UNIVERSIDAD ADOLFO IBÁÑEZ

Procesamiento de Lenguaje Natural

DIPLOMA/MAGISTER EN INTELIGENCIA ARTIFICIAL
UNIVERSIDAD ADOLFO IBÁÑEZ

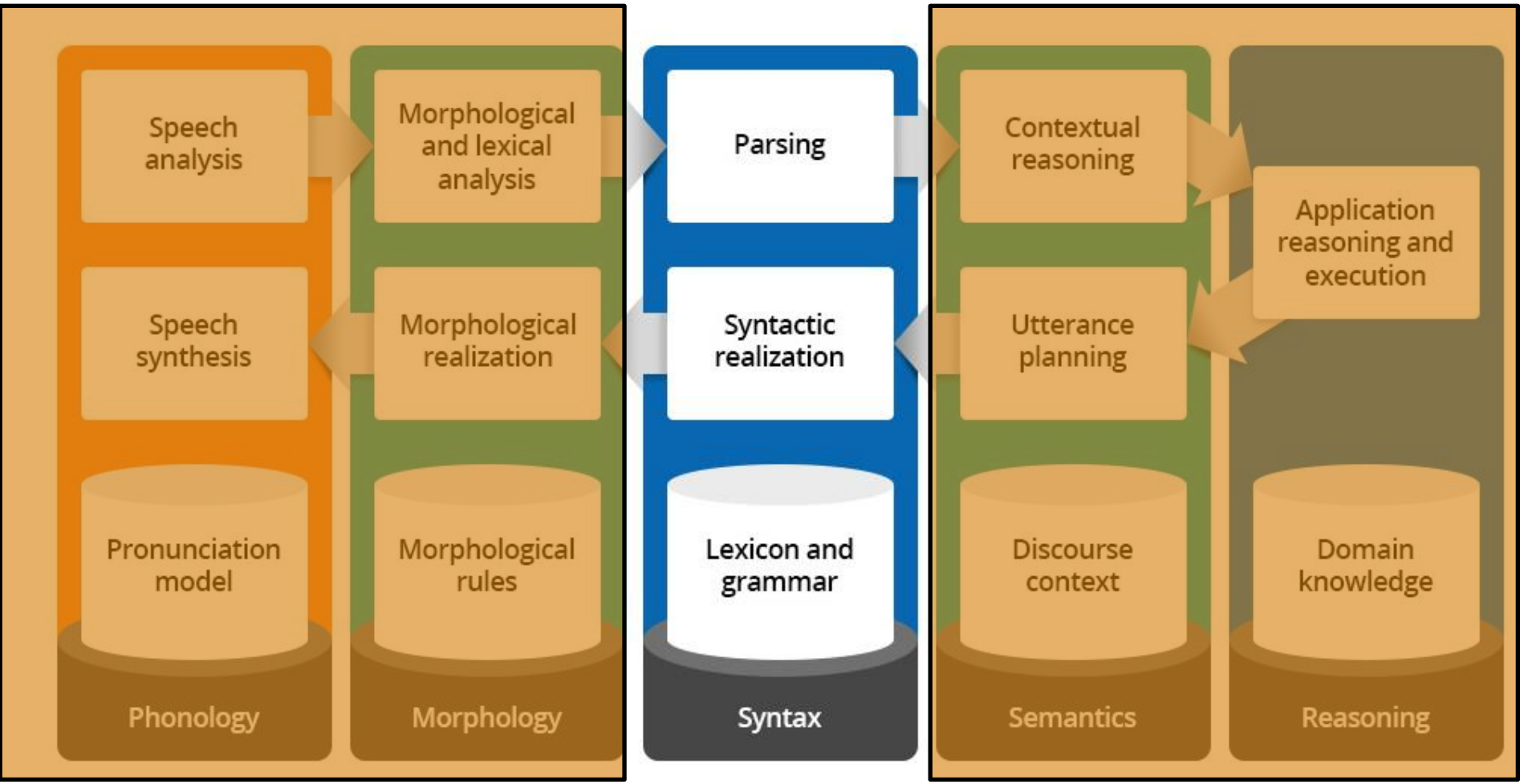
Profesor: Dr. John Atkinson

Análisis Sintáctico

OBJETIVO

Entender la manera en que se conectan las palabras entre sí para formar estructuras más complejas que permitan dar cuenta de su funciones/roles.

ETAPAS EN NLP



Análisis Sintáctico

- Tarea de generar automáticamente una estructura de relaciones (estructura sintáctica) que conectan las palabras de un texto en lenguaje natural.

- Usualmente esta tarea se conoce como ***Parsing***.

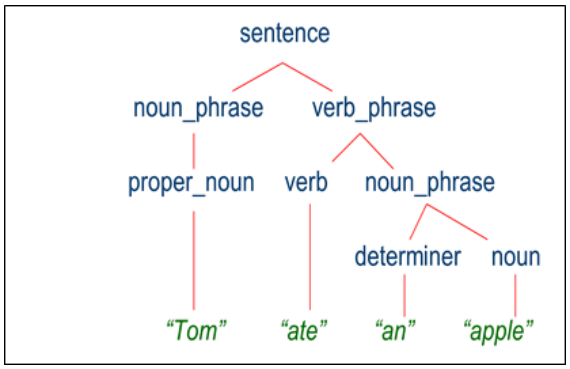
¿Qué hace un *Parser*?

Tom ate an Apple

Conocimiento Sintáctico

Reglas Gramaticales

Parser

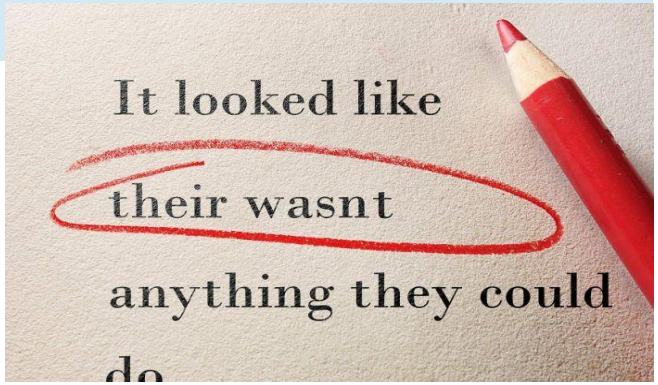


Estructura Sintáctica

Sintaxis

- La sintaxis se refiere a la forma en que las palabras se "*ordenan*" entre sí, y la "*relación*" que existe entre ellas.

- *Objetivo*: *modelar el conocimiento* de lo que la gente inconcientemente tiene acerca de la gramática en su lenguaje nativo.



¿Porqué es de Interés?

- *Paso previo para intentar comprender un texto en cualquier aplicación.*
- *Entender la estructura de una pregunta y una respuesta en sistemas de pregunta-respuesta (i.e., asistentes digitales, Alexa, atención a clientes)*
- *Extracción de Información específica desde documentos.*
- *Revisores gramaticales.*
- *Muchos más*

EJEMPLO

¿Cómo saber el tipo de respuesta para la siguiente pregunta?

¿Quién	es	Joe Biden?

EJEMPLO

¿Cómo saber la polaridad de la emoción de este texto de opinión en *twitter*?

```
El tiempo de recepción de la compra es muy bueno, aunque el servicio a clientes es de lo peor..
```

Ambigüedad Sintáctica

- Cada oración en un texto podría tener muchas *estructuras sintácticas* posibles: ambigüedad
- Generalmente, la estructura sintáctica que conecta las palabras en una oración tiene forma de árbol, de ahí que se denomina arbol sintáctico (***parse tree***).

¿Deberíamos encontrar un árbol o todos?
Si es uno, ¿Cómo sabemos cuál es válido?

Ambigüedad Sintáctica

Esta oración es ambigüa, ¿En qué forma?

I booked a flight from LA

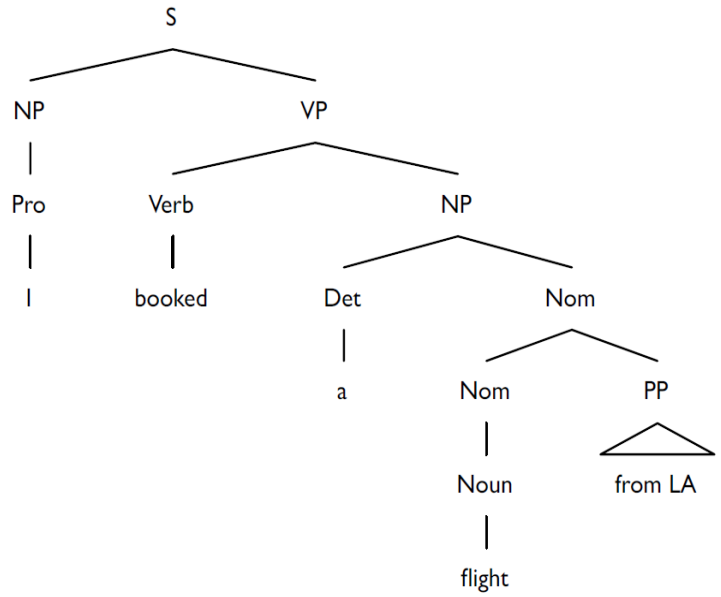
Ambigüedad Sintáctica

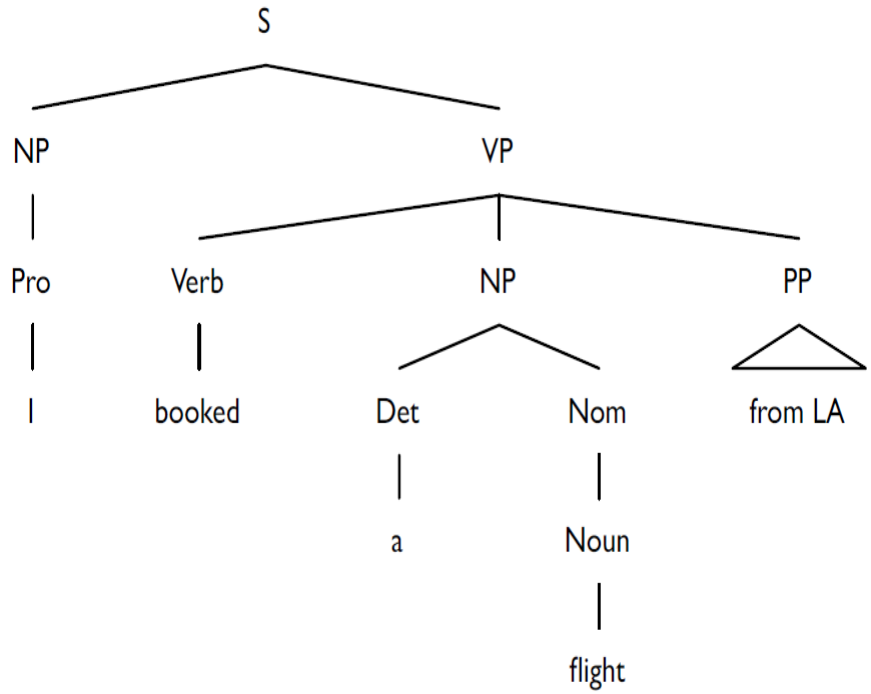
¿Qué debería pasar si analizamos la oración?

```
I booked (a flight from LA)
(I booked a flight) from LA
```

Se podrían generar 2 árboles de análisis!!

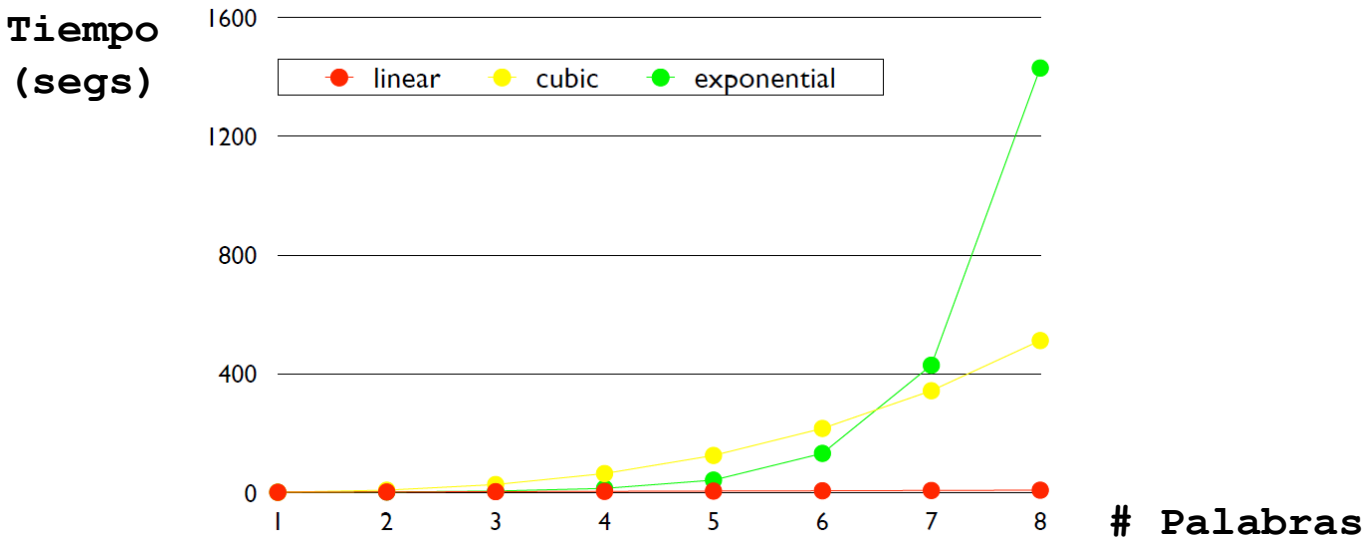
Árboles de Análisis Posibles





Búsqueda

- La tarea del parser se convierte en resolver un problema de búsqueda de estructuras válidas.
- Se requieren métodos eficientes, de lo contrario esto pasa a ser un problema combinatorial!!.



Gramática

Una *gramática* representa el conocimiento sintáctico formal que se tiene sobre un cierto lenguaje natural (o sea, las *reglas* que lo controlan).

Especificación de una Gramática

- Codificada por humanos en la forma de **reglas sintácticas**
- Aprendida *automáticamente* desde un corpus

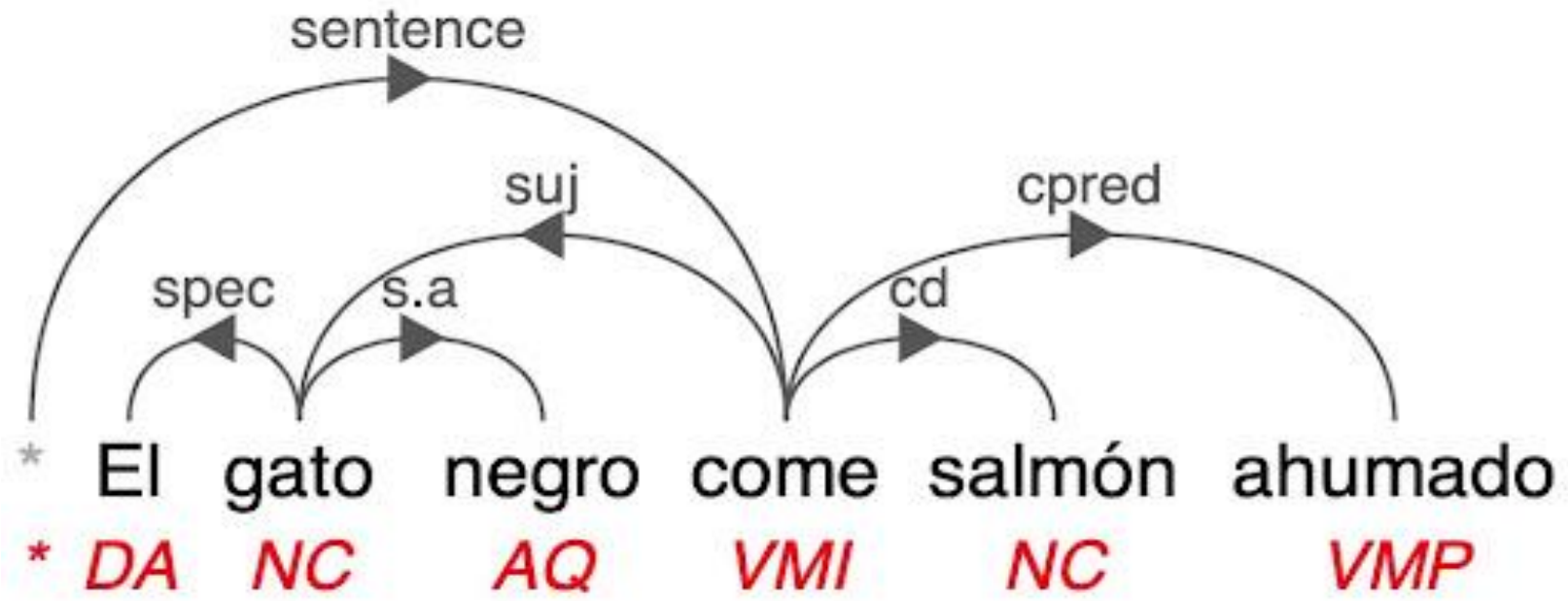
Gramática

- Existen varios formalismos gramaticales: gramática de dependencias, gramática categóricas, **gramática de estructura de frase**, etc.

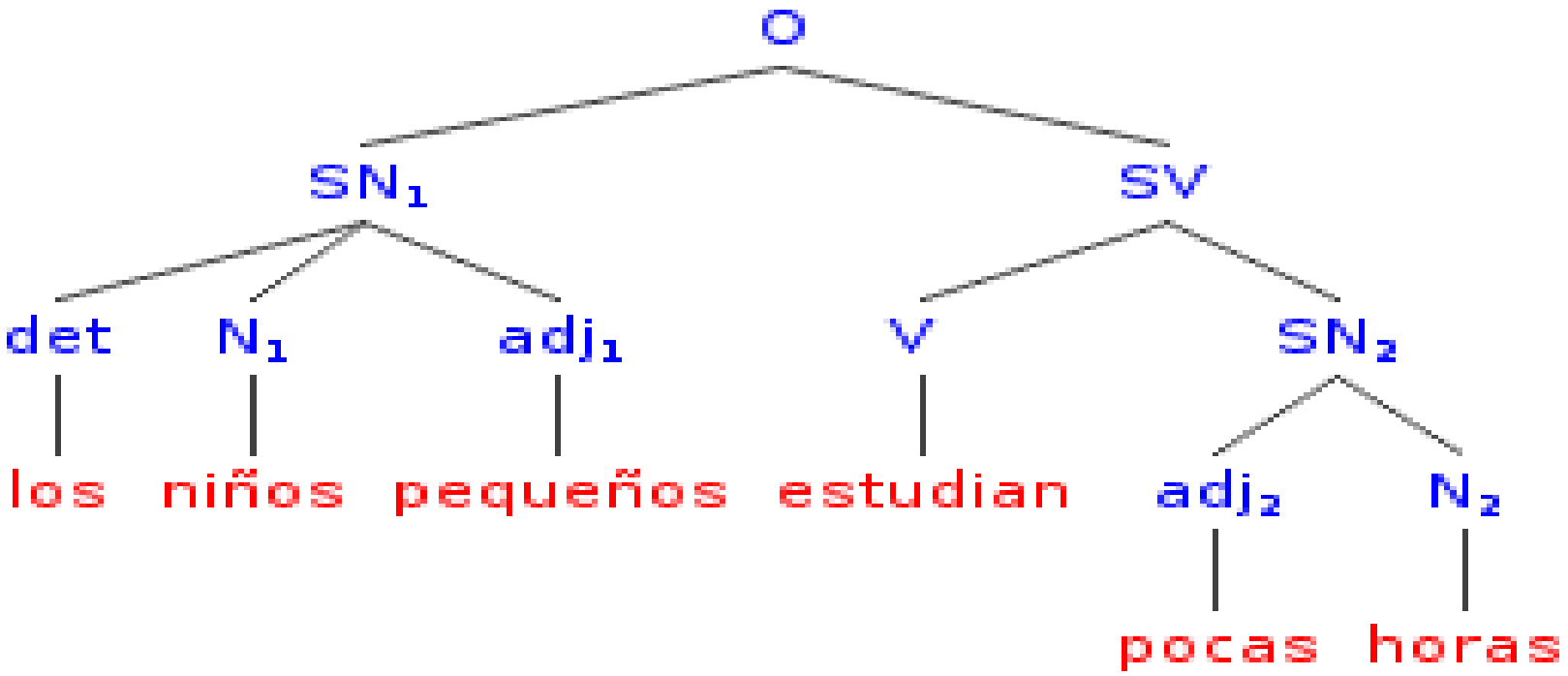
- Un tipo de gramática simple (basada en reglas) popular se denomina Gramática Independiente del Contexto (CFG).

- Ampliamente utilizada en la construcción de sistemas de NLP.

Una gramática de **dependencias**:



Una gramática de **estructura de frase**:



Lenguajes

¿Qué *máquina* es más adecuada para cada tipo de *lenguaje* y *gramática*?

Jerarquía de *Chomsky*

Gramática	Lenguaje	Modelo
Tipo 0: Irrestricta	Recursivamente enumerable (Nivel Pragmático)	Máquina de Turing (MT)
Tipo 1: Dependiente del Contexto	Dependiente del Contexto (Nivel Semántico)	Autómata Linealmente Limitado (ALL)
Tipo 2: Independiente del Contexto	Independiente del Contexto (Nivel Sintáctico)	Autómata de Pila (AP)
Tipo 3: Regular	Regular (Nivel Léxico)	Autómata Finito (AF)

CFG: Intuitivamente

Una gramática del tipo CFG permite especificar los grupos de elementos válidos en un lenguaje (*constituyentes*) y la forma en que estos se conectan (*ordenamiento*).

Ejemplo:

- En Español existe una *frase verbal* (constituyente) que viene después de una *frase nominal* (ordenamiento).
- *Una frase nominal se caracteriza por que tiene como "cabeza" un nombre.*

CFG: Formalmente

Una CFG es una 4-tuple formada por:

1) Un conjunto de símbolos no-terminales (variables): **N**
2) Un conjunto de símbolos terminales (alfabeto): **Σ**
3) Un conjunto de producciones **P** (reglas) de la forma:

   $A \rightarrow \alpha$

   Donde **A** es un no-terminal y α es un string de símbolos del conjunto infinito de strings: $(\Sigma \cup N)^*$
4) Un símbolo de comienzo (raíz): **S**

El símbolo * significa que un string de repite *0 o más veces!*.

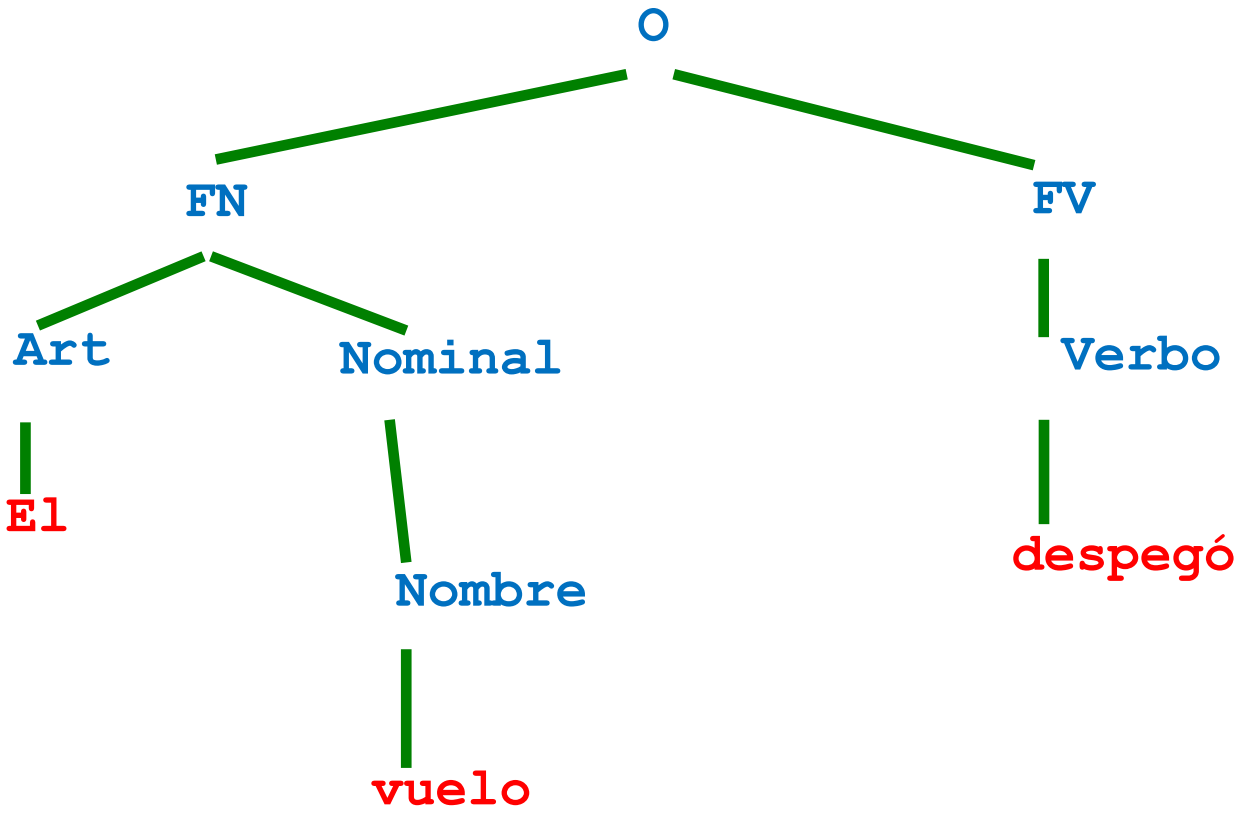
Ejemplo de una CFG

Parsing

- Dada una gramática ***G*** y un texto (o string) ***w***, un *parser* debe ser capaz de determinar si ***G*** acepta ***w.***
- Para esto, el parser debe buscar todos los *parse trees* asignados a ***w*** por ***G***.
- Pero podría generarse más de un *parse tree*.

¿Como derivamos el árbol para la oración "El vuelo despegó" utilizando la CFG previa?

O	→	FN FV
FN	→	Art NOMINAL
NOMINAL	→	Nombre
FV	→	Verbo
Art	→	un
Art	→	el
Nombre	→	vuelo
Verbo	→	despegó
Verbo	→	aterrizó



Parsing como método de búsqueda

Estrategias:

Top-down (*descendente*): comienza *buscando* desde la raíz del árbol e intenta llegar a las hojas terminales (donde están las palabras de la entrada).

Bottom-up (*ascendente*): comienza buscando desde las hojas del árbol (palabras de la entrada) e intenta buscar hacia arriba hasta llegar a la raíz.

Métodos de Parsing

- Algunos métodos populares bottom-up basados en "*memoria*" (*chart parsing*):
  - ✓ Algoritmo CKY
  - ✓ Algoritmo de *Earley*
  - ✓ *Muchos otros (deterministas, estocásticos)*
- Varios métodos que usan aprendizaje automático (requieren corpus de entrenamiento!!).

Método CKY para *full* Parsing

- Utiliza estrategia de búsqueda por *programación dinámica*.
- Utiliza una memoria (chart) para *almacenar* el trabajo hecho y no re-hacer el análisis cuando se debe revisar árboles alternativos (chart parser).
- Análisis sintáctico ***bottom-up*** eficiente (no rehace trabajo).
- Entrega la estructura sintáctica (si es que la entrada es correcta!!).

Parser: *A peras y manzanas…*

Dada una frase de entrada:

- Toma cada palabra de la entrada (símbolo terminal) y busca si es que existe una regla que la produce (se almacena en el chart).
- Luego, iterativamente buscar alguna otra regla que produzca la que se generó previamente (almacenada en el chart).
- Si requiere buscar otras alternativas para ir ascendiendo, almacenar temporalmente las reglas que ha encontrado hasta acá (hasta que se encuentre un "padre").
- El parser término cuando se logra llegar al símbolo raíz y estan cubiertos TODAS las palabras de la entrada (o sea, todas son generadas por alguna regla).
- El parser finalmente devuelve el chart donde está todo el camino recorrido (árbol).

EJEMPLO

Considere la siguiente gramática:

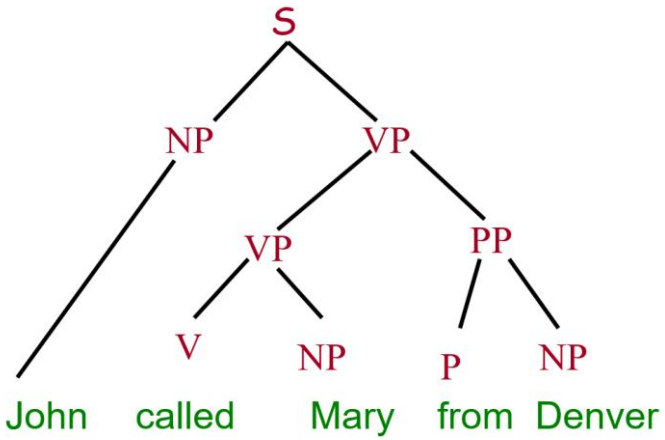
S → NP VP	NP → John
VP → V NP	NP → Mary
NP → NP PP	NP → Denver
VP → VP PP	V → called
PP → P NP	P → from

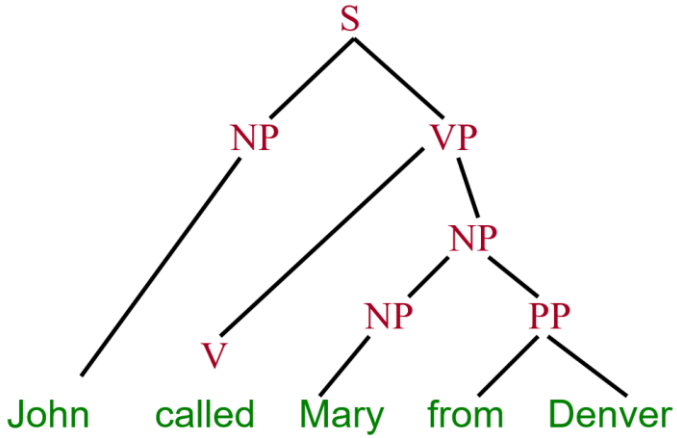
¿El parser reconoce la siguiente oración?

John called Mary from Denver

Observación

La CFG es ambigua!! Existen 2 árboles posibles





¿Cómo funciona?

				NP
			P	Denver
		NP	from	
	V	Mary		
NP	called			
John				

S → NP VP	NP → John
VP → V NP	NP → Mary
NP → NP PP	NP → Denver
VP → VP PP	V → called
PP → P NP	P → from

Consideraciones

- El parser genera finalmente el *chart* que representa el *parse tree* para la entrada.
- La entrada se reconoce, pero en este caso existe más de un árbol posible!.
- Alternativas de solución:
  - ✓ *Producir reglas probabilísticas*
  - ✓ *Modificar la gramática para que no sea ambigüa*
  - ✓ *Aprender automáticamente las reglas*

Variaciones

- Parsing se puede realizar con técnicas de aprendizaje automático.
- No siempre se necesita toda la información sintáctica.
- Podemos utilizer parsing parcial (*partial parsing*):
  ✓ Analizar y extraer sólo los grupos de interés.

Aplicación

- Recibimos textos de quejas de clientes en forma semanal.
- Debemos extraer información específica desde dichos textos: quién se queja, sobre qué, etc.
- El “target” de la queja probablemente se encuentra en alguna *frase nominal* del texto.
- Podríamos aplicar un analizador sintáctico (parser) para extraer la información:
  - Muy ineficiente pues solo requerimos una parte de la estructura sintáctica!.
- **Solución**: *partial parsing* (opuesto a *full parsing*).

Partial Parsing

(1) Chunking:

Tarea de identificar desde un texto, una secuencia de segmentos o chunks que NO se traslapan (no existe jerarquía!!).

(2) Named-Entity Recognition (NER):

Tarea de identificar las entidades mencionadas en un texto, según sus categorías (ej. persona, localización, organización).

Chunking

- Supongamos, que queremos sólo extraer las ***frases nominales*** (FN) desde una oración.
- No necesitamos realizar (full) parsing.
- Debemos detectar quién está dentro de un grupo de interés (FN) y quién no.
- Podríamos tener un *método de clasificación* que delimite lo que está al inicio de un grupo (**BEGIN**), dentro (**INSIDE**) y fuera (**OUTSIDE**).

EJEMPLO

Chunking de FN

Texto original:

He	saw	the	big	Dog

Texto etiquetado:

PRP	VBD	DT	JJ	NN

Texto segmentado:

BEGIN	OUTSIDE	BEGIN	INSIDE	INSIDE

Métodos

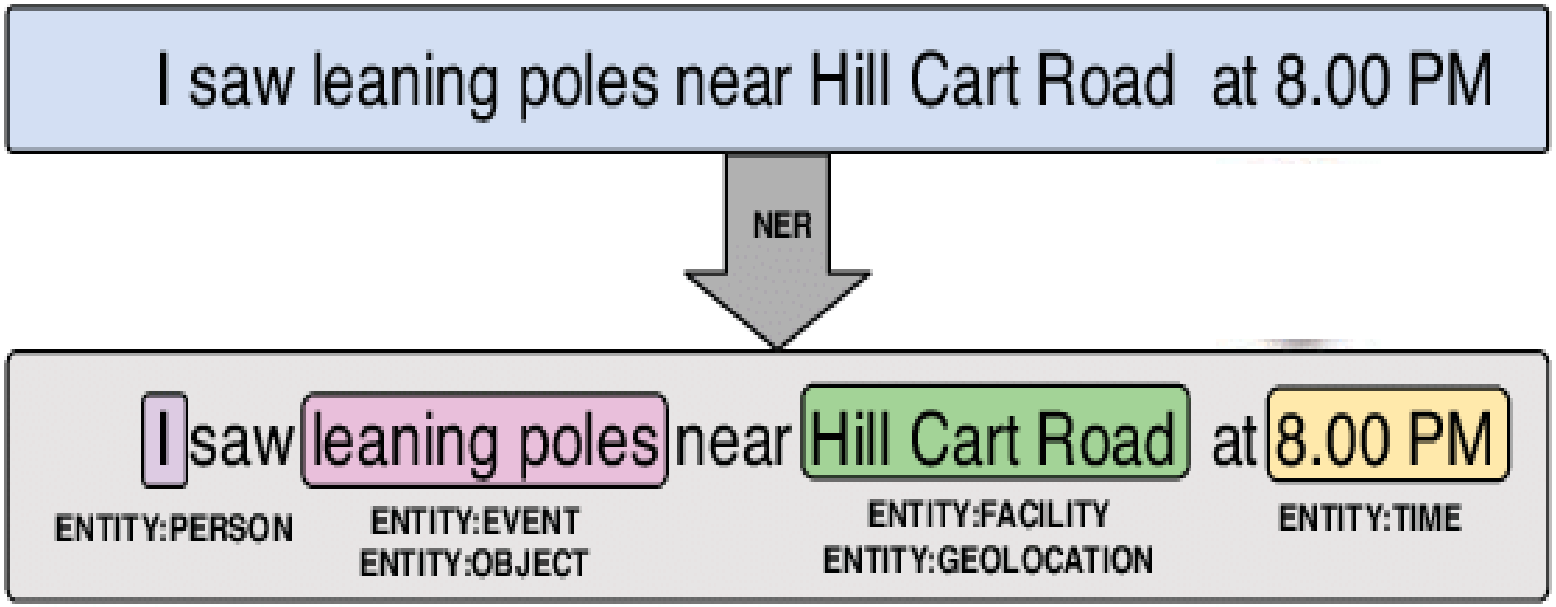
- Reglas sintácticas simples que detecten los chunks de interés (ej. FN, FV).
- Métodos de aprendizaje supervisado que etiquetan chunks respecto a un *corpus* de textos etiquetados de entrenamiento.
- Métodos estocásticos de predicción de chunks.

Named-Entity Recognition (NER)

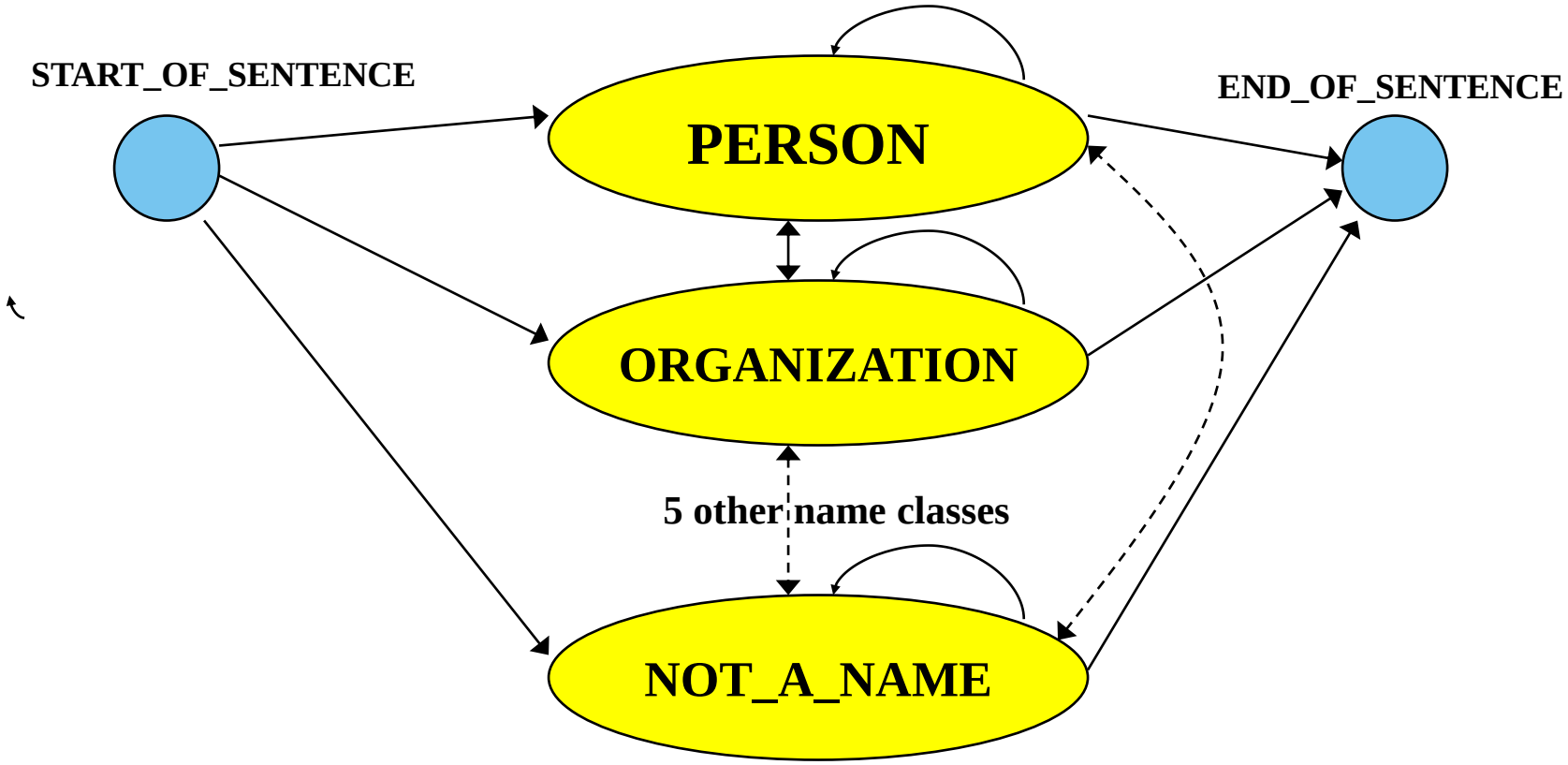
- **NER** es una tarea de análisis parcial que busca *localizar* y *clasificar* nombres de entidades mencionadas en un texto en categorías predefinidas: *nombres*, *organizaciones*, *localizaciones*, *códigos*, *fechas*, etc.

- **NER** es una tarea intrínseca de secuencia-a-secuencia:
  - Recibe un texto de entrada y genera una secuencia de etiquetas de entidades de salida.

Named-Entity Recognition



NER con HMM


START_OF_SENTENCE
PERSON
END_OF_SENTENCE
ORGANIZATION
5 other name classes
NOT_A_NAME

Métodos para NER

1. Patrones escritos manualmente basado en reglas.

2. Clasificadores automáticos: Bayesianos, Máxima Entropía, Redes Neuronales.

3. Modelos de predicción de secuencias: HMM, apredizaje profundo, redes recurrentes, LSTM, etc.

RESUMEN

- **Parsing** es la tarea de construir automáticamente una estructura gramatical a partir de oraciones en lenguaje natural.
- Existen varias técnicas de parsing tradicionales, y basadas en aprendizaje automático, del tipo top-down y bottom-up.
- Métodos robustos incluyen técnicas de parsing parcial como Chunking y Named-Entity Recognition.


Tiempo de Ejercicios

Ejercicio (1)

- ✓ Cargue en *Google Colab* el programa Python **`parsing.`**
- ✓ Cargue la gramática del archivo "aerolínea.cfg".
- ✓ Ejecute el parser paso a paso, y vea que entrega al ingresar diferentes consultas:

  quiero un vuelo
  quiero un vuelo desde Santiago a Concepcion

Ejercicio (2)

- ✓ Cargue en *Google Colab* el programa Python **NER**.
- ✓ Cargue el texto de ejemplo "sample.txt".
- ✓ Ejecute el programa para extraer entidades desde un texto de ejemplo.

Ejercicio (3)

- ✓ Cargue en *Google Colab* el programa Python **`chunking`**.
- ✓ Cargue el texto de ejemplo “sample.txt”.
- ✓ Ejecute el programa para extraer entidades desde un texto de ejemplo.