Procesamiento del Lenguaje Natural

# Tema 4. Análisis sintáctico

# Índice

Esquema

Ideas clave

4.1. Introducción y objetivos

4.2. Sintaxis

4.3. Gramáticas de estructura sintagmática

4.4. Ambigüedad en el análisis sintáctico

4.5. Métodos para el análisis sintáctico, basados en la programación dinámica

4.6. Métodos probabilistas en el análisis sintáctico

4.7. Gramáticas de dependencias o gramáticas valenciales

4.8. Referencias bibliográficas

A fondo

Modelos probabilísticos para el análisis sintáctico de dependencias

Análisis sintáctico de dependencias

Técnicas de procesamiento de lenguaje

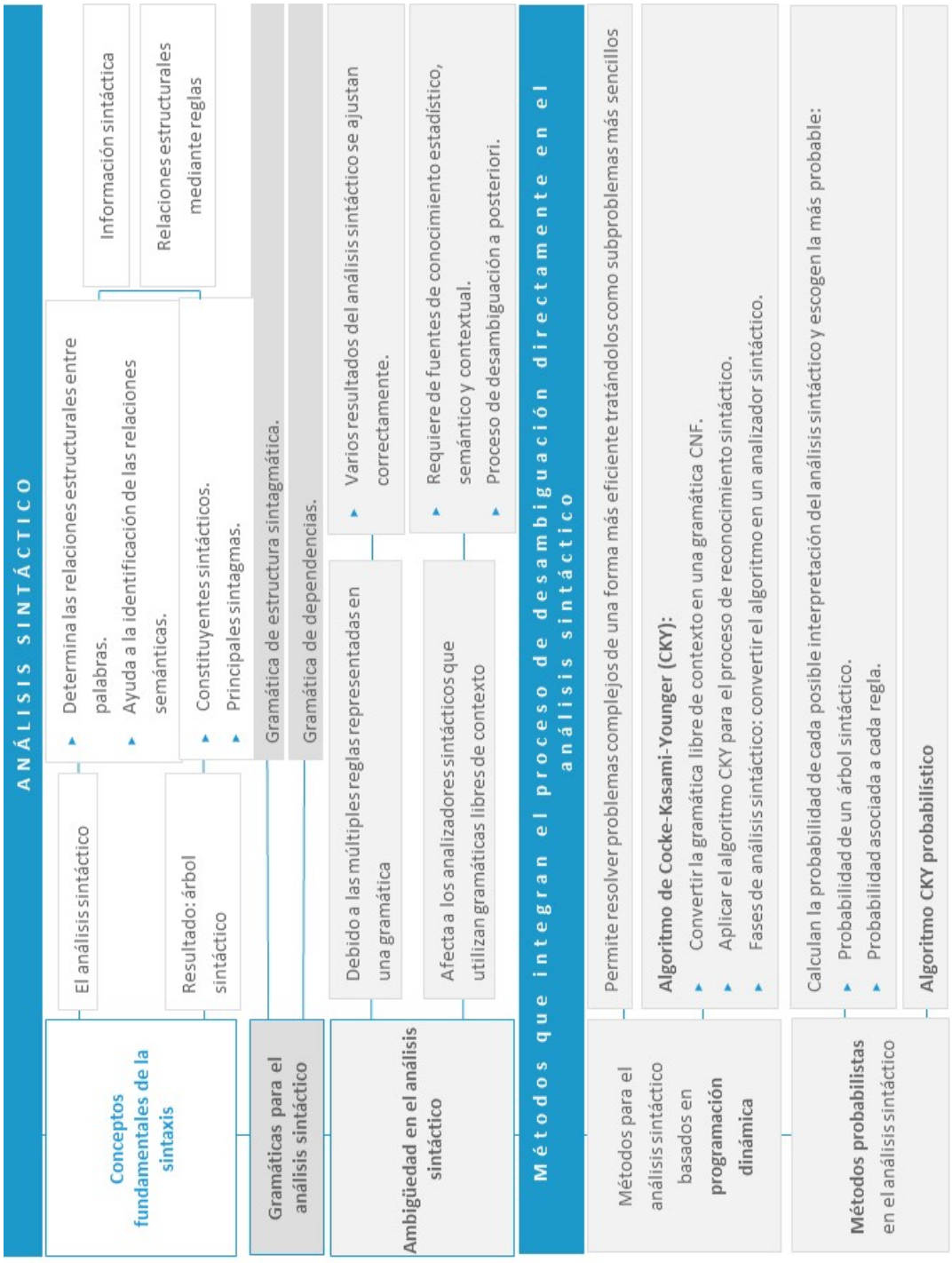
Análisis sintáctico: estrategia ascendente y búsqueda en profundidad

Análisis sintáctico: estrategia descendente y búsqueda en profundidad

Análisis sintáctico: estrategia ascendente y búsqueda en anchura

Test

# Esquema



## 4.1. Introducción y objetivos

Se profundizará en el análisis sintáctico y las relaciones estructurales entre las palabras. Esta información sintáctica se modela mediante las **gramáticas sintagmáticas,** de las cuales veremos las **gramáticas libres de contexto.** Tras esto, analizaremos el **concepto de ambigüedad estructural**, uno de los mayores problemas que se dan en el análisis sintáctico, y cuáles son las técnicas para solucionarla como son los métodos basados en programación dinámica y los métodos probabilistas.

### Objetivos

- Entender cómo el problema de la ambigüedad en la estructura de las frases afecta al funcionamiento de los analizadores sintácticos basados en búsqueda, ya sea ascendente o descendente.
- Describir el funcionamiento de algunas técnicas de programación dinámica que se utilizan en el análisis sintáctico porque son capaces de representar ambigüedades.
- Describir el concepto de gramática libre de contexto probabilística y modelar un problema de ambigüedad utilizando este tipo de gramática.
- Aplicar métodos basados en gramáticas libres de contexto probabilísticas para solventar problemas de ambigüedad.
- Definir el concepto de gramática de dependencias o gramática valencial e identificar sus diferencias respecto a las gramáticas de estructura sintagmática.

## 4.2. Sintaxis

La Real Academia Española (RAE, s. f.) en su diccionario de la lengua española define la palabra **sintaxis:**

> «Del lat. tardío *syntaxis*, y este del gr. *σ νταξις sýntaxis* , de *συντ σσειν syntássein* "disponer conjuntamente", "ordenar". 1. f. Gram. Parte de la gramática que estudia el modo en que se combinan las palabras y los grupos que estas forman para expresar significados, así como las relaciones que se establecen entre todas esas unidades».

### El análisis sintáctico

En el análisis sintáctico se **determinan las relaciones estructurales entre palabras** y es muy relevante en el procesamiento del lenguaje natural no solo por la información sintáctica que aporta, también porque es un paso esencial para la posterior identificación de las relaciones semánticas de las oraciones.

**Árbol sintáctico**

Es el resultado del análisis sintáctico. Sus nodos son los constituyentes sintácticos y las hojas, las palabras que componen la oración analizada. La estructura jerárquica del árbol permite observar que un constituyente está formado por una o varias palabras y por otros constituyentes.

Un constituyente sintáctico es una palabra o una secuencia de palabras que realizan una función conjunta dentro de la estructura jerárquica de la oración.

En la gramática tradicional se llama **sintagma** a l **constituyente sintáctico compuesto de dos a más elementos** y según la *Nueva gramática de la lengua española* (Real Academia Española, 2009) se denomina **grupo sintáctico.**

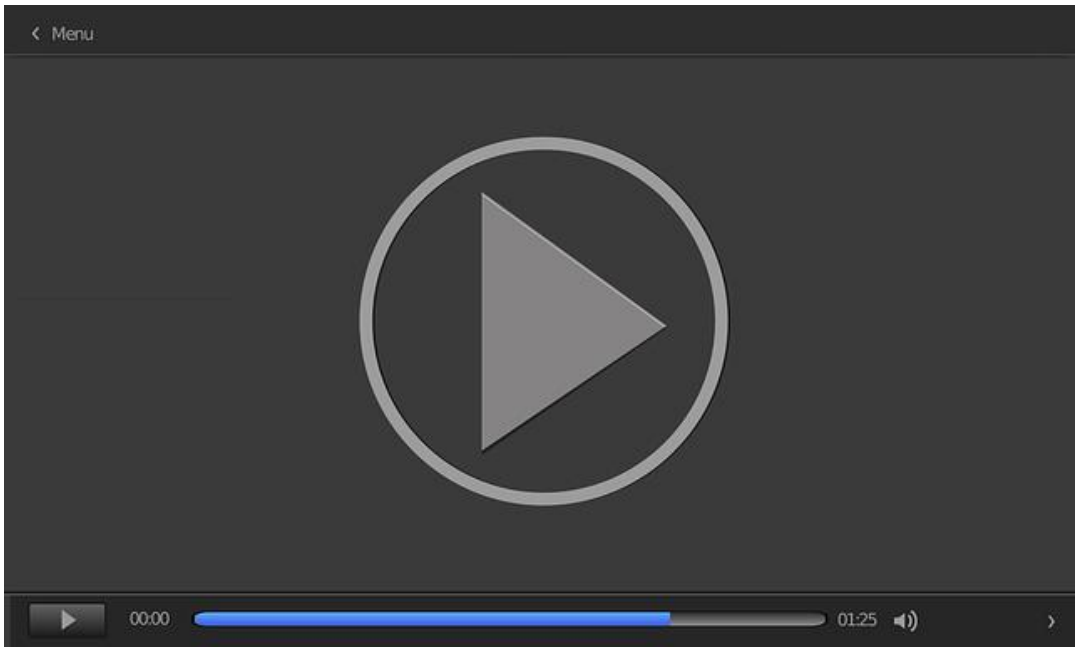
Los principales tipos de sintagma de la lengua española se presentan en la siguiente tabla:

| Sintagma | Núcleo | Función |
|---|---|---|
| **Sintagma Nominal (SN)** | Sustantivo<br>Pronombre | Sujeto.<br>Atributo.<br>Complemento directo.<br>Complemento indirecto (solo si el núcleo es un pronombre).<br>Complemento predicativo.<br>Complemento circunstancial. |
| **Sintagma Verbal (SV)** | Verbo | Predicado (predicado verbal o predicado nominal). |
| **Sintagma Preposicional (SP o SPrep)** | Preposición | Complemento de régimen.<br>Complemento directo.<br>Complemento preposicional.<br>Complemento circunstancial. |
| **Sintagma Adjetival (SAdj)** | Adjetivo | Complemento adyacente.<br>Complemento predicativo. |
| **Sintagma Adverbial (SAdv)** | Adverbio | Adjunto.<br>Complemento circunstancial. |

Tabla 1. Principales sintagmas de la lengua española. Fuente: elaboración propia.

# Ideas clave

En el vídeo *Sintaxis y constituyentes sintácticos* se presentará la definición de sintaxis (parte de la gramática que estudia cómo se combinan las palabras) y los diferentes constituyentes sintácticos que existen.




04.01. Sintaxis y constituyentes sintácticos

Accede al vídeo:

https://unir.cloud.panopto.eu/Panopto/Pages/Embed.aspx?id=53f64d15-0c96-4582-b013-ac6b00c1e214

**Información sintáctica**

Es información relativa a las relaciones estructurales entre palabras. Dicha información de una palabra permite determinar las combinaciones posibles de este elemento con el resto de los elementos de la oración. Por ejemplo, la información sintáctica de un verbo puede indicar si este puede tener asociado un objeto directo, un objeto indirecto o un complemento preposicional. En concreto, los verbos transitivos exigen la presencia de un objeto directo y este conocimiento lingüístico sobre la estructura sintáctica de los verbos transitivos debe modelarse correctamente para poder realizar el análisis sintáctico.

Las relaciones estructurales entre palabras se pueden formular mediante reglas

Para indicar que en español un elemento que sea un sintagma nominal, por ejemplo, está compuesto por dos elementos, un determinante y un nombre (colocados en este orden específico) se puede utilizar la siguiente regla:

$$SN \rightarrow Det\ N$$

Donde:

- $SN$ indica el sintagma nominal.
- $Det$: el determinante.
- $N$ : el nombre.

A partir de esta información sintáctica se podrá determinar que el elemento «la mesa» es un sintagma nominal correcto porque se compone del determinante «la» seguido del nombre «mesa».

# Ideas clave

Una definición formal de la estructura sintáctica de una lengua es imprescindible para poder realizar correctamente el análisis sintáctico. Las **gramáticas de estructura sintagmática** que se presentan a continuación modelan la información sintáctica y, por tanto, recogen la información relativa a las relaciones estructurales de una lengua necesaria en el análisis sintáctico realizado en las tareas de procesamiento del lenguaje natural.

Cabe destacar que, en la lingüística moderna, se están extendiendo las llamadas **gramáticas valenciales o gramáticas de dependencias.**

Estas gramáticas, a diferencia de las de estructura sintagmática, se basan en dependencias y no en los constituyentes sintácticos.

Por lo que en las gramáticas de dependencias se modelan los elementos léxicos y las relaciones existentes entre estos elementos, tal como se muestra en el último apartado.

## 4.3. Gramáticas de estructura sintagmática

Una gramática de estructura sintagmática **permite definir la estructura formal del lenguaje.** En esta sección se define el concepto y los elementos de una gramática de estructura sintagmática y, además, se identifican los diferentes tipos: gramáticas de estados finitos, gramáticas libres de contexto, gramáticas sensibles al contexto, gramáticas enumerables recursivamente y gramáticas generativas transformacionales.

Las páginas señaladas corresponden al apartado «1.2. Las gramáticas formales» del capítulo indicado para el estudio de esta sección».

## 4.4. Ambigüedad en el análisis sintáctico

Uno de los mayores problemas que se dan en el análisis sintáctico es la ambigüedad.

La ambigüedad estructural se debe a las múltiples reglas representadas en una gramática, que provienen del uso común de la lengua y que permiten que se puedan encontrar varios resultados del análisis sintáctico que se ajusten correctamente a la analizada.

En una frase en la que aparezca la coordinación «hombres y mujeres mayores», será difícil determinar en el análisis sintáctico si el emisor de la frase se refiere a que tanto los hombres como las mujeres son mayores o, por el contrario, se refiere a todos los hombres, sean o no mayores, y a las mujeres mayores.

- En el primer caso, el adjetivo «mayores» está asociado al sintagma nominal compuesto de la coordinación de los dos nombres: «hombres» y «mujeres». Esta relación se podría escribir utilizando corchetes como [[hombres y mujeres] mayores].
- En el segundo caso, el adjetivo «mayores» está asociado solo al nombre «mujeres» y el sintagma nominal que forman estas dos palabras está relacionado a través de la conjunción copulativa «y» al nombre «hombres». Por tanto, esta relación se podría expresar utilizando la notación de corchetes como [hombres y [mujeres mayores]].

El siguiente ejemplo ilustrativo presenta un caso de ambigüedad estructural en la que existen dos posibles resultados del análisis sintáctico de una oración y que se representan gráficamente en forma de árboles sintácticos.

Al final, el analizador sintáctico que obtiene esos dos resultados debe optar por resolver la ambigüedad eligiendo uno de los árboles sintácticos como salida del proceso de análisis.

Ejemplo ilustrativo 1

Ambigüedad en análisis sintáctico de la frase de Groucho Marx en la película Animal Crackers (1930): *«I shot an elephant in my pajamas».*

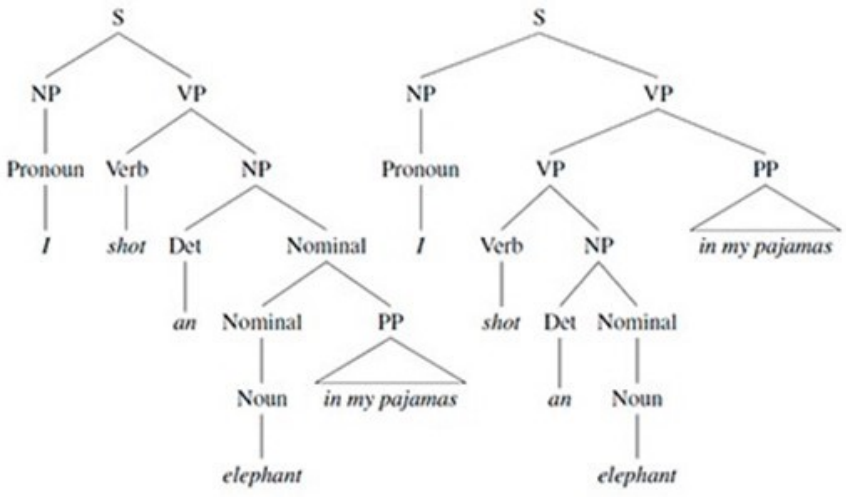



Figura 1. Árboles sintácticos que representan el resultado del análisis sintáctico. Fuente: Jurafsky y Martin, 2009.

El análisis sintáctico de la frase *«I shot an elephant in my pajamas»* puede derivar en dos posibles soluciones y, por tanto, se observa en este caso el problema de la ambigüedad estructural. Tal como se muestra en la Figura 1, el elemento *«in my pajamas»* puede ser parte del sintagma nominal (NP) cuyo núcleo es el nombre (*noun*) *«elephant»*, en el árbol sintáctico de la izquierda, o parte del predicado verbal (VP) cuyo núcleo es el verbo (*verb*) *«shot»,* en el árbol sintáctico de la derecha.

# Ideas clave

El problema de la ambigüedad en la estructura de las frases **afecta el funcionamiento de la mayoría de los sistemas de procesamiento del lenguaje natural.** Entonces, también afecta los analizadores sintácticos que utilizan gramáticas libres de contexto y que se basan en búsqueda, ya sea ascendente o descendente, presentados en el tema anterior.

Por lo tanto, esos analizadores deben ser capaces de elegir un único resultado correcto del análisis de entre la multitud de resultados posibles.

La selección del mejor resultado del análisis, es decir, del mejor árbol sintáctico, se realiza normalmente a través de un **proceso de desambiguación sintáctica** que requiere de fuentes de conocimiento estadístico, semántico y contextual.

Métodos más avanzados evitan el uso de conocimiento lingüístico para realizar la desambiguación *a posteriori* e integran el **proceso de desambiguación** directamente en el análisis sintáctico. Algunos de estos métodos se basan en programación dinámica y otros, en métodos probabilísticos que se presentan en este tema.

## 4.5. Métodos para el análisis sintáctico, basados en la programación dinámica

**Técnicas de programación dinámica** se utilizan en el análisis sintáctico para tratar el problema de la ambigüedad estructural. La programación dinámica **es un método** que sirve para **optimizar la resolución de problemas** que pueden ser discretizados y secuencializados. Así, la programación dinámica permite resolver problemas complejos de una forma más eficiente tratándolos como subproblemas más sencillos.

El principal fundamento de la programación dinámica es que las soluciones óptimas de subproblemas permiten encontrar la solución óptima del problema en su conjunto.

Basándose en esta idea, en la programación dinámica se completan de forma sistemática tablas de soluciones para los diferentes subproblemas y, cuando estas tablas están completas, contienen la solución a todos los subproblemas necesarios para resolver el problema global.

La programación dinámica se utiliza en el análisis sintáctico para tratar el problema de la ambigüedad. Para ello, los subproblemas representan los posibles árboles sintácticos para todos los constituyentes sintácticos detectados como entrada del análisis.

El algoritmo más utilizado para implementar el análisis sintáctico basado en programación dinámica es el **algoritmo de Cocke-Kasami-Younger (CKY)** (Kasami, 1965 y Younger, 1967). El algoritmo CKY se aplica cuando la gramática utilizada en el análisis sintáctico es del tipo *Chomsky Normal Form* (CNF) (Chomsky, 1963). En una gramática CNF las reglas solo pueden tener a la derecha:

- Dos símbolos no terminales (regla $A \rightarrow BC$).
- O un símbolo terminal (regla $A \rightarrow w$).

Sin embargo, esto no representa un problema ya que una gramática libre de contexto se puede transformar en una gramática CNF sin perder expresividad.

El primer paso del algoritmo CKY es **convertir la gramática libre de contexto en una gramática CNF**. Para ello, todas aquellas reglas que cumplen por defecto con las condiciones de una CNF (esto es, que a la derecha tengan dos símbolos no terminales o un símbolo terminal) son directamente copiadas a la nueva gramática. El resto de las reglas que no cumplen con las condiciones de una gramática CNF son transformadas como se describe a continuación.

Aquellas reglas que mezclan símbolos terminales y no terminales son alteradas con la introducción de un no símbolo terminal comodín que involucra únicamente el símbolo terminal original. Por ejemplo, una regla para un verbo en infinitivo en inglés como *I*
$NF - VP \rightarrow VP$ sería sustituida por las dos reglas
$INF - VP \rightarrow VP$ y
$\rightarrow$ . Ahora sí, estas nuevas reglas cumplen las condiciones de una gramática CNF.

Las reglas con un solo símbolo no terminal a la derecha, llamémosle **unitarias**, se pueden eliminar reescribiendo la parte derecha de las reglas originales con el lado derecho de todas aquellas reglas no unitarias. Más formalmente, si
$A \Rightarrow B$ es una cadena de una o más reglas unitarias y
$B \rightarrow \gamma$ es una regla no unitaria, entonces podemos añadir
$A \rightarrow \gamma$ para cada regla en la gramática y eliminar todas las reglas unitarias.

Se puede demostrar que esta operación puede conducir a un aplanamiento sustancial de la gramática y la consiguiente promoción de terminales a niveles bastante altos en los árboles resultantes.

Las reglas con más de dos términos en el lado derecho se normalizan a través de la introducción de nuevos símbolos no terminales que extienden las secuencias más largas en varias reglas nuevas. Formalmente:

$$A \rightarrow BC\gamma$$

Entonces, si tenemos una regla como la anterior, podemos reemplazar el par más a la izquierda de los símbolos no terminales con un nuevo símbolo no terminal e introducir las siguientes nuevas reglas:

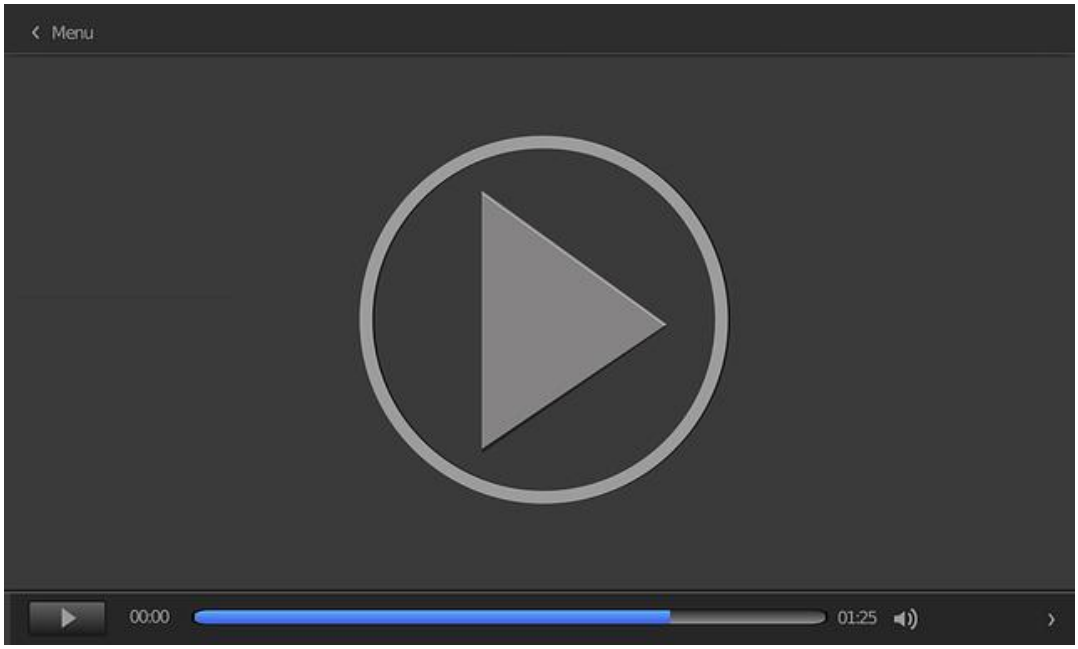
$$A \rightarrow X1\gamma$$

$$X1 \rightarrow BC$$

Este proceso se puede repetir tantas veces como sea necesario hasta alcanzar reglas de una longitud 2. La elección del par de símbolos no terminales a reemplazar es puramente arbitraria; cualquier procedimiento sistemático que resulte en reglas binarias es adecuado. Una vez se hayan convertido todas las reglas en binarias se añaden a la nueva gramática, finalizando aquí el proceso de conversión a CNF.

# Ideas clave

En el vídeo *Paso 1 del algoritmo CKY: conversión de la gramática libre de contexto en una gramática CNF* se verá el primer paso para aplicar el algoritmo CKY para el análisis sintáctico. Este consiste en convertir la gramática en un formato libre de contexto al formato CNF.




04.02. Paso 1 del algoritmo CKY: conversión de la gramática libre de contexto en una gramática CNF

Accede al vídeo:

https://unir.cloud.panopto.eu/Panopto/Pages/Embed.aspx?id=6a5ed687-9b9d-4e80-be95-ac6b00c1e293

Una vez tengamos la gramática en formato CNF, podemos aplicar el algoritmo CKY, el cual nos permite llevar a cabo el proceso de reconocimiento sintáctico. Con nuestra gramática en CNF, cada nodo no terminal por encima del nivel de categoría gramatical en un árbol sintáctico tendrá exactamente dos hijos. Se puede usar una matriz de dos dimensiones para codificar la estructura de todo un árbol.

# Ideas clave

Para una frase de longitud $n$, trabajaremos con la parte superior triangular de una matriz:

$$(n+1)\,x\,(n+1)$$

Cada celda $[i,j]$ de esta matriz contiene el conjunto de símbolos no terminales que representan todos los constituyentes sintácticos que abarcan posiciones de entrada desde $i$ hasta $j$. Ya que nuestro sistema de indexación comienza en 0, se deduce que la celda que representa la entrada completa reside en la posición $[0,n]$ de la matriz.

Dado que cada entrada no terminal en nuestra tabla tiene dos hijos en el análisis sintáctico, podemos decir que:

- Para cada constituyente sintáctico representado por una entrada $[i,j]$, tiene que haber una posición en la entrada $k$, que puede ser dividida en dos partes de tal manera que:

$$i < k < j$$

- Dada una posición $k$:

  - El primer constituyente sintáctico $[i,k]$ debe estar a la izquierda de la entrada $[i,j]$ en algún lugar a lo largo de la fila $i$.

  - Y la segunda entrada $[k,j]$ debe encontrarse por debajo de aquel, a lo largo de la columna $j$.

La superdiagonal en la matriz contiene las categorías gramaticales para cada una de las palabras de la oración. Las subsiguientes diagonales por encima de la superdiagonal contienen los constituyentes sintácticos para los diferentes tramos de longitud creciente en la oración.

Vista esta configuración, el reconocimiento CKY consiste en rellenar la tabla de análisis sintáctico de la manera correcta.

Para ello, se procederá de abajo hacia arriba de manera que, a la hora de rellenar la celda
$[i,j]$, las celdas que pueden contener partes que podrían contribuir a esta entrada de la tabla (las celdas a la izquierda y debajo) deben estar ya rellenas.

El pseudocódigo del algoritmo se muestra en la Figura 2. Dicho algoritmo rellena la matriz triangular superior operando por columnas de abajo a arriba y de izquierda a derecha como se muestra en la Figura 3. Este esquema garantiza que en cada momento tengamos toda la información necesaria.

```
function CKY-PARSE(words, grammar) returns table

for j ← from 1 to LENGTH(words) do
    for all {A | A → words[j] ∈ grammar}
            table[j − 1, j] ← table[j − 1, j] ∪ A
    for i ← from j − 2 downto 0 do
        for k ← i + 1 to j − 1 do
            for all {A | A → BC ∈ grammar and B ∈ table[i,k] and C ∈ table[k, j]}
                    table[i,j] ← table[i,j] ∪ A
```

Figura 2. Pseudocódigo del algoritmo CKY. Fuente: Jurafsky y Martin, 2009.

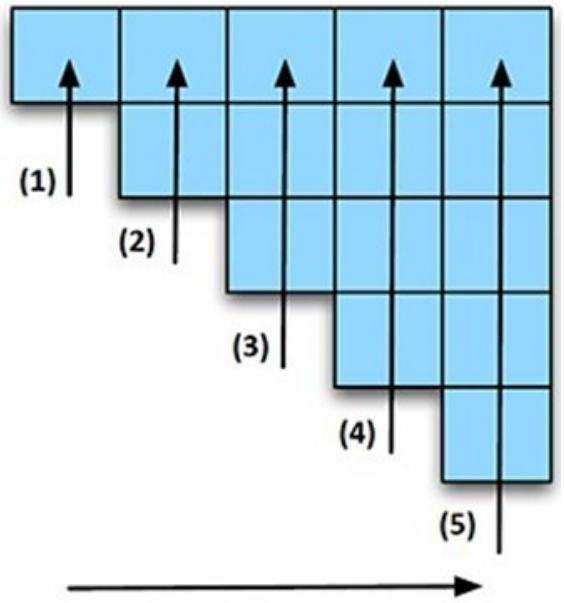



Figura 3. Secuencia seguida para rellenar la tabla en el proceso de reconocimiento CKY. Fuente: Jurafsky y Martin, 2009.

La Figura 4 muestra el caso general de rellenado de la celda $[i,j]$. En cada partición, el algoritmo considera si los contenidos de las dos celdas pueden ser combinados de modo que se cumpla alguna de las reglas de la gramática.

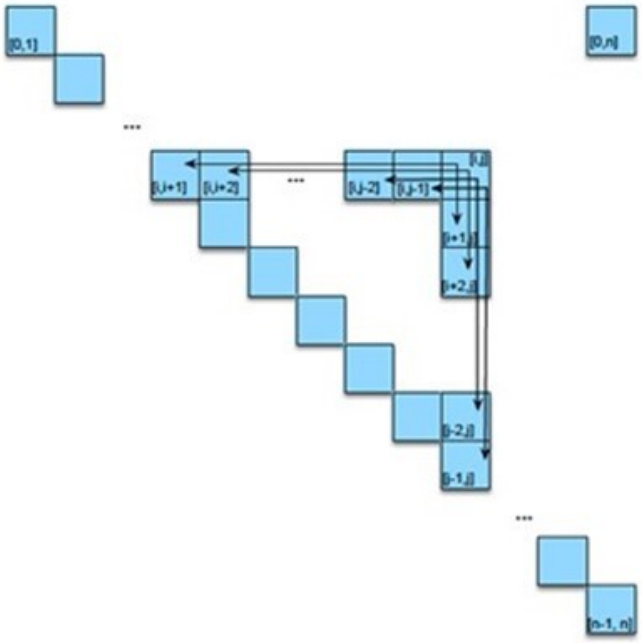



Figura 4. Todas las posibles formas de rellenar la celda $[i,j]$ en la tabla CKY. Fuente: Jurafsky y Martin, 2009.

Ejemplo ilustrativo 2

Algoritmo CKY para el reconocimiento sintáctico de la frase en inglés *«Book the flight through Houston»* (en español, «Reserva el vuelo a través de Houston»).

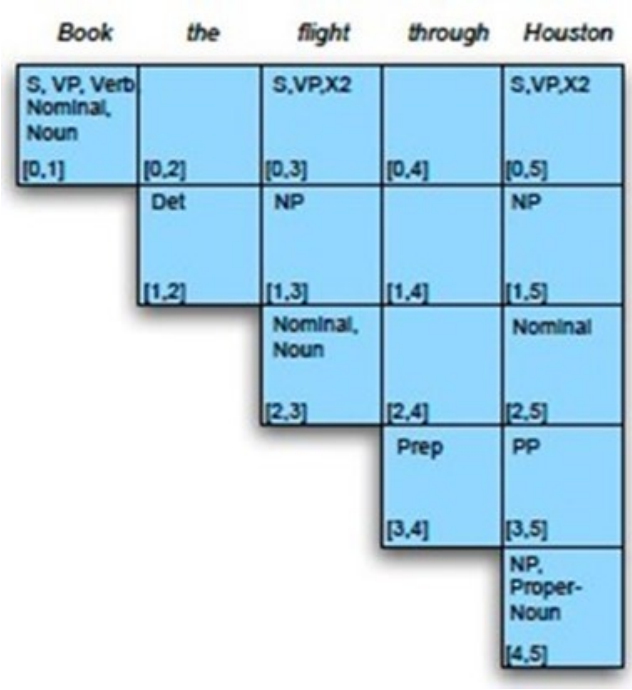



Figura 5. Matriz del análisis sintáctico al aplicar el algoritmo CKY. Fuente: Jurafsky y Martin, 2009.

La Figura 5 muestra la matriz del análisis sintáctico completo para la frase *«Book the flight through Houston»* cuando se utiliza la gramática en formato CNF presentada en la Figura 6 y 7 para realizar el reconocimiento sintáctico.

| $\mathscr{L}_1$ Grammar | $\mathscr{L}_1$ in CNF |
|---|---|
| $S \rightarrow NP\ VP$ | $S \rightarrow NP\ VP$ |
| $S \rightarrow Aux\ NP\ VP$ | $S \rightarrow X1\ VP$ |
| | $X1 \rightarrow Aux\ NP$ |
| $S \rightarrow VP$ | $S \rightarrow book \mid include \mid prefer$ |
| | $S \rightarrow Verb\ NP$ |
| | $S \rightarrow X2\ PP$ |
| | $S \rightarrow Verb\ PP$ |
| | $S \rightarrow VP\ PP$ |
| $NP \rightarrow Pronoun$ | $NP \rightarrow I \mid she \mid me$ |
| $NP \rightarrow Proper\text{-}Noun$ | $NP \rightarrow TWA \mid Houston$ |
| $NP \rightarrow Det\ Nominal$ | $NP \rightarrow Det\ Nominal$ |
| $Nominal \rightarrow Noun$ | $Nominal \rightarrow book \mid flight \mid meal \mid money$ |
| $Nominal \rightarrow Nominal\ Noun$ | $Nominal \rightarrow Nominal\ Noun$ |
| $Nominal \rightarrow Nominal\ PP$ | $Nominal \rightarrow Nominal\ PP$ |
| $VP \rightarrow Verb$ | $VP \rightarrow book \mid include \mid prefer$ |
| $VP \rightarrow Verb\ NP$ | $VP \rightarrow Verb\ NP$ |
| $VP \rightarrow Verb\ NP\ PP$ | $VP \rightarrow X2\ PP$ |
| | $X2 \rightarrow Verb\ NP$ |
| $VP \rightarrow Verb\ PP$ | $VP \rightarrow Verb\ PP$ |
| $VP \rightarrow VP\ PP$ | $VP \rightarrow VP\ PP$ |
| $PP \rightarrow Preposition\ NP$ | $PP \rightarrow Preposition\ NP$ |

Figura 6. Gramática libre de contexto (columna izquierda) y en formato CNF (columna derecha) y lexicón (abajo) para realizar el análisis sintáctico en lengua inglesa. Fuente: Jurafsky y Martin, 2009.

| Lexicon |
|---|
| $Det \rightarrow that \mid this \mid the \mid a$ |
| $Noun \rightarrow book \mid flight \mid meal \mid money$ |
| $Verb \rightarrow book \mid include \mid prefer$ |
| $Pronoun \rightarrow I \mid she \mid me$ |
| $Proper\text{-}Noun \rightarrow Houston \mid NWA$ |
| $Aux \rightarrow does$ |
| $Preposition \rightarrow from \mid to \mid on \mid near \mid through$ |

Figura 7. Gramática libre de contexto (columna izquierda) y en formato CNF (columna derecha) y lexicón (abajo) para realizar el análisis sintáctico en lengua inglesa. Fuente: Jurafsky y Martin, 2009.

El ejemplo concreto de cómo se rellenarían las celdas de la columna 5 de la matriz después de leer la palabra «Houston» se muestra en la Figura 8. Las flechas señalan los elementos de las dos celdas que se utilizan para agregar una entrada a la tabla.

Cabe destacar que en la celda [0; 5] se indica la presencia de tres posibles análisis alternativos para esta entrada:

- Uno, donde la preposición (PP) modifica la palabra *«flight»*.
- Otro, donde esta misma preposición modifica la palabra *«book»*.
- Y el tercero, que captura la regla *VP → X2 PP*.

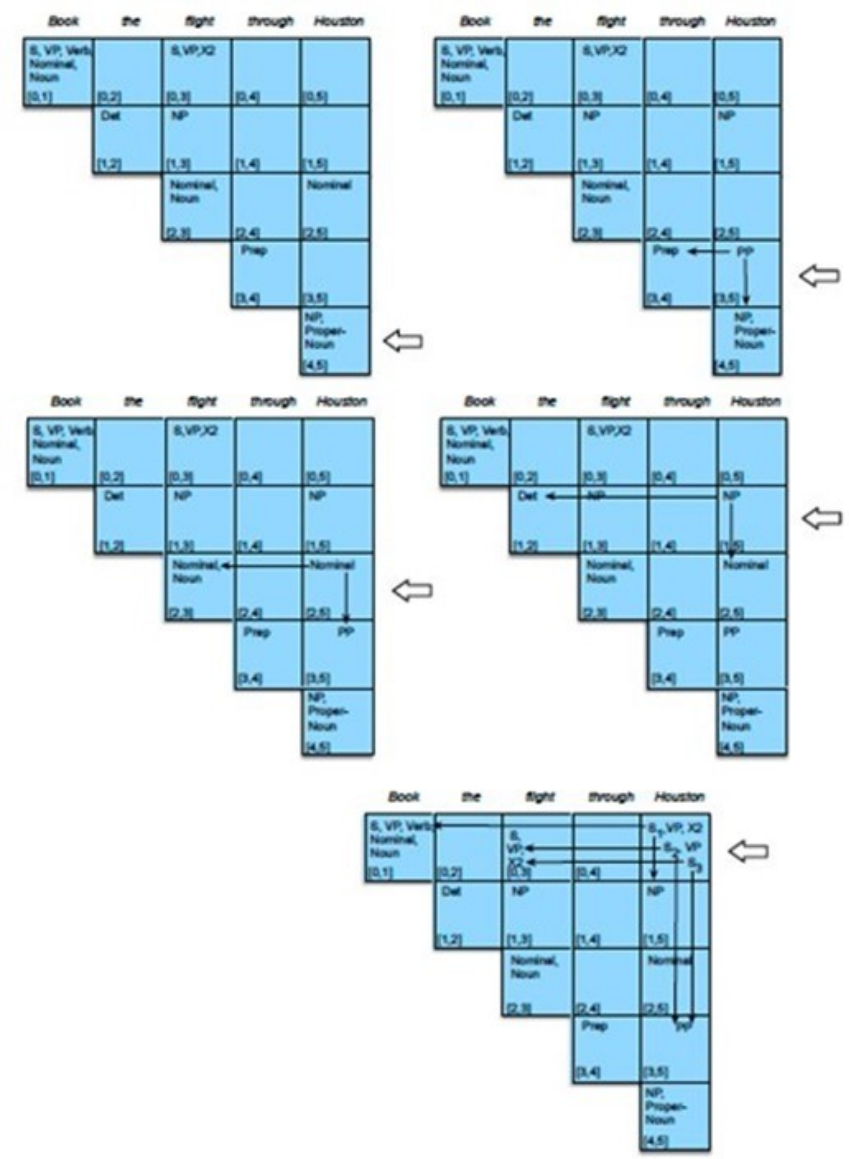
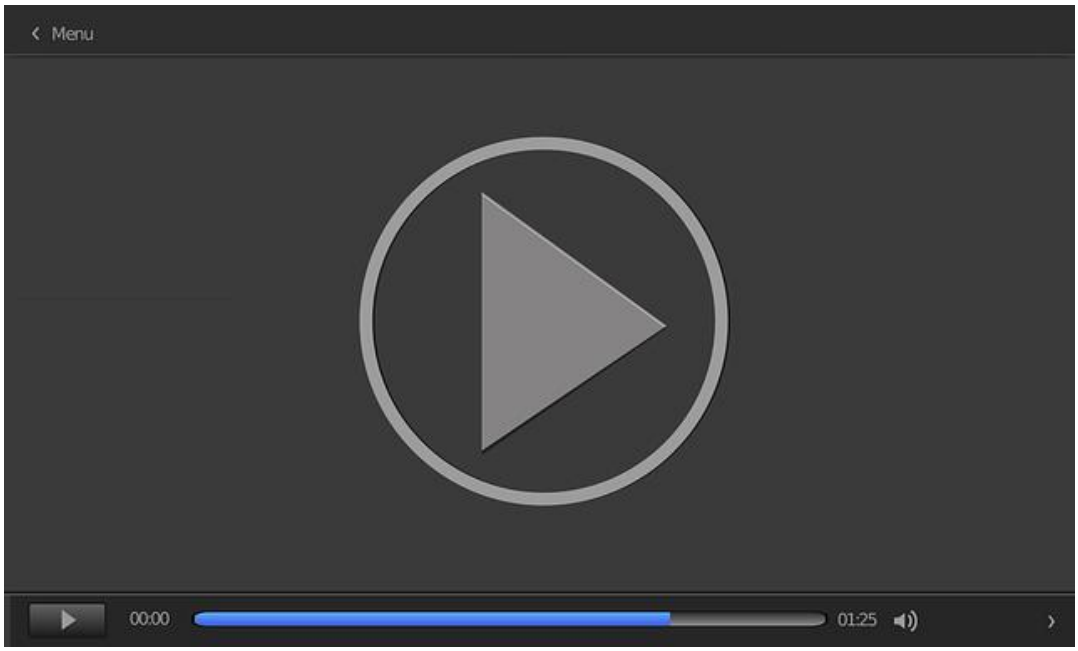



Figura 8. Rellenado de las celdas de la columna 5 después de leer la palabra «Houston».

Fuente: Jurafsky y Martin, 2009.

# Ideas clave

En el vídeo *Paso 2 del algoritmo CKY: reconocimiento sintáctico* se verá el segundo paso para aplicar el algoritmo CKY para el análisis sintáctico. Este consiste en crear la matriz que codifique los diferentes árboles sintácticos.




04.03. Paso 2 del algoritmo CKY: reconocimiento sintáctico

Accede al vídeo:

https://unir.cloud.panopto.eu/Panopto/Pages/Embed.aspx?id=84a46435-30e1-4463-918d-ac6b00c1e2ca

Finalmente, llegamos a la **fase de análisis sintáctico**. Hay que tener en cuenta que el algoritmo mostrado en la Figura 3 es un reconocedor sintáctico, no un analizador sintáctico. Por tanto, para convertirlo en un analizador capaz de devolver todos los análisis sintácticos posibles para una entrada dada, podemos hacer dos simples cambios en el algoritmo.

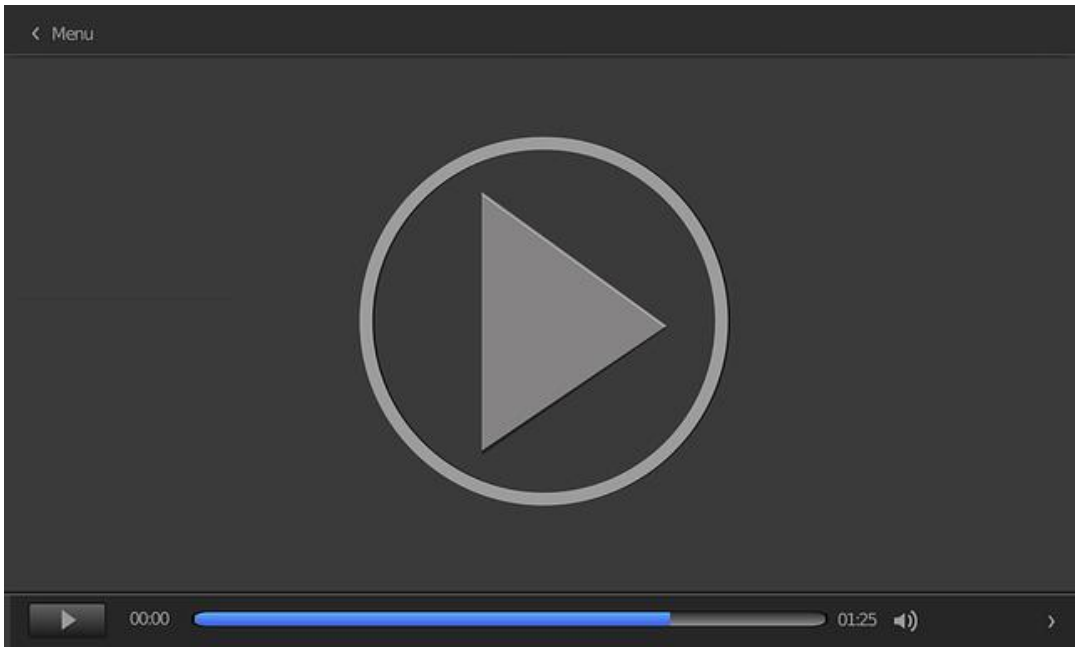
- El primer cambio es aumentar las entradas de la tabla de manera que cada símbolo no terminal esté emparejado con punteros a las entradas de la tabla de las que se derivaron (parecido a como se muestra en la Figura 8 del ejemplo ilustrativo 2).

- El segundo cambio consiste en permitir múltiples versiones del mismo símbolo no terminal para ser introducidos en la tabla (véase la Figura 8).

Con estos cambios, el cuadro completo contiene todos los posibles análisis sintácticos para una entrada dada. Devolver un único análisis sintáctico arbitrario consiste en la elección de una oración
$S$ de la celda
$[0,n]$ y luego recuperar de forma recursiva sus constituyentes sintácticos de la tabla.

# Ideas clave

En el vídeo *Paso 3 del algoritmo CKY: análisis sintáctico* se explicará el tercer paso del algoritmo CKY, donde se realiza la decodificación o la obtención de los distintos árboles sintácticos a partir de la matriz que se ha creado anteriormente.




04.04. Paso 3 del algoritmo CKY: análisis sintáctico

Accede al vídeo:

https://unir.cloud.panopto.eu/Panopto/Pages/Embed.aspx?id=3bd3b864-19b4-4849-8fdd-ac6b009a5b0a

## 4.6. Métodos probabilistas en el análisis sintáctico

La ambigüedad estructural se puede modelar de forma eficiente utilizando el algoritmo CKY presentado en el apartado anterior. Sin embargo, el analizador sintáctico va a devolver múltiples resultados del análisis y para poder resolver la ambigüedad y llegar a una única solución es necesario **implementar métodos probabilistas** en él. De hecho, la mayoría de los analizadores sintácticos utilizados hoy en día implementan métodos probabilistas como método de **desambiguación.**

Los analizadores sintácticos probabilistas calculan la probabilidad de cada posible interpretación del análisis sintáctico y escogen la más probable.

Formalmente, un analizador sintáctico probabilista tiene como objetivo producir el análisis sintáctico más probable
$\widehat{T}$ para una oración dada
$S$:

$$\widehat{T}(S) = \underset{Ts.t.S=yield(T)}{argmax} P(T)$$

Por lo tanto, el analizador sintáctico probabilista va a computar el árbol sintáctico
$T$ que maximice
$P(T)$, esto es, la probabilidad de dicho árbol. Dicho de otra forma, el analizado sintáctico va a buscar el árbol sintáctico más probable. Notar que la cadena de palabras
$S$ se denomina el *«yield»* de cualquier árbol sintáctico que se puede aplicar a la oración a analizar.

# Ideas clave

Para calcular la **probabilidad de un árbol sintáctico** $P(T)$ de una oración $S$ a partir de las probabilidades de las reglas de una gramática, se considera que las reglas son independientes y se calcula la probabilidad del árbol multiplicando las probabilidades de cada una de las reglas involucradas en el análisis de la oración.

Para calcular la **probabilidad asociada a una regla** representada en una gramática libre de contexto existen dos maneras. La forma más sencilla es utilizar un *treebank* (Marcus, Santorini y Marcinkiewicz, 1993), esto es un corpus de frases ya analizados. Dado un *treebank,* podemos calcular la probabilidad de cada expansión de un símbolo no terminal mediante el recuento del número de veces que se produce la expansión y luego normalizarlo, tal como se muestra en la siguiente fórmula:

$$P(\alpha \rightarrow \beta | \alpha) = \frac{Count(\alpha \rightarrow \beta)}{\sum_{\gamma} Count(\alpha \rightarrow \gamma)} = \frac{Count(\alpha \rightarrow \beta)}{Count(\alpha)}$$

Si no tenemos un *treebank,* pero sí tenemos un analizador sintáctico (no probabilístico), podemos generar los recuentos que necesitamos para el cálculo de las probabilidades asociadas a cada regla. Para ello, analizamos sintácticamente el corpus de frases con el analizador. Si las frases no son ambiguas, sería tan simple como analizar todo el corpus, incrementar un contador para cada regla dada en el análisis sintáctico y, luego, normalizar para obtener las probabilidades.

Como la mayor parte de las oraciones son ambiguas, es decir, tienen múltiples análisis sintácticos, tenemos que mantener una cuenta separada para cada análisis sintáctico de una oración y ponderar cada uno de estos recuentos parciales por la probabilidad del análisis sintáctico en el que aparece. Pero para obtener las probabilidades para ponderar las reglas, debemos tener ya un analizador sintáctico probabilístico, lo cual nos lleva a un problema recurrente.

La forma de resolver este problema es mejorar de forma creciente nuestras estimaciones:

- Comenzar con un analizador sintáctico con probabilidades iguales para cada regla.
- Seguidamente, analizar la frase.
- Calcular la probabilidad para cada análisis sintáctico.
- Utilizar estas probabilidades para ponderar los contadores.
- Reestimar las probabilidades de cada regla, y así sucesivamente, hasta que nuestras probabilidades converjan.

El algoritmo estándar para el cálculo de esta solución se llama *inside-outside* (Baker, 1979) y es un caso especial del algoritmo *Expectation-Maximization* (Manning y Schütze, 1999).

## Algoritmo CKY probabilístico

La mayoría de los analizadores sintácticos probabilistas modernos se basan en el **algoritmo CKY probabilístico** (Ney, 1991), una versión probabilística del algoritmo CKY presentado en la sección anterior.

El algoritmo CKY probabilístico extiende el algoritmo CKY para incluir información probabilística.

La versión probabilística del algoritmo CKY, igual que el algoritmo CKY básico, utiliza una gramática CNF (*Chomsky Normal Form*). Sin embargo, en la versión probabilística, cada regla está anotada con la probabilidad de que se cumpla dicha regla.

Por lo tanto, el **primer paso del algoritmo CKY probabilístico** consiste en convertir las reglas de una gramática libre de contexto anotada con probabilidades al **formato**

**CNF**. Para realizar este proceso se aplica un método análogo al utilizado para el algoritmo CKY básico. En la conversión de las reglas al formato CNF, además de aplicar los criterios explicados en la sección anterior, se deben recalcular las probabilidades de modo que la probabilidad asociada a cada posible árbol resultante del análisis sintáctico de la oración permanezca constante para la nueva gramática CNF.

En el **segundo paso del algoritmo CKY probabilístico,** y una vez se tienen las reglas en formato CNF, se crearía una **matriz** similar a la del CKY básico, pero **con una dimensión adicional.** Recordemos que en el CKY básico cada celda de la matriz contenía una lista con los constituyentes sintácticos para cada palabra. En el caso CKY probabilístico, cada celda tiene una dimensión adicional que se utiliza para indexar cada constituyente sintáctico. Específicamente, para una frase de longitud $n$ (palabras) y una gramática que contiene $V$ símbolos no terminales (constituyentes sintácticos), tendríamos una matriz $(n+1)\,x\,(n+1)\,xV$. El valor de cada una de las celdas corresponde en este caso a la probabilidad de cada constituyente o símbolo no terminal para cada palabra.

El pseudocódigo del algoritmo CKY probabilístico se presenta en la Figura 9.

```
function PROBABILISTIC-CKY(words,grammar) returns most probable parse
                                                and its probability
for j ← from 1 to LENGTH(words) do
    for all { A | A → words[j] ∈ grammar}
      table[j − 1, j, A] ← P(A → words[j])
    for i ← from j − 2 downto 0 do
        for k ← i + 1 to j − 1 do
              for all { A | A → BC ∈ grammar,
                              and table[i, k, B] > 0 and table[k, j, C] > 0 }
                    if (table[i,j,A] < P(A → BC) × table[i,k,B] × table[k,j,C]) then
                        table[i,j,A] ← P(A → BC) × table[i,k,B] × table[k,j,C]
                        back[i,j,A] ← {k, B, C}
return BUILD_TREE(back[1, LENGTH(words), S]), table[1, LENGTH(words), S]
```

Figura 9. Pseudocódigo del algoritmo CKY probabilístico. Fuente: Jurafsky y Martin, 2009.

Ejemplo ilustrativo 3

Algoritmo CKY probabilístico para el reconocimiento sintáctico de la frase en inglés «*The flight includes a meal*», en español significa «El vuelo incluye comida».

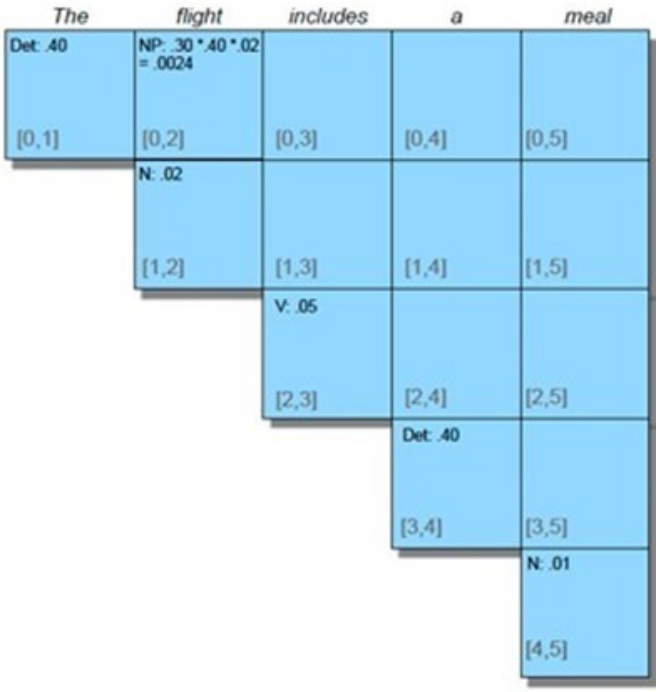


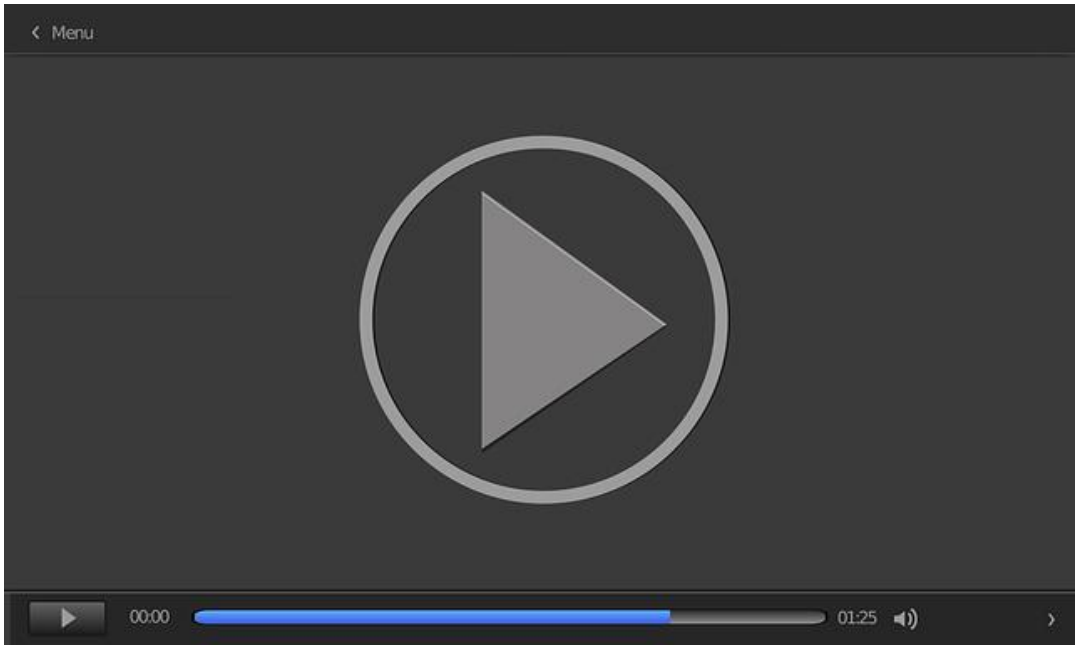

Figura 10. Matriz para el análisis sintáctico aplicando el algoritmo CKY probabilístico. Fuente: Jurafsky y Martin, 2009.

Los primeros pasos de aplicar el algoritmo probabilístico CKY para analizar la frase «*The flight includes a meal*», se muestran en la matriz de la Figura 10. La gramática en formato CNF que incluye las probabilidades de cada una de las reglas y que se utiliza para realizar el reconocimiento sintáctico se presenta en la Figura 11.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $S$ | $\rightarrow$ | $NP\ VP$ | .80 | $Det$ | $\rightarrow$ | *the* | .40 |
| $NP$ | $\rightarrow$ | $Det\ N$ | .30 | $Det$ | $\rightarrow$ | *a* | .40 |
| $VP$ | $\rightarrow$ | $V\ NP$ | .20 | $N$ | $\rightarrow$ | *meal* | .01 |
| $V$ | $\rightarrow$ | *includes* | .05 | $N$ | $\rightarrow$ | *flight* | .02 |

Figura 11. Gramática de la lengua inglesa en formato CNF donde cada regla está anotada con la probabilidad y que se utiliza para realizar el análisis sintáctico. Fuente: Jurafsky y Martin, 2009.

En el vídeo *Algoritmo CKY probabilístico* se explicará este, que amplía el CKY clásico y permite añadir probabilidades para obtener la probabilidad de cada uno de los árboles sintácticos codificados en la matriz.




04.05. Algoritmo CKY probabilístico

Accede al vídeo:

https://unir.cloud.panopto.eu/Panopto/Pages/Embed.aspx?id=86b6a09f-ef2f-4a2b-95ad-ac6b009a5b45

## 4.7. Gramáticas de dependencias o gramáticas valenciales

Una **gramática valencial modela las dependencias entre los elementos léxicos de una oración** y, de este hecho, proviene su nombre de gramática de dependencias.

A diferencia de las gramáticas de estructura sintagmática, que modelan los constituyentes sintácticos de una oración, las gramáticas de dependencias describen la estructura sintáctica de una oración únicamente en términos de las palabras (o lemas) que la componen y del conjunto de las relaciones gramaticales establecidas entre las palabras.

Ejemplo ilustrativo 1

Análisis sintáctico utilizando una gramática de dependencias.

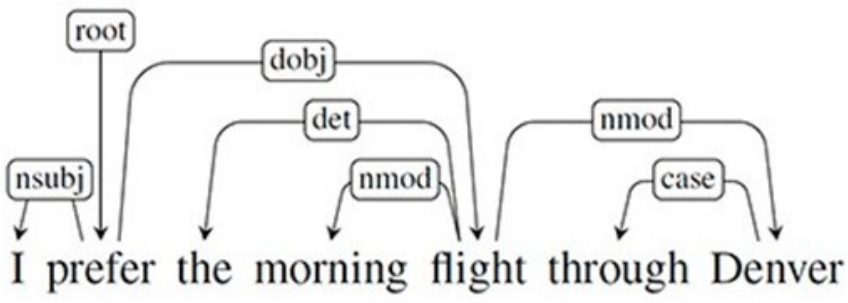



Figura 12. Resultado del análisis sintáctico utilizando una gramática de dependencias.
Fuente: Jurafsky y Martin, 2009.

El análisis sintáctico de la oración en inglés «*I prefer the morning flight through Denver*» (en español, «Prefiero un vuelo a través de Denver por la mañana»).

Utilizando una gramática de dependencias, produciría la salida presentada en la Figura 1. De la que se desprende que la palabra *«prefer»* (prefiero) es la raíz de la oración y que esta se relaciona a través de una dependencia del tipo *dobj* (objeto directo) con la palabra *«flight»* (vuelo). A su vez, la palabra *«flight»* se relaciona a través de una dependencia del tipo *det* (determinante) con la palabra *«flight».* Cada relación se da entre un origen *(head)* y su elemento dependiente *(dependent).*

El resultado del análisis sintáctico utilizando una gramática de dependencias (Figura 12) se puede comparar con el árbol sintáctico (Figura 13) resultante de realizar el análisis sintáctico de la misma oración utilizando una gramática de estructura sintagmática como podría ser una gramática libre de contexto.

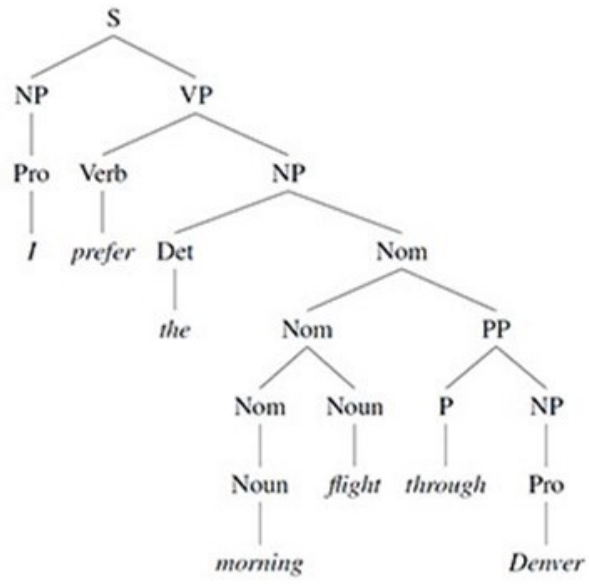



Figura 13. Árbol sintáctico que representan el resultado del análisis sintáctico utilizando una gramática de estructura sintagmática. Fuente: Jurafsky y Martin, 2009.

El análisis sintáctico utilizando gramáticas de dependencias se popularizó en la década de los 90. Por lo tanto, en las últimas **décadas han aparecido diferentes**

**analizadores sintácticos de dependencias** basados en principios diversos: en programación dinámica (Eisner, 1996), en enfoques transicionales deterministas (Covington, 2001) (Nivre y Scholz, 2004), en aprendizaje automático supervisado (Yamada y Matsumoto, 2003) o en representaciones gráficas (McDonald et al., 2005).

Entre las ventajas que tienen estas gramáticas aparece su habilidad para trabajar con idiomas que son morfológicamente ricos y dónde se tiene **libertad para ordenar las palabras de distinta manera** dentro de una misma oración. En estos casos, se pueden tener, por ejemplo, objetos gramaticales que pueden aparecer bien antes o bien después de un adverbio de lugar.

Una gramática de estructura sintagmática necesitaría reglas que modelasen todas las posibles combinaciones. En cambio, las gramáticas de dependencias son agnósticas a esto.

Otra ventaja de las gramáticas de dependencia es que las relaciones *head-dependent* tienen una similitud respecto a las relaciones semánticas entre predicados y sus argumentos. En estas gramáticas esta información se ve de manera directa, a diferencia de las gramáticas de estructura sintagmática, donde esta información se tiene que extraer mediante técnicas adicionales sobre los árboles generados.

Así, las gramáticas de dependencias permiten detectar las relaciones entre *-head-dependent-* identificándolas con una relación gramatical concreta según la función gramatical del *dependent* con respecto a su *head.*

**Algunas de estas categorías son:**

| Clausal Argument Relations | Description |
|---|---|
| NSUBJ | Nominal subject |
| DOBJ | Direct object |
| IOBJ | Indirect object |
| CCOMP | Clausal complement |
| XCOMP | Open clausal complement |
| **Nominal Modifier Relations** | **Description** |
| NMOD | Nominal modifier |
| AMOD | Adjectival modifier |
| NUMMOD | Numeric modifier |
| APPOS | Appositional modifier |
| DET | Determiner |
| CASE | Prepositions, postpositions and other case markers |
| **Other Notable Relations** | **Description** |
| CONJ | Conjunct |
| CC | Coordinating conjunction |

Figura 14. Lista de algunas de las posibles relaciones gramaticales que pueden aparecer con las gramáticas de dependencias. Fuente: Jurafsky y Martin, 2009.

Las gramáticas de dependencias se caracterizan por estar formadas según una estructura de grafos G = (V, A), donde se tiene una relación entre los vértices V (las palabras) mediante una serie de arcos (A). Estas relaciones cumplen que:

- Siempre hay **un único nodo raíz** que no tiene arcos que llevan a él.
- Excepto para el nodo raíz, los demás nodos tienen un **único arco que lleva a ellos.**
- Hay **un único camino** entre el nodo raíz y los demás vértices V.

Con esto se garantiza que cada palabra tenga un único *head,* que la estructura de dependencia esté conectada, y que haya un único nodo raíz desde el que se pueda partir para seguir un camino único hacia cualquiera de las palabras de la frase. Junto con esto, aparece otra propiedad en los arcos *head-dependent* denominada **proyectividad.** Esta se da si existe un camino entre el head y todas las palabras que aparecen entre él y su *dependent.* Por ejemplo, en la frase de la Figura 12 todos los arcos son proyectivos. Sin embargo, en la frase siguiente el arco entre *flight* y *was* no es proyectivo ya que no existe un camino entre *flight* y *this* o *morning* considerando sólo las palabras que están entre *head-dependent (flight-was).*

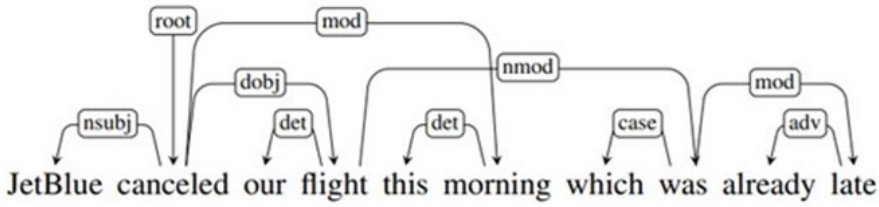



Figura 15. Ejemplo de salida de una gramática de dependencia donde no todos los arcos son proyectivos.

Fuente: Jurafsky y Martin, 2009.

Con ello, cuando todos los arcos de la gramática de dependencia son proyectivos se dice por extensión que el árbol de dependencias es proyectivo.

## Algoritmos para gramáticas de dependencias

De cara a construir un algoritmo que lleve a cabo un análisis con gramáticas de dependencias, es importante partir de *treebanks* donde ya exista información anotada respecto a frases y sus análisis correspondientes. Estos *treebanks* están disponibles para distintas lenguas, como por ejemplo el de (Weischedel et al., 2011).

Partiendo de esto, una de las aproximaciones más conocidas para las gramáticas de dependencias es el *transitioned-based dependency parsing*. Con este algoritmo,

ilustrado en la siguiente figura, se tienen distintos elementos: una pila donde están los elementos sobre los que se hace el análisis, un *buffer* con los tokens que todos los tokens sobre los que no se ha hecho aún el análisis, y un oráculo que dará como salida la acción concreta que hacer dentro de la iteración del algoritmo. Las acciones que puede llevar a cabo el oráculo se dividen en tres:

- **LeftArc:** Definir una relación *head-dependent* entre la palabra en la posición primera de la pila con respecto a la segunda palabra. Eliminar la segunda palabra de la pila.
- **RightArc:** Definir una relación *head-dependent* entre la segunda palabra de la pila y la primera. Eliminar la primera palabra de la pila.
- **Shift:** Pasar la primera palabra del *buffer* a la pila.

Con ello, la Figura 16 muestra el esquema del algoritmo, y la Figura 17 muestra un ejemplo de las salidas que se van obteniendo para cada iteración.

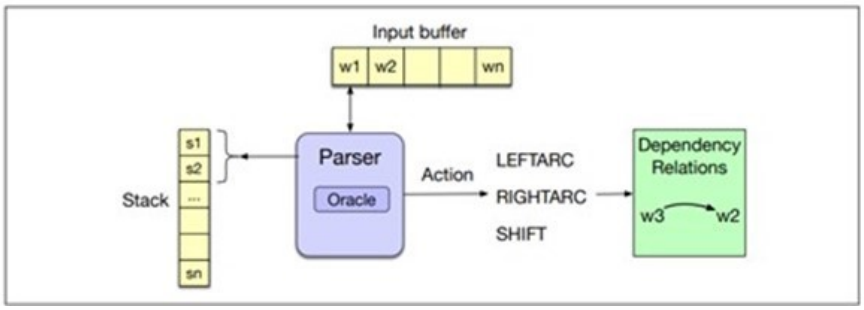



Figura 16. Esquema del algoritmo de *transitioned-based dependency parsing.* Fuente: Jurafsky y Martin, 2009.

| Step | Stack | Word List | Action | Relation Added |
|---|---|---|---|---|
| 0 | [root] | [book, me, the, morning, flight] | SHIFT | |
| 1 | [root, book] | [me, the, morning, flight] | SHIFT | |
| 2 | [root, book, me] | [the, morning, flight] | RIGHTARC | (book → me) |
| 3 | [root, book] | [the, morning, flight] | SHIFT | |
| 4 | [root, book, the] | [morning, flight] | SHIFT | |
| 5 | [root, book, the, morning] | [flight] | SHIFT | |
| 6 | [root, book, the, morning, flight] | [] | LEFTARC | (morning ← flight) |
| 7 | [root, book, the, flight] | [] | LEFTARC | (the ← flight) |
| 8 | [root, book, flight] | [] | RIGHTARC | (book → flight) |
| 9 | [root, book] | [] | RIGHTARC | (root → book) |
| 10 | [root] | [] | Done | |

Figura 17. Ejemplo de salida para cada iteración del algoritmo transitioned-based dependency parsing.

Fuente: Jurafsky y Martin, 2009.

El punto fundamental es la creación del oráculo; es decir, cómo tener un sistema que determine en cada paso cuál de las tres acciones se deben ejecutar. Este se puede generar mediante el uso de algoritmos de aprendizaje supervisado entrenados sobre la información que proviene de los *treebanks*.

## 4.8. Referencias bibliográficas

Baker, J. K. (1979). Trainable grammars for speech recognition. En D. H. Klatt y J. J. Wolf (Eds.), *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America* (pp. 547-550). Acoustical Society of America.

Chomsky, N. (1963). Formal properties of grammars. En R. D. Luce, R. Bush y E. Galanter (Eds.), *Handbook of Mathematical Psychology* (Vol. 2, pp. 323-418). Cambridge, Estados Unidos: Wiley.

Jurafsky, D. y Martin, J. H. (2021). *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition and Computational Linguistics.* Prentice-Hall.

Kasami, T. (1965). *An efficient recognition and syntax analysis algorithm for context-free languages.* (Informe No. R-257). Universidad de Illinois.

Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. The MIT Press.

Marcus, M. P., Santorini, B. y Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics, 19*(2), 313-330.

Ney, H. (1991). Dynamic programming parsing for contextfree grammars in continuous speech recognition. *IEEE Transactions on Signal Processing, 39*(2), 336-340.

RAE. (s. f.). Sintaxis. En *Diccionario de la lengua española* (actualización de la 23ª ed.). http://dle.rae.es/?id=XzfiT9q

Real Academia Española. (2009). *Nueva gramática de la lengua española.* Espasa.

Younger, D. H. (1967). Recognition and parsing of context-free languages in time n3. *Information and Control,* 10, 189-208.

# Modelos probabilísticos para el análisis sintáctico de dependencias

Eisner, J. (1996). Three new probabilistic models for dependency parsing: An exploration. Proceedings of the 16th International Conference on Computational Linguistics (COLING-96), 340-345. https://cs.jhu.edu/~jason/papers/eisner.coling96.pdf

El artículo presenta tres métodos probabilísticos para el análisis sintáctico basado en una gramática de dependencias. Los métodos propuestos se evalúan utilizando el conocido corpus Wall Street Journal (WSJ).

# Análisis sintáctico de dependencias

Jurafsky, D. y Martin, J. H. (2021). *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition and Computational Linguistics.* Prentice-Hall. https://web.stanford.edu/~jurafsky/slp3/14.pdf

El capítulo 14 de esta edición describe en detalle los aspectos relacionados con el análisis sintáctico de dependencias, comentando distintas maneras de construir el oráculo, incluso mediante el uso de modelos de aprendizaje profundo.

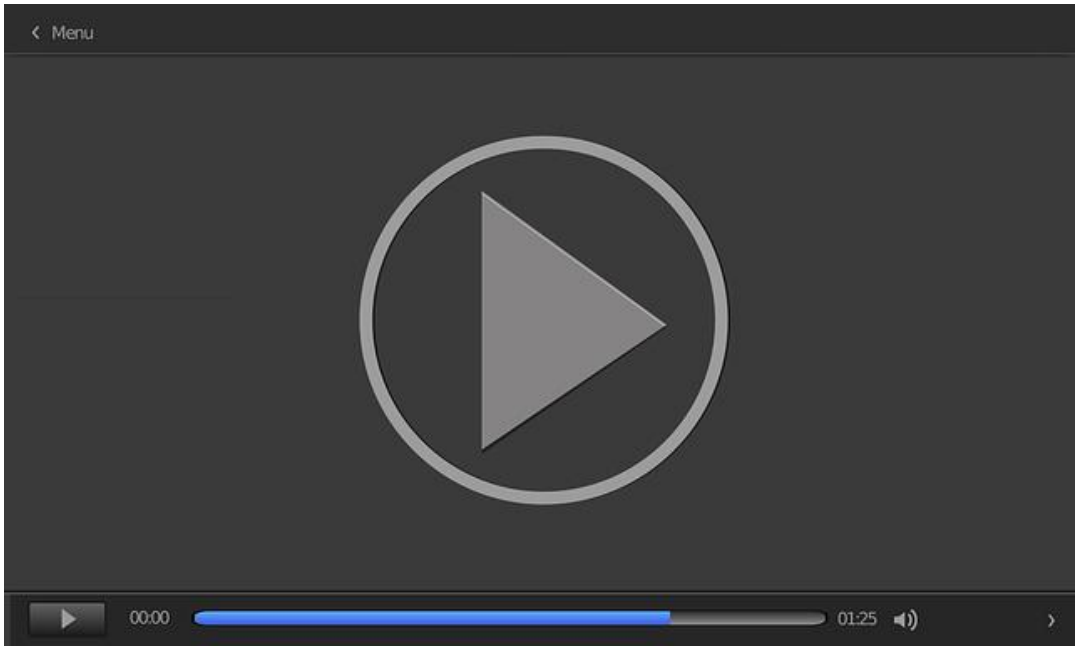
# Técnicas de procesamiento de lenguaje

Badia, T. (2003). Técnicas de procesamiento de lenguaje. En M. A. Martí (Coord.). *Tecnologías del lenguaje* (pp. 199-206). Editorial UOC. Disponible en la Biblioteca Virtual de UNIR.

Para completar el estudio de esta sección puedes leer las páginas 199-206 del siguiente libro: Técnicas de procesamiento de lenguaje.

## Análisis sintáctico: estrategia ascendente y búsqueda en profundidad

En este vídeo se verá desde donde parte el análisis sintáctico utilizando una estrategia ascendente y la búsqueda en profundidad. Se parte de las palabras que forman la oración y se va creando el árbol sintáctico hacia arriba.



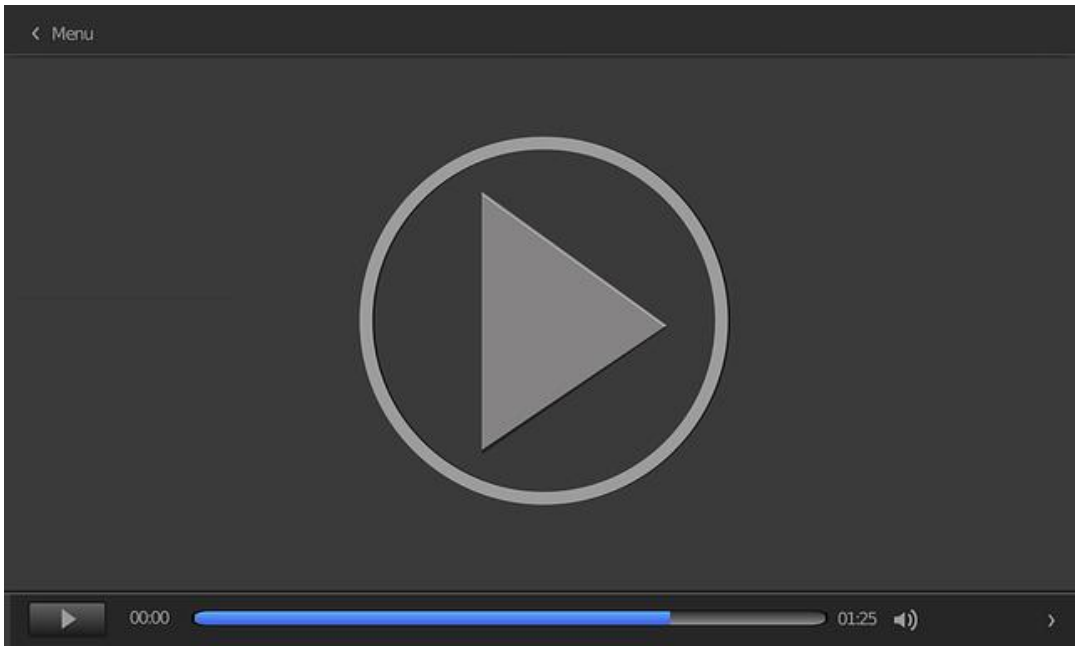

04.06. Análisis sintáctico: estrategia ascendente y búsqueda en profundidad

Accede al vídeo:

https://unir.cloud.panopto.eu/Panopto/Pages/Embed.aspx?id=2930ee9c-ab8f-40a7-b0b7-ac6b00a61c4e

## Análisis sintáctico: estrategia descendente y búsqueda en profundidad

En este vídeo se estudiará el análisis sintáctico usando una estrategia descendente y una búsqueda en profundidad (hacia abajo).



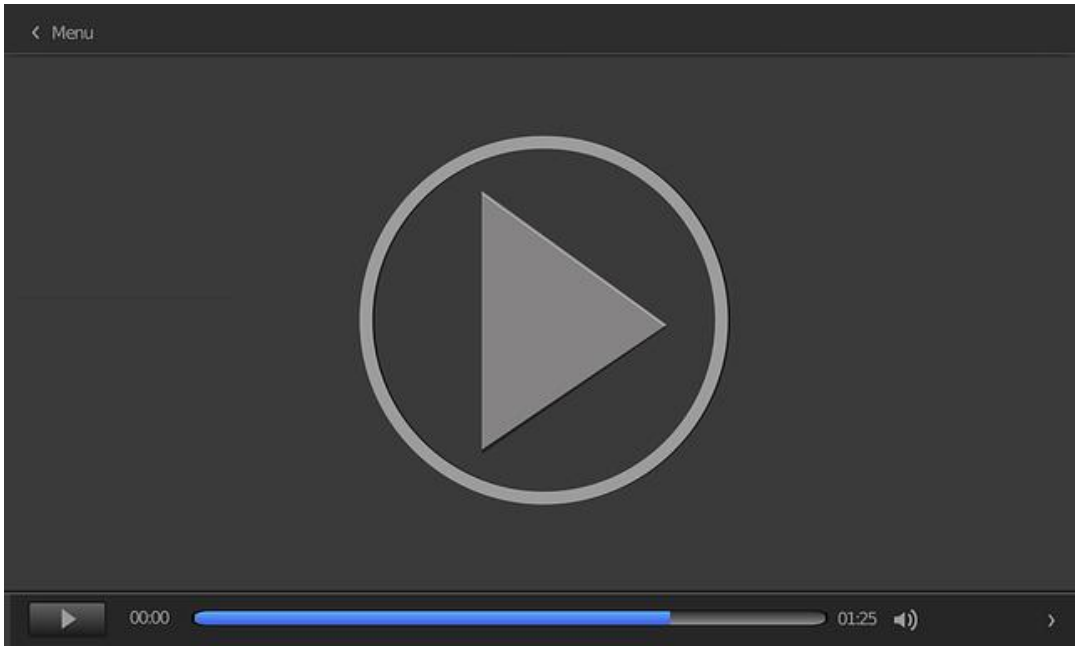

04.07. Análisis sintáctico: estrategia descendente y búsqueda en profundidad

Accede al vídeo:

https://unir.cloud.panopto.eu/Panopto/Pages/Embed.aspx?id=c8dbd3ca-f56d-4c04-a174-ac6b00a585d2

## Análisis sintáctico: estrategia ascendente y búsqueda en anchura

En este vídeo se verá cómo se crea el árbol de abajo hacia arriba a partir de las palabras hasta que llegamos a la composición completa. Al realizar la búsqueda en anchura se irá desarrollando el árbol por niveles.




04.08. Análisis sintáctico: estrategia ascendente y búsqueda en anchura

Accede al vídeo:

https://unir.cloud.panopto.eu/Panopto/Pages/Embed.aspx?id=566eff22-9c29-4a30-8819-ac6b00a7978c

**1.** Indica las afirmaciones correctas sobre el problema de la ambigüedad en el análisis sintáctico:

A. Solo afecta a un grupo reducido de analizadores sintácticos.

B. Se debe a un mal diseño de la gramática en la que aparecen múltiples reglas para una misma frase analizada.

C. Se da cuando el analizador sintáctico encuentra varios árboles sintácticos válidos y no es capaz de decidir cuál es el mejor.

D. Se puede resolver *a posteriori* una vez efectuado el análisis sintáctico.

**2.** Indica las afirmaciones correctas sobre los métodos para el análisis sintáctico-basados en programación dinámica:

A. Tratan el problema de la ambigüedad estructural.

B. Buscan soluciones óptimas a subproblemas que permiten encontrar la solución al problema en su conjunto.

C. Devuelven un único resultado para el análisis sintáctico.

D. Los posibles árboles sintácticos debidos a la ambigüedad estructural se representan como subproblemas.

**3.** Indica las afirmaciones correctas sobre el algoritmo de Cocke-Kasami-Younger (CKY):

A. Está basado en programación dinámica.

B. Puede incluir probabilidades para cada árbol sintáctico

C. Utiliza una gramática libre de contexto.

D. Utiliza una gramática del tipo Chomsky Normal Form (CNF).

**4.** Indica las afirmaciones correctas sobre una gramática CNF:

A. Las reglas solo pueden tener a la derecha dos símbolos no terminales o un símbolo terminal.

B. Cualquier gramática libre de contexto se podrá transformar en una gramática CNF.

C. Una regla en una gramática libre de contexto que mezcle símbolos terminales y no terminales se podrá transformar al formato CNF como una única regla con un símbolo comodín.

D. Las reglas unitarias en una gramática libre de contexto se podrán transformar al formato CNF reescribiendo la parte derecha de estas reglas originales con el lado derecho de todas aquellas reglas no unitarias.

**5.** Indica las afirmaciones correctas sobre el funcionamiento del algoritmo CKY:

A. Para rellenar la celda *[i,j ]* de la matriz, el algoritmo considera si los contenidos de una celda en la fila *i* y una celda en la columna *j* pueden ser combinados de modo que se cumpla alguna de las reglas de la gramática.

B. El algoritmo rellena la matriz triangular superior operando por columnas de abajo a arriba y de izquierda a derecha.

C. Una celda de la matriz puede contener varios posibles análisis sintácticos alternativos para la oración.

D. Para devolver un único análisis sintáctico se debe elegir de entre los símbolos presentes en la celda *[0,n]* de la matriz uno que represente la oración, por ejemplo *S* o *O* dependiendo del vocabulario utilizado en la gramática, y recuperar de forma recursiva sus constituyentes sintácticos.

**6.** Indica las afirmaciones correctas sobre la matriz de análisis sintáctico en el algoritmo CKY:

A. Para una frase de longitud *n,* se trabaja con una matriz de dimensión *n x n.*

B. Cada celda *[i, j]* de esta matriz contiene el conjunto de símbolos no terminales que representan todos los constituyentes sintácticos que abarcan posiciones de entrada desde *i* hasta *j.*

C. La superdiagonal en la matriz contiene las categorías gramaticales para cada una de las palabras de la oración.

D. Las subsiguientes diagonales por encima de la superdiagonal contienen los constituyentes sintácticos para los diferentes tramos de longitud decreciente en la oración.

**7.** Indica las afirmaciones correctas sobre la fase de reconocimiento sintáctico del algoritmo CKY:

A. El algoritmo aplicado en esta fase permite obtener un único árbol sintáctico.

B. Cada nodo del árbol sintáctico tiene exactamente dos hijos.

C. La estructura de un árbol sintáctico se puede codificar como una matriz de dos dimensiones.

D. El algoritmo aplicado en esta fase se puede modificar para convertirlo en un analizador sintáctico.

**8.** Indica las afirmaciones correctas sobre los analizadores sintácticos probabilistas:

A. Buscan el árbol sintáctico más probable para una oración maximizando la probabilidad de dicho árbol.

B. Tienen como objetivo producir el análisis sintáctico más probable para una oración.

C. Calculan la probabilidad de cada interpretación del análisis sintáctico para una oración y escogen la más probable.

D. Calculan la probabilidad de un árbol sintáctico de una oración a partir de las probabilidades de las reglas involucradas en el análisis de la oración.

**9.** Indica las afirmaciones correctas sobre el cálculo de la probabilidad asociada a una regla representada en una gramática libre de contexto:

A. Si no se dispone de un corpus de frases analizadas sintácticamente, no se puede calcular dicha probabilidad.

B. Si se dispone de un *treebank,* se recuenta el número de veces que se expande un símbolo no terminal y se normaliza.

C. Si no se dispone de un *treebank,* pero sí se dispone de un analizador sintáctico no probabilístico, se analiza sintácticamente el corpus de frases con el analizador y si estas no son ambiguas se recuenta el número de veces que se aplica cada regla y se normaliza.

D. Si no se dispone de un *treebank,* se estiman las probabilidades con un algoritmo basado en Expectation Maximization, donde se parte de unas probabilidades que se van ajustando iterativamente hasta que se dé el criterio de convergencia.

**10.** Indica las afirmaciones correctas sobre el algoritmo CKY probabilístico:

A. Extiende el algoritmo CKY añadiendo información sobre la probabilidad de que se cumpla cada regla.

B. En la conversión de las reglas de una gramática libre de contexto al formato CNF, las probabilidades de las reglas permanecen constantes.

C. Crea una matriz cuadrada con tantas celdas como la longitud de la frase y una dimensión adicional del tamaño del número de símbolos no terminales.

D. El valor de cada una de las celdas de la matriz se corresponde con la probabilidad de cada constituyente o símbolo no terminal para cada palabra.