ::= \\ V_{21} ::= * \\ V_{22} ::= \text{ascii} \\ V_{23} ::= ; \\ V_{24} ::= . \\ V_{25} ::= \text{stopwords} \\ V_{26} ::= \text{symbols} \\ V_{27} ::= \text{labels} \\ V_{28} ::= \text{and} \\ V_{29} ::= \text{to} \\ V_{30} ::= \text{from} \\ V_{31} ::= 1 \\ V_{32} ::= 2 \\ V_{33} ::= 3 \\ V_{34} ::= 4 \\ V_{35} ::= 5 \\ V_{36} ::= 6 \\ V_{37} ::= 7 \\ V_{39} ::= 8 \\ V_{40} ::= 9 \\ V_{41} ::= \text{a} \\ V_{42} ::= \text{b} \\ V_{43} ::= \text{c} \\ V_{44} ::= \text{d} \end{cases} \tag{A.6}$$

$$P^{3'} = \begin{cases} V_{45} ::= \text{e} \\ V_{46} ::= \text{f} \\ V_{47} ::= \text{g} \\ V_{48} ::= \text{h} \\ V_{49} ::= \text{i} \\ V_{50} ::= \text{j} \\ V_{51} ::= \text{k} \\ V_{52} ::= \text{l} \\ V_{53} ::= \text{m} \\ V_{54} ::= \text{n} \\ V_{55} ::= \text{o} \\ V_{56} ::= \text{p} \\ V_{57} ::= \text{q} \\ V_{58} ::= \text{r} \\ V_{59} ::= \text{s} \\ V_{60} ::= \text{t} \\ V_{61} ::= \text{u} \\ V_{62} ::= \text{v} \\ V_{63} ::= \text{w} \\ V_{64} ::= \text{x} \\ V_{65} ::= \text{y} \\ V_{66} ::= \text{z} \\ V_{67} ::= \text{A} \\ V_{68} ::= \text{B} \\ V_{69} ::= \text{C} \\ V_{70} ::= \text{D} \\ V_{71} ::= \text{E} \\ V_{72} ::= \text{F} \\ V_{73} ::= \text{G} \\ V_{74} ::= \text{H} \\ V_{75} ::= \text{I} \\ V_{76} ::= \text{J} \\ V_{77} ::= \text{K} \\ V_{78} ::= \text{L} \\ V_{79} ::= \text{M} \\ V_{80} ::= \text{N} \\ V_{81} ::= \text{O} \\ V_{82} ::= \text{P} \\ V_{83} ::= \text{Q} \\ V_{84} ::= \text{R} \\ V_{85} ::= \text{S} \\ V_{86} ::= \text{T} \\ V_{87} ::= \text{U} \\ V_{88} ::= \text{V} \\ V_{89} ::= \text{W} \\ V_{90} ::= \text{X} \end{cases} \tag{A.7}$$

$$P^{3'} = \begin{cases} V_{91} ::= \mathrm{Y} \\ V_{92} ::= \mathrm{Z} \\ V_{93} ::= 0 \\ V_{94} ::= < ID > V_{23} \\ V_{95} ::= < NOMARCH > V_{23} \\ V_{100} ::= V_{18} V_{23} \\ V_{101} ::= < CICLOS > V_{18} \\ V_{102} ::= V_{24} < ID > \\ V_{103} ::= \{DICCIONARIOS\} V_{28} \end{cases} \tag{A.8}$$

$$
P^{4'} = \begin{cases}
< SENTENCIA > ::= V_1 V_2 V_{104} \\
\quad | \; V_3 V_{105} \\
\quad | \; V_4 V_{21} V_{106} \\
\quad | \; V_{13} < ID > V_{107} \\
\quad | \; V_5 < DICCIONARIOS > V_{107} \\
\quad | \; V_5 < DICCIONARIOS > V_{19} \{ < STRING > \} V_{108} \\
\quad | \; V_7 < DICCIONARIOS > V_{109} \\
\quad | \; V_7 < DICCIONARIOS > V_{19} \{ V_{110} \} V_{100} \\
\quad | \; V_8 < ID > V_{107} \\
\quad | \; V_{10} < ID > V_{24} < NOMARCH >< USING > V_{111} \\
\quad | \; V_{12} < STRING > V_{29} < STRING >< ID > V_{24} < NOMARCH > V_{111} \\
\quad | \; V_{14} < STRING > V_{16} < STRING >< ID > V_{24} < NOMARCH > V_{111} \\
\quad | \; V_{11} V_{17} < ID > V_{24} < NOMARCH > V_{111} \\
\quad | \; V_{15} < ID > V_{24} < NOMARCH > V_{111} \\
< CREATE > ::= V_1 V_2 V_{104} \\
< DROP > ::= V_3 V_{105} \\
< LIST > ::= V_4 V_{21} V_{106} \\
< DISPLAY > := V_{13} < ID > V_{107} \\
< ADD > ::= V_5 < DICCIONARIOS > V_{107} \\
\quad | \; V_5 < DICCIONARIOS > V_{19} \{ < STRING > \} V_{108} \\
< DELETE > ::= V_7 < DICCIONARIOS > V_{109} \\
\quad | \; V_7 < DICCIONARIOS > V_{19} \{ V_{110} \} V_{100} \\
< INSERT > ::= V_8 < ID > V_{107} \\
< PREPARE > ::= V_{10} < ID > V_{24} < NOMARCH >< USING > V_{111} \\
< EXTRACT > ::= V_{12} < STRING > V_{29} < STRING >< ID > V_{24} < NOMARCH > V_{111} \\
< REPLACEALL > ::= V_{14} < STRING > V_{16} < STRING >< ID > V_{24} < NOMARCH > V_{111} \\
< REMOVE > ::= V_{11} V_{17} < ID > V_{24} < NOMARCH > V_{111} \\
< INFOALL > ::= V_{15} < ID > V_{24} < NOMARCH > V_{111} \\
< USING > ::= V_{19} V_{101} \mid STOPWORDS \mid SYMBOLS \mid LABELS \\
< CICLOS > ::= < DICCIONARIOS > \{ V_{103} \} \\
< NOMARCH > ::= < ID > V_{102} \\
< ID > ::= < LETRA > \{ V_{102} \} \\
< STRING > ::= < LETRA > \{ < LETRA > \} \mid < DIGITO > \{ < DIGITO > \} \\
\quad | < DIGITO > \{ < LETRA > \} \mid < SIMBOLO > \{ < LETRA > \mid < SIMBOLO > \} \\
< LETRA > ::= a \mid b \mid c ... z \mid A \mid B \mid C ... Z \\
< DIGITO > ::= 0 \mid 1 \mid 2.., 9 \\
< DICCIONARIOS > ::= STOPWORDS \mid SYMBOLS \mid LABELS \\
< SIMBOLO > ::= \text{ascii}
\end{cases} \tag{A.9}
$$

$$
P^{4'} = \begin{cases}
V_1 ::= \text{create} \\
V_2 ::= \text{corpus} \\
V_3 ::= \text{drop} \\
V_4 ::= \text{list} \\
V_5 ::= \text{add} \\
V_6 ::= \text{from} \\
V_7 ::= \text{delete} \\
V_9 ::= \text{put} \\
V_{10} ::= \text{prepare} \\
V_{11} ::= \text{remove} \\
V_{12} ::= \text{extract} \\
V_{13} ::= \text{display} \\
V_{14} ::= \text{replaceall} \\
V_{15} ::= \text{infoall} \\
V_{16} ::= \text{for} \\
V_{17} ::= \text{acutes} \\
V_{18} ::=) \\
V_{19} ::= (\\
V_{20} ::= \\
V_{21} ::= * \\
V_{22} ::= \text{ascii} \\
V_{23} ::= ; \\
V_{24} ::= . \\
V_{25} ::= \text{stopwords} \\
V_{26} ::= \text{symbols} \\
V_{27} ::= \text{labels} \\
V_{28} ::= \text{and} \\
V_{29} ::= \text{to} \\
V_{30} ::= \text{from} \\
V_{31} ::= 1 \\
V_{32} ::= 2 \\
V_{33} ::= 3 \\
V_{34} ::= 4 \\
V_{35} ::= 5 \\
V_{36} ::= 6 \\
V_{37} ::= 7 \\
V_{39} ::= 8 \\
V_{40} ::= 9 \\
V_{41} ::= \text{a} \\
V_{42} ::= \text{b} \\
V_{43} ::= \text{c} \\
V_{44} ::= \text{d}
\end{cases} \tag{A.10}
$$

$$P^{4'} = \begin{cases} V_{45} ::= \text{e} \\ V_{46} ::= \text{f} \\ V_{47} ::= \text{g} \\ V_{48} ::= \text{h} \\ V_{49} ::= \text{i} \\ V_{50} ::= \text{j} \\ V_{51} ::= \text{k} \\ V_{52} ::= \text{l} \\ V_{53} ::= \text{m} \\ V_{54} ::= \text{n} \\ V_{55} ::= \text{o} \\ V_{56} ::= \text{p} \\ V_{57} ::= \text{q} \\ V_{58} ::= \text{r} \\ V_{59} ::= \text{s} \\ V_{60} ::= \text{t} \\ V_{61} ::= \text{u} \\ V_{62} ::= \text{v} \\ V_{63} ::= \text{w} \\ V_{64} ::= \text{x} \\ V_{65} ::= \text{y} \\ V_{66} ::= \text{z} \\ V_{67} ::= \text{A} \\ V_{68} ::= \text{B} \\ V_{69} ::= \text{C} \\ V_{70} ::= \text{D} \\ V_{71} ::= \text{E} \\ V_{72} ::= \text{F} \\ V_{73} ::= \text{G} \\ V_{74} ::= \text{H} \\ V_{75} ::= \text{I} \\ V_{76} ::= \text{J} \\ V_{77} ::= \text{K} \\ V_{78} ::= \text{L} \\ V_{79} ::= \text{M} \\ V_{80} ::= \text{N} \\ V_{81} ::= \text{O} \\ V_{82} ::= \text{P} \\ V_{83} ::= \text{Q} \\ V_{84} ::= \text{R} \\ V_{85} ::= \text{S} \\ V_{86} ::= \text{T} \\ V_{87} ::= \text{U} \\ V_{88} ::= \text{V} \\ V_{89} ::= \text{W} \\ V_{90} ::= \text{X} \end{cases} \tag{A.11}$$

$$P^{4'} = \begin{cases} V_{91} ::= \mathrm{Y} \\ V_{92} ::= \mathrm{Z} \\ V_{93} ::= 0 \\ V_{95} ::= < NOMARCH > V_{23} \\ V_{100} ::= V_{18} V_{23} \\ V_{101} ::= < CICLOS > V_{18} \\ V_{102} ::= V_{24} < ID > \\ V_{103} ::= \{DICCIONARIOS\} V_{28} \\ V_{104} ::= V_{24} V_{94} \\ V_{105} ::= V_{2} V_{94} \\ V_{106} ::= V_{30} V_{100} \\ V_{107} ::= V_{24} V_{95} \\ V_{108} ::= V_{20} V_{100} \\ V_{109} ::= V_{30} V_{95} \\ V_{110} ::= < STRING > V_{28} \\ V_{111} ::= V_{9} V_{95} \end{cases} \tag{A.12}$$

$$
P^{5'} = \left\{
\begin{array}{rl}
< SENTENCIA > & ::= V_1 V_{112} \\
& \mid V_3 V_{105} \\
& \mid V_4 V_{113} \\
& \mid V_{13} V_{114} \\
& \mid V_5 V_{114} \\
& \mid V_5 < DICCIONARIOS > V_{19} V_{115} \\
& \mid V_7 V_{118} \\
& \mid V_7 < DICCIONARIOS > V_{19} V_{116} \\
& \mid V_8 V_{114} \\
& \mid V_{10} < ID > V_{24} < NOMARCH > V_{119} \\
& \mid V_{12} < STRING > V_{29} < STRING >< ID > V_{24} V_{117} \\
& \mid V_{14} < STRING > V_{16} < STRING >< ID > V_{24} V_{117} \\
& \mid V_{11} V_{17} < ID > V_{24} V_{117} \\
& \mid V_{15} < ID > V_{24} < NOMARCH > V_{117} \\
< CREATE > & ::= V_1 V_{112} \\
< DROP > & ::= V_3 V_{105} \\
< LIST > & ::= V_4 V_{113} \\
< DISPLAY > & := V_{13} V_{114} \\
< ADD > & ::= V_5 V_{114} \\
& \mid V_5 < DICCIONARIOS > V_{19} V_{115} \\
< DELETE > & ::= V_7 V_{118} \\
& \mid V_7 < DICCIONARIOS > V_{19} V_{116} \\
< INSERT > & ::= V_8 V_{114} \\
< PREPARE > & ::= V_{10} < ID > V_{24} < NOMARCH > V_{119} \\
< EXTRACT > & ::= V_{12} < STRING > V_{29} < STRING >< ID > V_{24} V_{117} \\
< REPLACEALL > & ::= V_{14} < STRING > V_{16} < STRING >< ID > V_{24} V_{117} \\
< REMOVE > & ::= V_{11} V_{17} < ID > V_{24} V_{117} \\
< INFOALL > & ::= V_{15} < ID > V_{24} < NOMARCH > V_{117} \\
< USING > & ::= V_{19} V_{101} \mid STOPWORDS \mid SYMBOLS \mid LABELS \\
< CICLOS > & ::=< DICCIONARIOS > \{V_{103}\} \\
< NOMARCH > & ::=< ID > V_{102} \\
< ID > & ::=< LETRA > \{V_{102}\} \\
< STRING > & ::=< LETRA > \{< LETRA >\} \mid< DIGITO > \{< DIGITO >\} \\
& \mid< DIGITO > \{< LETRA >\} \mid< SIMBOLO > \{< LETRA > \mid < SIMBOLO >\} \\
< LETRA > & ::= a \mid b \mid c...z \mid A \mid B \mid C...Z \\
< DIGITO > & ::= 0 \mid 1 \mid 2..,9 \\
< DICCIONARIOS > & ::= STOPWORDS \mid SYMBOLS \mid LABELS \\
< SIMBOLO > & ::= \text{ascii}
\end{array}
\right. \tag{A.13}
$$

$$
P^{5'} = \begin{cases}
V_1 ::= \text{create} \\
V_2 ::= \text{corpus} \\
V_3 ::= \text{drop} \\
V_4 ::= \text{list} \\
V_5 ::= \text{add} \\
V_6 ::= \text{from} \\
V_7 ::= \text{delete} \\
V_9 ::= \text{put} \\
V_{10} ::= \text{prepare} \\
V_{11} ::= \text{remove} \\
V_{12} ::= \text{extract} \\
V_{13} ::= \text{display} \\
V_{14} ::= \text{replaceall} \\
V_{15} ::= \text{infoall} \\
V_{16} ::= \text{for} \\
V_{17} ::= \text{acutes} \\
V_{18} ::=) \\
V_{19} ::= (\\
V_{20} ::= \\
V_{21} ::= * \\
V_{22} ::= \text{ascii} \\
V_{23} ::= ; \\
V_{24} ::= . \\
V_{25} ::= \text{stopwords} \\
V_{26} ::= \text{symbols} \\
V_{27} ::= \text{labels} \\
V_{28} ::= \text{and} \\
V_{29} ::= \text{to} \\
V_{30} ::= \text{from} \\
V_{31} ::= 1 \\
V_{32} ::= 2 \\
V_{33} ::= 3 \\
V_{34} ::= 4 \\
V_{35} ::= 5 \\
V_{36} ::= 6 \\
V_{37} ::= 7 \\
V_{39} ::= 8 \\
V_{40} ::= 9 \\
V_{41} ::= \text{a} \\
V_{42} ::= \text{b} \\
V_{43} ::= \text{c} \\
V_{44} ::= \text{d}
\end{cases} \tag{A.14}
$$

$$P^{5'} = \begin{cases} V_{45} ::= \text{e} \\ V_{46} ::= \text{f} \\ V_{47} ::= \text{g} \\ V_{48} ::= \text{h} \\ V_{49} ::= \text{i} \\ V_{50} ::= \text{j} \\ V_{51} ::= \text{k} \\ V_{52} ::= \text{l} \\ V_{53} ::= \text{m} \\ V_{54} ::= \text{n} \\ V_{55} ::= \text{o} \\ V_{56} ::= \text{p} \\ V_{57} ::= \text{q} \\ V_{58} ::= \text{r} \\ V_{59} ::= \text{s} \\ V_{60} ::= \text{t} \\ V_{61} ::= \text{u} \\ V_{62} ::= \text{v} \\ V_{63} ::= \text{w} \\ V_{64} ::= \text{x} \\ V_{65} ::= \text{y} \\ V_{66} ::= \text{z} \\ V_{67} ::= \text{A} \\ V_{68} ::= \text{B} \\ V_{69} ::= \text{C} \\ V_{70} ::= \text{D} \\ V_{71} ::= \text{E} \\ V_{72} ::= \text{F} \\ V_{73} ::= \text{G} \\ V_{74} ::= \text{H} \\ V_{75} ::= \text{I} \\ V_{76} ::= \text{J} \\ V_{77} ::= \text{K} \\ V_{78} ::= \text{L} \\ V_{79} ::= \text{M} \\ V_{80} ::= \text{N} \\ V_{81} ::= \text{O} \\ V_{82} ::= \text{P} \\ V_{83} ::= \text{Q} \\ V_{84} ::= \text{R} \\ V_{85} ::= \text{S} \\ V_{86} ::= \text{T} \\ V_{87} ::= \text{U} \\ V_{88} ::= \text{V} \\ V_{89} ::= \text{W} \\ V_{90} ::= \text{X} \end{cases} \tag{A.15}$$

$$P^{5'} = \begin{cases} V_{91} ::= \text{Y} \\ V_{92} ::= \text{Z} \\ V_{93} ::= 0 \\ V_{95} ::= < NOMARCH > V_{23} \\ V_{100} ::= V_{18} V_{23} \\ V_{101} ::= < CICLOS > V_{18} \\ V_{102} ::= V_{24} < ID > \\ V_{103} ::= \{DICCIONARIOS\} V_{28} \\ V_{104} ::= V_{24} V_{94} \\ V_{105} ::= V_{2} V_{94} \\ V_{106} ::= V_{30} V_{100} \\ V_{107} ::= V_{24} V_{95} \\ V_{108} ::= V_{20} V_{100} \\ V_{109} ::= V_{30} V_{95} \\ V_{110} ::= < STRING > V_{28} \\ V_{111} ::= V_{9} V_{95} \\ V_{112} ::= V_{2} V_{104} \\ V_{113} ::= V_{21} V_{106} \\ V_{114} ::= < ID > V_{107} \\ V_{115} ::= \{STRING\} V_{108} \\ V_{116} ::= \{V_{110}\} V_{100} \\ V_{117} ::= < NOMARCH > V_{111} \\ V_{118} ::= < DICCIONARIOS > V_{109} \\ V_{119} ::= < USING > V_{111} \end{cases} \tag{A.16}$$

$$
P^{6'} = \begin{cases}
< SENTENCIA > ::= V_1 V_{112} \\
\quad | \; V_3 V_{105} \\
\quad | \; V_4 V_{113} \\
\quad | \; V_{13} V_{114} \\
\quad | \; V_5 V_{114} \\
\quad | \; V_5 < DICCIONARIOS > V_{120} \\
\quad | \; V_7 V_{118} \\
\quad | \; V_7 < DICCIONARIOS > V_{121} \\
\quad | \; V_8 V_{114} \\
\quad | \; V_{10} < ID > V_{24} V_{122} \\
\quad | \; V_{12} < STRING > V_{29} < STRING >< ID > V_{123} \\
\quad | \; V_{14} < STRING > V_{16} < STRING >< ID > V_{123} \\
\quad | \; V_{11} V_{17} < ID > V_{24} V_{117} \\
\quad | \; V_{15} < ID > V_{123} \\
< CREATE > ::= V_1 V_{112} \\
< DROP > ::= V_3 V_{105} \\
< LIST > ::= V_4 V_{113} \\
< DISPLAY > := V_{13} V_{114} \\
< ADD > ::= V_5 V_{114} \\
\quad | \; V_5 < DICCIONARIOS > V_{120} \\
< DELETE > ::= V_7 V_{118} \\
\quad | \; V_7 < DICCIONARIOS > V_{121} \\
< INSERT > ::= V_8 V_{114} \\
< PREPARE > ::= V_{10} < ID > V_{24} V_{122} \\
< EXTRACT > ::= V_{12} < STRING > V_{29} < STRING >< ID > V_{123} \\
< REPLACEALL > ::= V_{14} < STRING > V_{16} < STRING >< ID > V_{123} \\
< REMOVE > ::= V_{11} V_{17} < ID > V_{24} V_{117} \\
< INFOALL > ::= V_{15} < ID > V_{123} \\
< USING > ::= V_{19} V_{101} \mid STOPWORDS \mid SYMBOLS \mid LABELS \\
< CICLOS > ::=< DICCIONARIOS > \{V_{103}\} \\
< NOMARCH > ::=< ID > V_{102} \\
< ID > ::=< LETRA > \{V_{102}\} \\
< STRING > ::=< LETRA > \{< LETRA >\} \mid < DIGITO > \{< DIGITO >\} \\
\quad \mid < DIGITO > \{< LETRA >\} \mid < SIMBOLO > \{< LETRA > \mid < SIMBOLO >\} \\
< LETRA > ::= a \mid b \mid c...z \mid A \mid B \mid C...Z \\
< DIGITO > ::= 0 \mid 1 \mid 2..,9 \\
< DICCIONARIOS > ::= STOPWORDS \mid SYMBOLS \mid LABELS \\
< SIMBOLO > ::= \text{ascii}
\end{cases} \tag{A.17}
$$

$$P^{6'} = \begin{cases} V_1 ::= \text{create} \\ V_2 ::= \text{corpus} \\ V_3 ::= \text{drop} \\ V_4 ::= \text{list} \\ V_5 ::= \text{add} \\ V_6 ::= \text{from} \\ V_7 ::= \text{delete} \\ V_9 ::= \text{put} \\ V_{10} ::= \text{prepare} \\ V_{11} ::= \text{remove} \\ V_{12} ::= \text{extract} \\ V_{13} ::= \text{display} \\ V_{14} ::= \text{replaceall} \\ V_{15} ::= \text{infoall} \\ V_{16} ::= \text{for} \\ V_{17} ::= \text{acutes} \\ V_{18} ::=) \\ V_{19} ::= (\\ V_{20} ::= \\ V_{21} ::= * \\ V_{22} ::= \text{ascii} \\ V_{23} ::= ; \\ V_{24} ::= . \\ V_{25} ::= \text{stopwords} \\ V_{26} ::= \text{symbols} \\ V_{27} ::= \text{labels} \\ V_{28} ::= \text{and} \\ V_{29} ::= \text{to} \\ V_{30} ::= \text{from} \\ V_{31} ::= 1 \\ V_{32} ::= 2 \\ V_{33} ::= 3 \\ V_{34} ::= 4 \\ V_{35} ::= 5 \\ V_{36} ::= 6 \\ V_{37} ::= 7 \\ V_{39} ::= 8 \\ V_{40} ::= 9 \\ V_{41} ::= \text{a} \\ V_{42} ::= \text{b} \\ V_{43} ::= \text{c} \\ V_{44} ::= \text{d} \end{cases} \tag{A.18}$$

$$P^{6'} = \begin{cases} V_{45} ::= \text{e} \\ V_{46} ::= \text{f} \\ V_{47} ::= \text{g} \\ V_{48} ::= \text{h} \\ V_{49} ::= \text{i} \\ V_{50} ::= \text{j} \\ V_{51} ::= \text{k} \\ V_{52} ::= \text{l} \\ V_{53} ::= \text{m} \\ V_{54} ::= \text{n} \\ V_{55} ::= \text{o} \\ V_{56} ::= \text{p} \\ V_{57} ::= \text{q} \\ V_{58} ::= \text{r} \\ V_{59} ::= \text{s} \\ V_{60} ::= \text{t} \\ V_{61} ::= \text{u} \\ V_{62} ::= \text{v} \\ V_{63} ::= \text{w} \\ V_{64} ::= \text{x} \\ V_{65} ::= \text{y} \\ V_{66} ::= \text{z} \\ V_{67} ::= \text{A} \\ V_{68} ::= \text{B} \\ V_{69} ::= \text{C} \\ V_{70} ::= \text{D} \\ V_{71} ::= \text{E} \\ V_{72} ::= \text{F} \\ V_{73} ::= \text{G} \\ V_{74} ::= \text{H} \\ V_{75} ::= \text{I} \\ V_{76} ::= \text{J} \\ V_{77} ::= \text{K} \\ V_{78} ::= \text{L} \\ V_{79} ::= \text{M} \\ V_{80} ::= \text{N} \\ V_{81} ::= \text{O} \\ V_{82} ::= \text{P} \\ V_{83} ::= \text{Q} \\ V_{84} ::= \text{R} \\ V_{85} ::= \text{S} \\ V_{86} ::= \text{T} \\ V_{87} ::= \text{U} \\ V_{88} ::= \text{V} \\ V_{89} ::= \text{W} \\ V_{90} ::= \text{X} \end{cases} \tag{A.19}$$

$$P^{6'} = \begin{cases} V_{91} ::= \mathrm{Y} \\ V_{92} ::= \mathrm{Z} \\ V_{93} ::= 0 \\ V_{95} ::= < NOMARCH > V_{23} \\ V_{100} ::= V_{18} V_{23} \\ V_{101} ::= < CICLOS > V_{18} \\ V_{102} ::= V_{24} < ID > \\ V_{103} ::= \{DICCIONARIOS\} V_{28} \\ V_{104} ::= V_{24} V_{94} \\ V_{105} ::= V_{2} V_{94} \\ V_{106} ::= V_{30} V_{100} \\ V_{107} ::= V_{24} V_{95} \\ V_{108} ::= V_{20} V_{100} \\ V_{109} ::= V_{30} V_{95} \\ V_{110} ::= < STRING > V_{28} \\ V_{111} ::= V_{9} V_{95} \\ V_{112} ::= V_{2} V_{104} \\ V_{113} ::= V_{21} V_{106} \\ V_{114} ::= < ID > V_{107} \\ V_{115} ::= \{STRING\} V_{108} \\ V_{116} ::= \{V_{110}\} V_{100} \\ V_{117} ::= < NOMARCH > V_{111} \\ V_{118} ::= < DICCIONARIOS > V_{109} \\ V_{119} ::= < USING > V_{111} \\ V_{120} ::= V_{19} V_{115} \\ V_{121} ::= V_{19} V_{116} \\ V_{122} ::= < NOMARCH > V_{119} \\ V_{123} ::= V_{24} V_{117} \end{cases} \tag{A.20}$$

$$
P^{7'} = \left\{
\begin{array}{rl}
< SENTENCIA > & ::= V_1 V_{112} \\
& \mid V_3 V_{105} \\
& \mid V_4 V_{113} \\
& \mid V_{13} V_{114} \\
& \mid V_5 V_{114} \\
& \mid V_5 V_{124} \\
& \mid V_7 V_{118} \\
& \mid V_7 V_{125} \\
& \mid V_8 V_{114} \\
& \mid V_{10} < ID > V_{126} \\
& \mid V_{12} < STRING > V_{29} < STRING > V_{127} \\
& \mid V_{14} < STRING > V_{16} < STRING > V_{127} \\
& \mid V_{11} V_{17} V_{127} \\
& \mid V_{15} < ID > V_{127} \\
< CREATE > & ::= V_1 V_{112} \\
< DROP > & ::= V_3 V_{105} \\
< LIST > & ::= V_4 V_{113} \\
< DISPLAY > & := V_{13} V_{114} \\
< ADD > & ::= V_5 V_{114} \\
& \mid V_5 V_{124} \\
< DELETE > & ::= V_7 V_{118} \\
& \mid V_7 V_{125} \\
< INSERT > & ::= V_8 V_{114} \\
< PREPARE > & ::= V_{10} < ID > V_{126} \\
< EXTRACT > & ::= V_{12} < STRING > V_{29} < STRING > V_{127} \\
< REPLACEALL > & ::= V_{14} < STRING > V_{16} < STRING > V_{127} \\
< REMOVE > & ::= V_{11} V_{17} V_{127} \\
< INFOALL > & ::= V_{15} < ID > V_{123} \\
< USING > & ::= V_{19} V_{101} \mid STOPWORDS \mid SYMBOLS \mid LABELS \\
< CICLOS > & ::= < DICCIONARIOS > \{V_{103}\} \\
< NOMARCH > & ::= < ID > V_{102} \\
< ID > & ::= < LETRA > \{V_{102}\} \\
< STRING > & ::= < LETRA > \{< LETRA >\} \mid < DIGITO > \{< DIGITO >\} \\
& \mid < DIGITO > \{< LETRA >\} \mid < SIMBOLO > \{< LETRA > \mid < SIMBOLO >\} \\
< LETRA > & ::= a \mid b \mid c...z \mid A \mid B \mid C...Z \\
< DIGITO > & ::= 0 \mid 1 \mid 2..,9 \\
< DICCIONARIOS > & ::= STOPWORDS \mid SYMBOLS \mid LABELS \\
< SIMBOLO > & ::= \text{ascii}
\end{array}
\right. \tag{A.21}
$$

$$P^{7'} = \begin{cases} V_1 ::= \text{create} \\ V_2 ::= \text{corpus} \\ V_3 ::= \text{drop} \\ V_4 ::= \text{list} \\ V_5 ::= \text{add} \\ V_6 ::= \text{from} \\ V_7 ::= \text{delete} \\ V_9 ::= \text{put} \\ V_{10} ::= \text{prepare} \\ V_{11} ::= \text{remove} \\ V_{12} ::= \text{extract} \\ V_{13} ::= \text{display} \\ V_{14} ::= \text{replaceall} \\ V_{15} ::= \text{infoall} \\ V_{16} ::= \text{for} \\ V_{17} ::= \text{acutes} \\ V_{18} ::=) \\ V_{19} ::= (\\ V_{20} ::= \\ V_{21} ::= * \\ V_{22} ::= \text{ascii} \\ V_{23} ::= ; \\ V_{24} ::= . \\ V_{25} ::= \text{stopwords} \\ V_{26} ::= \text{symbols} \\ V_{27} ::= \text{labels} \\ V_{28} ::= \text{and} \\ V_{29} ::= \text{to} \\ V_{30} ::= \text{from} \\ V_{31} ::= 1 \\ V_{32} ::= 2 \\ V_{33} ::= 3 \\ V_{34} ::= 4 \\ V_{35} ::= 5 \\ V_{36} ::= 6 \\ V_{37} ::= 7 \\ V_{39} ::= 8 \\ V_{40} ::= 9 \\ V_{41} ::= \text{a} \\ V_{42} ::= \text{b} \\ V_{43} ::= \text{c} \\ V_{44} ::= \text{d} \end{cases} \tag{A.22}$$

$$P^{7'} = \begin{cases} V_{45} ::= \text{e} \\ V_{46} ::= \text{f} \\ V_{47} ::= \text{g} \\ V_{48} ::= \text{h} \\ V_{49} ::= \text{i} \\ V_{50} ::= \text{j} \\ V_{51} ::= \text{k} \\ V_{52} ::= \text{l} \\ V_{53} ::= \text{m} \\ V_{54} ::= \text{n} \\ V_{55} ::= \text{o} \\ V_{56} ::= \text{p} \\ V_{57} ::= \text{q} \\ V_{58} ::= \text{r} \\ V_{59} ::= \text{s} \\ V_{60} ::= \text{t} \\ V_{61} ::= \text{u} \\ V_{62} ::= \text{v} \\ V_{63} ::= \text{w} \\ V_{64} ::= \text{x} \\ V_{65} ::= \text{y} \\ V_{66} ::= \text{z} \\ V_{67} ::= \text{A} \\ V_{68} ::= \text{B} \\ V_{69} ::= \text{C} \\ V_{70} ::= \text{D} \\ V_{71} ::= \text{E} \\ V_{72} ::= \text{F} \\ V_{73} ::= \text{G} \\ V_{74} ::= \text{H} \\ V_{75} ::= \text{I} \\ V_{76} ::= \text{J} \\ V_{77} ::= \text{K} \\ V_{78} ::= \text{L} \\ V_{79} ::= \text{M} \\ V_{80} ::= \text{N} \\ V_{81} ::= \text{O} \\ V_{82} ::= \text{P} \\ V_{83} ::= \text{Q} \\ V_{84} ::= \text{R} \\ V_{85} ::= \text{S} \\ V_{86} ::= \text{T} \\ V_{87} ::= \text{U} \\ V_{88} ::= \text{V} \\ V_{89} ::= \text{W} \\ V_{90} ::= \text{X} \end{cases} \tag{A.23}$$

$$P^{7'} = \begin{cases} V_{91} ::= \text{Y} \\ V_{92} ::= \text{Z} \\ V_{93} ::= 0 \\ V_{95} ::=< NOMARCH > V_{23} \\ V_{100} ::= V_{18}V_{23} \\ V_{101} ::=< CICLOS > V_{18} \\ V_{102} ::= V_{24} < ID > \\ V_{103} ::= \{DICCIONARIOS\}V_{28} \\ V_{104} ::= V_{24}V_{94} \\ V_{105} ::= V_{2}V_{94} \\ V_{106} ::= V_{30}V_{100} \\ V_{107} ::= V_{24}V_{95} \\ V_{108} ::= V_{20}V_{100} \\ V_{109} ::= V_{30}V_{95} \\ V_{110} ::=< STRING > V_{28} \\ V_{111} ::= V_{9}V_{95} \\ V_{112} ::= V_{2}V_{104} \\ V_{113} ::= V_{21}V_{106} \\ V_{114} ::=< ID > V_{107} \\ V_{115} ::= \{STRING\}V_{108} \\ V_{116} ::= \{V_{110}\}V_{100} \\ V_{117} ::=< NOMARCH > V_{111} \\ V_{118} ::=< DICCIONARIOS > V_{109} \\ V_{119} ::=< USING > V_{111} \\ V_{120} ::= V_{19}V_{115} \\ V_{121} ::= V_{19}V_{116} \\ V_{122} ::=< NOMARCH > V_{119} \\ V_{124} ::=< DICCIONARIOS > V_{120} \\ V_{125} ::=< DICCIONARIOS > V_{121} \\ V_{126} ::= V_{24}V_{122} \\ V_{127} ::=< ID > V_{123} \end{cases} \tag{A.24}$$

$$
P^{8'} = \left\{
\begin{array}{rl}
< SENTENCIA > & ::= V_1 V_{112} \\
& \mid V_3 V_{105} \\
& \mid V_4 V_{113} \\
& \mid V_{13} V_{114} \\
& \mid V_5 V_{114} \\
& \mid V_5 V_{124} \\
& \mid V_7 V_{118} \\
& \mid V_7 V_{125} \\
& \mid V_8 V_{114} \\
& \mid V_{10} V_{128} \\
& \mid V_{12} < STRING > V_{29} V_{129} \\
& \mid V_{14} < STRING > V_{16} V_{129} \\
& \mid V_{11} V_{130} \\
& \mid V_{15} V_{131} \\
< CREATE > & ::= V_1 V_{112} \\
< DROP > & ::= V_3 V_{105} \\
< LIST > & ::= V_4 V_{113} \\
< DISPLAY > & := V_{13} V_{114} \\
< ADD > & ::= V_5 V_{114} \\
& \mid V_5 V_{124} \\
< DELETE > & ::= V_7 V_{118} \\
& \mid V_7 V_{125} \\
< INSERT > & ::= V_8 V_{114} \\
< PREPARE > & ::= V_{10} < ID > V_{126} \\
< EXTRACT > & ::= V_{12} < STRING > V_{29} V_{129} \\
< REPLACEALL > & ::= V_{14} < STRING > V_{16} V_{129} \\
< REMOVE > & ::= V_{11} V_{17} V_{127} \\
< INFOALL > & ::= V_{15} < ID > V_{123} \\
< USING > & ::= V_{19} V_{101} \mid STOPWORDS \mid SYMBOLS \mid LABELS \\
< CICLOS > & ::= < DICCIONARIOS > \{V_{103}\} < NOMARCH > \quad ::= < ID > V_{102} \\
< ID > & ::= < LETRA > \{V_{102}\} \\
< STRING > & ::= < LETRA > \{< LETRA >\} \mid < DIGITO > \{< DIGITO >\} \\
& \mid < DIGITO > \{< LETRA >\} \mid < SIMBOLO > \{< LETRA > \mid < SIMBOLO >\} \\
< LETRA > & ::= a \mid b \mid c ... z \mid A \mid B \mid C ... Z \\
< DIGITO > & ::= 0 \mid 1 \mid 2..,9 \\
< DICCIONARIOS > & ::= STOPWORDS \mid SYMBOLS \mid LABELS \\
< SIMBOLO > & ::= \text{ascii}
\end{array}
\right. \tag{A.25}
$$

$$
P^{8'} = \begin{cases}
V_1 ::= \text{create} \\
V_2 ::= \text{corpus} \\
V_3 ::= \text{drop} \\
V_4 ::= \text{list} \\
V_5 ::= \text{add} \\
V_6 ::= \text{from} \\
V_7 ::= \text{delete} \\
V_9 ::= \text{put} \\
V_{10} ::= \text{prepare} \\
V_{11} ::= \text{remove} \\
V_{12} ::= \text{extract} \\
V_{13} ::= \text{display} \\
V_{14} ::= \text{replaceall} \\
V_{15} ::= \text{infoall} \\
V_{16} ::= \text{for} \\
V_{17} ::= \text{acutes} \\
V_{18} ::=) \\
V_{19} ::= (\\
V_{20} ::= \\
V_{21} ::= * \\
V_{22} ::= \text{ascii} \\
V_{23} ::= ; \\
V_{24} ::= . \\
V_{25} ::= \text{stopwords} \\
V_{26} ::= \text{symbols} \\
V_{27} ::= \text{labels} \\
V_{28} ::= \text{and} \\
V_{29} ::= \text{to} \\
V_{30} ::= \text{from} \\
V_{31} ::= 1 \\
V_{32} ::= 2 \\
V_{33} ::= 3 \\
V_{34} ::= 4 \\
V_{35} ::= 5 \\
V_{36} ::= 6 \\
V_{37} ::= 7 \\
V_{39} ::= 8 \\
V_{40} ::= 9 \\
V_{41} ::= \text{a} \\
V_{42} ::= \text{b} \\
V_{43} ::= \text{c} \\
V_{44} ::= \text{d}
\end{cases} \tag{A.26}
$$

$$P^{8'} = \begin{cases} V_{45} ::= \text{e} \\ V_{46} ::= \text{f} \\ V_{47} ::= \text{g} \\ V_{48} ::= \text{h} \\ V_{49} ::= \text{i} \\ V_{50} ::= \text{j} \\ V_{51} ::= \text{k} \\ V_{52} ::= \text{l} \\ V_{53} ::= \text{m} \\ V_{54} ::= \text{n} \\ V_{55} ::= \text{o} \\ V_{56} ::= \text{p} \\ V_{57} ::= \text{q} \\ V_{58} ::= \text{r} \\ V_{59} ::= \text{s} \\ V_{60} ::= \text{t} \\ V_{61} ::= \text{u} \\ V_{62} ::= \text{v} \\ V_{63} ::= \text{w} \\ V_{64} ::= \text{x} \\ V_{65} ::= \text{y} \\ V_{66} ::= \text{z} \\ V_{67} ::= \text{A} \\ V_{68} ::= \text{B} \\ V_{69} ::= \text{C} \\ V_{70} ::= \text{D} \\ V_{71} ::= \text{E} \\ V_{72} ::= \text{F} \\ V_{73} ::= \text{G} \\ V_{74} ::= \text{H} \\ V_{75} ::= \text{I} \\ V_{76} ::= \text{J} \\ V_{77} ::= \text{K} \\ V_{78} ::= \text{L} \\ V_{79} ::= \text{M} \\ V_{80} ::= \text{N} \\ V_{81} ::= \text{O} \\ V_{82} ::= \text{P} \\ V_{83} ::= \text{Q} \\ V_{84} ::= \text{R} \\ V_{85} ::= \text{S} \\ V_{86} ::= \text{T} \\ V_{87} ::= \text{U} \\ V_{88} ::= \text{V} \\ V_{89} ::= \text{W} \\ V_{90} ::= \text{X} \end{cases} \tag{A.27}$$

$$P^{8'} = \begin{cases} V_{91} ::= \mathrm{Y} \\ V_{92} ::= \mathrm{Z} \\ V_{93} ::= 0 \\ V_{95} ::= < NOMARCH > V_{23} \\ V_{100} ::= V_{18} V_{23} \\ V_{101} ::= < CICLOS > V_{18} \\ V_{102} ::= V_{24} < ID > \\ V_{103} ::= \{DICCIONARIOS\} V_{28} \\ V_{104} ::= V_{24} V_{94} \\ V_{105} ::= V_{2} V_{94} \\ V_{106} ::= V_{30} V_{100} \\ V_{107} ::= V_{24} V_{95} \\ V_{108} ::= V_{20} V_{100} \\ V_{109} ::= V_{30} V_{95} \\ V_{110} ::= < STRING > V_{28} \\ V_{111} ::= V_{9} V_{95} \\ V_{112} ::= V_{2} V_{104} \\ V_{113} ::= V_{21} V_{106} \\ V_{114} ::= < ID > V_{107} \\ V_{115} ::= \{STRING\} V_{108} \\ V_{116} ::= \{V_{110}\} V_{100} \\ V_{117} ::= < NOMARCH > V_{111} \\ V_{118} ::= < DICCIONARIOS > V_{109} \\ V_{119} ::= < USING > V_{111} \\ V_{120} ::= V_{19} V_{115} \\ V_{121} ::= V_{19} V_{116} \\ V_{122} ::= < NOMARCH > V_{119} \\ V_{124} ::= < DICCIONARIOS > V_{120} \\ V_{125} ::= < DICCIONARIOS > V_{121} \\ V_{126} ::= V_{24} V_{122} \\ V_{127} ::= < ID > V_{123} \\ V_{128} ::= < ID > V_{126} \\ V_{129} ::= < STRING > V_{127} \\ V_{130} ::= V_{17} V_{127} \\ V_{131} ::= < ID > V_{127} \end{cases} \tag{A.28}$$

$$
P^{9'} = \left\{
\begin{array}{rl}
< SENTENCIA > & ::= V_1 V_{112} \\
& \mid V_3 V_{105} \\
& \mid V_4 V_{113} \\
& \mid V_{13} V_{114} \\
& \mid V_5 V_{114} \\
& \mid V_5 V_{124} \\
& \mid V_7 V_{118} \\
& \mid V_7 V_{125} \\
& \mid V_8 V_{114} \\
& \mid V_{10} V_{128} \\
& \mid V_{12} V_{134} \\
& \mid V_{14} V_{135} \\
& \mid V_{11} V_{130} \\
& \mid V_{15} V_{131} \\
< CREATE > & ::= V_1 V_{112} \\
< DROP > & ::= V_3 V_{105} \\
< LIST > & ::= V_4 V_{113} \\
< DISPLAY > & := V_{13} V_{114} \\
< ADD > & ::= V_5 V_{114} \\
& \mid V_5 V_{124} \\
< DELETE > & ::= V_7 V_{118} \\
& \mid V_7 V_{125} \\
< INSERT > & ::= V_8 V_{114} \\
< PREPARE > & ::= V_{10} < ID > V_{126} \\
< EXTRACT > & ::= V_{12} < STRING > V_{29} V_{129} \\
< REPLACEALL > & ::= V_{14} < STRING > V_{16} V_{129} \\
< REMOVE > & ::= V_{11} V_{17} V_{127} \\
< INFOALL > & ::= V_{15} < ID > V_{123} \\
< USING > & ::= V_{19} V_{101} \mid STOPWORDS \mid SYMBOLS \mid LABELS \\
< CICLOS > & ::= < DICCIONARIOS > \{V_{103}\} \\
< NOMARCH > & ::= < ID > V_{102} \\
< ID > & ::= < LETRA > \{V_{102}\} \\
< STRING > & ::= < LETRA > \{< LETRA >\} \mid < DIGITO > \{< DIGITO >\} \\
& \mid < DIGITO > \{< LETRA >\} \mid < SIMBOLO > \{< LETRA > \mid < SIMBOLO >\} \\
< LETRA > & ::= a \mid b \mid c...z \mid A \mid B \mid C...Z \\
< DIGITO > & ::= 0 \mid 1 \mid 2..,9 \\
< DICCIONARIOS > & ::= STOPWORDS \mid SYMBOLS \mid LABELS \\
< SIMBOLO > & ::= \text{ascii}
\end{array}
\right. \tag{A.29}
$$

$$P^{9'} = \begin{cases} V_1 ::= \text{create} \\ V_2 ::= \text{corpus} \\ V_3 ::= \text{drop} \\ V_4 ::= \text{list} \\ V_5 ::= \text{add} \\ V_6 ::= \text{from} \\ V_7 ::= \text{delete} \\ V_9 ::= \text{put} \\ V_{10} ::= \text{prepare} \\ V_{11} ::= \text{remove} \\ V_{12} ::= \text{extract} \\ V_{13} ::= \text{display} \\ V_{14} ::= \text{replaceall} \\ V_{15} ::= \text{infoall} \\ V_{16} ::= \text{for} \\ V_{17} ::= \text{acutes} \\ V_{18} ::=) \\ V_{19} ::= (\\ V_{20} ::= \\ V_{21} ::= * \\ V_{22} ::= \text{ascii} \\ V_{23} ::= ; \\ V_{24} ::= . \\ V_{25} ::= \text{stopwords} \\ V_{26} ::= \text{symbols} \\ V_{27} ::= \text{labels} \\ V_{28} ::= \text{and} \\ V_{29} ::= \text{to} \\ V_{30} ::= \text{from} \\ V_{31} ::= 1 \\ V_{32} ::= 2 \\ V_{33} ::= 3 \\ V_{34} ::= 4 \\ V_{35} ::= 5 \\ V_{36} ::= 6 \\ V_{37} ::= 7 \\ V_{39} ::= 8 \\ V_{40} ::= 9 \\ V_{41} ::= \text{a} \\ V_{42} ::= \text{b} \\ V_{43} ::= \text{c} \\ V_{44} ::= \text{d} \end{cases} \tag{A.30}$$

$$P^{9'} = \begin{cases} V_{45} ::= \text{e} \\ V_{46} ::= \text{f} \\ V_{47} ::= \text{g} \\ V_{48} ::= \text{h} \\ V_{49} ::= \text{i} \\ V_{50} ::= \text{j} \\ V_{51} ::= \text{k} \\ V_{52} ::= \text{l} \\ V_{53} ::= \text{m} \\ V_{54} ::= \text{n} \\ V_{55} ::= \text{o} \\ V_{56} ::= \text{p} \\ V_{57} ::= \text{q} \\ V_{58} ::= \text{r} \\ V_{59} ::= \text{s} \\ V_{60} ::= \text{t} \\ V_{61} ::= \text{u} \\ V_{62} ::= \text{v} \\ V_{63} ::= \text{w} \\ V_{64} ::= \text{x} \\ V_{65} ::= \text{y} \\ V_{66} ::= \text{z} \\ V_{67} ::= \text{A} \\ V_{68} ::= \text{B} \\ V_{69} ::= \text{C} \\ V_{70} ::= \text{D} \\ V_{71} ::= \text{E} \\ V_{72} ::= \text{F} \\ V_{73} ::= \text{G} \\ V_{74} ::= \text{H} \\ V_{75} ::= \text{I} \\ V_{76} ::= \text{J} \\ V_{77} ::= \text{K} \\ V_{78} ::= \text{L} \\ V_{79} ::= \text{M} \\ V_{80} ::= \text{N} \\ V_{81} ::= \text{O} \\ V_{82} ::= \text{P} \\ V_{83} ::= \text{Q} \\ V_{84} ::= \text{R} \\ V_{85} ::= \text{S} \\ V_{86} ::= \text{T} \\ V_{87} ::= \text{U} \\ V_{88} ::= \text{V} \\ V_{89} ::= \text{W} \\ V_{90} ::= \text{X} \end{cases} \tag{A.31}$$

$$P^{9'} = \begin{cases} V_{91} ::= \mathrm{Y} \\ V_{92} ::= \mathrm{Z} \\ V_{93} ::= 0 \\ V_{95} ::=< NOMARCH > V_{23} \\ V_{100} ::= V_{18}V_{23} \\ V_{101} ::=< CICLOS > V_{18} \\ V_{102} ::= V_{24} < ID > \\ V_{103} ::= \{DICCIONARIOS\}V_{28} \\ V_{104} ::= V_{24}V_{94} \\ V_{105} ::= V_{2}V_{94} \\ V_{106} ::= V_{30}V_{100} \\ V_{107} ::= V_{24}V_{95} \\ V_{108} ::= V_{20}V_{100} \\ V_{109} ::= V_{30}V_{95} \\ V_{110} ::=< STRING > V_{28} \\ V_{111} ::= V_{9}V_{95} \\ V_{112} ::= V_{2}V_{104} \\ V_{113} ::= V_{21}V_{106} \\ V_{114} ::=< ID > V_{107} \\ V_{115} ::= \{STRING\}V_{108} \\ V_{116} ::= \{V_{110}\}V_{100} \\ V_{117} ::=< NOMARCH > V_{111} \\ V_{118} ::=< DICCIONARIOS > V_{109} \\ V_{119} ::=< USING > V_{111} \\ V_{120} ::= V_{19}V_{115} \\ V_{121} ::= V_{19}V_{116} \\ V_{122} ::=< NOMARCH > V_{119} \\ V_{124} ::=< DICCIONARIOS > V_{120} \\ V_{125} ::=< DICCIONARIOS > V_{121} \\ V_{126} ::= V_{24}V_{122} \\ V_{127} ::=< ID > V_{123} \\ V_{128} ::=< ID > V_{126} \\ V_{129} ::=< STRING > V_{127} \\ V_{130} ::= V_{17}V_{127} \\ V_{131} ::=< ID > V_{127} \\ V_{132} ::= V_{29}V_{129} \\ V_{133} ::= V_{16}V_{129} \\ V_{134} ::=< STRING > V_{132} \\ V_{135} ::=< STRING > V_{133} \end{cases} \tag{A.32}$$

Apéndice B

Expresiones Regulares en Java

Las expresiones regulares son algo que se usa desde hace años en lenguajes de programación como *Perl, Sed o Awk*. En la versión 1.4 del JDK[1] de Sun, se incluye el paquete *java.util.regex*[2] , que proporciona una serie de clases para poder hacer uso de la potencia de este tipo de expresiones en Java.

Una expresión regular es un patrón que describe a una cadena de caracteres. Las expresiones regulares se rigen por una serie de normas y hay una construcción para cualquier patrón de caracteres. Una expresión regular sólo puede contener (además de letras y números) los siguientes caracteres:

$$< \$, ^{,} ., *, +, ?, [,], \backslash . >$$

Una expresión regular sirve para buscar patrones en una cadena de texto, por ejemplo encontrar cuantas veces se repite una palabra en un texto, para comprobar que una cadena tiene una determinada estructura, por ejemplo que el nombre del archivo que se propone tiene una determinada extensión[1]. El paquete *java.util.regex* está formado por dos clases, la clase *Matcher* y la clase *Pattern* y por una excepción, *PatternSyntaxException*.

[1] JDK son las siglas de Java Developers Kit. Es decir, conjunto de herramientas para desarrollar aplicaciones en Java.

[2] http://java.sun.com/1.4/docs/api/java/util/regex/package-summary.html

B.1. El uso de la expresiones regulares

B.1.1. La Clase Pattern

La clase *Pattern* (según la documentación del jdk1.4) es la representación compilada de una expresión regular, o lo que es lo mismo, representa a la expresión regular, que en el paquete *java.util.regex* necesita estar compilada. En castellano significa patrón. La clase Matcher es un tipo de objeto que se crea a partir de un patrón mediante la invocación del método *Pattern.matcher*, este objeto permite realizar las operaciones sobre la secuencia de caracteres que se desean validar o la secuencia de caracteres en que se requieren buscar. En castellano lo más parecido a esto es la palabra encajador o empatador (de empatamiento). Por lo tanto se tienen patrones que deben ser compilados, a partir de estos se crean objetos *Matcher* (encajadores) para poder realizar las operaciones sobre la cadena en cuestión.
Con la clase *Pattern*, para crear un patrón se necesita compilar una expresión regular, esto se consigue con el método *compile*:

Pattern patron = Pattern.compile(“camión”);

Métodos

A continuación se describen una serie de métodos de la clase *Pattern*.
Método pattern: devuelve la expresión regular que hemos compilado.
Método matcher: crea un objeto Matcher a partir del patrón.
Método split: divide una cadena dada en partes que cumplan el patrón compilado.
Método matches: compila una expresión regular y comprueba una cadena de caracteres contra ella.

B.1.2. La Clase *Matcher*

Esta clase se utiliza para comprobar cadenas contra el patrón indicado. Un objeto *Matcher* se genera a partir de un objeto *Pattern* por medio del método *matcher*:

Pattern patron = Pattern.compile("camion");

Matcher encaja = patron.matcher();

Métodos

Una vez que se tiene el objeto creado, se pueden realizar tres tipos de operaciones sobre una cadena de caracteres (a través de diferentes métodos). Estas operaciones se describen a continuación.

Método matches: que intenta encajar toda la secuencia en el patrón (para el patrón "camion" la cadena "camion" encajaría, la cadena "mi camion es verde" no encajaría).

Método lookingAt: intenta encajar el patrón en la cadena (para el patrón "camion" tanto la cadena "camion" como la cadena "mi camion es verde" encajaria).

Método find: busca subcadenas dentro de la cadena de caracteres que cumplan el patrón compilado (una vez encontrada una ocurrencia, se puede inspeccionar por medio de los métodos *start* que marca el primer carácter de la ocurrencia en la secuencia y el método *end* que marca el último carácter de la ocurrencia). Todos estos métodos devuelven un *booleano* que indica si la operación ha tenido éxito. Todo lo anterior está orientado a la búsqueda de patrones en cadenas de caracteres, pero puede que se quiera llegar más

allá, que lo que se desee sea reemplazar una cadena de caracteres que se corresponda con un patrón por otra cadena. Por ejemplo un método que consigue esto es *replaceAll* que reemplaza toda ocurrencia del patrón en la cadena por la cadena que se le suministra.

B.2. Ejemplos

En esta sección se presentan algunos ejemplos en Java, haciendo uso de la clase *Matcher* y la clase *Pattern*.

El ejemplo B.1 sustituye todas las apariciones que concuerdan con el patrón "$a*$ b$''porlacadena$"$*$ ", arrojando como salida $*mm*$.

El ejemplo B.2 busca sub-cadenas que cumplan con el patron, y sustituye todas las apariciones que concuerdan con el patrón $[A - D]$, arrojando como salida $ABCD$.

Ejemplo B.1. // cuenta.java

```
import java.util.regex.*;
public class cuenta
{
public static void main(String args[ ])
Pattern patron = Pattern.compile("a * b");
Matcher encaja = patron.matcher("aaabmmaab");
String resultado = encaja.replaceAll("*");
System.out.print(resultado);
}
```

Ejemplo B.2. // subcad.java

```
import java.util.regex.*;
public class subcad
{
public static void main(String args[ ])
Pattern p = Pattern.compile("([A-D])");
Matcher m = p.matcher("ABCDEFGHI");
while(m.find())
{
System.out.print(m.group(1));
}
}
```

B.3. Conclusión

Las expresiones regulares cubren un hueco en el JDK de Sun que venía siendo solicitado desde hace mucho tiempo. Con la inclusión de las expresiones regulares Java se convierte, en este tema, en un lenguaje de programación tan flexible como otros más tradicionales en el tema de las expresiones regulares, *Perl, Awk, etc.* Hasta ahora la unica opción para conseguir un efecto parecido era el uso de *StringTokenizer* en conjunción con llamadas repetidas al modo *charAt* que producé un código demasiado complicado.

Lo que viene a continuación no es mas que la traducción de la documentaciónde una parte de la clase*Pattern*[3] .

[3] http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html

Expresión	Encaja con
x	El caracter x
\\	El caracter \
\t	El tabulador
\n	Nueva linea
\r	Retorno de carro
\f	Nueva pagina
\a	Una beep de alerta
\e	Escape
\cx	El caracter de control corresponde a x

Tabla B.1: Caracteres

Expresión	Encaja con
$[abc]$	a,b,o c
$[\wedge abc]$	Cualquier caracter excepto a,b,o c
$[a - zA - Z]$	Desde la a hasta la z o desde la A a la Z
$[a - d[m - p]]$	Desde la a hasta la d, o desde la a la p

Tabla B.2: Intervalos de caracteres

Expresión	Encaja con
.	Cualquier caracter
\d	Un numero
\D	Todo menos un numero
\s	Un espacio en blanco
\S	Todo menos un espacio en blanco
\w	Una letra
\W	Todo menos letras

Tabla B.3: Intervalos de caracteres predefinidos

Expresión	Encaja con
$\wedge$	Comienzo de una línea
$	Fin de una línea
\b	Fin de palabra
\B	No es fin de palabra
\A	El principio de la cadena de entrada
\G	El final del último patron encajado

Tabla B.4: Intervalos de caracteres POSIX

Expresión	Encaja con
X?	X, una o ninguna vez
X*	X, cero o ninguna vez
X+	una o mas veces
X $\{n\}$	X exactamente n veces
X $\{n,\}$	X por lo menos n veces
X $\{n,m\}$	X por lo menos n veces

Tabla B.5: Cuantificadores de cantidad

Expresión	Encaja con
XY	X seguido de Y
$x \mid Y$	X o Y
(X)	X, como un grupo

Tabla B.6: Operadores lógicos

Expresión	Encaja con
\n	Lo que haya encajado en el $n^{ésimo}$ grupo
\,	Escape, y entrecomillado

Tabla B.7: Referencias hacia atrás

Apéndice C

Programación en *Flex* y *Bison*

Flex, el cual es el sucesor de *lex*, es un programa que genera un archivo en lenguaje C, que representa el analizador léxico.
Al igual que con *flex*; *bison*, sucesor de *yacc*; genera un archivo en C, pero a diferencia de *flex*, lo que representa es el analizador sintáctico.

El analizador léxico puede funcionar por sí solo, mientras que el analizador sintáctico se apoya (hace llamadas) en el analizador léxico, por lo que es necesario incluir este último en el analizador sintáctico (como se verá más adelante).

En la figura C.1 se muestra un esquema de los archivos que deben ser pasados a los respectivos programas y los archivos que se generan. Más adelante en el documento se analizan las inclusiones de los archivos y la forma de compilarlos.

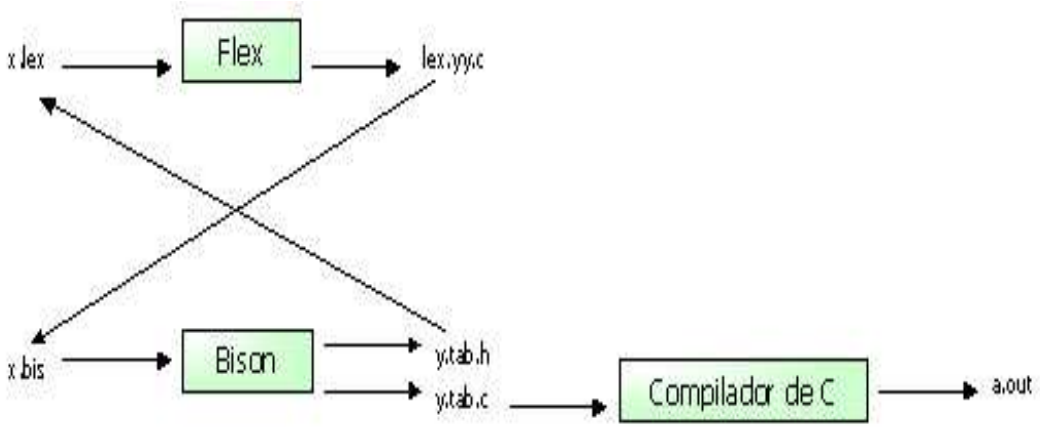


Figura C.1: Flujo de Archivos

Ambos archivos, el que será compilado en flex y el de bison, tienen una estructura en común.

declaraciones

%%

reglas de traducción flex-bison

%%

implementación de funciones de apoyo en c

básicamente consta de tres partes las cuales están separadas por el símbolo "%%" y la primera parte a su vez está separada en declaraciones propias de cada programa y en declaraciónes para C, donde estas últimas están limitadas por los símbolos: "%{" y "%}":

%{

declaraciones para C

%}

declaraciones para el programa *flex-bison*

C.0.1. Estructura específica para el archivo *flex*

Declaraciones para C: es la parte donde se hacen los *includes* y la declaración de variables en código C que sirven de apoyo en la creación del compilador[1] .

Declaraciones para el programa *flex*: Aquí se pueden incluir las definiciones regulares que son proposiciones que se utilizan como componentes en las expresiones regulares que aparecen en las reglas de traducción.

Reglas de traducción *flex*: Son proposiciones de la forma: p_n $\{acciones\}$, donde p_n es una expresión regular y las acciones son un fragmento de programa escrito en c que describe que instrucciones realizará el analizador léxico cuando el patrón p_n concuerda con la cadena leida[2] .

Implementaciones de funciones de apoyo en C: Contiene las funciones auxiliares en C que pueden necesitar las acciones, y donde se define la función main. Para que el analizador léxico se ejecute debe ponerse en alguna parte del código, según la necesidad del programa, la llamada a la función: yylex()[3] .

[1] Cuando el analizador léxico se conjunta con el analizador sintáctico, es necesario incluir el archivo y.tab.h

[2] Cuando el analizador léxico se conjunta con el analizador sintáctico, es necesario agregar la línea de código return $\{token\}$, a cada proposición léxica(reglas de traducción en *flex*), donde *token* son los delcarados en el archivo para bison

[3] Cuando el analizador léxico se conjunta con el analizador sintáctico no se debe de incluir la

C.1. Estructura específica para el archivo *bison*

Declaraciones para C: es la parte donde se hacen los *includes* y declaración de variables en código C que sirven de apoyo en la creación del compilador[4] .

Declaraciones para el programa *bison*: es donde se declara el símbolo inicial de la gramática: %start simbolo, donde símbolo es el BNF inicial de la gramática. También se declaran los componentes léxicos (*tokens*) de nuestro lenguaje, de la forma: %token tok_1 tok_2 ... tok_n, donde tok_n representa cada uno de los *tokens* de la gramática.

Reglas de traducción *bison*: es aquí donde se ponen las reglas del lenguaje (en BNF's), pero con características diferentes (ver capítulo 2).

Implementaciones de funciones de apoyo en C: Contiene las funciones auxiliares que se pueden necesitar, y donde se define la función *main*. Para que el analizador sintáctico se ejecute deberá escribirse en alguna parte del código, según la necesidad del programa, la llamada a función: yyparse.
A continuación se incluye el archivo para *flex* y para *bison*.

función main, ni hacer la llamada al analizador léxico yylex(); ya que el analizador sintáctico es el que se encarga de hacer este trabajo

[4] Es necesario incluir el archivo: lex.yy.c

Archivo para *flex* (x.lex)

```
%{
#include ‘‘y.tab.h"
%}
DIGITO [0-9]
LETRA [a-z A-Z]
ID {LETRA}({LETRA}|{DIGITO})*
NOMARCH {ID}.{ID}
STRING {LETRA}({LETRA})*| {DIGITO}({LETRA}|{DIGITO})*
BLANCO [ \ t]
BLANCOS {BLANCO}+
%%
‘‘⋆"{printf(“ASTERISCO\ t"); return(asterisco);}
‘‘+" | ‘‘-" {printf(‘‘MAS \ t"); return(suma);}
‘‘CREATE" | ‘‘create" {printf(‘‘CREATE \ t "); return(create);}
‘‘CORPUS"| ‘‘corpus" {printf(‘‘CORPUS \ t"); return(corpus);}
‘‘DROP" | ‘‘drop" {printf(‘‘DROP \ t"); return(drop);}
‘‘LIST" | ‘‘list" {printf("LIST \ t"); return(list);}
‘‘FROM" | ‘‘from" {printf(‘‘FROM \ t"); return(from);}
‘‘DISPLAY" | ‘‘display" {printf(‘‘DISPLAY \ t"); return(display);}
‘‘INSERT" | ‘‘insert" {printf(‘‘INSERT \ t"); return(insert);}
‘‘PREPARE" | ‘‘prepare" {print(‘‘PREPARE \ t"); return(prepare);}
‘‘EXTRACT" | ‘‘extract" {print(‘‘EXTRACT \ t"); return(extract);}
‘‘REPLACEALL" | ‘‘replaceall" {print(‘‘REPLACEALL \ t"); return(replaceall);}
‘‘INFOALL" | ‘‘infoall" {print(‘‘INFOALL \ t"); return(infoall);}
‘‘DELETE" | ‘‘delete" {printf(‘‘DELETE \ t"); return(delete);}
```

```
‘‘REMOVE" | ‘‘remove" {print(
‘‘REMOVE \ t"); return(remove);}
‘‘PUT" | ‘‘put" {print(‘‘PUT \ t"); return(put);}
‘‘AND" | ‘‘and" {print(‘‘AND \ t"); return(and);}
‘‘ADD" | ‘‘add" {printf(‘‘ADD \ t"); return(add);}
‘‘TO" | ‘‘to" {print(‘‘TO \ t"); return(to);}
‘‘FOR" | ‘‘for" {print(‘‘FOR \ t"); return(for);}
‘‘ACUTES" | ‘‘acutes" {print(‘‘ACUTES \ t"); return(acutes);}
‘‘STOPWORDS" | ‘‘stopwords" {printf(‘‘STOPWORDS \ t"); return(cerradas);}
‘‘SYMBOLS" | ‘‘symbols" {printf(‘‘SYMBOLS \ t"); return(simbolos);}
‘‘LABELS" | ‘‘labels" {printf(‘‘LABELS \ t"); return(etiquetas);}
‘‘;" {printf(‘‘PUNTO-COMA \ t"); return(pyc);}
{ID} {printf(‘‘IDENTIFICADOR \ t"); return(id);}
{STRING} {printf(‘‘STRING \ t"); return(string);}
‘‘(" {printf(‘‘PARENTESIS-APERTURA \ t"); return(pa);}
‘‘)" {printf(‘‘PARENTESIS-CIERRE \ t"); return(pc);}
‘‘." {printf(‘‘PUNTO \ t"); return(punto);}
{BLANCOS} {printf(‘‘ ");}
%%
```

Archivo para *bison* (x.bis)

```
%{
int errors=0;
#include ‘‘lex.yy.c"
%}
%start EXPRESION
%token
asterisco
suma
create
corpus
drop
list
from
display
insert
prepare
extract
replaceall
infoall
delete
remove
put
and
add
to
```

```
for
acutes
etiquetas
simbolos
cerradas
pyc
id
pa
pc
punto
entero
real
string
%%
EXPRESION :CREATE | DROP | LIST | DISPLAY | INSERT | ADD | DELETE
                | INFOALL | REMOVE | EXTRACT | REPLACEALL;
CREATE :create corpus id pyc ;
DROP :drop corpus id pyc;
LIST :list asterisco from id pyc;
DISPLAY :display id punto ARCHIVO pyc;
INSERT :insert id punto ARCHIVO pyc;
ADD :add DICCIONARIOS from ARCHIVO pyc;
DELETE :delete DICCIONARIOS from ARCHIVO pyc;
INFOALL :infoall id punto ARCHIVO put ARCHIVO pyc;
REMOVE :remove acutes id punto ARCHIVO put ARCHIVO pyc;
EXTRACT : extract string to string id punto ARCHIVO put ARCHIVO pyc ;
```

```
REPLACEALL : replaceall string to string id punto ARCHIVO put ARCHIVO pyc;
DICCIONARIOS :cerradas
              | simbolos
              | etiquetas
              ;
ARCHIVO :id punto id;

%%
yyerror(char *msg) { errors++; }
main()
{
yyparse();
if (errors==0)
          printf("\ n \ n La expresion es correcta \ n ");
else
          printf("\ n \ n La expresion es incorrecta");
}
```

C.2. Descripción del proceso

EL proceso de la generación del compilador resulta sencillo después de haber creado los archivos necesarios para ello. Solo queda compilar los archivos con su respectivo programa. A continuación se listan los comandos necesarios:

flex -v x.lex donde **"-v"** es el parámetro que indica a *flex* que si hay errores en

la compilación lo muestre, **"x.lex"** es el archivo que se escribió para el analizador léxico.

bison -dvy x.bis donde *"-dvy"* es el parámetro que indica a bison si hay errores en la compilación los muestre, *"x.bis"* es el archivo que se escribió para el analizador sintáctico.

cc y.tab.c -lfl donde **"y.tab.c"** es el archivo que se generó al compilar con bison; aquí es importante asegurarse que también se compiló el archivo para *flex*. El parámetro **-lfl** es para indicarle al compilador de C, que debe incluir la librería especial para generar un compilador.

Y por último para poder compilar con el compilador se tiene la siguiente instrucción.

./a.out < entrada donde ./ es para indicarle a linux que se quiere ejecutar un programa fuente del path. **"a.out"** es el compilador, **" < "** es para indicar que vamos a dirigir un archivo como entrada a el compilador y "entrada" es donde se tiene escrito el conjunto de sentencias a analizar.