Procesamiento de Lenguaje Natural

TEMA 3

Análisis sintáctico

Enrique Alfonseca
Pilar Rodríguez

Índice

- **Análisis parcial**
  - Introducción
  - Listas de transformación
  - Aprendizaje basado en memoria
  - Otros sistemas
- **Formalismos**
  - CFG
  - DCG
  - HPSG
  - PCFG
  - Otros
- **Algoritmos de análisis**
  - Introducción
  - Top-down
  - Bottom-up
  - Shift-reduce
  - Left-corner
  - Chart parsing

Introducción

Ciertas agrupaciones de palabras se comportan como *constituyentes*:

- Pueden ocurrir en distintas posiciones de las oraciones.
- Muestran posibilidades sintácticas uniformes.

Algunas pruebas de *constituencia*:

- Movimiento:

  a. Ya estudié *la lección 3*.
  b. *La lección 3*, ya la estudié.
  c. ?Ya *la lección 3* estudié.
  d. *Ya lección 3 estudié la.

- Sustitución

  a. Pedro quiere *una manzana*.
  b. Pedro quiere *un lápiz de otro color*.
  c. Pedro quiere *el coche que vimos ayer*.

Introducción (II)

Sintagma nominal (noun phrase) Van encabezados por nombres.

El perro triste que me encontré ayer en la calle

Sintagma preposicional (prepositional phrase) Van encabezadas por una preposición, y tienen un complemento *sintagma nominal*.

para los niños pobres de todo el mundo

Introducción (III)

Sintagma verbal (verb phrase) Van encabezados por un verbo. En general, organiza todos los demás elementos de la oración.

Llegaremos a tiempo al cine

Sintagma adjetival (adjectival phrase) Van encabezados por adjetivos.

especialmente seguro de sí mismo

Introducción (IV) - Argumentos y adjuntos

Adjuntos o modificadores

- Modifican el significado de aquello que modifican.
- Generalmente lo hacen más específico, sin alterar su significado básico.

  El libro sobre la mesa, Marchamos rápidamente

- Puede haber cualquier número de ellos

  El libro rojo sobre la mesa con dibujos

Argumentos o complementos

- La interpretación del argumento viene completamente determinada por la cabeza a la que complementa.

  discusión con Juan sobre el menú

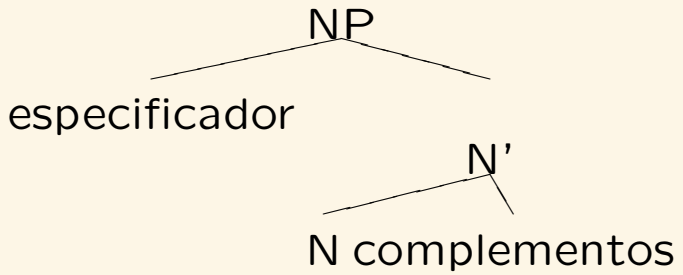
- El número de complementos viene también determinado.

  rey de Francia, retrato de Juan

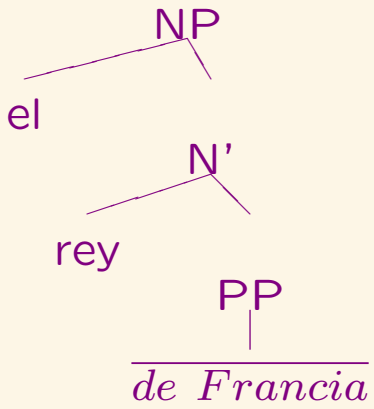
Introducción (V)

Sintagmas nominales

Suelen comenzar con un determinante, seguido de un nombre y sus complementos.



Ejemplo:

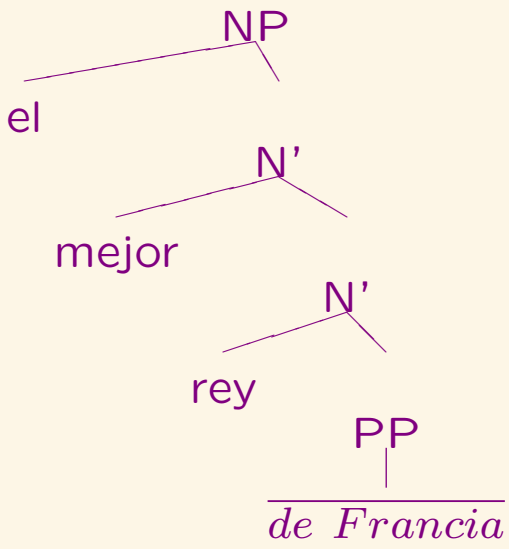


Introducción (VI)

Sintagmas nominales

Opcionalmente, puede tener modificadores a la derecha o a la izquierda de la cabeza.

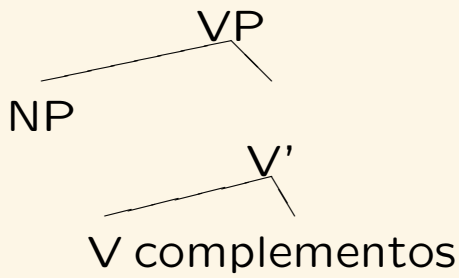
Ejemplo:



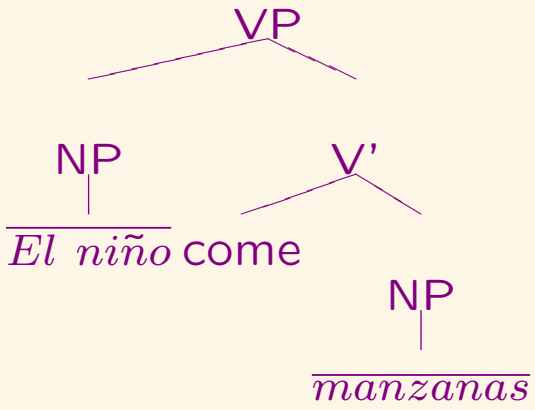
Introducción (VII)

Sintagmas verbales

Suelen comenzar con el sujeto (como especificador), seguido de un verbo y sus argumentos. Puede haber también adjuntos.

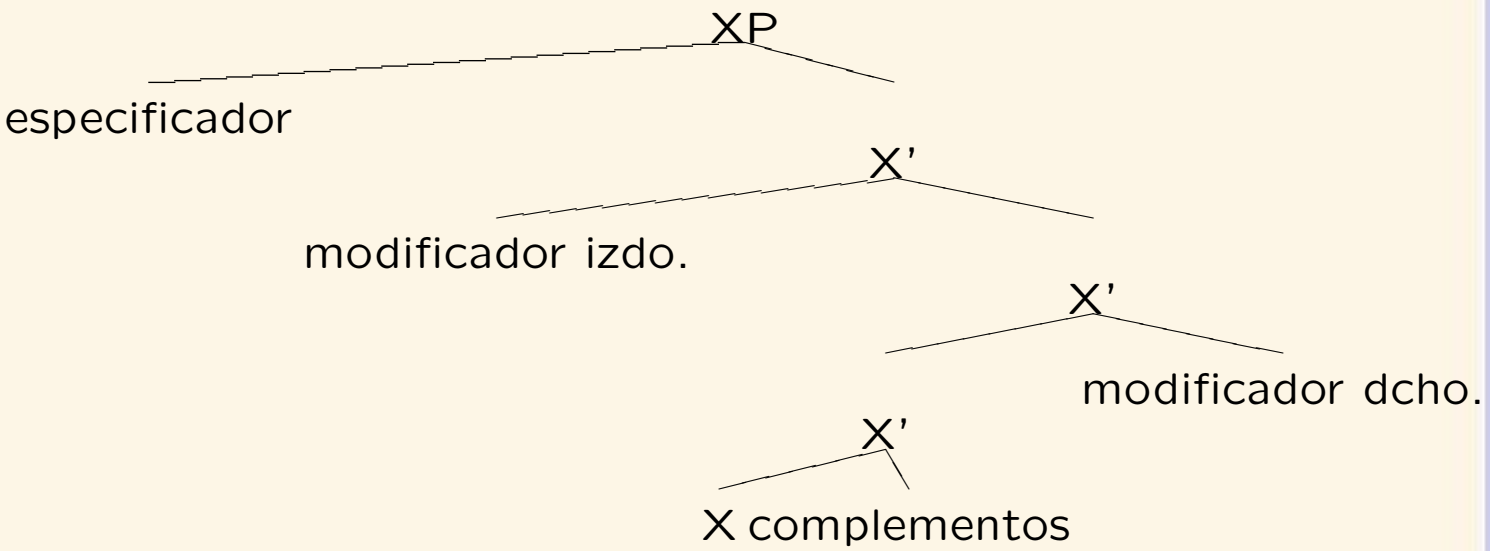


Ejemplo:



Introducción (VIII)

Teoría de la X-barra



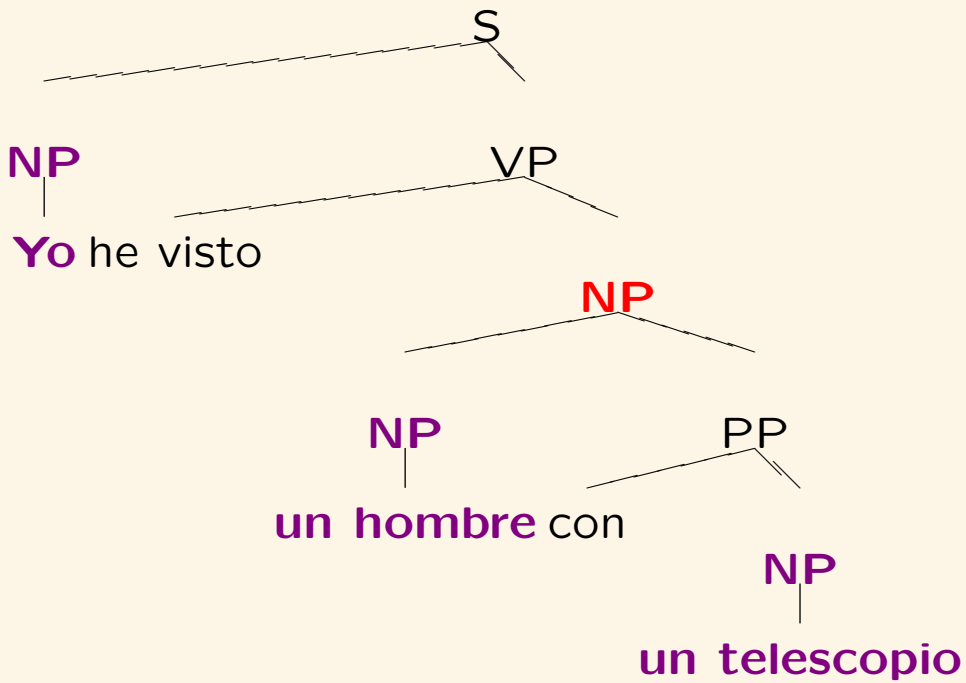
Introducción (IX)

- Abney (1991) propuso implementar análizadores completos como una cascada de pequeños analizadores.
- Cada uno puede especializarse en detectar un tipo particular de constituyente: sintagmas nominales, verbos compuestos, cuantificadores, sintagmas adjetivales, etc.
- Previamente, Church (1988) había construido a mano un detector de sintagmas nominales.

Introducción (X)

NP chunking

- Sintagma nominal básico: sintagma nominal no recursivo y sin solapamiento.
- *NP chunking*: identificación de sintagmas nominales básicos en un texto.



Listas de transformación (I)

(Ramshaw and Marcus, 1995)

- Lista de reglas usada para clasificar objetos en distintas clases.
- Las reglas se aplican en orden, de modo que los efectos de una regla influyen en las siguientes.

$\Rightarrow$ ¡¡¡Igual que las utilizadas en etiquetado de las partes del lenguaje!!!

Listas de transformación (II)

Representación del problema

† Las palabras serán clasificadas en tres conjuntos, llamados `I`, `O` y `B`.

- `I` todas aquellas palabras que están dentro de un sintagma nominal básico.
- `O` todas aquellas palabras que están fuera de un sintagma nominal básico.
- `B` todas aquellas palabras que están al comienzo de un sintagma nominal básico, siempre y cuando la palabra anterior está al final de otro sintagma nominal.

Ejemplo

```
[Dio] a [Francisca] [un abrazo]
Dio/I a/O Francisca/I un/B abrazo/I
```

Listas de transformación (III)

Procedimiento

1. Aprender una Lista de Transformación para etiquetar cada palabra con una de las tres etiquetas, {I, O, B}.
2. Cada palabra recibirá inicialmente una etiqueta. Por ejemplo consideramos que todos los nombres, pronombres, adjetivos y determinantes constituyen por sí mismos sintagmas nominales.
3. Cada regla, en función de la palabra actual, las palabras que la rodean, y sus etiquetas decidirá si se cambia la etiqueta o no.

Listas de transformación (IV)

Ejemplo

El hombre dio a un amigo un regalo

El/I hombre/I dio/O a/O un/I amigo/I un/B regalo/I

Etiquetado inicial:

El hombre dio a un amigo un regalo

El/I hombre/B dio/O a/O un/I amigo/B un/B regalo/B

Regla: Si la palabra actual es un nombre común etiquetado como B, y va precedido por un determinante, cambiar la etiqueta del nombre a I.

Resultado:

El hombre dio a un amigo un regalo

El/I hombre/I dio/O a/O un/I amigo/I un/B regalo/I

Listas de transformación (V)

Algoritmo de aprendizaje

Inicializar las etiquetas `IOB`.

Repetir

- Sea x un ejemplo arbitrario con la etiqueta errónea.
- Generar todas las reglas que arreglan la etiqueta de x
- Para cada una de ellas, calcular la mejora que produciría aplicada a la totalidad del corpus:
  $Ganancia(r) = Errs\ arreglados - Errs\ producidos$
- Tomar la regla $Best$ que maximiza la ganancia.
- Añadir $Best$ al final de la Lista de Transformación, y aplicarla a los datos de entrenamiento
- Recalcular el conjunto de ejemplos aún no cubiertos por la Lista de Transformación.

Listas de transformación (VI)

Algoritmo de etiquetado

1. Inicializar las etiquetas `IOB`.

2. Para cada regla r de la lista,
   - Aplicarla al texto.
3. Devolver el texto anotado.

Listas de transformación (VII)

Evaluación

$$Precision = \frac{NPs\ correctos\ encontrados}{NPs\ encontrados}$$

$$Recall = \frac{NPs\ correctos\ encontrados}{NPs\ totales}$$

$$F(\beta) = \frac{(\beta^2 + 1) \cdot P \cdot R}{\beta^2 \cdot P + R}$$

Correcta: El hombre dio a un amigo un regalo

Obtenida: El hombre dio a un amigo un regalo

Precision = 0.5, Recall = 0.33, F(1) = 0.4

Aprendizaje basado en memoria (I)

(Argamon, 1998; Veenstra, 1998; Tjong Kim Sang, 1999)

- Se almacena en memoria todo el corpus de entrenamiento, ya etiquetado.
- A la hora de encontrar sintagmas nominales básicos en un texto nuevo, se buscan fragmentos similares del corpus de entrenamiento.
  $\rightarrow$ se pueden usar heurísticas para escoger entre varias opciones.
- Cada fragmento se etiqueta igual que alguna de sus apariciones en el corpus.

Aprendizaje basado en memoria (II)

- El aprendizaje es muy rápido (simplemente cargar el corpus de entrenamiento).
- Ocupa mucha más memoria que las listas de transformación.

- Se puede implementar con algoritmos de búsqueda muy rápidos, pero nunca puede igualar a las listas de transformación (que se pueden compilar como autómatas finitos, que se ejecutan en tiempo lineal).

Otros procedimientos (I)

- Aprendizaje de gramáticas para sintagmas nominales (Cardie y Pierce, 1998).
- Redes neuronales (Muñoz, 1999).

- Combinación de los anteriores (Tjong Kim Sang, 2000)
- Support Vector Machines (Kudo, 2001)

Todos ellos han ido aumentando la F-score($\beta = 1$) de 92.03% (Ramshaw y Marcus, 1994) a 94.22% (Kudo, 2001).

Otros procedimientos (II)

Otra posibilidad es subdividir la identificación de sintagmas nominales aún más, en varios *chunkers* diferentes:

1. Identificación de expresiones cuantificadoras.
2. Identificación de sintagmas adjetivales.
3. Identificación de los sintagmas nominales.

Aplicado al aprendizaje con listas de transformación, y entrenado en un corpus depurado, se consigue una F($\beta = 1$) de 94.94% utilizando un PoS-tagger basado en cadenas de Markov.

También puede utilizarse para localizar otros tipos de constituyentes, como verbos compuestos

Otros procedimientos (III)

Demostración

Índice

- **Análisis parcial**
  - Introducción
  - Listas de transformación
  - Aprendizaje basado en memoria
  - Otros sistemas
- **Formalismos**
  - CFG
  - DCG
  - HPSG
  - PCFG
  - Otros
- **Algoritmos de análisis**
  - Introducción
  - Top-down
  - Bottom-up
  - Shift-reduce
  - Left-corner
  - Chart parsing

Context-Free Grammars (I)

Una gramática independiente del contexto $\mathcal{G}$ está formada por:

- Un conjunto de símbolos terminales T (en el caso del lenguaje, son las palabras).
- Un conjunto de símbolos no terminales N, que representan los diferentes tipos de *constituyentes*.
- Un símbolo inicial S.
- Un conjunto de reglas de producción, que indican las maneras en que podemos derivar oraciones válidas a partir del símbolo inicial.

Context-Free Grammars (II)

Ejemplo

```
S ::= NP VP
NP ::= DT NNS
 | DT NN
VP ::= VB NP
 | VB
DT ::= los | un
NNS ::= niños | hombres
VB ::= comen | cantan | miran
NN ::= kiwi | canto | coche
```

Context-Free Grammars (III)

Utilizando las reglas de la gramática, podemos derivar oraciones:

- Una regla se aplica sustituyendo el símbolo que aparece en la parte izquierda por los que aparecen en su parte derecha.

  $\texttt{S} \Rightarrow \texttt{NP VP}$

- Una derivación es la aplicación sucesiva de reglas hasta obtener una expresión que sólo contiene símbolos terminales.

Context-Free Grammars (IV)

Ejemplo

$$
\begin{aligned}
\texttt{S} &\Rightarrow \texttt{NP VP} \\
&\Rightarrow \texttt{DT NNS VP} \\
&\Rightarrow \texttt{los NNS VP} \\
&\Rightarrow \texttt{los niños VP} \\
&\Rightarrow \texttt{los niños VB NP} \\
&\Rightarrow \texttt{los niños miran NP} \\
&\Rightarrow \texttt{los niños miran DT NN} \\
&\Rightarrow \texttt{los niños miran un NN} \\
&\Rightarrow \texttt{los niños miran un coche}
\end{aligned}
$$

Podemos abreviar la derivación:

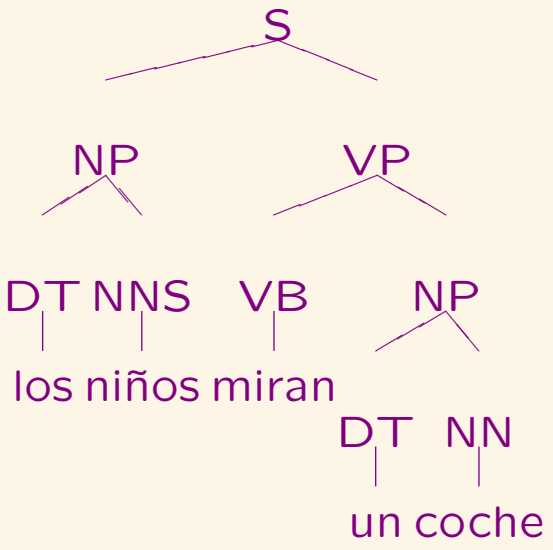
$\texttt{S} \Rightarrow^{\star} \texttt{los niños miran un coche}$

Context-Free Grammars (V)

Ejemplo

La manera en que se ha realizado la derivación se puede capturar con un árbol de derivación sintáctica:

$\mathtt{S} \Rightarrow^{\star} \mathtt{los\ niños\ miran\ un\ coche}$



Context-Free Grammars (VI)

Lenguaje generado por una gramática

Se dice que una gramática $\mathcal{G}$ genera una cadena s si

$$S \Rightarrow^{\star} s$$

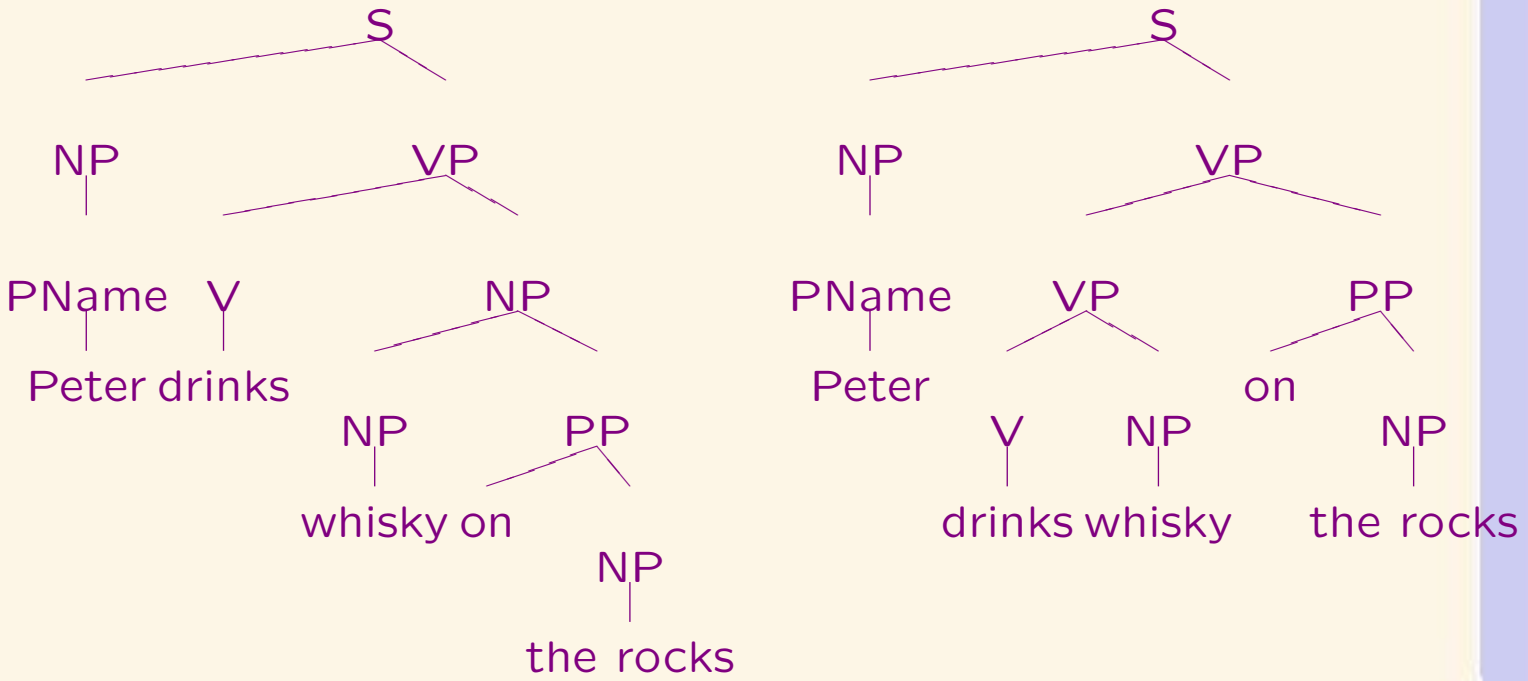
El lenguaje generado por una gramática $\mathcal{G}$ se define como

$$L(\mathcal{G}) = \{s \epsilon T^* | S \Rightarrow^{\star} s\}$$

Context-Free Grammars (VII)

Ambigüedad

```
S ::= NP VP          VP ::= V NP
NP ::= PName         VP ::= V
NP ::= whisky        VP ::= VP PP
NP ::= the rocks     PName ::= Peter
NP ::= NP PP         V ::= drinks
PP ::= on NP
```



Context-Free Grammars (VIII)

Utilidad

- Generación de texto, para...
  - Guías de museos.
  - Sistemas de recomendación.
  - Interfaces de diálogo.
  - etc.
- Análisis de texto, para...
  - Correctores gramaticales
  - Generación de resúmenes
  - Búsqueda de respuestas a preguntas
  - Extracción de Información
  - etc.

Context-Free Grammars (X)

Problemas

Para añadir concordancia, habría que multiplicar el número de reglas por todas las posibilidades de concordancia:

```
S ::= NP_1sg VP_1sg
S ::= NP_2sg VP_2sg
S ::= NP_3sg VP_3sg
S ::= NP_1pl VP_1pl
S ::= NP_2pl VP_2pl
S ::= NP_3pl VP_3pl
VP_1sg ::= VB_intrans_1sg
VP_1sg ::= VB_trans_1sg NP
VP_1sg ::= VB_ditrans_1sg NP NP
...
```

El número de reglas crece de forma exponencial sobre el número de características que queremos incorporar a la gramática.

Definite-Clause Grammars (I)

- Es igual que una CFG, pero permite generalizar las reglas.
- Los no terminales pueden ser predicados de lógica de primer orden, con argumentos.
- En esos argumentos, se pueden codificar los parámetros para implementar concordancia y otros fenómenos lingüísticos (movimiento, topicalización, etc.)

Definite-Clause Grammars (II)

Ejemplo

```
S(Per,Num) ::= NP(Per,Num) VP(Per,Num)
VP(Per,Num) ::= VB_trans(Per,Num) NP(_,_)
NP(Per,Num) ::= PRON(Per,Num)
NP(3,Num) ::= DT(Num), NN(Num)

VB_trans(1, sg) ::= como
VB_trans(2, sg) ::= comes
VB_trans(3, sg) ::= come
VB_trans(1, pl) ::= comemos
VB_trans(2, pl) ::= coméis
VB_trans(3, pl) ::= comen

PRON(1, sg) ::= yo
PRON(1, pl) ::= nosotros

DT(sg) ::= el
DT(pl) ::= los
NN(sg) ::= niño | plátano
NN(pl) ::= niños | plátanos
```

$S \Rightarrow^{\star}$ los niños comen el plátano

Definite-Clause Grammars (III)

- Cada regla en una DCG equivale a un número arbitrario de reglas de una CFG, que se obtienen sustituyendo las variables por cualquier combinación de constantes.
- El lenguaje generado por una DCG es el lenguaje generado por esas reglas CFG.
- El número de reglas CFG a que equivale una DCG puede ser infinito (por ejemplo, si un argumento es numérico y admite cualquier valor).
  $\Rightarrow$ La clase de lenguajes que se pueden representar con una DCG es estrictamente mayor que con una CFG.

Definite-Clause Grammars (IV)

Ejemplo con tiempos y casos

```
S(Per,Num) ::= NP(Per,Num,_,nom) VP(Per,Num,_)
VP(Per,Num,Tiempo) ::= VB_trans(Per,Num,Tiempo) NP(_,_,_,acc)
NP(Per,Num,Gen,Caso) ::= PRON(Per,Num,Gen,Caso)
NP(3,Num,Gen,Caso) ::= DT(Num,Gen), NN(Num,Gen)

VB_trans(1, sg, pres) ::= como
VB_trans(2, sg, pres) ::= comes
VB_trans(3, sg, pres) ::= come
VB_trans(1, pl, pres) ::= comemos
VB_trans(2, pl, pres) ::= coméis
VB_trans(3, pl, pres) ::= comen

PRON(1, sg, _, nom) ::= yo
PRON(1, sg, _, acc) ::= me
PRON(1, pl, _, nom) ::= nosotros

DT(sg, masc) ::= el
DT(pl, masc) ::= los
NN(sg, masc) ::= niño | plátano
NN(pl, masc) ::= niños | plátanos
```

Definite-Clause Grammars (V)

Demostración

HPSG (I)

Introduction (I)

Movimiento:

Varios fenómenos lingüísticos conllevan un movimiento de constituyentes:

- Preguntas: el sintagma sobre el que se pregunta se mueve al comienzo. En inglés, el auxiliar se adelanta también:

  ¿A quién$_i$ viste t_i en el teatro?

  Whom$_i$ did you see t_i at the theatre?

- Topicalización:

  Este libro$_i$, yo lo$_i$ quiero.

  This book$_i$, I want t_i.

El constituyente que se ha movido deja una *trace* (huella) en el lugar donde se originó.

HPSG (II)
Introduction (II)

Papeles semánticos:
Los argumentos de nombs y verbos se pueden clasificar en función de los papeles semánticos que realizan:

- Agente: aquello que realiza una acción.
- Paciente: aquello sobre lo que se realiza una acción.
- Instrumento: aquello con lo que la acción se lleva a cabo.

HPSG (III)
Introduction (III)

En general, el papel que realiza un constituyente vendrá determinado por la función sintáctica que realiza.

Enviar

- `sujeto` = `agente`
- `objeto` = `paciente`
- `objeto indirecto` = `receptor`

Recibir

- `sujeto` = `receptor`
- `objeto` = `paciente`
- `pp(de)` = `agente`

Voz activa vs. pasiva

HPSG (IV)

Introduction (IV)

Subcategorización:

Los verbos se pueden clasificar según el número de argumentos que requieren, y su tipo.
La mayoría de los argumentos se representan con sintagmas nominales, pero hay excepciones. Algunos ejemplos son:

- Comer necesita dos NPs: sujeto y complemento directo.

  Mafalda comió la sopa

- Dar necesita dos NPs (sujeto y complemento directo) y un PP(a) (complemento indirecto).

  Gustavo le dio el micrófono a Peggy

HPSG (V)
Introduction (V)

Subcategorización:

- Despojar necesita un NP (sujeto), un PP(a) (complemento indirecto) y un PP(de).

  Los orcos despojaron a los hobbits de su libertad

- Decir necesita un NP (sujeto) y un S(that) (complemento directo).

  Dice la leyenda que el anillo se perdió

- Volverse necesita un NP (sujeto) y un adjetivo (predicativo).

  Grendel se volvió loco.

Se llama subcategorización de un verbo a la descripción de sus argumentos.

HPSG (VI)
Gramáticas lexicalizadas (I)

Una solución elegante para incorporar información de movimiento y subcategorización en la gramática es proporcionarla para cada palabra.
Ejemplos:

```
v([vform:fin,tns:pres|_],np([per:3,num:sg,case:nom|_]))
--> [sleeps].
```

She sleeps

```
n([lex:plus|N],s([vform:bse,tns:_,comp:that|_]))
--> [desire].
```

The desire that he be granted asylum

HPSG (VII)

Gramáticas lexicalizadas (II)

- Ahora, la estructura de las oraciones las proporcionan las reglas que generan los símbolos terminales (las palabras).
- Las reglas entre símbolos no terminales simplemente han de ir rellenando los argumentos de los terminales.

```
s([Vform,TNS,comp:minus|V]) -->
  np(Subj),
  vp([Vform,TNS|V],np(Subj)).

vp(V,Subj) -->
  v(V,Subj,np(Obj)), % transitive
  np(Obj).

vp(V,Subj) -->
  v(V,Subj,np).  % intransitive
```

HPSG (VIII)

Gramáticas lexicalizadas (III)

Demostración

HPSG (IX)

- Feature Structure Grammars (FSG) son una formalización de la gramática lexicalizada.
- Cada palabra tiene atributos y valores.
- Se representa con una AVM (Attribute-Value Matrix)

```
v([vform:fin,tns:pres|_],np([per:3,num:sg|_]),np(_)) -->
[writes].
```

$$\begin{bmatrix} \text{lex} & \textit{writes} \\ \text{cat} & \textbf{verb} \\ \text{vform} & \textbf{fin} \\ \text{tns} & \textbf{pres} \\ \text{subj} & \begin{bmatrix} \text{cat} & \textbf{np} \\ \text{per} & 3 \\ \text{num} & \text{sg} \end{bmatrix} \\ \text{obj1} & \begin{bmatrix} \text{cat} & \textbf{np} \end{bmatrix} \end{bmatrix}$$

HPSG (X)

Head-Driven Phrase Structure Grammars (HPSG)

- Estructuras de rasgos para representar las palabras.
- Las reglas de la gramática consisten en combinar las distintas estructuras, unificando los argumentos de unas palabras con las estructuras de otros constituyentes.

PCFG (I)

Una gramática independiente del contexto probabilística es:

- Un conjunto de símbolos terminales T (palabras).
- Un conjunto de símbolos no terminales N.
- Un símbolo inicial S.
- Un conjunto de reglas de producción,

  $$N_i \rightarrow \mathcal{G}_j$$

  donde $\mathcal{G}_j$ es una secuencia de símbolos terminales y no terminales.
- Un conjunto de probabilidades sobre las reglas

  $$P(N_i \rightarrow \mathcal{G}_j | N_i)$$

  tal que

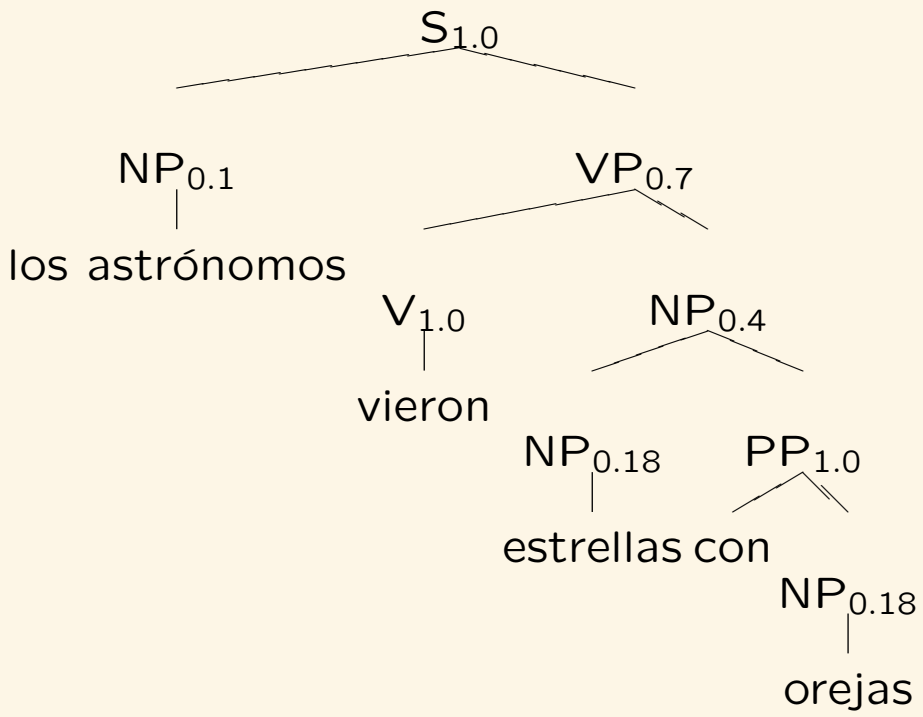
  $$\forall i \sum_j P(N_i \rightarrow \mathcal{G}_j | N_i) = 1$$

PCFG (II)

- Hasta ahora, dada una oración, podíamos obtener muchas derivaciones posibles, y nada nos permitía discriminar entre ellas.
- La probabilidad de una derivación sintáctica es el producto de las probabilidades de todas las reglas que se han aplicado.
- De este modo, podemos escoger el árbol de derivación con la mayor probabilidad.
- Las probabilidades se pueden estimar a partir de un corpus anotado.

PCFG (III)

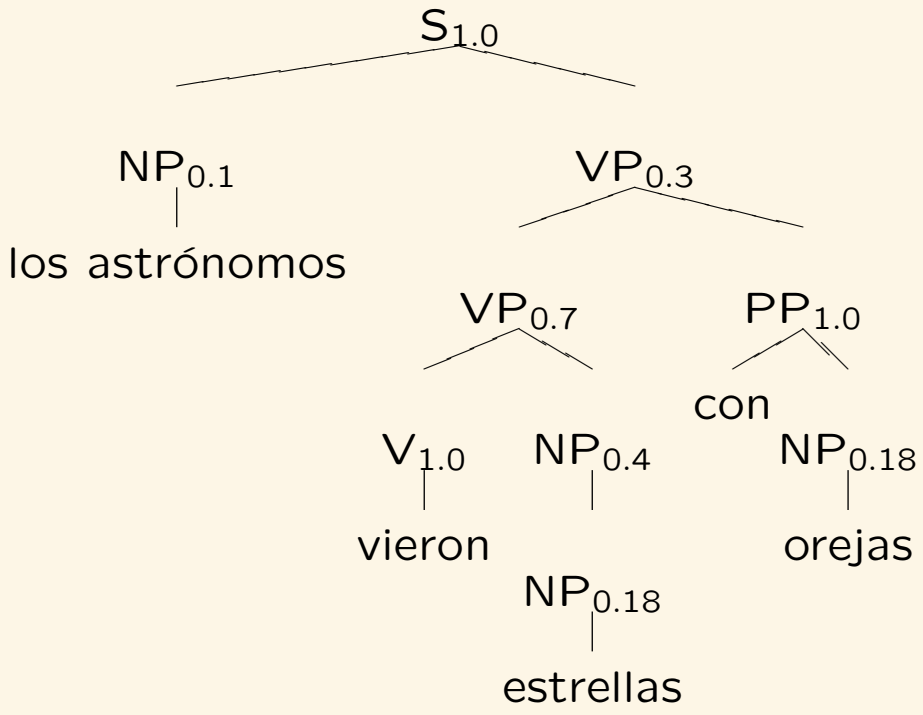
S ::= NP VP	1.0	NP ::= NP PP	0.4
PP ::= P NP	1.0	NP ::= los astrónomos	0.1
VP ::= V NP	0.7	NP ::= estrellas	0.18
VP ::= VP PP	0.3	NP ::= orejas	0.18
P ::= con	1.0	NP ::= telescopios	0.1
V ::= vieron	1.0	NP ::= espejos	0.04



$$P = 1.0 \times 0.1 \times 0.7 \times 1.0 \times 0.4 \times 0.18 \times 1.0 \times 0.18 = 0.0009072$$

PCFG (IV)

S ::= NP VP	1.0	NP ::= NP PP	0.4
PP ::= P NP	1.0	NP ::= los astrónomos	0.1
VP ::= V NP	0.7	NP ::= estrellas	0.18
VP ::= VP PP	0.3	NP ::= orejas	0.18
P ::= con	1.0	NP ::= telescopios	0.1
V ::= vieron	1.0	NP ::= espejos	0.04



$$P = 1.0 \times 0.1 \times 0.3 \times 0.7 \times 1.0 \times 0.18 \times 1.0 \times 0.18 = 0.0006804$$

Otros formalismos (I)

Tree Adjoining Grammars (TAG)

- Cada palabra se representa con la porción del árbol de derivación sintáctica que le corresponde, con huecos para sus argumentos.
- Analizar una frase consiste en combinar los árboles asociados a cada una de las palabras de la frase unos con otros.
- A cada árbol se le pueden añadir probabilidades.

Otros formalismos (I)

Combinatory Categorial Grammar (CCG)

- Cada palabra se representa como el constituyente del cual es raíz (los verbos representan oraciones).
- Además, para cada una se añade información de los argumentos que le faltan por la derecha o por la izquierda.
  Un verbo transitivo sería:
  $$S\backslash NP/NP$$
- Para analizar una frase, se combinan las representaciones de las palabras de la frase unas con otras.
- A cada posible representación de una palabra se le pueden añadir probabilidades.

Índice

- **Análisis parcial**
  - Introducción
  - Listas de transformación
  - Aprendizaje basado en memoria
  - Otros sistemas
- **Formalismos**
  - CFG
  - DCG
  - HPSG
  - PCFG
  - Otros
- **Algoritmos de análisis**
  - Introducción
  - Top-down
  - Bottom-up
  - Shift-reduce
  - Left-corner
  - Chart parsing

Introducción (I)

- Un analizador sintáctico busca posibles derivaciones de una frase a partir de la gramática.
- Un analizador es completo si para toda frase produce todas las derivaciones posibles de ese árbol.
- En el caso de gramáticas probabilísticas, puede interesarnos conocer tan solo los K análisis más probables.
- En otro caso, a veces basta con conocer una sola derivación correcta.

Introducción (II)

El número de posibles derivaciones suele ser exponencial:

Ejemplo:

```
S ::= X | Y | λ
X ::= aS
Y ::= aS
```

Hay 2^n maneras de generar a^n con esta gramática.

Analizador Top-down

- Mantiene un conjunto de constituyentes que tiene que formar con las palabras de la oración.
- Inicialmente, es el símbolo inicial, S.
- Va sustituyendo los constituyentes aplicando las reglas de la gramática, hasta que genera las palabras buscadas.

En ciertos momentos habrá que tomar una decisión:

1. Cuando haya varias reglas con la misma parte izquierda (reglas aplicables).
2. El orden en el cual aplicar las reglas.

Analizador Top-down

Ejemplo:

Pendiente	Palabras	Regla aplicada
s	Pedro quiere comida	
np vp	Pedro quiere comida	s ::= np vp
Pedro vp	Pedro quiere comida	np ::= Pedro
vp	quiere comida	
v np	quiere comida	vp ::= v np
quiere np	quiere comida	v ::= quiere
np	comida	
comida	comida	np ::= comida
λ	λ	

Analizador Top-down

Características:

- Si un camino de análisis no lleva a la oración buscada, se hace *backtracking* y se intenta otro.
- Un analizador top-down no termina nunca si la gramática contiene reglas recursivas por la izquierda

  ```
  NP ::= NP PP,
  ```

  incluso aunque sean recursivas como combinación de varias reglas:

  ```
  NP ::= Det NN
  Det ::= NP 's
  ```

Analizador Bottom-up

- Comienza con las palabras de la oración.
- Si una sub-cadena de símbolos coincide con la parte derecha de una regla, la reemplaza por la parte izquierda.
- Se termina cuando se han reducido todas al símbolo inicial, S.

En ciertos momentos habrá que tomar una decisión:

1. Cuando haya varias reglas con la misma parte derecha (reglas aplicables).
2. El orden en el cual aplicar las reglas.

Analizador Bottom-up

Ejemplo:

Pendiente	**Regla aplicada**
Pedro quiere comida	np ::= Pedro
np quiere comida	v ::= quiere
np v np	np ::= comida
np v np	vp ::= v (*intransitivo*)
np vp np	s ::= np vp
s np	*backtrack*
np v np	vp ::= v np (*transitivo*)
np vp	s ::= np vp
s	

Analizador Bottom-up

Características:

- Poco eficiente si hay mucha ambigüedad léxica.

Analizador Shift-reduce

- Es un caso particular del bottom-up.
- La lista de palabras pendientes se divide en dos tipos: completadas y originales.
- Las operaciones se dividen en dos tipos:
  1. shift: cuando una palabra *original* se transforma en un constituyente *completado*.
  2. reduce: una operación entre palabras completadas.

Analizador Shift-reduce

Completadas	Originales	Operación
	Pedro quiere comida	
np	quiere comida	shift
np v	comida	shift
np v np		shift
np vp		reduce
s		reduce

Analizador Left corner

Definición: un símbolo A es left-corner de un símbolo B si

- $A = B$.
- Existe una regla `B ::= A ....`
- Existe un símbolo C tal que A es left-corner de C y C es left-corner de B.

Analizador Left corner

Ejemplos:

```
S ::= NP VP
NP ::= Det N
VP ::= V PP | V NP
```

Los siguientes son left-corners directamente por las reglas: `lc(NP, S). lc(Det, NP). lc(V, VP).`

Los siguientes lo son por la propiedad transitiva: `lc(Det, S).`

Por último, todos son left-corner de sí mismos:

`lc(S, S). lc(NP, NP). lc(VP, VP).`

`lc(Det, Det). lc(PP, PP). lc(N, N).`

`lc(V, V).`

Analizador Left corner (III)

- El analizador left-corner es un analizador *bottom-up*, pero la información de left-corner se utiliza para reducir el número de decisiones equivocadas.
- Sólo se aplican las reglas cuyo primer elemento es un left-corner del constituyente que queremos reconocer.

  $\rightarrow$ ¿Por qué?

Chart parsing (tabulated parsing)

- Los algoritmos que hemos visto hasta ahora pueden tardar un tiempo exponencial en analizar una frase (dependiendo de la gramática).
- Es posible mejorarlo bastante utilizando programación dinámica.

Chart parsing (II)
Forma Normal de Chomsky

Una gramática está en Forma Normal de Chomsky si todas las reglas son de una de las siguientes dos formas:

1. Reglas binarias con símbolos no terminales:

   ```
   A ::= B C
   ```

2. Reglas unarias que generan símbolos terminales:

   ```
   A ::= w
   ```

Teorema: Para toda CFG existe una gramática equivalente en Forma Normal de Chomsky.

Chart parsing (III)

Algoritmo de Cocke-Kasami-Younger

El algoritmo CKY funciona del siguiente modo:

1. La gramática debe estar en Forma Normal de Chomsky.
2. Define una función *multiplicación*, que toma dos conjuntos de no terminales $A = \{a_1, ..., a_m\}$ y $B = \{b_1, ..., b_n\}$ y devuelve un conjunto $G = \{g_1, ..., g_k\}$ tal que

$$\forall g_i \exists a_j \epsilon A \wedge \exists b_k \epsilon B such\ that [g_i ::= a_j b_k] \epsilon \mathcal{G}$$

Por ejemplo, si `NP` pertenece a A y `VP` pertenece a B, entonces `S` pertenece a G, dado que `S ::= NP VP` es una regla de la gramática.

Chart parsing (IV)
Algoritmo de CKY

3. Se crea una matriz de tamaño $(n+1)^2$, donde n es la longitud de la oración a analizar.
4. Se va completando la matriz, de modo que la casilla (i, j) es

$$chart(i,j) = \cup_{i<k<j} chart(i,k) * chart(k,j)$$

Chart parsing (V)

Algoritmo de CKY

Para j desde 1 hasta n,
  $chart(j-1,j) = \{A | A ::= word_j\}$
  Para i desde $j-2$ hasta 0,
    Para k desde $i+1$ hasta $j-1$,
      $chart(i,j) = chart(i,j) \cup (chart(i,k) * chart(k,j))$
Si $s \epsilon chart(0,n)$ return $true$ else return $false$.

La complejidad es $O(n^3)$.

Chart parsing (VI)

Ejemplo

$_0$ *Pedro* $_1$ *usa* $_2$ *el* $_3$ *coche* $_4$ *de* $_5$ *Paco* $_6$

```
S ::= NP VP
NP ::= NP PP
VP ::= V NP | VP PP
NP ::= Det N
NP ::= pedro | paco
PP ::= P NP
Det ::= el
N ::= coche | pato
V ::= usa | ve
P ::= de
```

	0	1	2	3	4	5	6
0		NP	-	-	S		S,S
1			V	-	VP		VP,VP
2				Det	NP		NP
3					N		-
4						P	PP
5							NP
6							