

# Introduction au Traitement Automatique des Langues

## **5 – Les niveaux de traitement – Le niveau Syntaxique**

# Introduction au traitement automatique des langues

## Contenu de la matière :

- 1) Introduction Générale
- 2) Les applications du TAL
- 3) Les niveaux de traitement - Traitements de «bas niveau»
- 4) Les niveaux de traitement - Le niveau lexical
- 5) Les niveaux de traitement - Le niveau syntaxique**
- 6) Les niveaux de traitement - Le niveau sémantique
- 7) Les niveaux de traitement - Le niveau pragmatique

# Plan du cours

1. Syntaxe - Définition
2. Analyse Syntaxique – Définitions
3. Chunking : Constituants – Syntagmes
4. Grammaire Hors Contexte CFG et FNC
5. Arbre syntaxique et algorithme CYK
6. Arbre de dépendence
7. Named Entity Recognition – NER

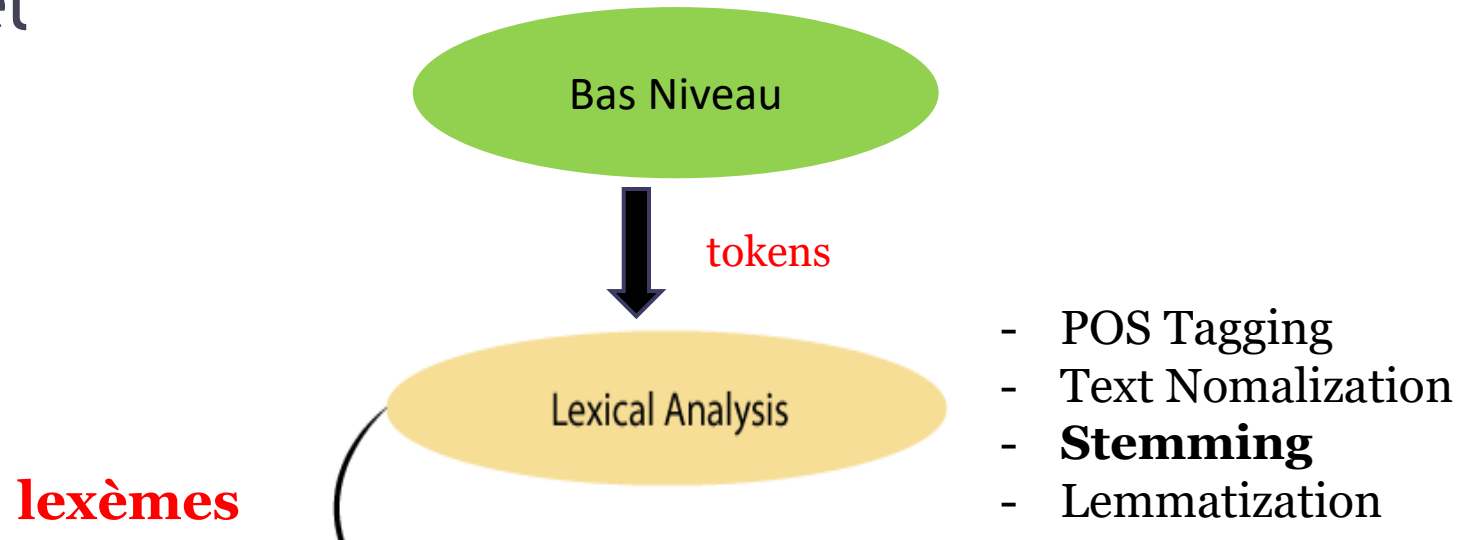
# Rappel



**tokens**

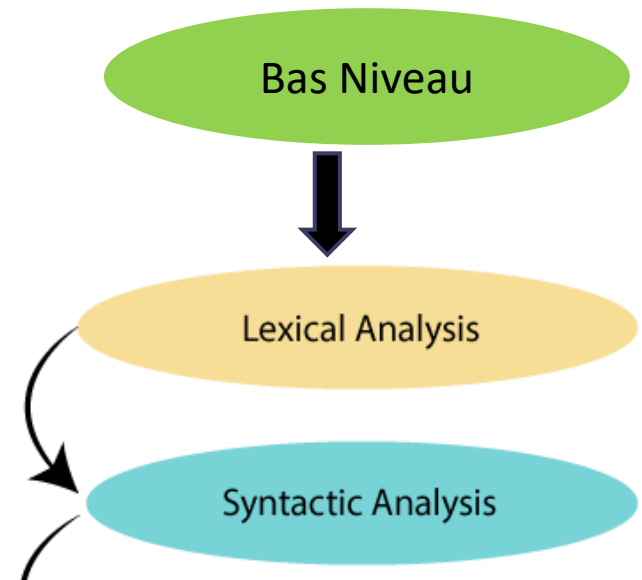
- Encodage de caractères
- **Tokenization**
- Filtrage simple

# Rappel



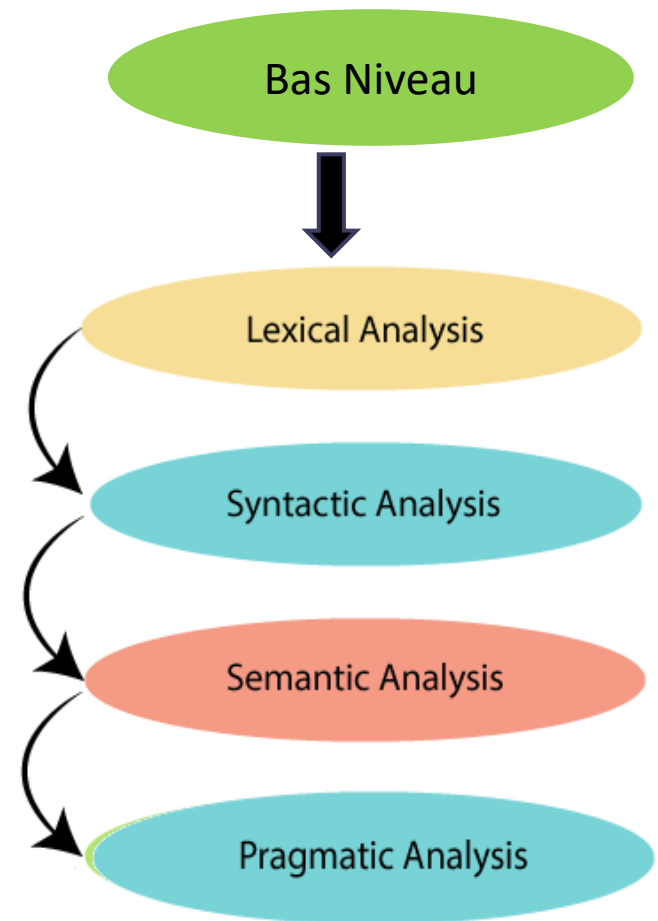
# Analyse Syntaxique

- Optionnel, selon l'application/projet.
- POS Tagging - Etiquetage morpho-syntaxique:
  - Trouver les catégories des mots (nom, verbe, etc.) d'une phrase.
- Analyse syntaxique de surface (**Chunking**):
  - Trouver les **syntagmes** d'une phrase sans structure d'arbre.
- **Parse tree** and **Dependency tree**.
- Reconnaissance **d'entités nommées (NER)**:
  - Trouver les personnes, les organisations, les places, les nombres, etc. dans une phrase.



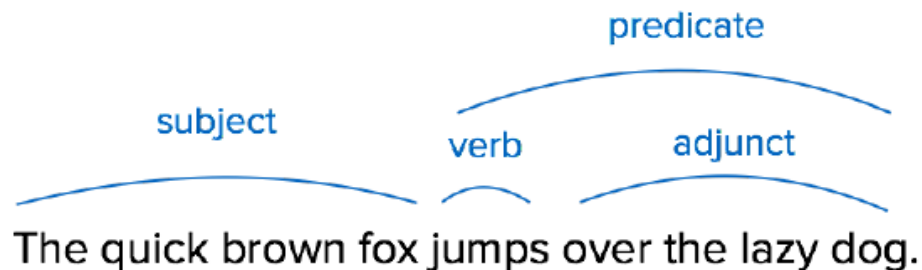
# Définitions - Syntaxe

- La **syntaxe** est la branche de la linguistique qui étudie la façon dont les **mots se combinent** pour former des **phrases** ou des **énoncés** dans une langue.
- La **syntaxe** est l'étude de la façon avec laquelle les **mots sont composés et ordonnés** pour construire des **phrases** et **énoncés** bien formés .
- La **syntaxe** est l'ensemble des **règles** de composition de phrases à partir de mots.
- La syntaxe ne s'intéresse **pas** à la **sémantique** (sens, signification) des mots.
- Par contre, une analyse syntaxique peut apporter de l'information utile à la compréhension d'une phrase.



# Définitions - Analyse Syntaxique

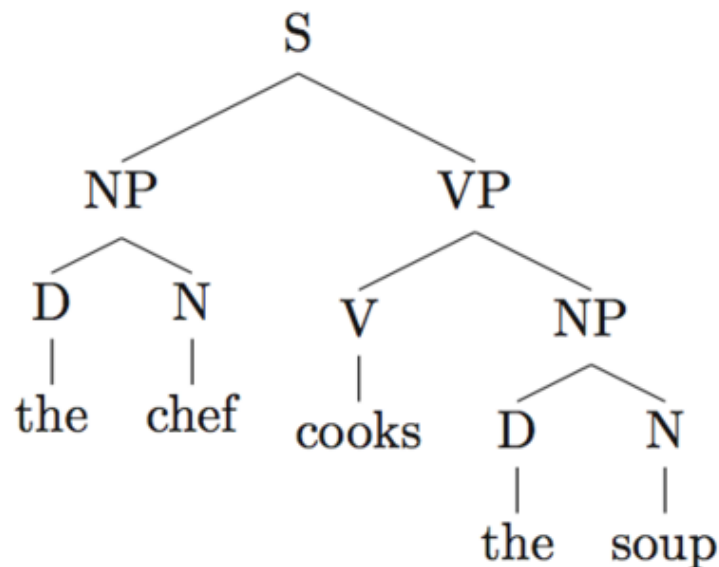
- **L'analyse syntaxique (parsing)** consiste à mettre en évidence la **structure d'un texte**, généralement une phrase écrite dans une **langue naturelle** (ou d'un **programme informatique**).
- Elle vérifie et calcule la **validité** de certaines **séquences de mots**, et l'**ordre des mots** dans une **phrases**.
- Elle peut donc être utilisé pour signaler les erreurs de syntaxe et de **grammaire**, en suivant les **règles** définies par la langue.
- L'analyse syntaxique vise à **produire** une **représentation** (**arbre**) des **relations grammaticales** entre les mots.





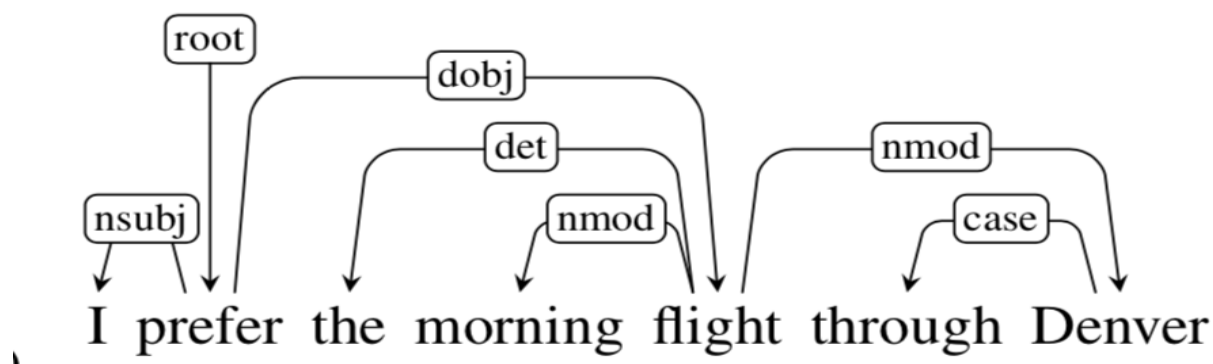
# Définitions - Analyse Syntaxique

- **L'analyse syntaxique (parsing)** consiste à mettre en évidence la **structure d'un texte**, généralement une phrase écrite dans une **langue naturelle** (ou d'un **programme informatique**).
- L'analyseur syntaxique (**parser**, en anglais) est le programme informatique qui réalise cette tâche.
- L'analyse syntaxique vise à **produire** une **représentation (arbre)** des **relations grammaticales** entre les mots.
- Deux types : **arbre syntaxique** – parse tree



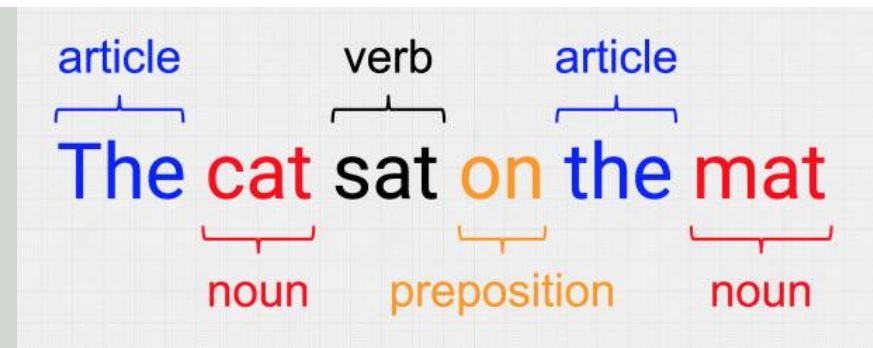
# Définitions - Analyse Syntaxique

- **L'analyse syntaxique (parsing)** consiste à mettre en évidence la structure d'un texte, généralement une phrase écrite dans une langue naturelle (ou d'un programme informatique).
- L'analyseur syntaxique (**parser**, en anglais) est le programme informatique qui réalise cette tâche.
- L'analyse syntaxique vise à produire une représentation (**arbre**) des relations grammaticales entre les mots.
- Deux types : **arbre de dépendance** – dependency tree



# Etiquetage Morpho-syntactique - Rappel

- Part-of-Speech Tagging – **POS Tagging**
- Est le processus de **classification** automatique des mots dans leurs **catégorie grammaticale**, et de les **étiqueter** (annoter) en conséquence.
- Assigne chaque mot d'un texte à sa catégorie grammaticale.
- Par exemple, le mot ferme peut être un **verbe (V)** dans « il **ferme** la porte » et un **nom (N)** dans « il va à la **ferme** ».
- Par exemple : **drink** peut être breuvage (nom) ou **boire** (verbe).



# Etiquetage Morpho-syntaxique

=> **Limitation** :

- En utilisant le POS tagging, nous pourrions obtenir le résultat ci-dessous pour

«**le parapluie bleu** dans le sac»

«**le**»: Déterminant, «**parapluie**»: Nom, «**bleu**»: Adjectif, ... , «**sac**»: Nom

- Il n'est nulle part mentionné que le terme «**bleu**» est utilisé (lié) pour «**parapluie**».
- Par conséquent, nous avons besoin d'une sorte de **groupement de mots** afin de retrouver la relation entre les mots d'une phrase.
- Traiter « le parapluie bleu » comme unité lexicale à part entière.

➔ Analyse syntaxique : **Chunking**

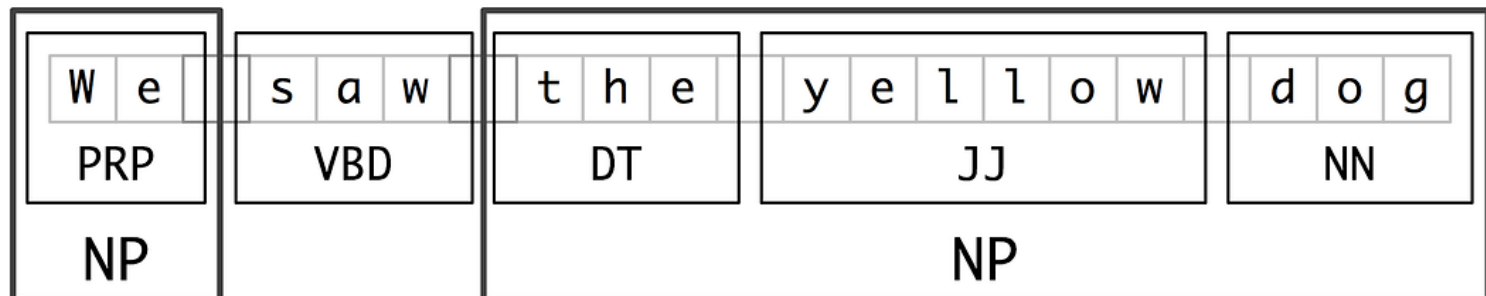
# Analyse Syntaxique - Chunking

- Le **Chunking** permet d'exposer et d'identifier de **nouveaux concepts linguistiques** utiles (division d'une phrase en blocs plus grands):
  1. les **constituants (syntagmes)**, qui sont des **groupes de mots** agissant comme une unité à part entière, tels les phrases de nom.
  2. les **relations grammaticales** telles les relations d'objet et de sujet d'un verbe.
  3. d'autres propriétés sur la relation entre les mots d'une phrase telles les **relations de dépendance**.
- **Chunking** veut dire groupement de mots/tokens en **chunks (constituants, syntagmes)**.
- Fonctionne après le POS Tagging. Il utilise les POS-tags comme entrée et fournit les chunks en sortie.

# Analyse Syntaxique - Chunking

**Constituant** (constituent, chunk, syntagme)

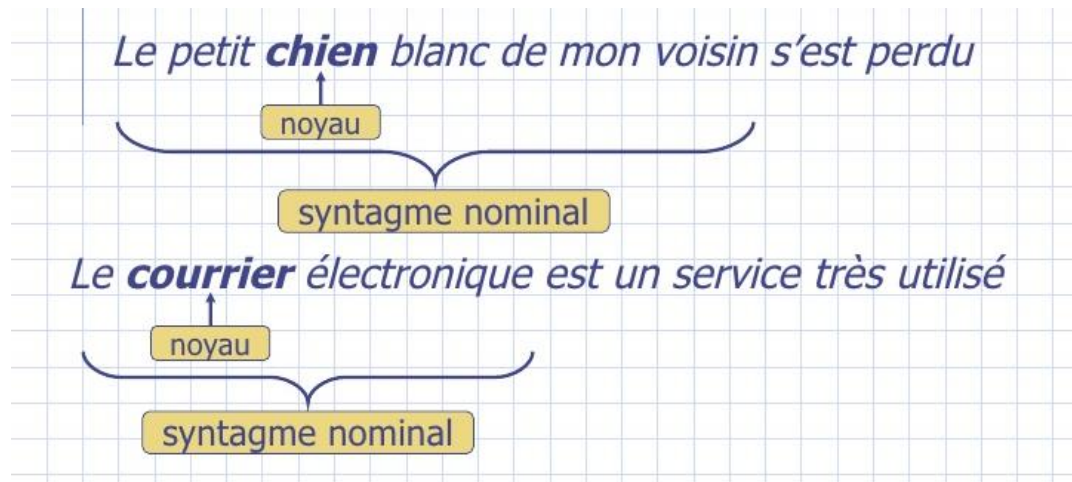
- Les énoncés naturels ne sont pas simplement des **suites de mots**, mais sont organisés en **constituants** de **taille supérieure au mot** (les **syntagmes**), qui entretiennent entre eux des **relations**.
- Il est composé d'un ou plusieurs mots allant jusqu'à la phrase simple.
- Le syntagme est composé d'un **noyau**, appelé aussi « **tête** ».
- **L'un des buts de l'analyse syntaxique est donc d'associer, à chaque énoncé, sa structure de constituants (Chunking).**



# Analyse Syntaxique - Chunking

**Constituant** (constituent, chunk, syntagme)

- Types : **Syntagme Nominal** (Noun Phrase - **NP**)
- Est un nom ou un groupe de mots entourant un nom, décrivant normalement l'objet ou le sujet d'un verbe.
- Est **un groupe de mots** dont l'élément central (**noyau**) est un **nom**.
- Un groupe de mots agissant comme un nom dans une phrase.



# Analyse Syntaxique - Chunking

**Constituant** (constituent, chunk, syntagme)

- Types : **Syntagme Verbal** (Verbal Phrase - **VP**)
- Le syntagme verbal est formé d'un verbe seule ou du verbe et de ses éventuels compléments.
- un **groupe de mots** agissant comme un verbe dans une phrase.
- Un syntagme verbal a pour **noyau** un **verbe**.
- *Il a travaillé courageusement toute la fin de semaine* : Le syntagme verbal « a travaillé courageusement toute la fin de semaine » a pour noyau le verbe « a travaillé ».

*Zoé offre un cadeau à Max*

*Eve met les fraises dans le panier*

*Marie discute de cette question avec Jean*

*Jean dort*

*Marie chante*

*Max nage*



# Analyse Syntaxique - Chunking

**Constituant** (constituent, chunk, syntagme)

- Types : **Syntagme Prépositionnel** (Prepositional Phrase - **PP**)
- **préposition** (à, de, pour, sur, dans, avec, en, par, parmi, depuis, après, avant, etc.) **suivi d'un syntagme nominal**.
- La **préposition n'est pas le noyau** dur du syntagme prépositionnel.
- *Je pars **à** trois **heures*** : Le syntagme prépositionnel « à trois heures » a pour noyau le nom "heures".

## Prepositional Phrases

- |                                 |                                |
|---------------------------------|--------------------------------|
| • <i>In</i> the beginning       | • <i>Around</i> the bend       |
| • <i>Before</i> the fall        | • <i>Down</i> in the sand trap |
| • <i>After</i> the brutal fight | • <i>Into</i> the dark woods   |
| • <i>At</i> school              | • <i>Against</i> the wind      |
| • <i>Down</i> the aisle         | • <i>Near</i> the mouse        |
| • <i>Across</i> the street      | • <i>Through</i> the tunnel    |
| • <i>Inside</i> your ear        | • <i>To</i> school             |
| • <i>Outside</i> the house      | • <i>Like</i> Larry's uncle    |
| • <i>Between</i> two girls      | • <i>Except</i> my friend      |

# Analyse Syntaxique - Chunking

## Exemple:

*« the blue umbrella is in the bag »*

### 1 – POS-Tagging, Penn Treebank Tagset :

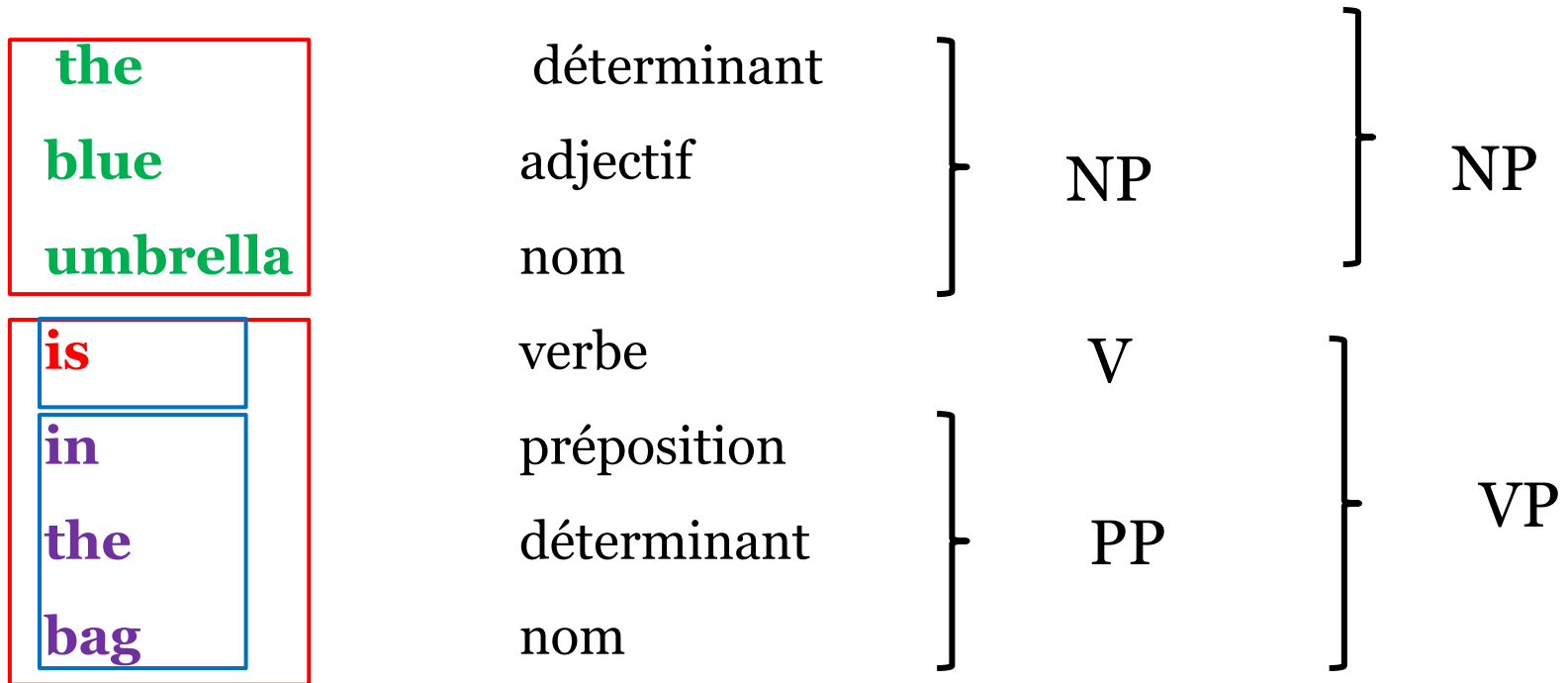
the	déterminant	DT
blue	adjectif	JJ
umbrella	nom	NN
is	verbe	VBP
in	préposition	IN
the	déterminant	DT
bag	nom	NN

# Analyse Syntaxique - Chunking

## Exemple:

« *the blue umbrella is in the bag* »

**2 – Chunking :** identifier les constituants, en fonction de règles grammaticales



# Analyse Syntaxique - Chunking

## Exemple:

« *the blue umbrella is in the bag* »

**2 – Chunking:** identifier les constituants en fonction de **règles grammaticales** :

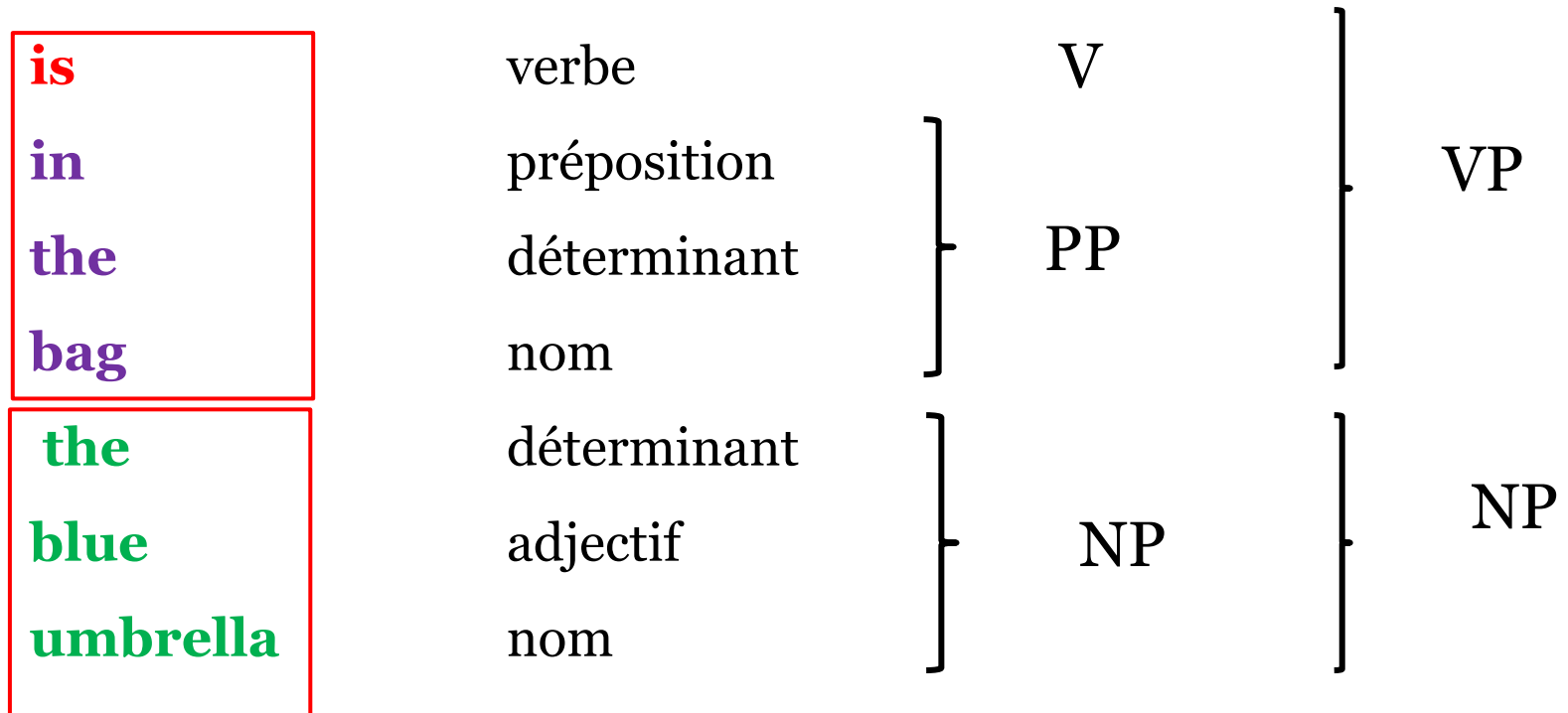
- **Exemple : Regex based chunking** : Introduire une grammaire où **NP** est combiné par :
  - DT? → one or zero determiner
  - JJ\* → zero or more adjectives
  - NN → Noun

# Analyse Syntaxique - Chunking : Limitations

## Exemple:

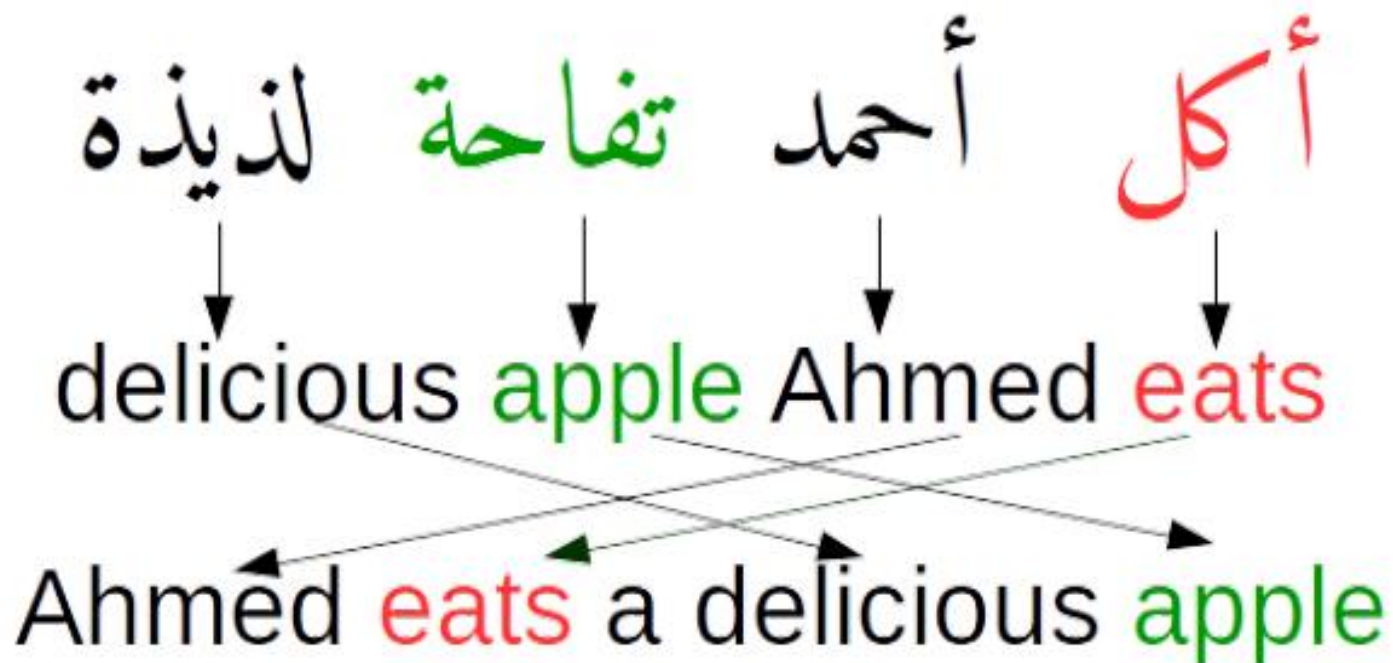
« *the blue umbrella is in the bag* »

## 2 – Chunking : Ordre ??



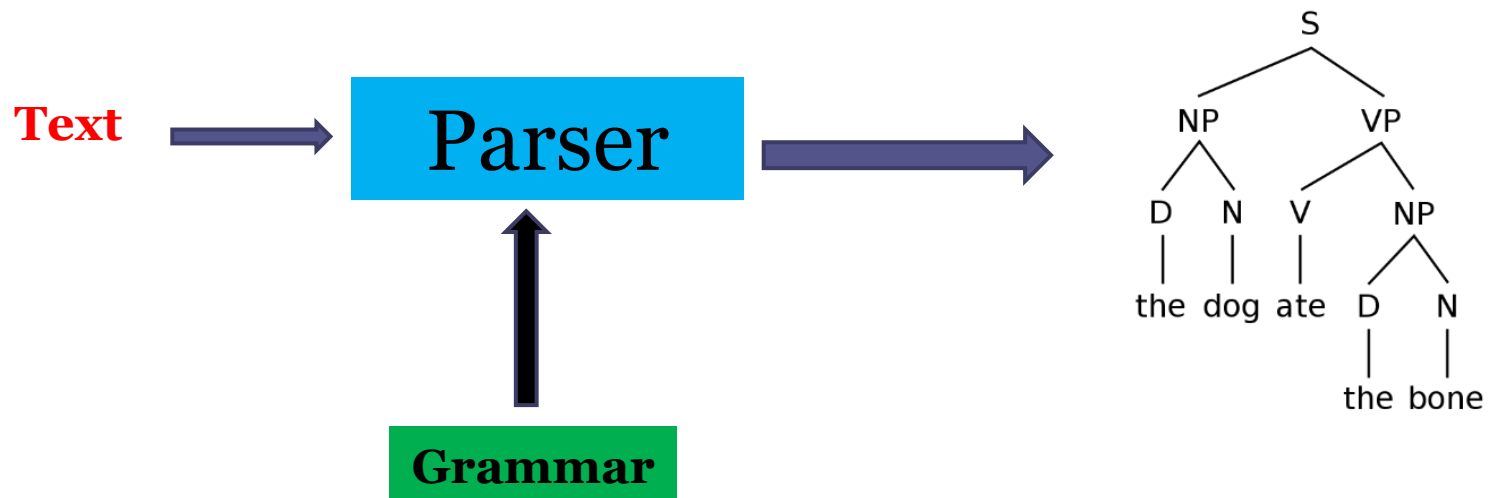
# Analyse Syntaxique - Chunking : Limitations

- Relations, ordres, et dépendances entres ces constituants ?
- Validité d'une phrase ?



# Analyse Syntaxique - Parse Tree

- **L'analyse syntaxique** (parsing) est le processus de prise en input d'un **texte/phrased** et d'une **grammaire** et de renvoyer comme output un (ou plusieurs) **arbre (s) syntaxique (s)** pour ce texte/phrased.
- C'est le processus de détermination de la **structure syntaxique** d'un texte en analysant ses **mots** et ses **constituants** sur la base d'une **grammaire** sous-jacente (de la langue).
- **Input : Texte + CFG => Output : structure (arbre) syntaxique**



# Grammaire hors Contexte

- La manière la plus courante de **modéliser** la *constituency* => **CFG**
- **Une grammaire hors contexte** (Context Free Grammar): une **liste de règles** qui définissent l'ensemble de toutes les **phrases bien formées** dans une langue.
- Chaque règle a un côté **gauche**, qui identifie une **catégorie syntaxique (constituants)**, et un côté **droit**, qui définit ses **composants alternatifs (constituants ou mots)**.
- Par exemple, la règle **S -> NP VP** signifie qu'une phrase (S) est définie comme une phrase nominale NP suivie d'une phrase verbale VP.
- Exemples règles pour l'Anglais : S -> NP VP, VP -> V VP, VP -> VP PP, VP -> V, VP -> V NP, NP -> NP PP, PP -> P NP, etc.



# Grammaire hors Contexte

- Une **grammaire hors contexte**  $G$  est définie par
  - $N$  : un ensemble de **symboles non-terminaux**
  - $\Sigma$  : un ensemble de **symboles terminaux** (différents de  $N$ )
  - $R$  : un ensemble de **règles de dérivations** ou **productions** de la forme

$$A \rightarrow \beta$$

où  $A$  est non-terminal et  $\beta$  est une chaîne dans  $(N \cup \Sigma)^*$

- $S$  : le symbole de départ

Non terminaux : constituants

Terminaux : mots

# Arbre Syntaxique - Parse Tree

- Le résultat de l'analyse syntaxique est représenté sous la forme d'un arbre => **Arbre Syntaxique**.
- Un arbre syntaxique convertit la phrase en un arbre dont les **feuilles** contiendront des **POS tags** (qui correspondent aux mots de la phrase), et le **reste de l'arbre** dira comment ces mots se **rejoignent** pour former la phrase globale.
- Exemple : **CFG**

$$\left\{ \begin{array}{l} S \Rightarrow NP \ VP \\ VP \Rightarrow V \ NP \\ NP \Rightarrow D \ N \\ PP \Rightarrow P \ NP \end{array} \right.$$

# Arbre Syntaxique - Parse Tree

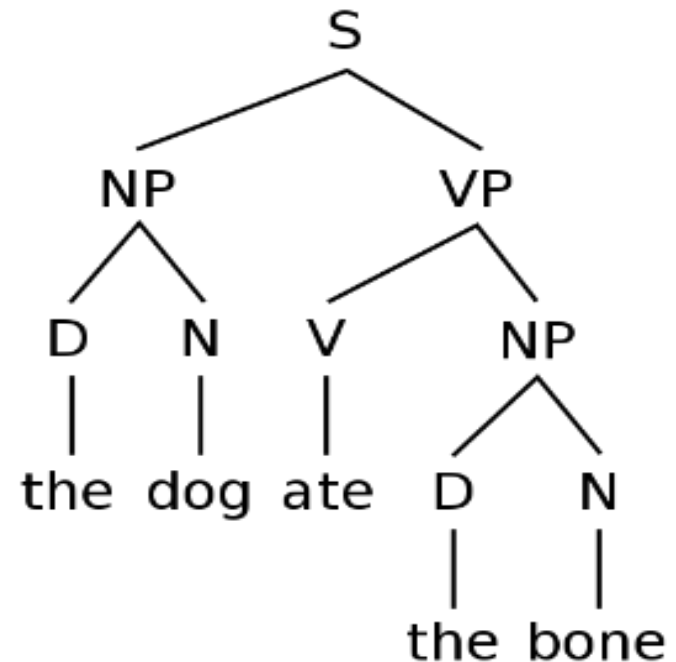
- Le résultat de l'analyse syntaxique est représenté sous la forme d'un arbre => **Arbre Syntaxique**.
- Un arbre syntaxique convertit la phrase en un arbre dont les **feuilles** contiendront des **POS tags** (qui correspondent aux mots de la phrase), et le **reste de l'arbre** dira comment ces mots se **rejoignent** pour former la phrase globale.
- Exemple : **CFG**

$$\left\{ \begin{array}{l} S \Rightarrow NP \ VP \\ VP \Rightarrow V \ NP \\ NP \Rightarrow D \ N \\ D \Rightarrow \text{the} \\ N \Rightarrow \text{dog} \mid \text{bone} \\ V \Rightarrow \text{ate} \end{array} \right.$$

# Arbre Syntaxique - Parse Tree

- Le résultat de l'analyse syntaxique est représenté sous la forme d'un arbre => **Arbre Syntaxique**.
- Un arbre syntaxique convertit la phrase en un arbre dont les **feuilles** contiendront des **POS tags** (qui correspondent aux mots de la phrase), et le **reste de l'arbre** dira comment ces mots se **rejoignent** pour former la phrase globale.
- Exemple **texte** : *The dog ate the bone*

$S \Rightarrow NP \ VP$   
 $VP \Rightarrow V \ NP$   
 $NP \Rightarrow D \ N$   
 $D \Rightarrow the$   
 $N \Rightarrow dog \mid bone$   
 $V \Rightarrow ate$



# Arbre Syntaxique - Parse Tree

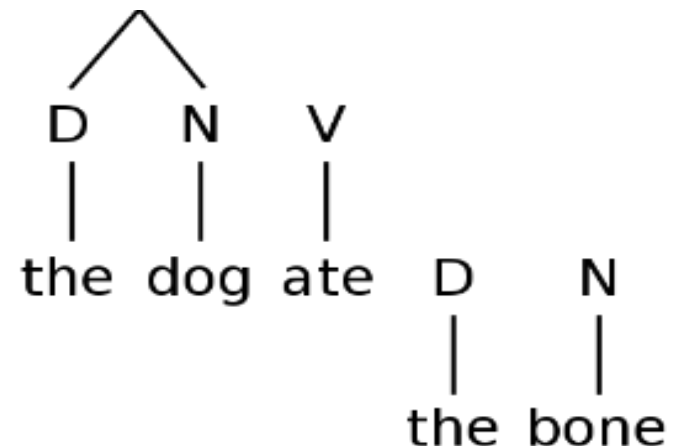
- Le résultat de l'analyse syntaxique est représenté sous la forme d'un arbre => **Arbre Syntaxique**.
- Un arbre syntaxique convertit la phrase en un arbre dont les **feuilles** contiendront des **POS tags** (qui correspondent aux mots de la phrase), et le **reste de l'arbre** dira comment ces mots se **rejoignent** pour former la phrase globale.
- Exemple **texte** : ***The dog ate the bone***

POS  
Tagging

[

- $S \Rightarrow NP \ VP$
- $VP \Rightarrow V \ NP$
- $NP \Rightarrow D \ N$
- $D \Rightarrow \text{the}$
- $N \Rightarrow \text{dog} \mid \text{bone}$
- $V \Rightarrow \text{ate}$

]

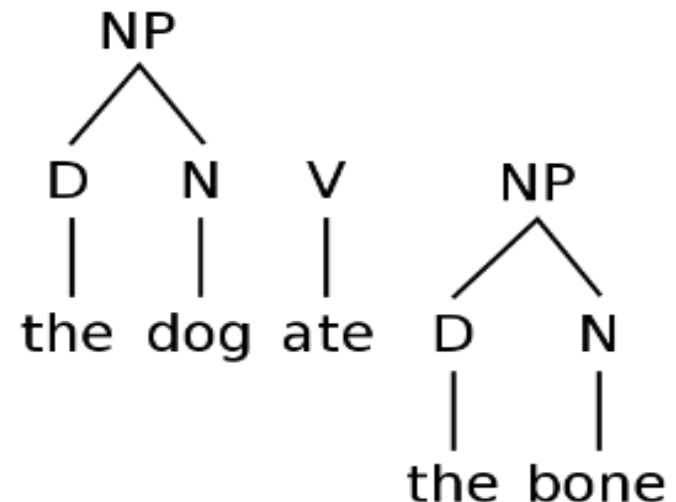


# Arbre Syntaxique - Parse Tree

- Le résultat de l'analyse syntaxique est représenté sous la forme d'un arbre => **Arbre Syntaxique**.
- Un arbre syntaxique convertit la phrase en un arbre dont les **feuilles** contiendront des **POS tags** (qui correspondent aux mots de la phrase), et le **reste de l'arbre** dira comment ces mots se **rejoignent** pour former la phrase globale.
- Exemple **texte** : ***The** **dog** **ate** **the** **bone***

Chunking

{	<b>S =&gt; NP VP</b>
	<b>VP =&gt; V NP</b>
	<b>NP =&gt; D N</b>
	<b>D =&gt; the</b>
	<b>N =&gt; dog   bone</b>
	<b>V =&gt; ate</b>

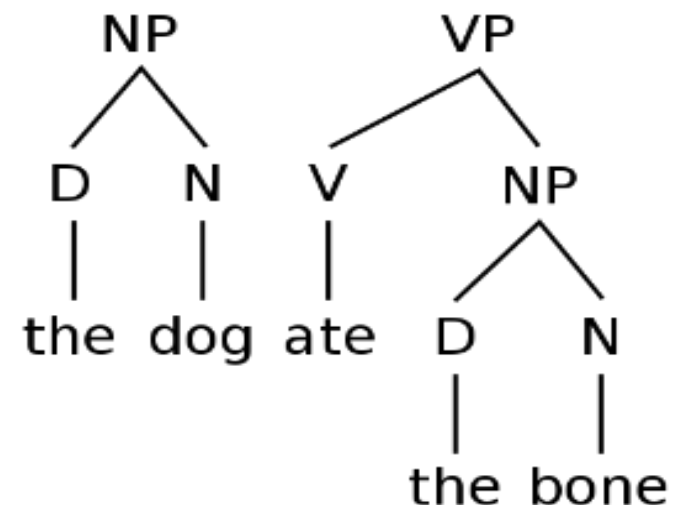


# Arbre Syntaxique - Parse Tree

- Le résultat de l'analyse syntaxique est représenté sous la forme d'un arbre => **Arbre Syntaxique**.
- Un arbre syntaxique convertit la phrase en un arbre dont les **feuilles** contiendront des **POS tags** (qui correspondent aux mots de la phrase), et le **reste de l'arbre** dira comment ces mots se **rejoignent** pour former la phrase globale.
- Exemple **texte** : *The dog ate the bone*

Chunking

{	<b>S =&gt; NP VP</b>
	<b>VP =&gt; V NP</b>
	<b>NP =&gt; D N</b>
	<b>D =&gt; the</b>
	<b>N =&gt; dog   bone</b>
	<b>V =&gt; ate</b>



# Arbre Syntaxique - Parse Tree

- Le résultat de l'analyse syntaxique est représenté sous la forme d'un arbre => **Arbre Syntaxique**.
- Un arbre syntaxique convertit la phrase en un arbre dont les **feuilles** contiendront des **POS tags** (qui correspondent aux mots de la phrase), et le **reste de l'arbre** dira comment ces mots se **rejoignent** pour former la phrase globale.
- Exemple **texte** : *The dog ate the bone*

Chunking

**S => NP VP**

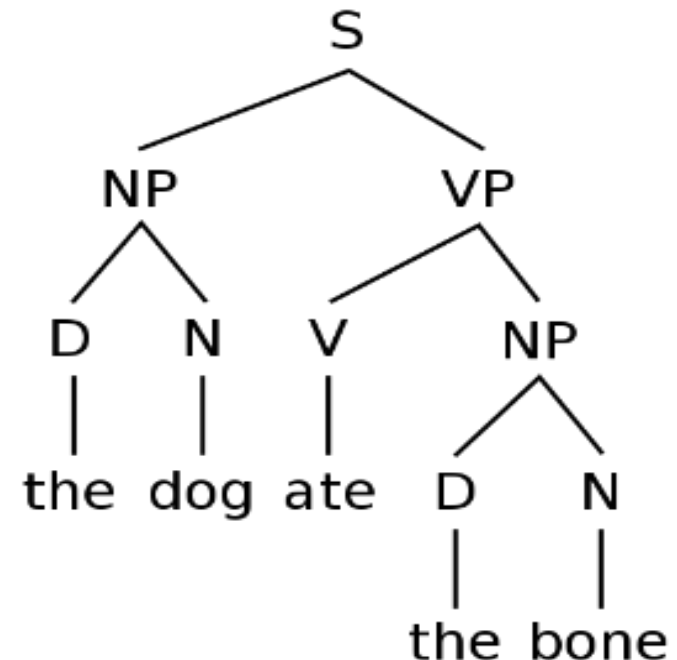
**VP => V NP**

**NP => D N**

**D => the**

**N => dog | bone**

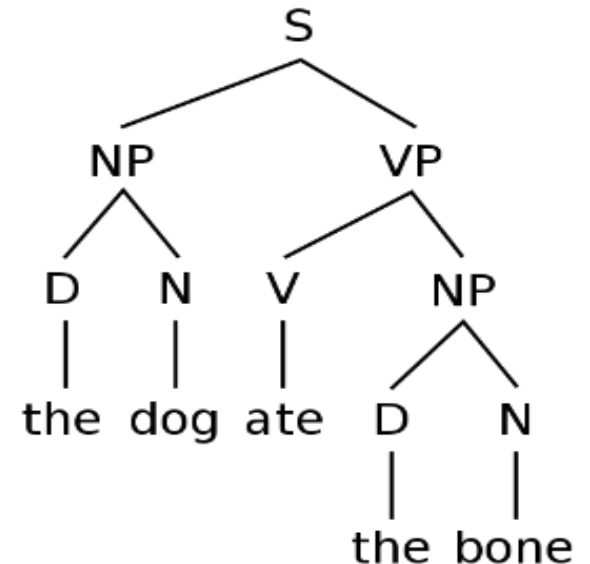
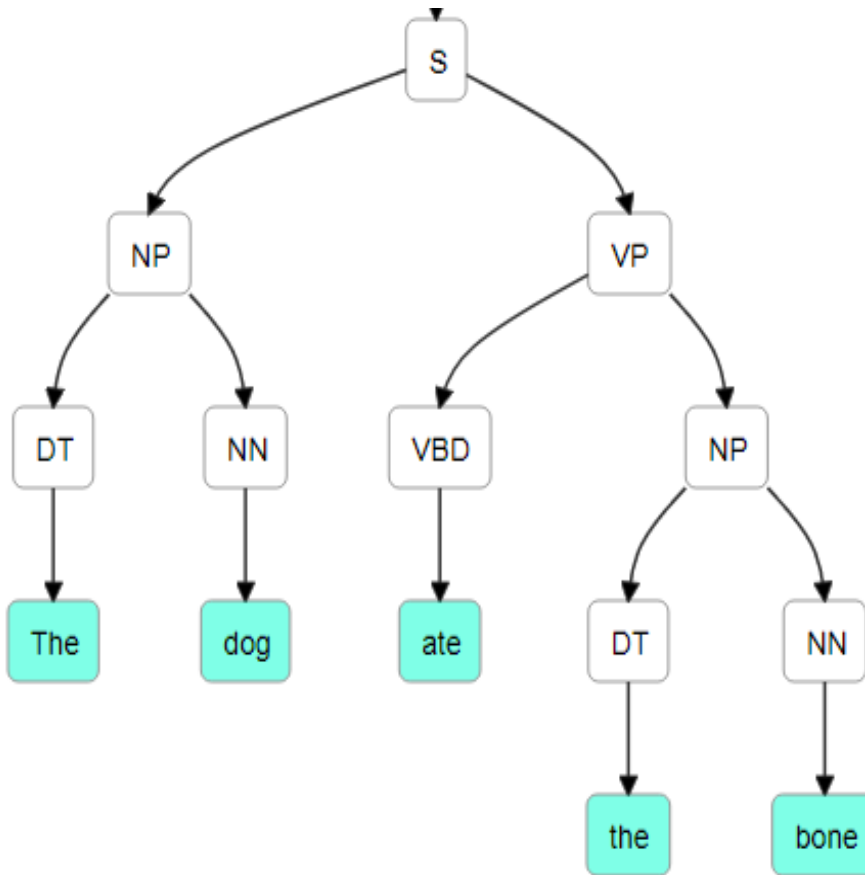
**V => ate**





# Arbre Syntaxique - Parse Tree

- Demo: <http://nlpviz.bpodgursky.com/>
- CoreNLP : <https://stanfordnlp.github.io/CoreNLP/>



# Arbre Syntaxique - Parse Tree

- **Bracketed notation - Label bracketing**

[S [NP [D The] [N dog] ] [VP [V ate] [NP [D The] [N bone] ] ] ]

- <http://mshang.ca/syntaxtree/>

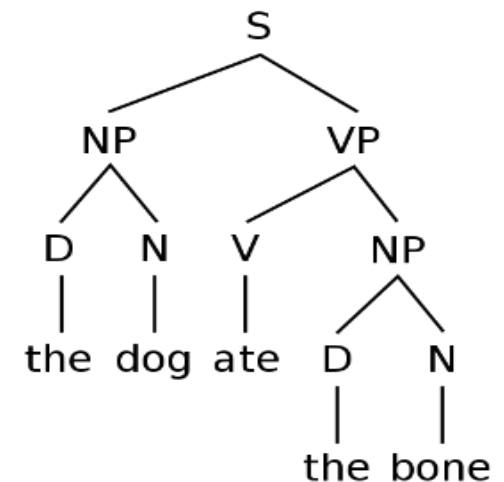
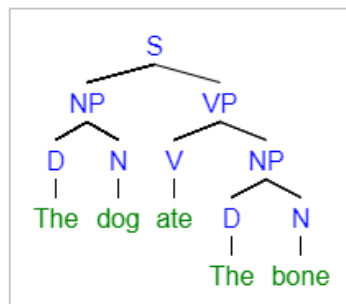
Syntax Tree Generator

[S [NP [D The] [N dog]] [VP [V ate] [NP [D The] [N bone]] ] ]

(C) 2011 by [Miles Shang](#), see [license](#).

Options

Help



# Arbre Syntaxique - Parse Tree

<http://nlp.stanford.edu:8080/parser/index.jsp>

## Stanford Parser

Please enter a sentence to be parsed:

The dog ate the bone

Language: English

[Sample Sentence](#)

[Parse](#)

### Your query

*The dog ate the bone*

### Tagging

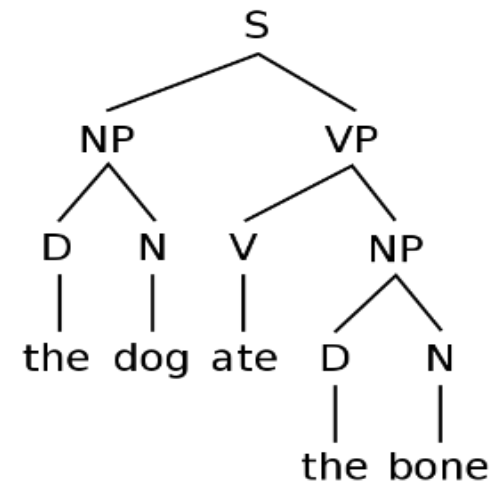
The/DT dog/NN ate/VBD the/DT bone/NN

### Parse

```
(ROOT
  (S
    (NP (DT The) (NN dog))
    (VP (VBD ate)
      (NP (DT the) (NN bone))))))
```

### Universal dependencies

```
det(dog-2, The-1)
nsubj(ate-3, dog-2)
root(ROOT-0, ate-3)
det(bone-5, the-4)
obj(ate-3, bone-5)
```



### Parse

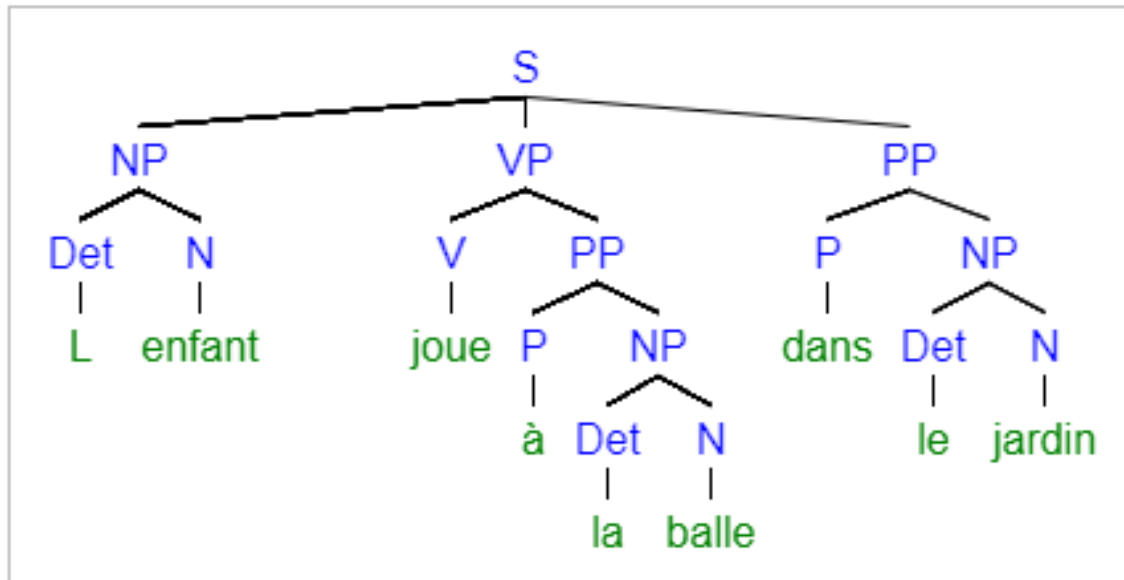
```
(ROOT
  (S
    (NP (DT The) (NN dog))
    (VP (VBD ate)
      (NP (DT the) (NN bone))))))
```

# Arbre Syntaxique - Parse Tree

**Exemple 2 :** Tracer l'arbre(s) syntaxique(s) pour la phrase suivante :

*L'enfant joue à la balle dans le jardin*

**Réponse :** [S [NP [Det L] [N enfant]]  
[VP [V joue] [PP [P à] [NP [Det la] [N balle]]]]  
[PP [P dans] [NP [Det le] [N jardin]] ]

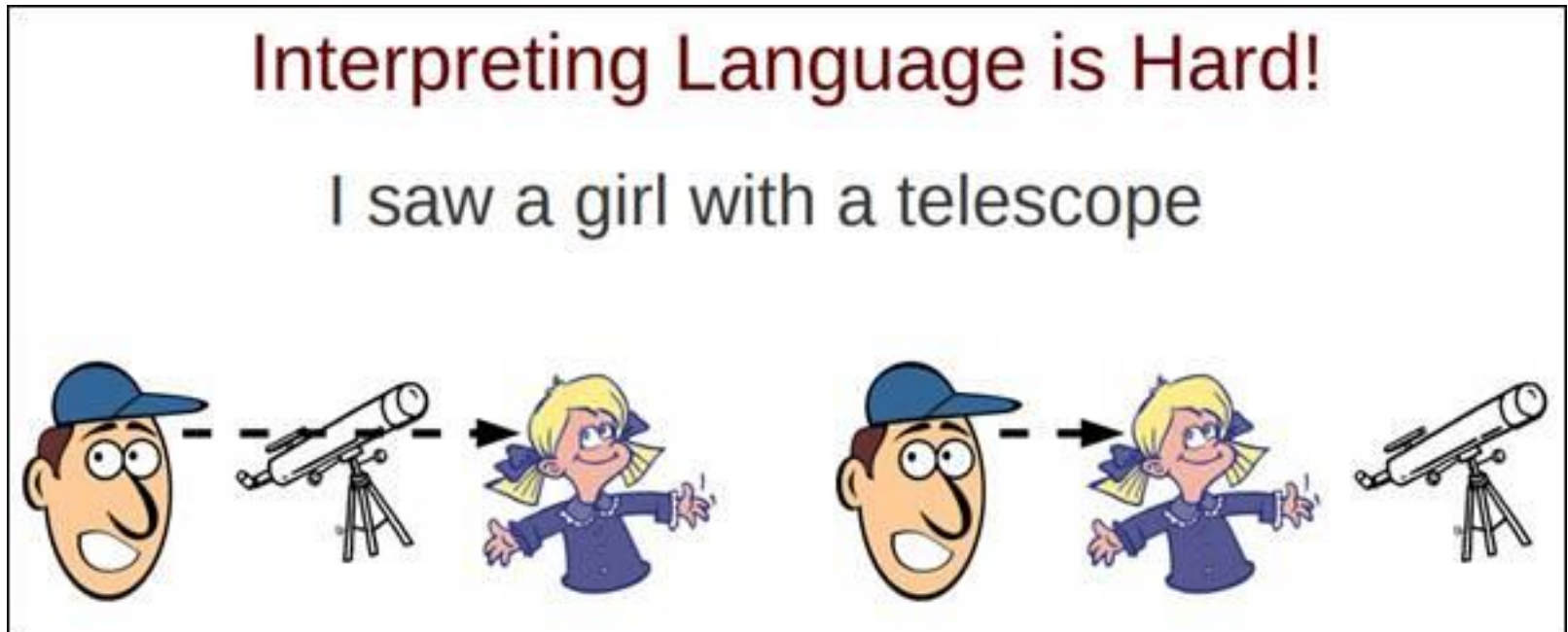


S => NP VP PP  
PP => P NP  
VP => V NP  
VP => V PP  
NP => D N  
...

# Arbre Syntaxique - Parse Tree

## Ambiguïté

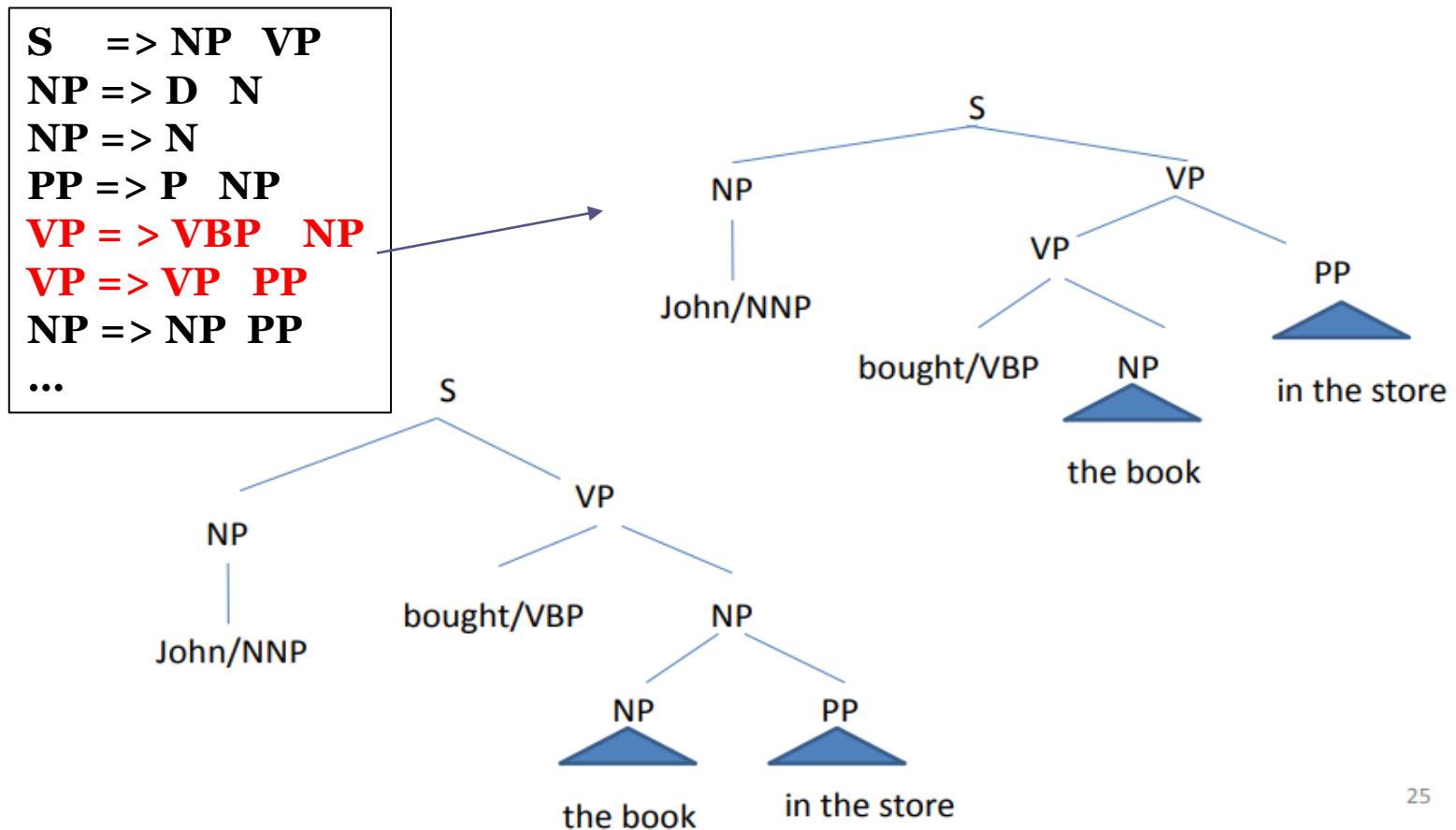
- **Ambiguïté structurelle:** une phrase peut avoir plus d'un arbre syntaxique.
- Ex : *I saw a girl with a telescope* | *I saw a girl with a telescope*



# Arbre Syntaxique - Parse Tree

## Ambiguïté

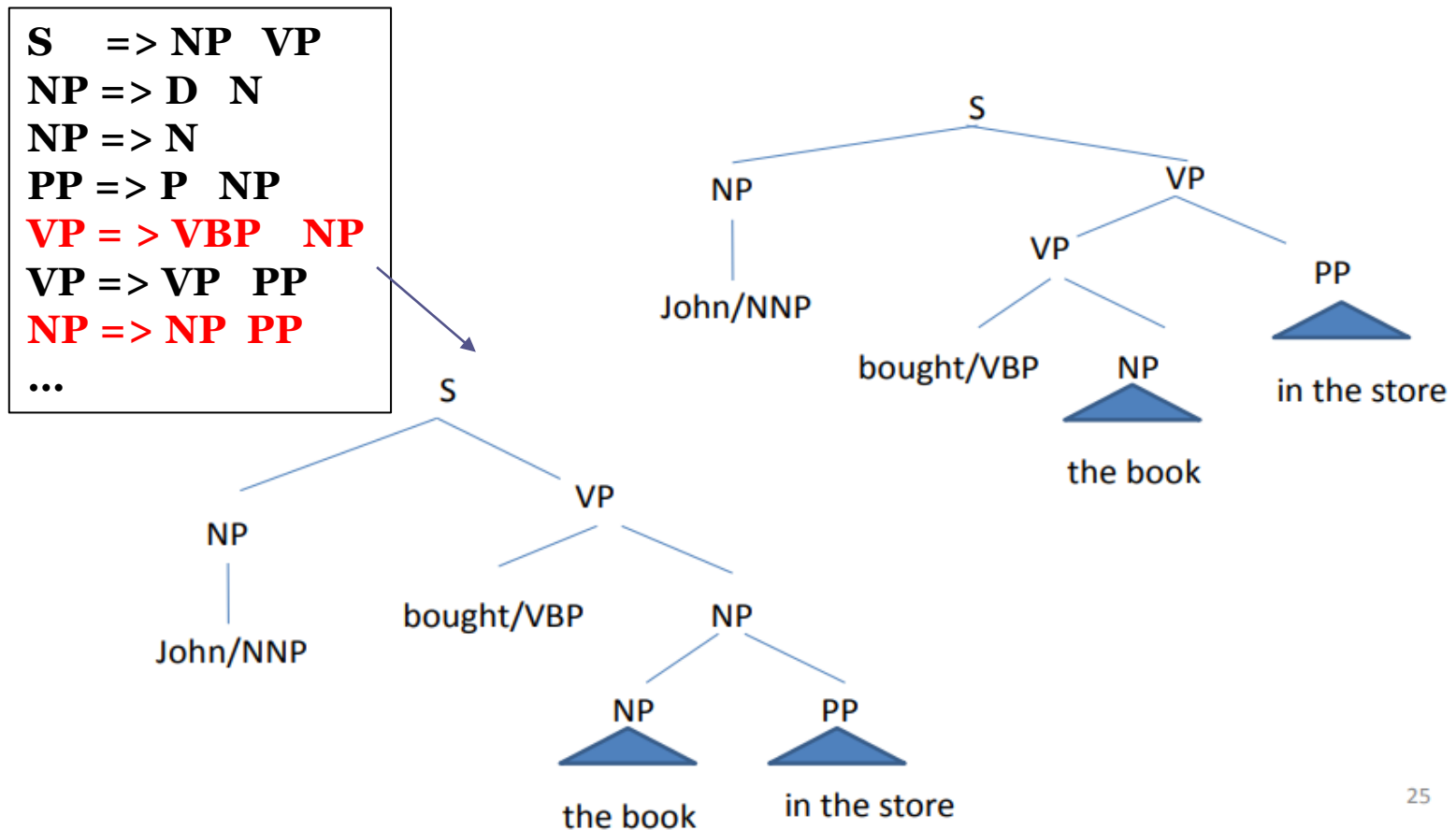
- **Ambiguïté structurelle:** une phrase peut avoir **plus d'un arbre syntaxique**.
- Ex : *John bought the book in the store* | *John bought the book in the store*



# Arbre Syntaxique - Parse Tree

## Ambiguïté

- **Ambiguïté structurelle:** une phrase peut avoir **plus d'un arbre syntaxique**.
- Ex : *John bought the book in the store* | *John bought the book in the store*



# Arbre Syntaxique - Parse Tree

## Ambiguïté

- A différencier avec : Ambiguïté **lexicale** : la phrase contient un **mot** qui a plus d'un sens.

- je regarde **l'homme** avec les jumelles



je regarde l'homme en utilisant des jumelles

- je regarde **l'homme avec les jumelles**

– « je regarde l'homme qui a des jumelles »



- je regarde **l'homme** avec les **jumelles**

– « je regarde l'homme qui est accompagné de sœurs jumelles ».

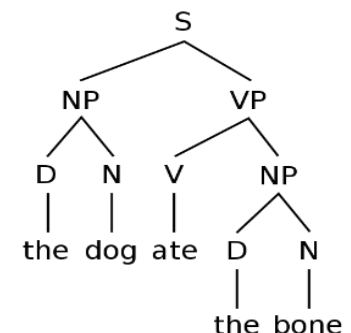




# Arbre Syntaxique - Parse Tree

## Inférence et évaluation de l'arbre syntaxique, parsing

- Il existe plusieurs **algorithmes** pour inférer et extraire l'arbre(s) syntaxique(s) d'une phrase (algorithme de parsing).
- Ex : Cocke-Younger-KAsami (CYK)
- Deux **stratégies** pour extraire l'arbre syntaxique : Top Down et Bottom Up.
- Calcul de scores et de **probabilités** pour résoudre les ambiguïtés.
- Plusieurs méthodes d'**évaluation** du parsing résultant.
- Ex : Precision, Recall, F1-Score.



# Arbre Syntaxique - Parse Tree

## Algorithme Cocke-Younger-Kasami (CYK)

- L'approche de **programmation dynamique** standard pour l'analyse syntaxique.
- L'algorithme CKY ci-dessous est conçu pour gérer efficacement ambiguïtés structurelles.
- L'algorithme **Vanilla CYK** renvoie une représentation efficace de l'ensemble des arbres d'analyse d'une phrase, mais ne nous dit pas quel arbre d'analyse est le bon.
- Pour cela, nous devons augmenter CKY avec des scores de probabilités pour chaque constituant possible : **Probabilistic CYK**.
- L'algorithme CYK nécessite que les grammaires soient d'abord en forme normale de Chomsky (**CNF**).

# Arbre Syntaxique - Parse Tree

## Forme Normale de Chomsky

- La forme normale de Chomsky (CNF) impose des contraintes sur la forme de  $\beta$ , qui doit être une des 2 possibilités suivantes :

$$A \rightarrow \beta$$

- deux symboles non-terminaux - exemple :  $A \rightarrow B C$
  - un symbole terminal - exemple :  $A \rightarrow a$
- La forme normale de Chomsky implique des **arbres syntaxiques binaires** (jusqu'aux règles produisant un seul terminal).
- Elle va simplifier le problème de trouver l'arbre syntaxique d'une phrase ▸ on a seulement à explorer l'espace des arbres binaires.
- On peut facilement **convertir** une grammaire hors contexte en CNF.

# Arbre Syntaxique - Parse Tree

## 1 - Convertir une CFG en CNF.

- pour les règles **ne respectant pas** la forme normale de Chomsky :
- remplace les terminaux (mots) par des non-terminaux (constituants) et ajouter des règles  $A \rightarrow a$  pour compenser :

$$\begin{array}{ccc} INF-VP \rightarrow to VP & \longrightarrow & \begin{array}{l} INF-VP \rightarrow TO VP \\ TO \rightarrow to \end{array} \end{array}$$

- remplace les règles avec un seul non-terminal par ce qui peut être produit par ce non-terminal :

$$\begin{array}{ccc} \begin{array}{l} S \rightarrow VP \\ VP \rightarrow Verb NP PP \end{array} & \longrightarrow & \begin{array}{l} S \rightarrow Verb NP PP \\ VP \rightarrow Verb NP PP \end{array} \end{array}$$

- remplace chaque paire de non-terminaux par un seul symbole non-terminal (et sa règle correspondante), jusqu'à ce que toutes les règles soient binaires:

$$\begin{array}{ccc} S \rightarrow Aux NP VP & \longrightarrow & \begin{array}{l} S \rightarrow XI VP \\ XI \rightarrow Aux NP \end{array} \end{array}$$

# Arbre Syntaxique - Parse Tree

## Algorithme Cocke-Younger-Kasami (CYK)

- Exemple :
  - Texte : « *The flight includes a meal* »
  - **Grammaire** - FNC:

**S** => **NP VP**

**NP** => **D N**

**VP** => **V NP**

**V** => **includes**

**D** => **the | a**

**N** => **meal | flight**

# Algorithme Cocke-Younger-Kasami (CYK)

## 2 - CKY Recognition

- Exemple : **0** *The* **1** *flight* **2** *includes* **3** *a* **4** *meal* **5**

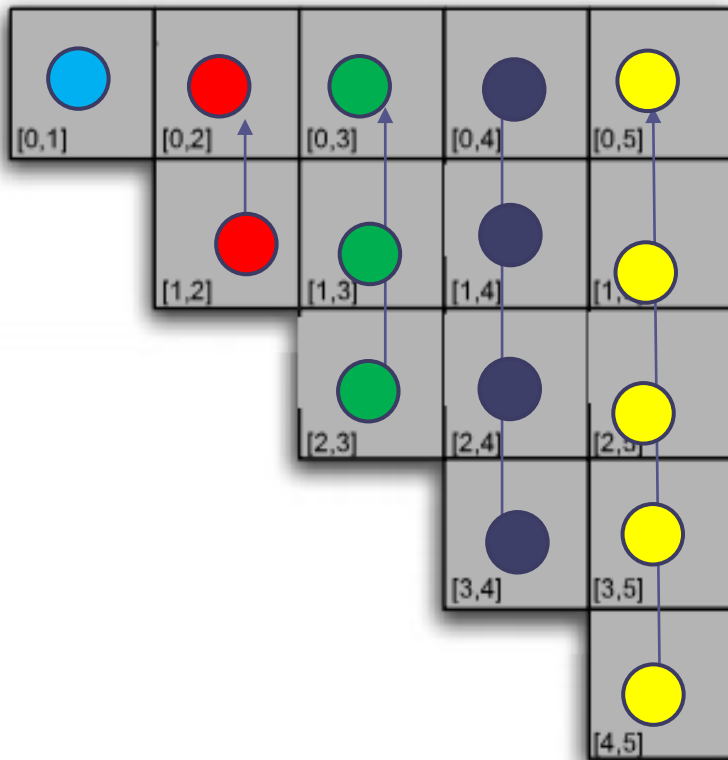
*The   flight   includes   a   meal*

[0,1]	[0,2]	[0,3]	[0,4]	[0,5]
	[1,2]	[1,3]	[1,4]	[1,5]
		[2,3]	[2,4]	[2,5]
			[3,4]	[3,5]
				[4,5]

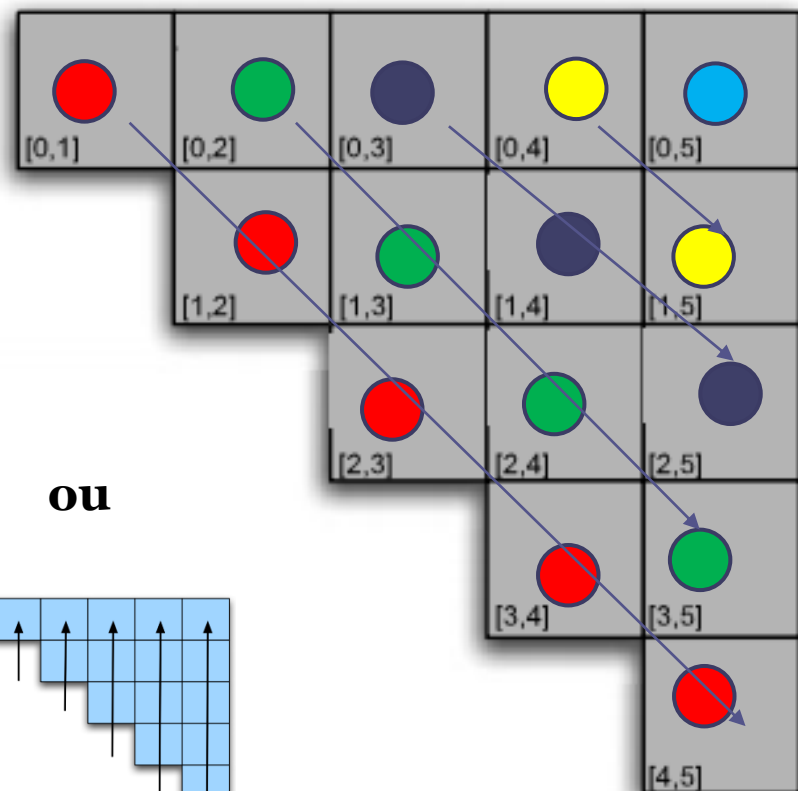
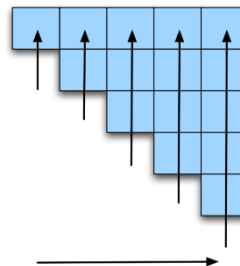
idée : on construit l'arbre couvrant les **mots** entre les positions **i** et **j** en cherchant deux sous arbres couvrant les mots de **i à k** et de **k à j** (pour un certain k).

## Algorithme Cocke-Younger-Kasami (CYK)

- Exemple : **0** The **1** flight **2** includes **3** a **4** meal **5**



ou



# Algorithme Cocke-Younger-Kasami (CYK)

## 3 – CKY Parsing

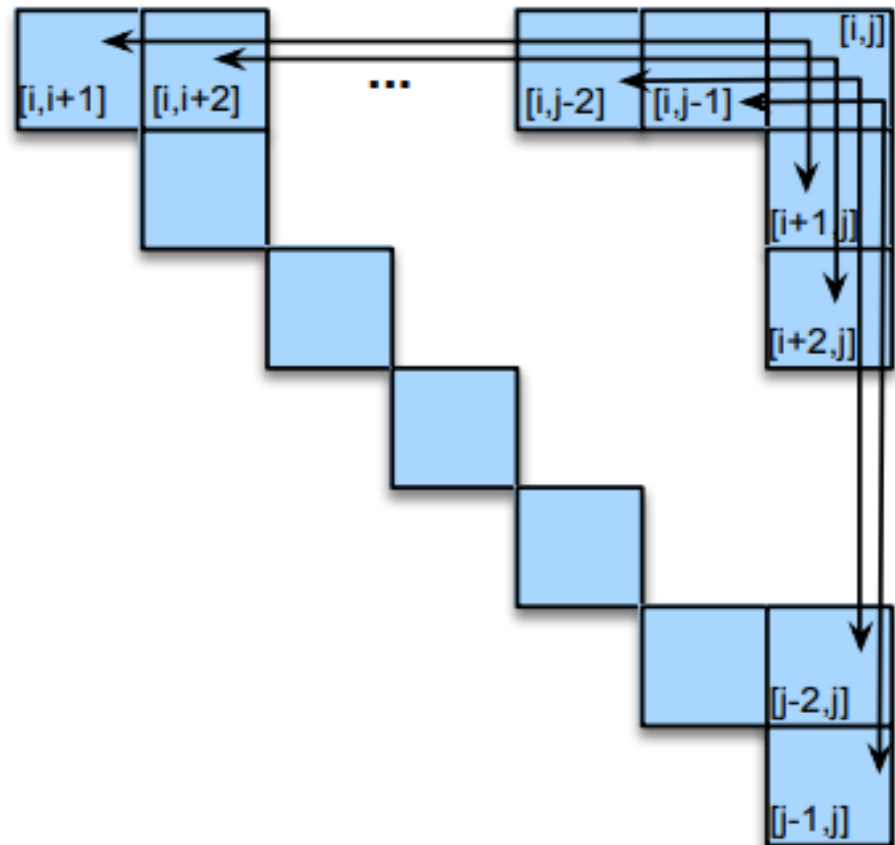
- Exemple : **o** The **1** flight **2** includes **3** a **4** meal **5**

Générer l'arbre(s) syntaxique(s) :

**Cellules comparées** pour une cellule donnée :

Ex : pour une case  $[i, j] \Rightarrow$  on compare les case  $[i, i+1]$  et  $[i+1, j]$

Ex : pour une case  $[i, j] \Rightarrow$  on compare les case  $[i, i+2]$  et  $[i+2, j]$





# Algorithme Cocke-Younger-Kasami (CYK)

## 3 – CKY Parsing

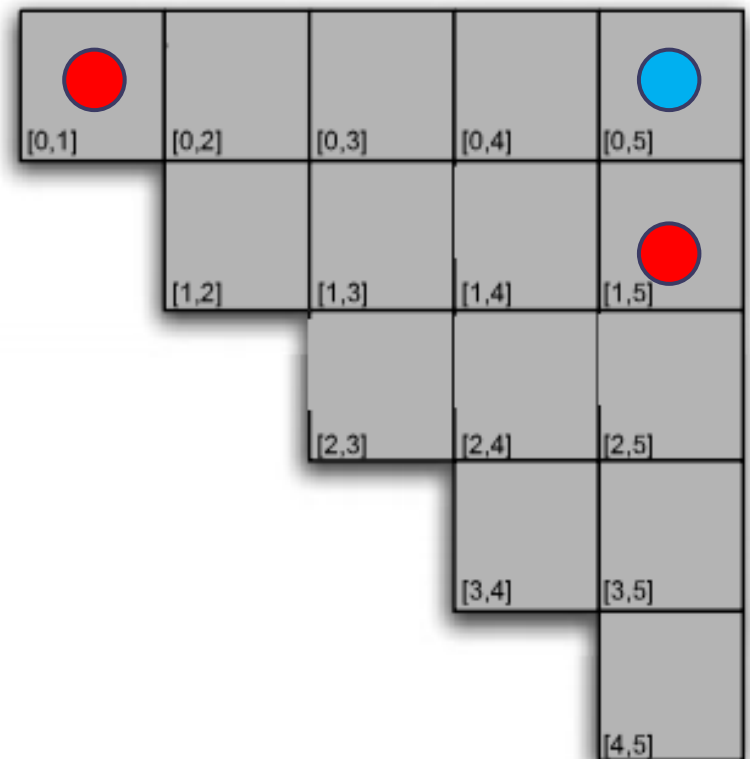
- Exemple : **o** *The* **1** *flight* **2** *includes* **3** *a* **4** *meal* **5**

Générer l'arbre(s) syntaxique(s) :

**Cellules comparées** pour une cellule donnée :

Ex : pour une case  $[i, j] \Rightarrow$  on compare les case  $[i, i+1]$  et  $[i+1, j]$

Ex : pour une case  $[i, j] \Rightarrow$  on compare les case  $[i, i+2]$  et  $[i+2, j]$



# Algorithme Cocke-Younger-Kasami (CYK)

## 3 – CKY Parsing

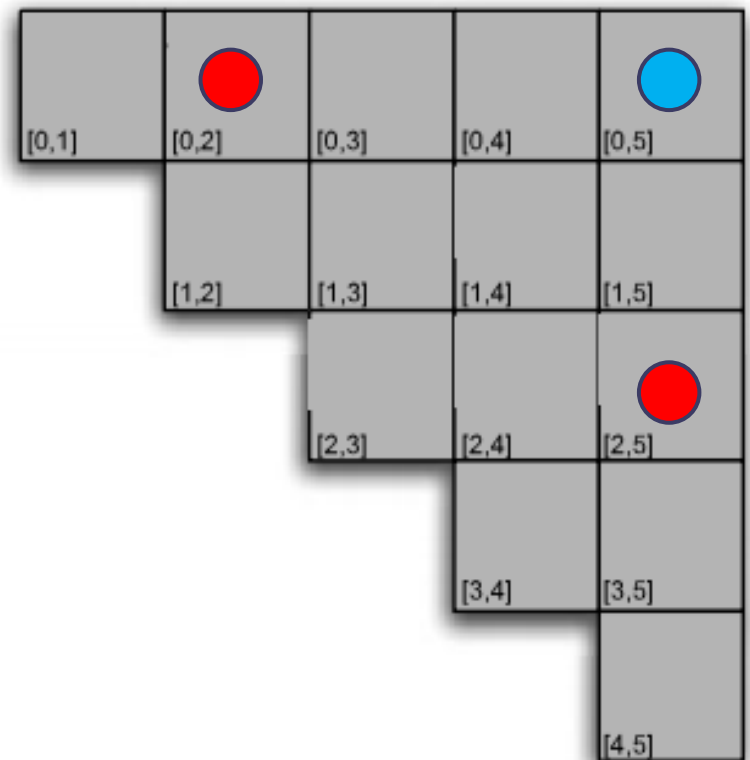
- Exemple : **o** The **1** flight **2** includes **3** a **4** meal **5**

Générer l'arbre(s) syntaxique(s) :

**Cellules comparées** pour une cellule donnée :

Ex : pour une case  $[i, j] \Rightarrow$  on compare les case  $[i, i+1]$  et  $[i+1, j]$

Ex : pour une case  $[i, j] \Rightarrow$  on compare les case  $[i, i+2]$  et  $[i+2, j]$



# Algorithme Cocke-Younger-Kasami (CYK)

## 3 – CKY Parsing

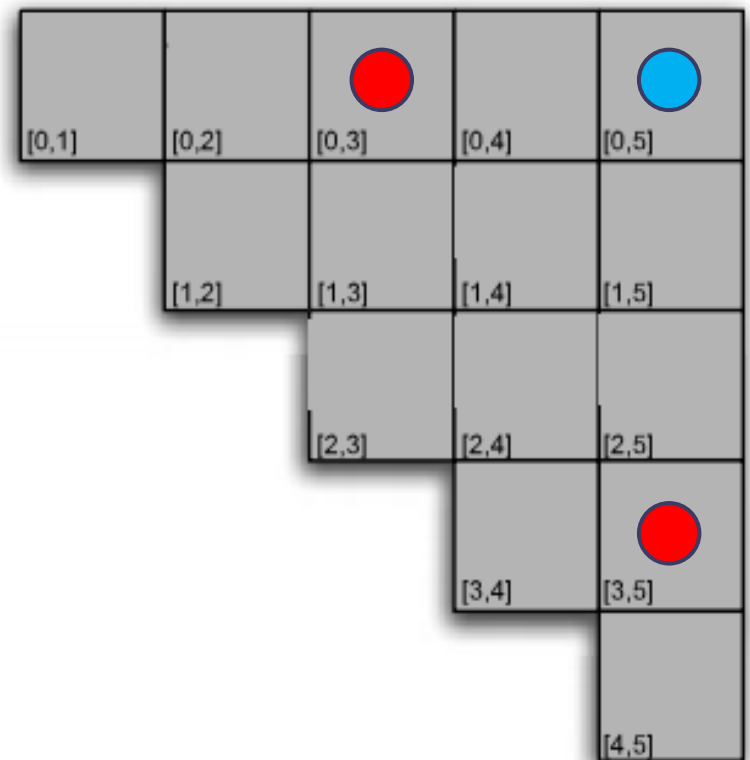
- Exemple : **o** *The* **1** *flight* **2** *includes* **3** *a* **4** *meal* **5**

Générer l'arbre(s) syntaxique(s) :

**Cellules comparées** pour une cellule donnée :

Ex : pour une case  $[i, j] \Rightarrow$  on compare les case  $[i, i+1]$  et  $[i+1, j]$

Ex : pour une case  $[i, j] \Rightarrow$  on compare les case  $[i, i+2]$  et  $[i+2, j]$



# Algorithme Cocke-Younger-Kasami (CYK)

## 3 – CKY Parsing

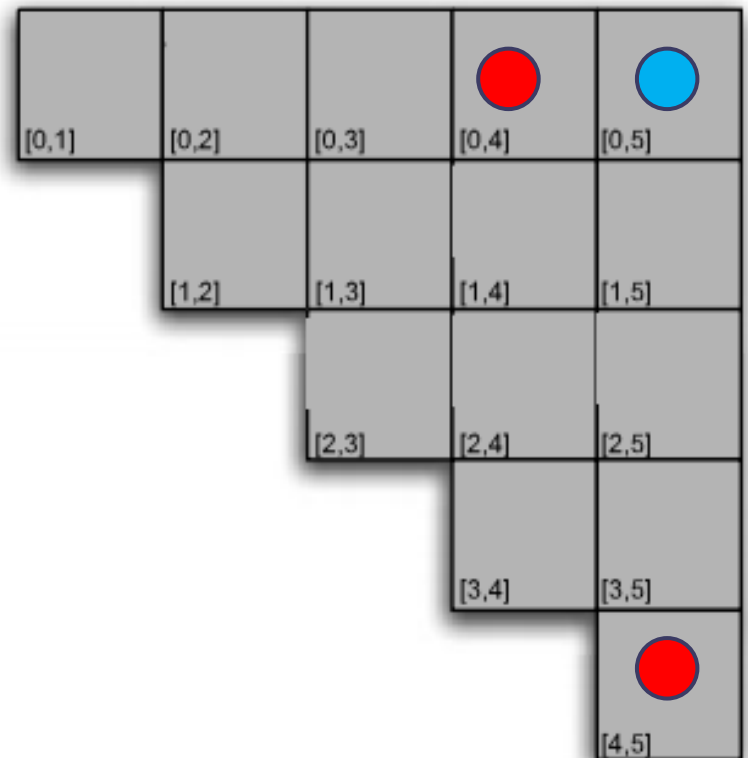
- Exemple : **o** The **1** flight **2** includes **3** a **4** meal **5**

Générer l'arbre(s) syntaxique(s) :

**Cellules comparées** pour une cellule donnée :

Ex : pour une case  $[i, j] \Rightarrow$  on compare les case  $[i, i+1]$  et  $[i+1, j]$

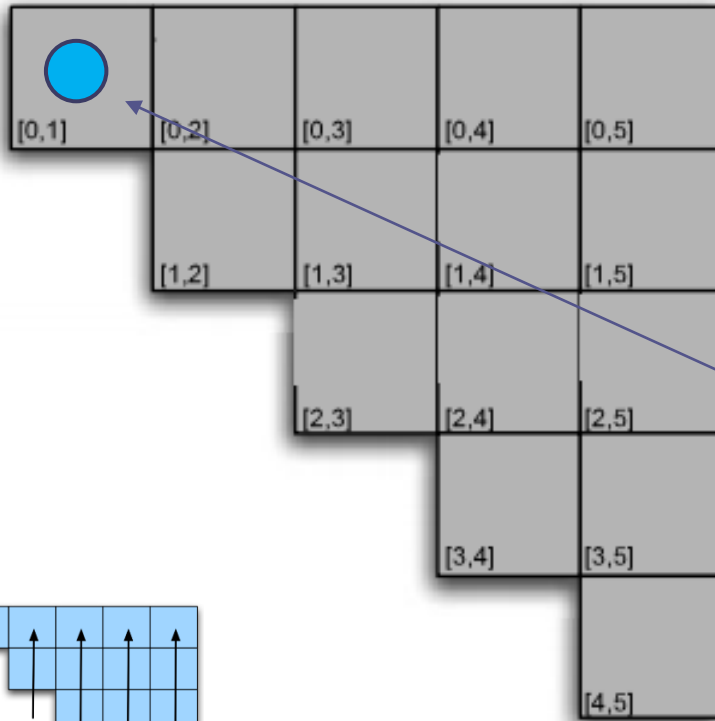
Ex : pour une case  $[i, j] \Rightarrow$  on compare les case  $[i, i+2]$  et  $[i+2, j]$



## Algorithme Cocke-Younger-Kasami (CYK)

- Exemple : **0** *The* **1** *flight* **2** *includes* **3** *a* **4** *meal* **5**

*The flight includes a meal*



**Grammaire - FNC:**

**S**  $\Rightarrow$  **NP VP**

**NP**  $\Rightarrow$  **D N**

**VP**  $\Rightarrow$  **V NP**

**V**  $\Rightarrow$  **includes**

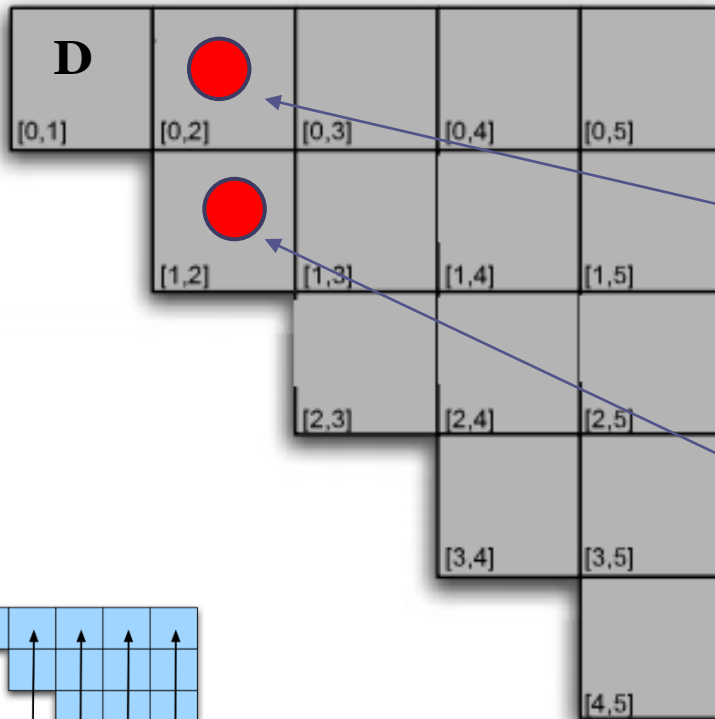
**D**  $\Rightarrow$  **the | a**

**N**  $\Rightarrow$  **meal | flight**

## Algorithme Cocke-Younger-Kasami (CYK)

- Exemple : **0** *The* **1** *flight* **2** *includes* **3** *a* **4** *meal* **5**

*The flight includes a meal*



**Grammaire - FNC:**

**S**  $\Rightarrow$  **NP VP**

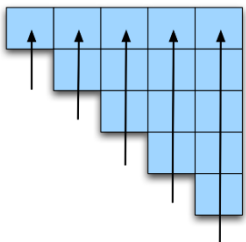
**NP**  $\Rightarrow$  **D N**

**VP**  $\Rightarrow$  **V NP**

**V**  $\Rightarrow$  **includes**

**D**  $\Rightarrow$  **the | a**




**N**  $\Rightarrow$  **meal | flight**



## Algorithme Cocke-Younger-Kasami (CYK)

- Exemple : **0** *The* **1** *flight* **2** *includes* **3** *a* **4** *meal* **5**

*The flight includes a meal*

<b>D</b> [0,1]	<b>NP</b> [0,2]	 [0,3]	[0,4]	[0,5]
	<b>N</b> [1,2]	 [1,3]	[1,4]	[1,5]
		 [2,3]	[2,4]	[2,5]
			[3,4]	[3,5]
				[4,5]

**Grammaire - FNC:**

**S** => **NP VP**

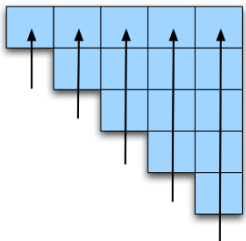
**NP** => **D N**

**VP** => **V NP**

**V** => **includes**

**D** => **the | a**

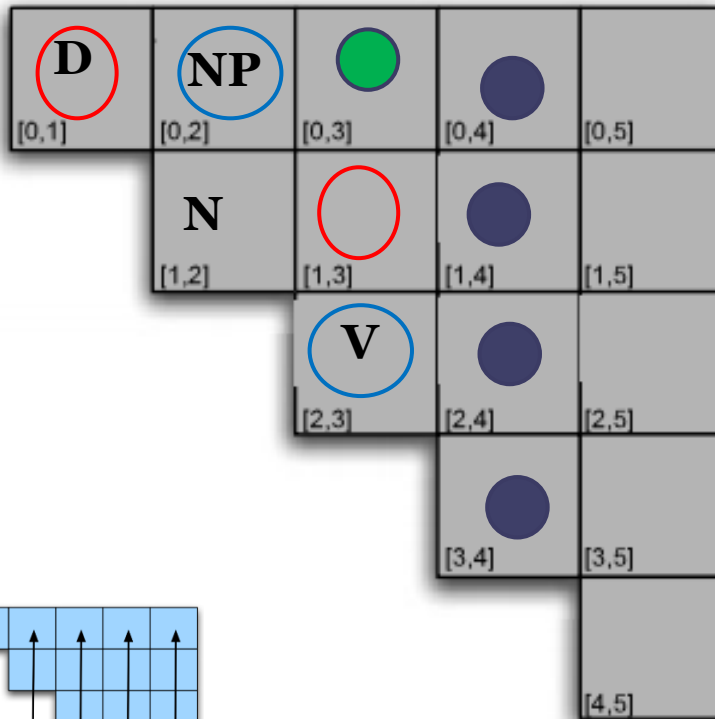
**N** => **meal | flight**



## Algorithme Cocke-Younger-Kasami (CYK)

- Exemple : **0** *The* **1** *flight* **2** *includes* **3** *a* **4** *meal* **5**

*The flight includes a meal*



**Grammaire - FNC:**

**S**  $\Rightarrow$  **NP VP**

**NP**  $\Rightarrow$  **D N**

**VP**  $\Rightarrow$  **V NP**

**V**  $\Rightarrow$  **includes**

**D**  $\Rightarrow$  **the | a**

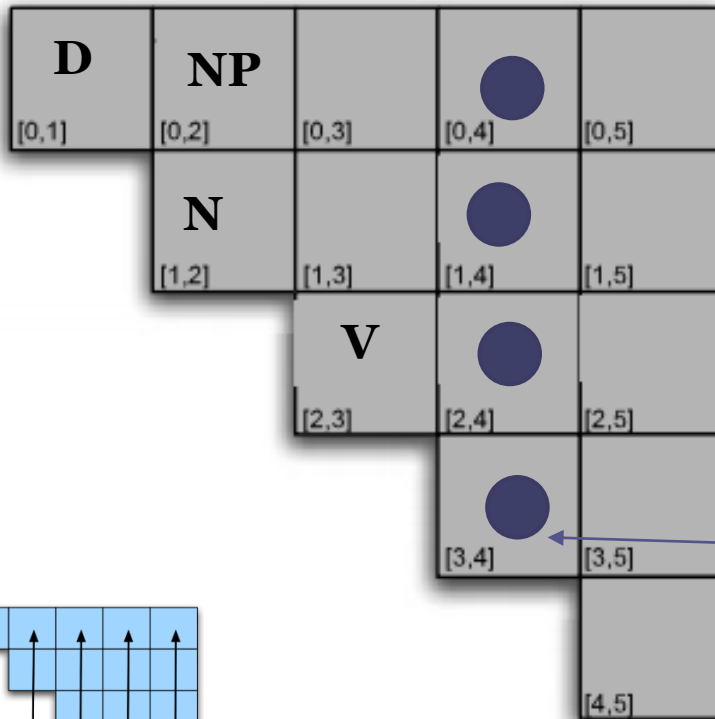
**N**  $\Rightarrow$  **meal | flight**



## Algorithmme Cocke-Younger-Kasami (CYK)

- Exemple : **0** *The* **1** *flight* **2** *includes* **3** *a* **4** *meal* **5**

*The flight includes a meal*



## Grammaire - FNC:

**S => NP VP**

**NP => D N**

**VP => V NP**

**V => includes**

- **D** => the | a

**N => meal | flight**

## Algorithme Cocke-Younger-Kasami (CYK)

- Exemple : **0** *The* **1** *flight* **2** *includes* **3** *a* **4** *meal* **5**

*The flight includes a meal*

<b>D</b> [0,1]	<b>NP</b> [0,2]			●
	<b>N</b> [1,2]			●
		<b>V</b> [2,3]		●
			<b>D</b> [3,4]	●
				● [4,5]

**Grammaire - FNC:**

**S**  $\Rightarrow$  **NP VP**

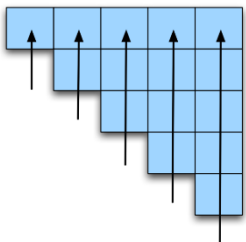
**NP**  $\Rightarrow$  **D N**

**VP**  $\Rightarrow$  **V NP**

**V**  $\Rightarrow$  **includes**

**D**  $\Rightarrow$  **the | a**

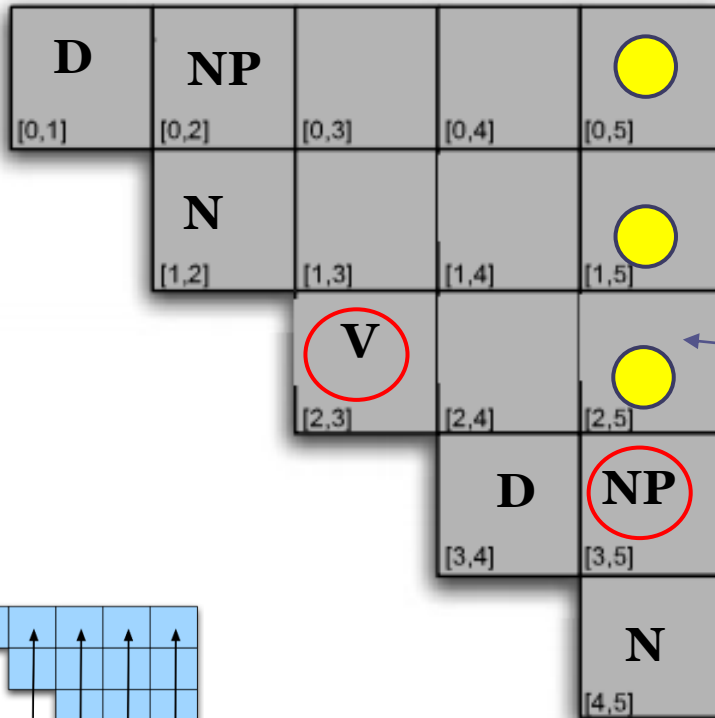
**N**  $\Rightarrow$  **meal | flight**



## Algorithme Cocke-Younger-Kasami (CYK)

- Exemple : **0** *The* **1** *flight* **2** *includes* **3** *a* **4** *meal* **5**

*The flight includes a meal*



**Grammaire - FNC:**

**S**  $\Rightarrow$  NP VP

**NP**  $\Rightarrow$  D N

**VP**  $\Rightarrow$  V NP

**V**  $\Rightarrow$  includes

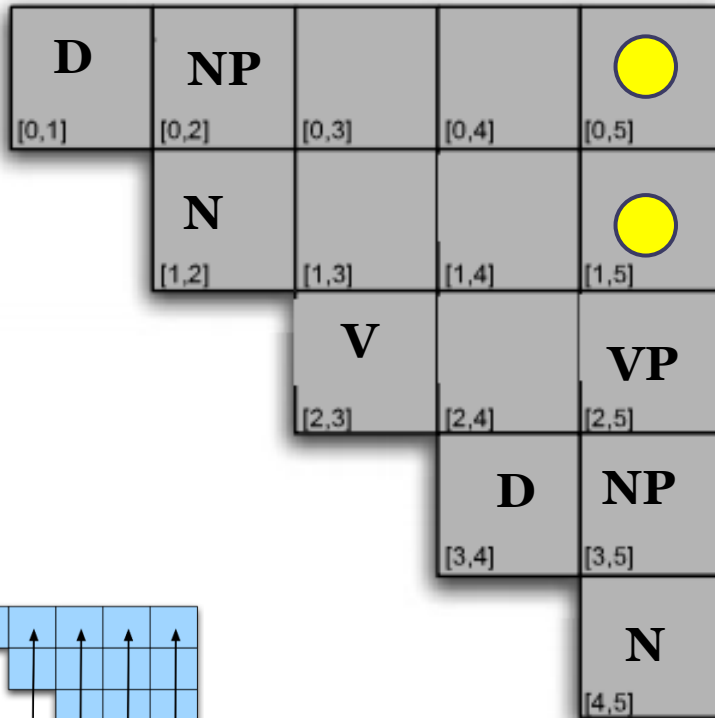
**D**  $\Rightarrow$  the | a

**N**  $\Rightarrow$  meal | flight

## Algorithme Cocke-Younger-Kasami (CYK)

- Exemple : **0** *The* **1** *flight* **2** *includes* **3** *a* **4** *meal* **5**

*The flight includes a meal*



**Grammaire - FNC:**

**S**  $\Rightarrow$  **NP VP**

**NP**  $\Rightarrow$  **D N**

**VP**  $\Rightarrow$  **V NP**

**V**  $\Rightarrow$  **includes**

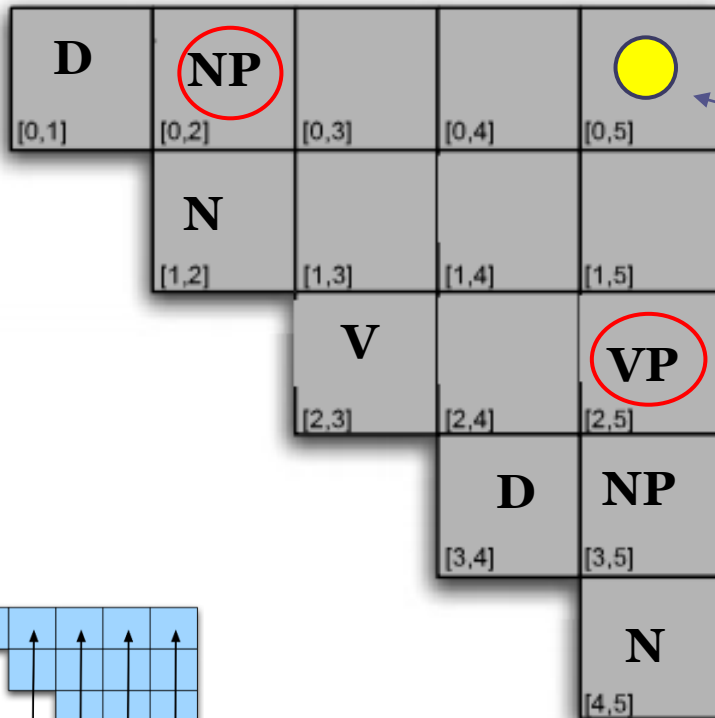
**D**  $\Rightarrow$  **the | a**

**N**  $\Rightarrow$  **meal | flight**

## Algorithme Cocke-Younger-Kasami (CYK)

- Exemple : **0** *The* **1** *flight* **2** *includes* **3** *a* **4** *meal* **5**

*The flight includes a meal*



Grammaire - FNC:

**S**  $\Rightarrow$  NP VP

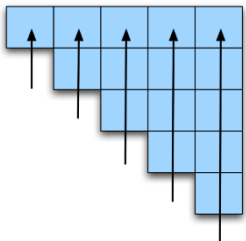
**NP**  $\Rightarrow$  D N

**VP**  $\Rightarrow$  V NP

**V**  $\Rightarrow$  includes

**D**  $\Rightarrow$  the | a

**N**  $\Rightarrow$  meal | flight

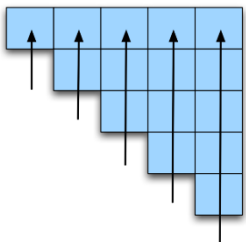


## Algorithme Cocke-Younger-Kasami (CYK)

- Exemple : **0** *The* **1** *flight* **2** *includes* **3** *a* **4** *meal* **5**

*The flight includes a meal*

<b>D</b> [0,1]	<b>NP</b> [0,2]			<b>S</b> [0,5]
	<b>N</b> [1,2]			
		<b>V</b> [2,3]		<b>VP</b> [2,5]
			<b>D</b> [3,4]	<b>NP</b> [3,5]
				<b>N</b> [4,5]



**Grammaire - FNC:**

**S** => **NP VP**

**NP** => **D N**

**VP** => **V NP**

**V** => **includes**

**D** => **the | a**

**N** => **meal | flight**

# Algorithme Cocke-Younger-Kasami (CYK)

## 3 – CKY Parsing

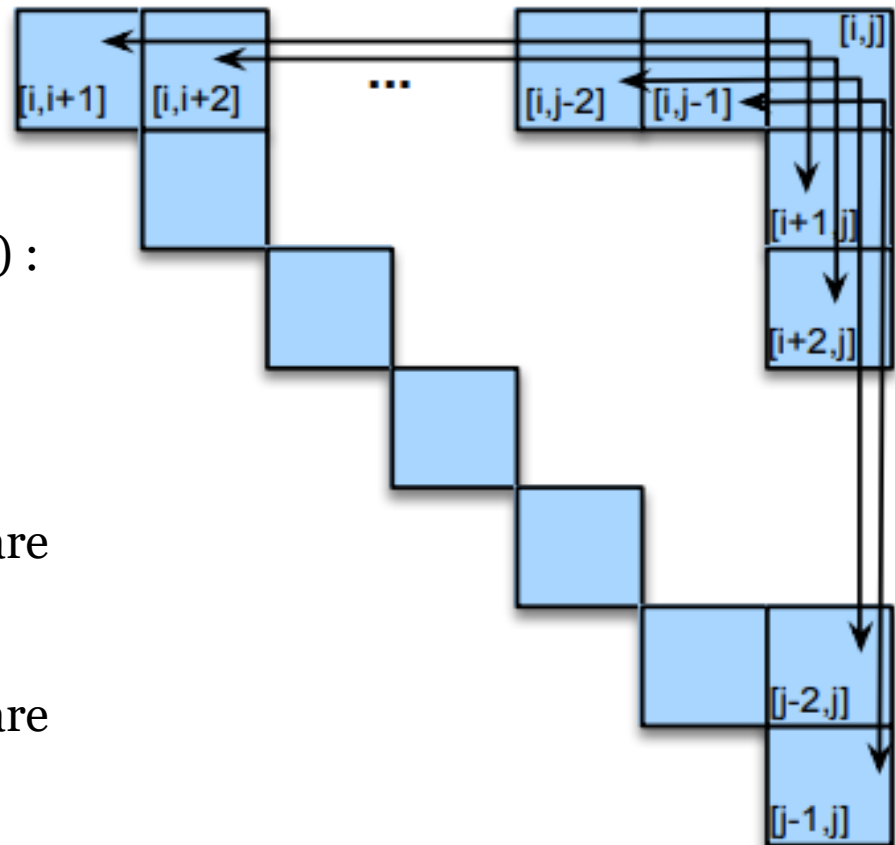
- Exemple : **o** *The* **1** *flight* **2** *includes* **3** *a* **4** *meal* **5**

**Générer l'arbre(s) syntaxique(s) :**

Cellules comparées pour une cellule donnée :

Ex : pour une case  $[i, j] \Rightarrow$  on compare les case  $[i, i+1]$  et  $[i+1, j]$

Ex : pour une case  $[i, j] \Rightarrow$  on compare les case  $[i, i+2]$  et  $[i+2, j]$

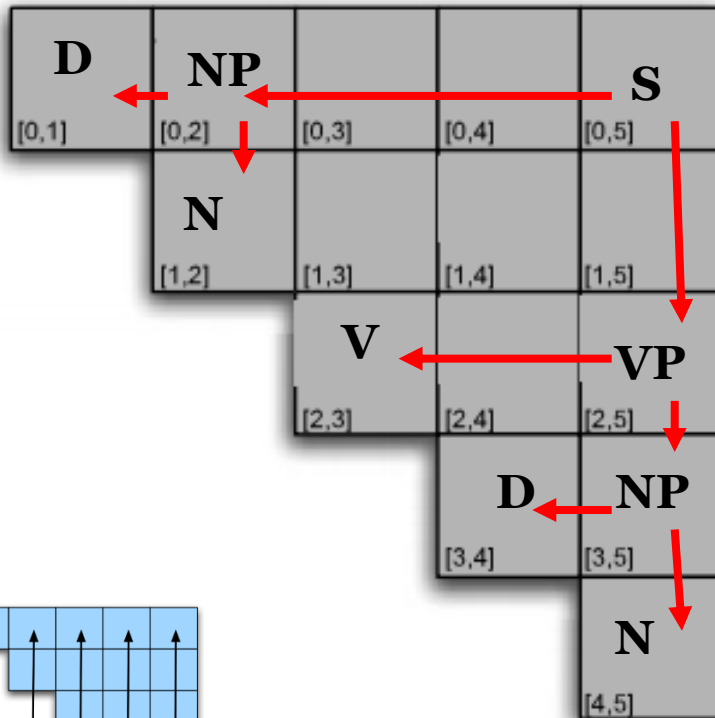


# Algorithme Cocke-Younger-Kasami (CYK)

## 3 – CKY Parsing

- Exemple : **0** *The* **1** *flight* **2** *includes* **3** *a* **4** *meal* **5**

*The flight includes a meal*



**S** => **NP VP**

**NP** => **D N**

**VP** => **V NP**

**V** => **includes**

**D** => **the | a**

**N** => **meal | flight**

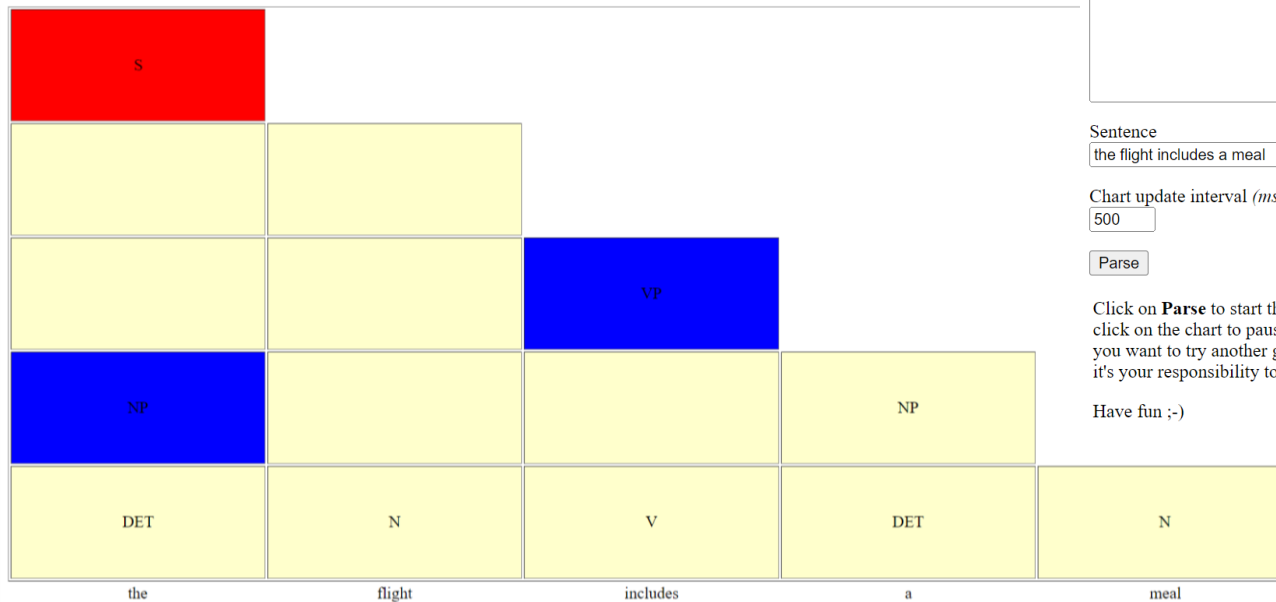
Un seul Arbre syntaxique



# Algorithme Cocke-Younger-Kasami (CYK)

## 3 – CKY Parsing

- Exemple : **0** *The* **1** *flight* **2** *includes* **3** *a* **4** *meal* **5**
- Démo - Animation** : <https://martinlaz.github.io/demos/cky.html>



Grammar (needs to be in CNF)

```
S -> NP VP
NP -> DET N
VP -> V NP
DET -> the
DET -> a
N -> meal
N -> flight
V -> includes
```

Sentence

the flight includes a meal

Chart update interval (ms)

500

Parse

Click on **Parse** to start the CKY parsing animation. Once the animation is running, you can click on the chart to pause/resume it. When it's done, click again on the chart to restart it. If you want to try another grammar/sentence, press Escape (or just refresh the page). And, yes, it's your responsibility to put the grammar in Chomsky normal form.

Have fun ;-)

# Arbre Syntaxique - Parse Tree

## Algorithme Cocke-Younger-Kasami (CYK)

- Exemple 2 : « *Book the flight through Houston* »

### 1 - Convertir une CFG en CNF.

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

# Algorithme Cocke-Younger-Kasami (CYK)

## 2 - CKY Recognition

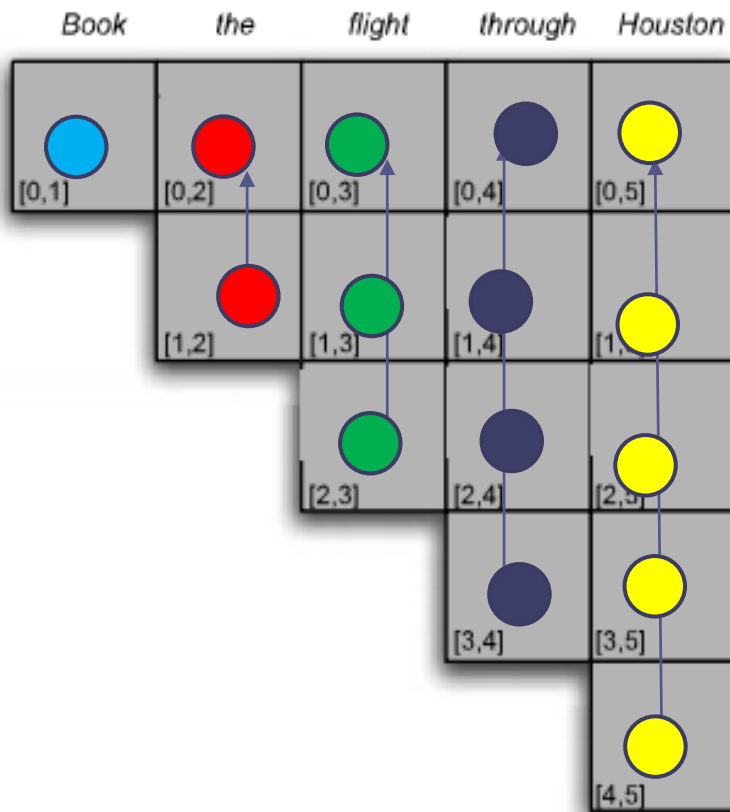
- Exemple : **0** *Book* **1** *the* **2** *flight* **3** *through* **4** *Houston* **5**

	<i>Book</i>	<i>the</i>	<i>flight</i>	<i>through</i>	<i>Houston</i>
	[0,1]	[0,2]	[0,3]	[0,4]	[0,5]
		[1,2]	[1,3]	[1,4]	[1,5]
			[2,3]	[2,4]	[2,5]
				[3,4]	[3,5]
					[4,5]

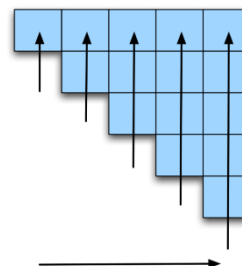
**idée** : on construit l'arbre couvrant les **mots** entre les positions **i** et **j** en cherchant deux sous arbres couvrant les mots de **i à k** et de **k à j** (pour un certain k).

## Algorithme Cocke-Younger-Kasami (CYK)

- Exemple : **0** *Book* **1** *the* **2** *flight* **3** *through* **4** *Houston* **5**



ou



# Algorithme Cocke-Younger-Kasami (CYK)

## 3 – CKY Parsing

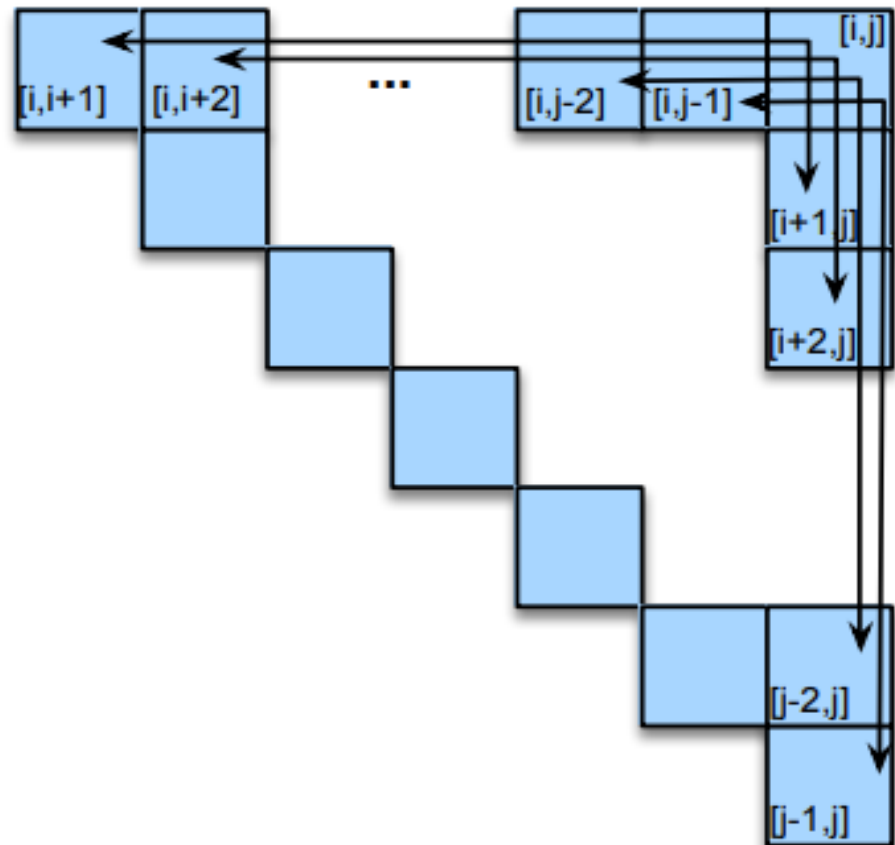
- Exemple : **0** *Book* **1** *the* **2** *flight* **3** *through* **4** *Houston* **5**

Générer l'arbre(s) syntaxique(s) :

**Cellules comparées** pour une cellule donnée :

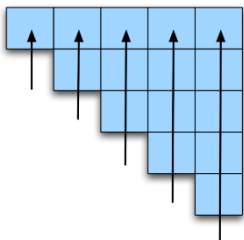
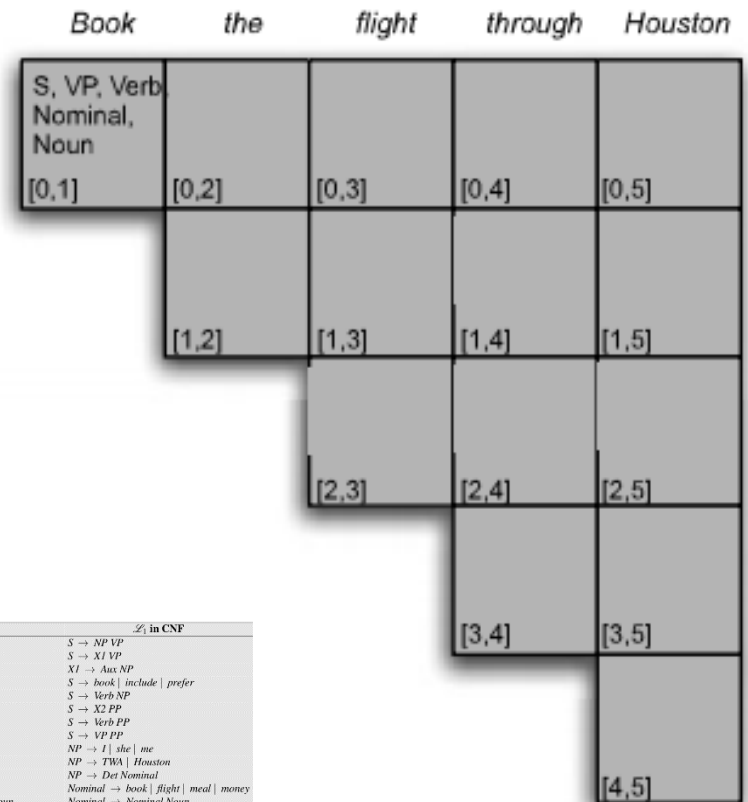
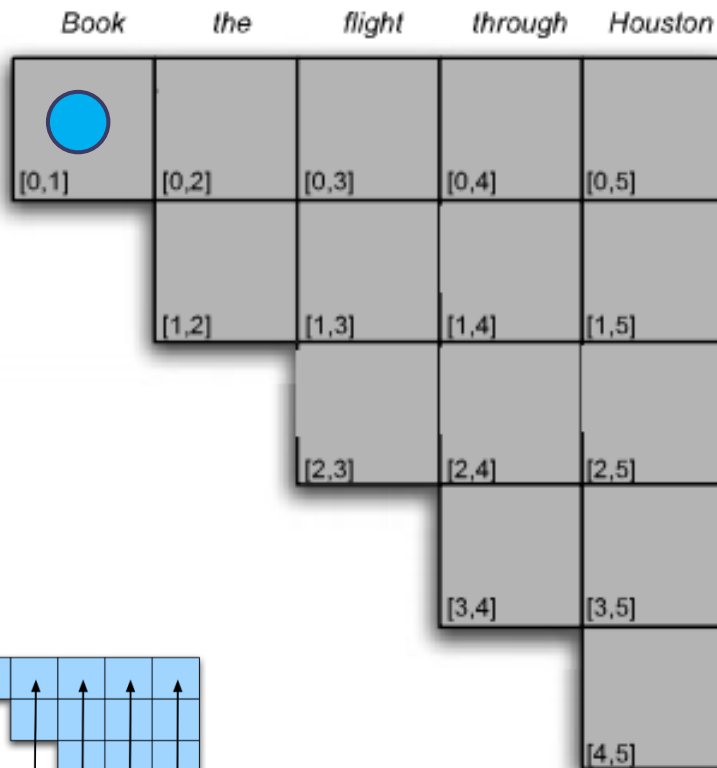
Ex : pour une case  $[i, j] \Rightarrow$  on compare les case  $[i, i+1]$  et  $[i+1, j]$

Ex : pour une case  $[i, j] \Rightarrow$  on compare les case  $[i, i+2]$  et  $[i+2, j]$



# Algorithme Cocke-Younger-Kasami (CYK)

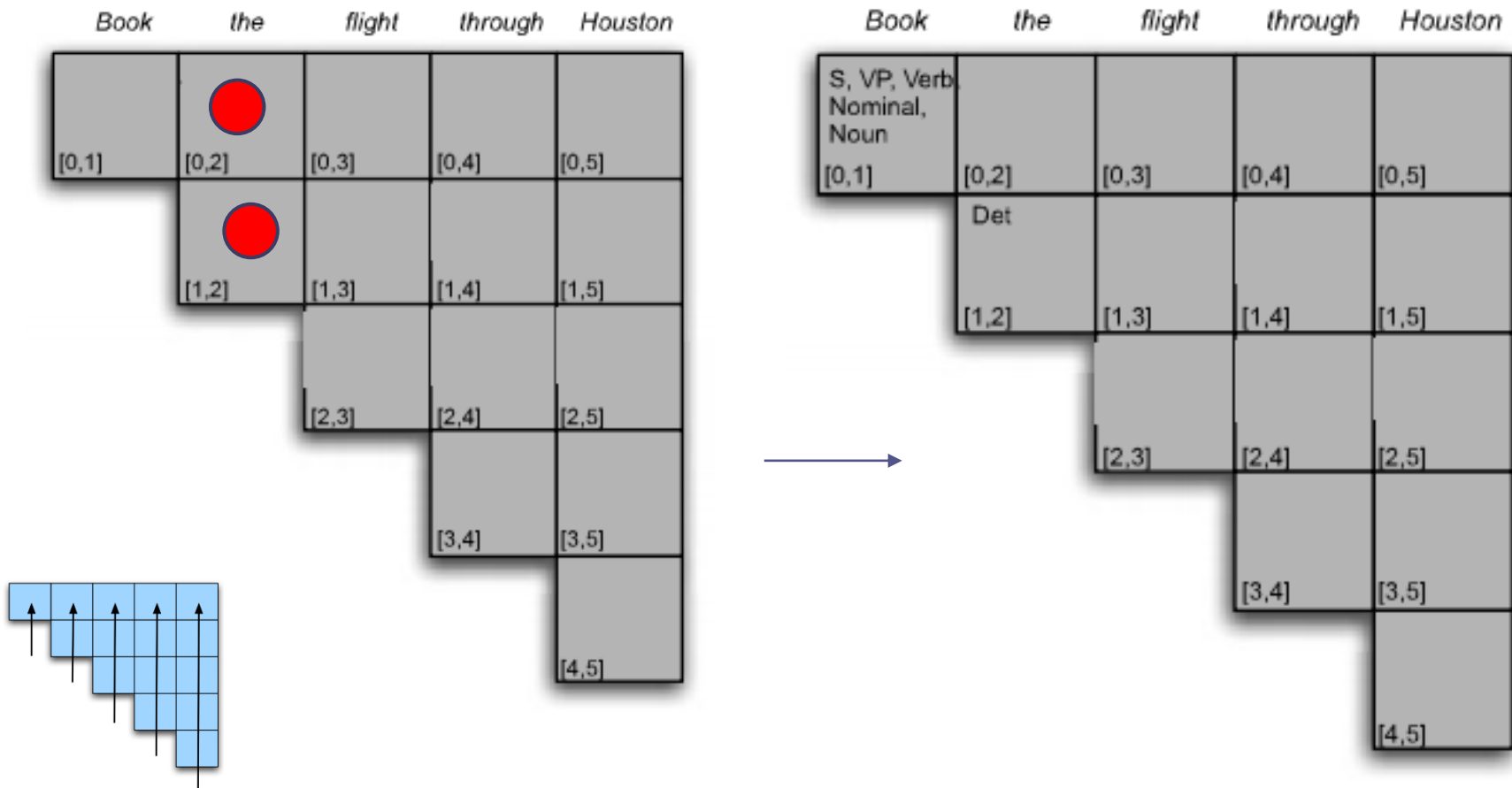
- Exemple : **0** Book **1** the **2** flight **3** through **4** Houston **5**



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

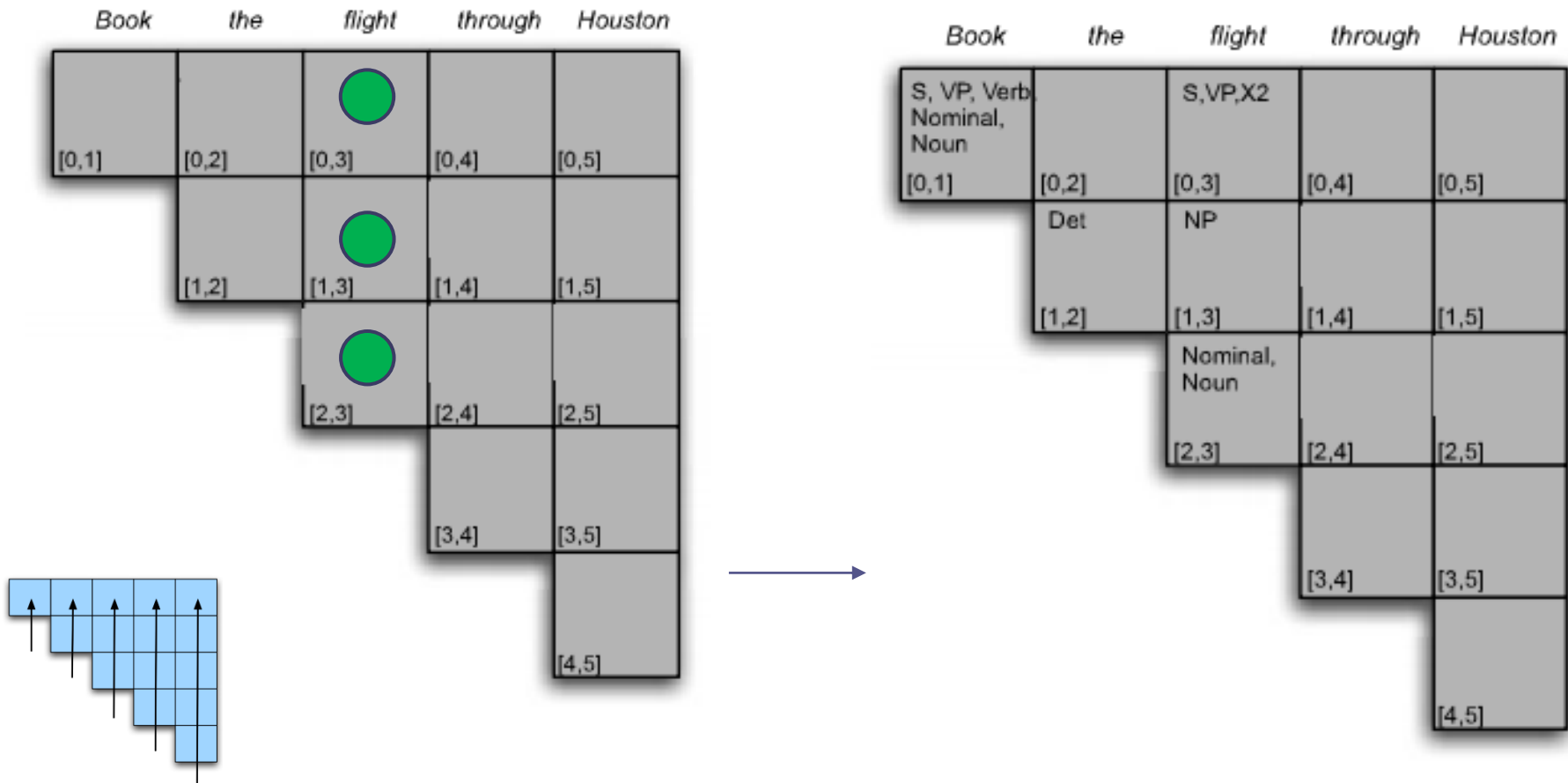
## Algorithme Cocke-Younger-Kasami (CYK)

- Exemple : **0** *Book* **1** *the* **2** *flight* **3** *through* **4** *Houston* **5**



## Algorithme Cocke-Younger-Kasami (CYK)

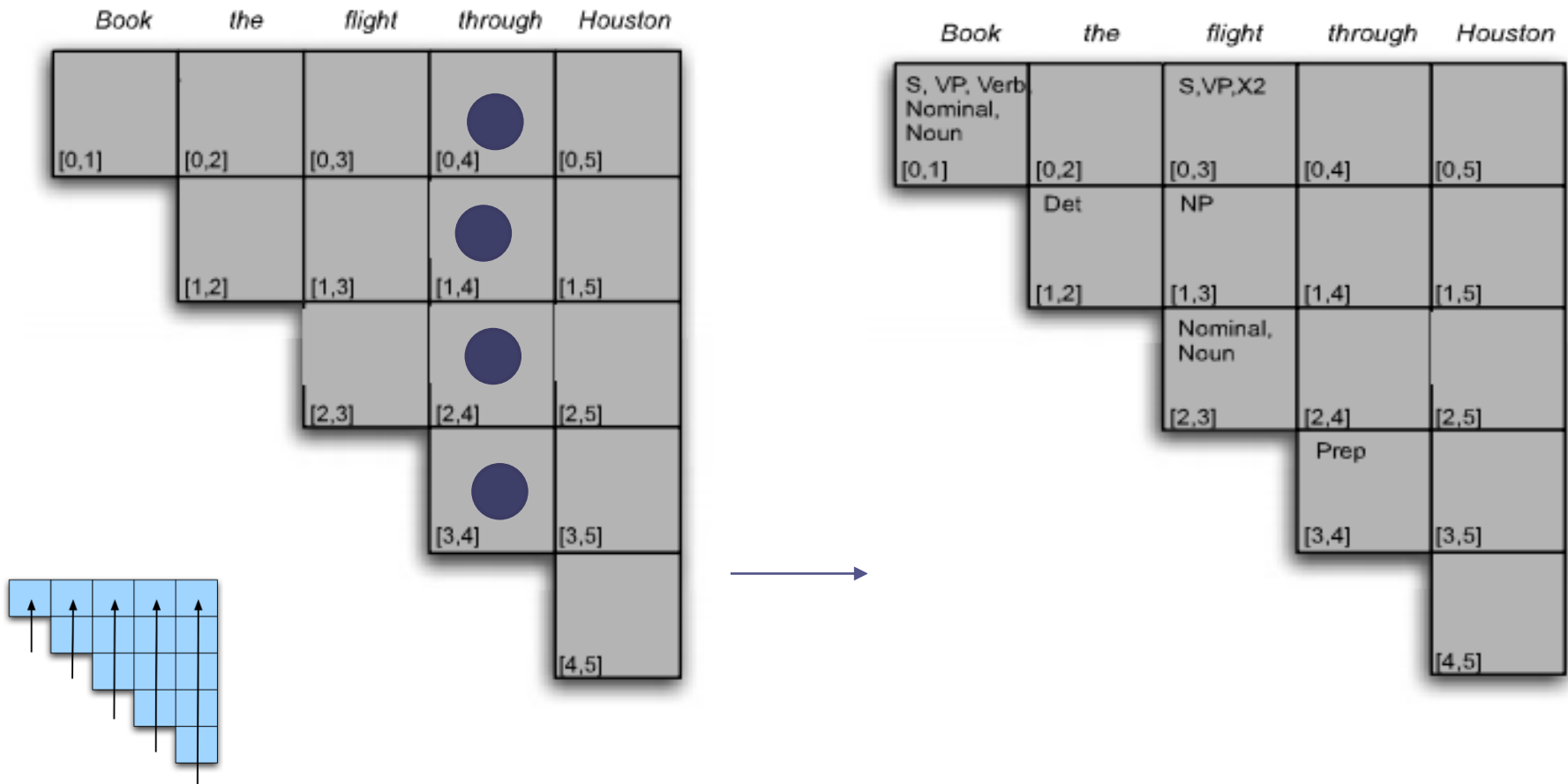
- Exemple : **0** *Book* **1** *the* **2** *flight* **3** *through* **4** *Houston* **5**





## Algorithme Cocke-Younger-Kasami (CYK)

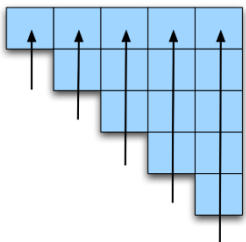
- Exemple : **0** *Book* **1** *the* **2** *flight* **3** *through* **4** *Houston* **5**



## Algorithme Cocke-Younger-Kasami (CYK)

- Exemple : **0** *Book* **1** *the* **2** *flight* **3** *through* **4** *Houston* **5**

Book	the	flight	through	Houston
[0,1]	[0,2]	[0,3]	[0,4]	[0,5]
	[1,2]	[1,3]	[1,4]	[1,5]
		[2,3]	[2,4]	[2,5]
			[3,4]	[3,5]
				[4,5]

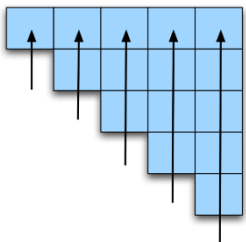


Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun [0,1]	[0,2]	S, VP, X2 [0,3]	[0,4]	S, VP, X2 [0,5]
	Det [1,2]	NP [1,3]	[1,4]	NP [1,5]
		Nominal, Noun [2,3]	[2,4]	Nominal [2,5]
			Prep [3,4]	PP [3,5]
				NP, Proper-Noun [4,5]

## Algorithme Cocke-Younger-Kasami (CYK)

- Exemple : **0** *Book* **1** *the* **2** *flight* **3** *through* **4** *Houston* **5**

Book	the	flight	through	Houston
[0,1]	[0,2]	[0,3]	[0,4]	[0,5]
	[1,2]	[1,3]	[1,4]	[1,5]
		[2,3]	[2,4]	[2,5]
			[3,4]	[3,5]
				[4,5]



Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun [0,1]	[0,2]	S,VP,X2 [0,3]	[0,4]	S,VP,X2 [0,5]
	Det [1,2]	NP [1,3]	[1,4]	NP [1,5]
		Nominal, Noun [2,3]	[2,4]	Nominal [2,5]
			Prep [3,4]	PP [3,5]
				NP, Proper- Noun [4,5]

## Algorithme Cocke-Younger-Kasami (CYK)

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun [0,1]	[0,2]	S,VP,X2 [0,3]	[0,4]	[0,5]
	Det [1,2]	NP [1,3]	[1,4]	[1,5]
		Nominal, Noun [2,3]	[2,4]	[2,5]
			Prep ← PP [3,4]	[3,5]
				NP, Proper- Noun [4,5]

### $\mathcal{L}_1$ in CNF

$S \rightarrow NP VP$   
 $S \rightarrow XI VP$   
 $XI \rightarrow Aux NP$   
 $S \rightarrow book \mid include \mid prefer$   
 $S \rightarrow Verb NP$   
 $S \rightarrow X2 PP$   
 $S \rightarrow Verb PP$   
 $S \rightarrow VP PP$   
 $NP \rightarrow I \mid she \mid me$   
 $NP \rightarrow TWA \mid Houston$   
 $NP \rightarrow Det Nominal$   
 $Nominal \rightarrow book \mid flight \mid meal \mid money$   
 $Nominal \rightarrow Nominal Noun$   
 $Nominal \rightarrow Nominal PP$   
 $VP \rightarrow book \mid include \mid prefer$   
 $VP \rightarrow Verb NP$   
 $VP \rightarrow X2 PP$   
 $X2 \rightarrow Verb NP$   
 $VP \rightarrow Verb PP$   
 $VP \rightarrow VP PP$   
 $PP \rightarrow Preposition NP$

## Algorithme Cocke-Younger-Kasami (CYK)

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun [0,1]		S,VP,X2 [0,3]		
	Det [1,2]	NP [1,3]		
		Nominal Noun [2,3]		
			Prep [3,4]	
				PP [3,5]
				NP, Proper- Noun [4,5]

### $\mathcal{L}_1$ in CNF

$S \rightarrow NP VP$   
 $S \rightarrow X1 VP$   
 $X1 \rightarrow Aux NP$   
 $S \rightarrow book \mid include \mid prefer$   
 $S \rightarrow Verb NP$   
 $S \rightarrow X2 PP$   
 $S \rightarrow Verb PP$   
 $S \rightarrow VP PP$   
 $NP \rightarrow I \mid she \mid me$   
 $NP \rightarrow TWA \mid Houston$   
 $NP \rightarrow Det Nominal$   
 $Nominal \rightarrow book \mid flight \mid meal \mid money$   
 $Nominal \rightarrow Nominal Noun$   
 $Nominal \rightarrow Nominal PP$   
 $VP \rightarrow book \mid include \mid prefer$   
 $VP \rightarrow Verb NP$   
 $VP \rightarrow X2 PP$   
 $X2 \rightarrow Verb NP$   
 $VP \rightarrow Verb PP$   
 $VP \rightarrow VP PP$   
 $PP \rightarrow Preposition NP$

## Algorithme Cocke-Younger-Kasami (CYK)

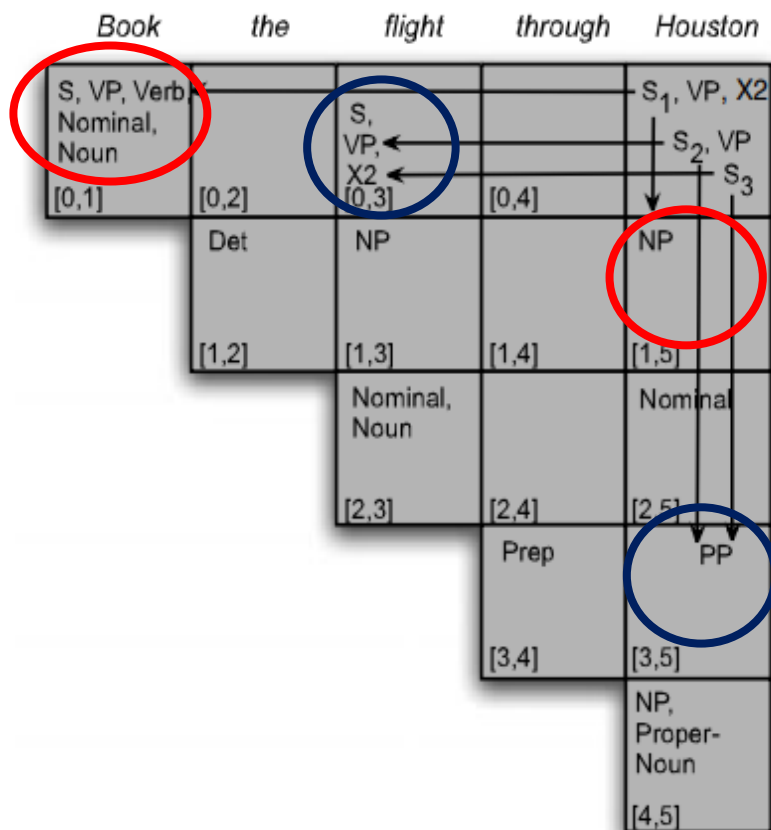
Book	the	flight	through	Houston
S, VP, Verb Nominal, Noun [0,1]		S,VP,X2 [0,3]		
	Det [1,2]	NP [1,3]		NP [1,5]
		Nominal, Noun [2,3]		Nominal [2,5]
			Prep [3,4]	PP [3,5]
				NP proper- Noun [4,5]

### $\mathcal{L}_1$ in CNF

$S \rightarrow NP VP$   
 $S \rightarrow XI VP$   
 $XI \rightarrow Aux NP$   
 $S \rightarrow book \mid include \mid prefer$   
 $S \rightarrow Verb NP$   
 $S \rightarrow X2 PP$   
 $S \rightarrow Verb PP$   
 $S \rightarrow VP PP$   
 $NP \rightarrow I \mid she \mid me$   
 $NP \rightarrow TWA \mid Houston$   
 $NP \rightarrow Det Nominal$   
 $Nominal \rightarrow book \mid flight \mid meal \mid money$   
 $Nominal \rightarrow Nominal Noun$   
 $Nominal \rightarrow Nominal PP$   
 $VP \rightarrow book \mid include \mid prefer$   
 $VP \rightarrow Verb NP$   
 $VP \rightarrow X2 PP$   
 $X2 \rightarrow Verb NP$   
 $VP \rightarrow Verb PP$   
 $VP \rightarrow VP PP$   
 $PP \rightarrow Preposition NP$

# Algorithme Cocke-Younger-Kasami (CYK)

3 arbres  
syntaxiques  
potentiels



$\mathcal{L}_1$  in CNF

$S \rightarrow NP VP$   
 $S \rightarrow XI VP$   
 $XI \rightarrow Aux NP$   
 $S \rightarrow book \mid include \mid prefer$   
 $S \rightarrow Verb NP$   
 $S \rightarrow X2 PP$   
 $S \rightarrow Verb PP$   
 $S \rightarrow VP PP$   
 $NP \rightarrow I \mid she \mid me$   
 $NP \rightarrow TWA \mid Houston$   
 $NP \rightarrow Det Nominal$   
 $Nominal \rightarrow book \mid flight \mid meal \mid money$   
 $Nominal \rightarrow Nominal Noun$   
 $Nominal \rightarrow Nominal PP$   
 $VP \rightarrow book \mid include \mid prefer$   
 $VP \rightarrow Verb NP$   
 $VP \rightarrow X2 PP$   
 $X2 \rightarrow Verb NP$   
 $VP \rightarrow Verb PP$   
 $VP \rightarrow VP PP$   
 $PP \rightarrow Preposition NP$

# Arbre Syntaxique - Parse Tree

## Ambiguïté structurelle

- Une phrase peut avoir une interprétation ambiguë et avoir plus d'un arbre syntaxique :
  - En ayant plusieurs paires par cellules, on pourrait tous les générer.
  - On n'a pas de façon de préférer un arbre parmi les autres.
  - Utiliser une version probabiliste des grammaires pour exprimer de telles préférences.
- Une phrase pourrait ne pas avoir d'arbre (i.e. n'appartient pas au langage de la grammaire):
  - le cas si aucune règle pour le symbole S est présente dans la cellule finale.



# Arbre Syntaxique - Parse Tree

## Algorithme CYK

- L'arbre syntaxique qui sera généré sera binaire et suivra la grammaire en forme normale de Chomsky
- Il serait possible de faire un post-traitement pour obtenir un arbre suivant la grammaire hors contexte originale : doit inverser les transformations utilisées pour la forme CNF.
- Certains algorithmes sont applicables directement à une grammaire non CNF: Earley algorithm.

# Arbre Syntaxique - Parse Tree

## Evaluer le parsing

- L'outil standard pour évaluer les analyseurs syntaxiques qui attribuent un seul arbre à une phrase est la métrique **PARSEVAL** (Black et al., 1991).
- La métrique PARSEVAL mesure à quel point les constituants de l'arbre syntaxique généré (d'hypothèse) ressemblent aux constituants d'une analyse (arbre) syntaxique (de référence) étiquetée à la main par les experts.
- Un constituant dans une analyse d'hypothèse d'une phrase  $S$  est étiqueté correct s'il y a un constituant dans l'analyse de référence avec le même point de départ, le même point de fin et le même symbole non terminal.

$$\text{labeled recall} = \frac{\# \text{ of correct constituents in hypothesis parse of } s}{\# \text{ of correct constituents in reference parse of } s}$$

$$\text{labeled precision} = \frac{\# \text{ of correct constituents in hypothesis parse of } s}{\# \text{ of total constituents in hypothesis parse of } s}$$

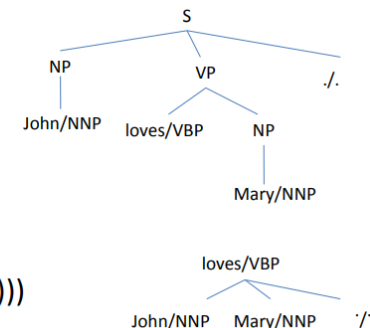
$$F_1 = \frac{2PR}{P + R}$$

# Arbre Syntaxique

## Treebank (syntaxique)

- Un treebank est un corpus de texte où chaque phrase a été **annotée** avec une **structure syntaxique** (dependency structure or phrase structure). i.e. chaque phrase a un arbre syntaxique qui lui correspond.
- Treebank, banque d'arbres, corpus **arboré**. Utilisé notamment pour extraire les règles d'une CFG.
- Les phrases sont annotées manuellement (ou semi-automatiquement).
- À partir d'un treebank, il est alors possible de mesurer statistiquement différents phénomènes syntaxiques.

- (S (NP (NNP John))  
    (VP (VBP loves)  
        (NP (NNP Mary))))  
  (. .))



# Arbre Syntaxique - Corpus / Datasets

## **Project Penn Treebank** (PTB)

- Le projet **Penn Treebank** a produit plusieurs corpus annotés.
- 1960s: Brown Corpus
- Début 1990s: The English Penn Treebank
- Fin 1990s: Prague Dependency Treebank
- 1990s – aujourd'hui: Arabic, Chinese, Dutch, Finnish, French, German, Greek, Hebrew, Hindi, Hungarian, Icelandic, Italian, Japanese, Korean, Latin, Norwegian, Polish, Spanish, Turkish, etc.
- différentes sources de texte : Brown, Switchboard, ATIS, Wall Street Journal..
- différentes versions : Treebank I, Treebank II, Treebank III.

## **Projet Universal Dependencies** (UD)

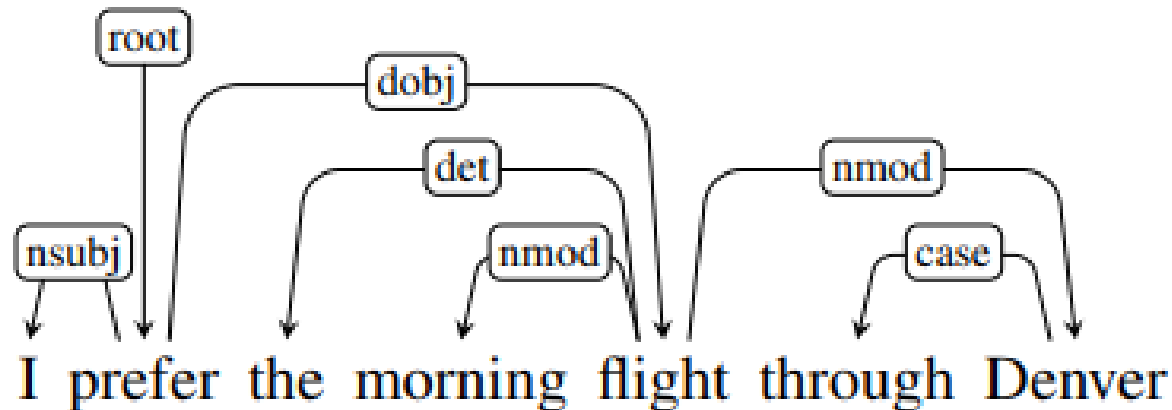
- est une initiative visant à créer Treebanks pour un large éventail de langues en utilisant le même schéma d'annotation. 2015.
- Version 2.7 sortie en 2020 consiste en 183 treebanks et 104 languages.

# Dependency Parsing - Arbre de dépendance

- Dans les **grammaires dé dépendance**, les constituants et les règles de structure des phrases ne jouent pas un rôle direct.
- Elles se concentrent sur la façon dont les mots se rapportent à d'autres mots (i.e. les **relations** entre les mots).
- La structure syntaxique d'une phrase est décrite uniquement en termes de mots (ou lemmes) dans une phrase et un ensemble associé de relations grammaticales binaires dirigées qui existent entre les mots.
- La dépendance est une relation asymétrique binaire qui existe entre une **tête** et ses **dépendants**.
- La **tête syntaxique (root)** d'une phrase est généralement considérée comme le verbe, et chaque autre **mot dépend** de la tête de la phrase ou s'y connecte via un **chemin de dépendances**.

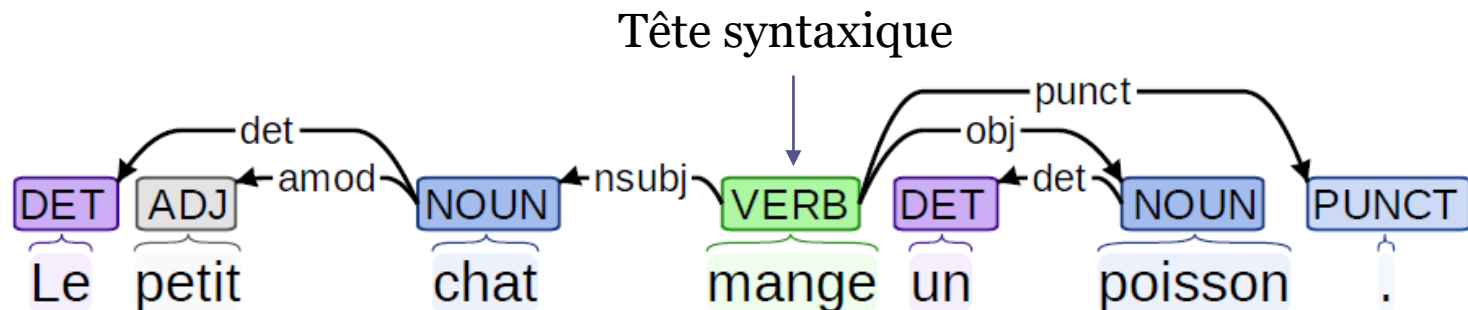
# Dependency Parsing - Arbre de dépendance

- Une **représentation (arbre) de dépendance** est une structure orientée étiquetée, où les **nœuds** sont les **mots** et les **arcs** étiquetés représentent les **relations de dépendance** des têtes aux dépendants.
- Les arcs sont étiquetés avec la **fonction grammaticale** qui existe entre un dépendant et sa tête (sujet, objet, etc.).
- Les liens syntaxiques entre les mots d'une phrase sont appelés : **dépendances**.
- Exemple : I prefer the morning flight through Denver



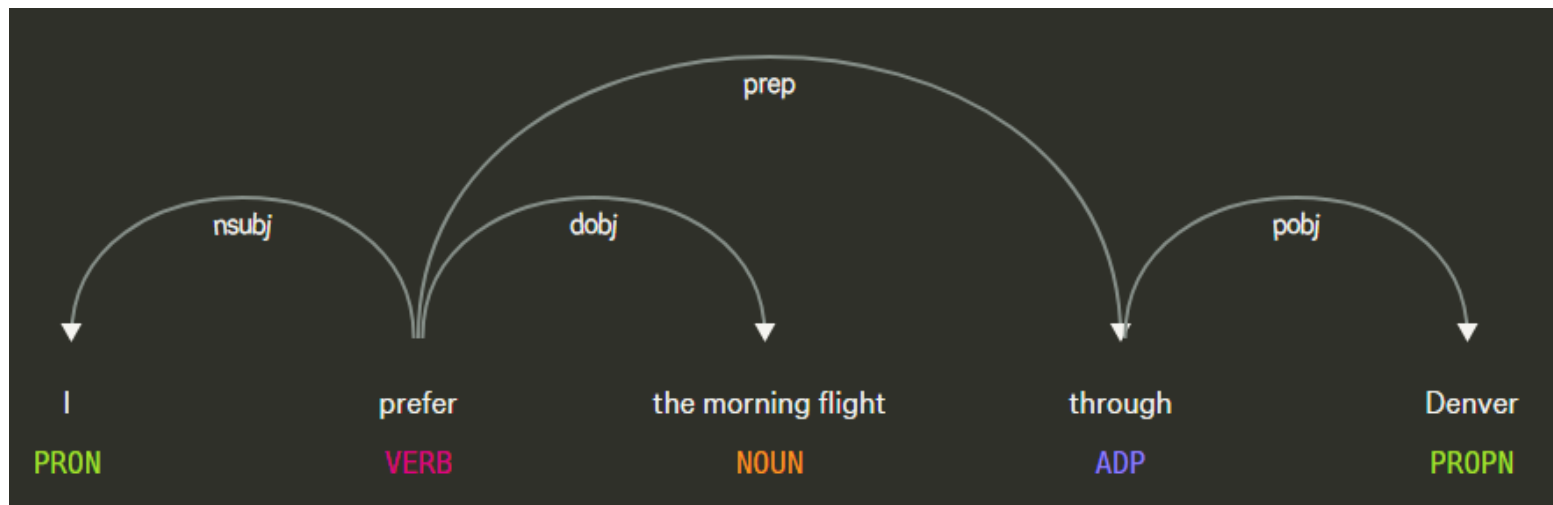
# Dependency Parsing - Arbre de dépendance

- Une **représentation (arbre) de dépendance** est une structure orientée étiquetée, où les **nœuds** sont les **mots** et les **arcs** étiquetés représentent les **relations de dépendance** des têtes aux dépendants.
- Les arcs sont étiquetés avec la **fonction grammaticale** qui existe entre un dépendant et sa tête (sujet, objet, etc.).
- Les liens syntaxiques entre les mots d'une phrase sont appelés : **dépendances**.
- Exemple : Le petit chat mange un poisson.



# Dependency Parsing - Arbre de dépendance

- **Une représentation (arbre) de dépendance** est une structure orientée étiquetée, où les **nœuds** sont les **mots** et les **arcs** étiquetés représentent les **relations de dépendance** des têtes aux dépendants.
- Les arcs sont étiquetés avec la **fonction grammaticale** qui existe entre un dépendant et sa tête.
- Exemple : I prefer the morning flight through Denver
- Test online – SpaCy : <https://explosion.ai/demos/displacy> | <https://corenlp.run/>





# Dependency Parsing

<https://universaldependencies.org/u/dep/index.html>

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

**Figure 14.2** Selected dependency relations from the Universal Dependency set. (de Marneffe et al., 2014)

# Dependency Parsing

Relation	Examples with <i>head</i> and <b>dependent</b>
NSUBJ	<b>United</b> <i>canceled</i> the flight.
DOBJ	United <i>diverted</i> the <b>flight</b> to Reno. We <i>booked</i> her the first <b>flight</b> to Miami.
IOBJ	We <i>booked</i> <b>her</b> the flight to Miami.
NMOD	We took the <b>morning</b> <i>flight</i> .
AMOD	Book the <b>cheapest</b> <i>flight</i> .
NUMMOD	Before the storm JetBlue canceled <b>1000</b> <i>flights</i> .
APPOS	<i>United</i> , a <b>unit</b> of UAL, matched the fares.
DET	<b>The</b> <i>flight</i> was canceled. <b>Which</b> <i>flight</i> was delayed?
CONJ	We <i>flew</i> to Denver and <b>drove</b> to Steamboat.
CC	We flew to Denver <b>and</b> <i>drove</i> to Steamboat.
CASE	Book the flight <b>through</b> <i>Houston</i> .

**Figure 14.3** Examples of core Universal Dependency relations.

# Dependency Parsing

Dép. de base	Description	Exemple
nsubj	sujet nominal	Le <u>peuple</u> <b>gagne</b>
obj	objet direct	On <b>présente</b> le <u>cours</u>
iobj	objet indirect	Il <u>m'</u> <b>envoie</b>
csubj	sujet propositionnel	<u>Suivre</u> le cours <b>permet</b> ...
Dép. des noms	Description	Exemple
amod	modificateur adjectival	La <b>fil</b> <u>le</u> <u>modeste</u>
det	déterminant	<u>La</u> <b>fil</b> <u>le</u>
nmod	modificateur nominal	Le <u>résultat</u> de la <b>cours</b> <u>e</u>
nummod	modificateur numérique	J'ai mangé <u>3</u> <b>bonbons</b>

TABLE – Quelques relations de dépendances universelles de Stanford

[de Marneffe et al., 2014]

(<https://universaldependencies.org/u/dep/index.html>)

# Dependency Parsing

[https://downloads.cs.stanford.edu/nlp/software/dependencies\\_manual.pdf](https://downloads.cs.stanford.edu/nlp/software/dependencies_manual.pdf)

Stanford Dependencies representation : 50 grammatical relations

## **Stanford typed dependencies manual**

Marie-Catherine de Marneffe and Christopher D. Manning

September 2008

Revised for the Stanford Parser v. 3.7.0 in September 2016

Please note that this manual describes the original Stanford Dependencies representation. As of version 3.5.2, the default representation output by the Stanford Parser and Stanford CoreNLP is the new Universal Dependencies (UD) representation, and we no longer maintain the original Stanford Dependencies representation. For a description of the UD representation, take a look at the Universal Dependencies documentation at <http://www.universaldependencies.org> and the discussion of the *enhanced* and *enhanced++* UD representations by Schuster and Manning (2016).

# Named Entity Recognition

- Une partie de l'étiquetage morphosyntaxique peut nous dire que des mots comme Camus, Stanford University, et Colorado sont tous des noms propres; être un nom propre est une propriété grammaticale de ces mots.
- Mais d'un point de vue **sémantique**, ces noms propres font référence à différents types d'entités:
- ✓ Camus est une **personne**, Stanford University est une **organisation**, et le Colorado est une **localisation**.
- Une **entité nommée** est tout ce qui peut être référencé avec un nom propre: une personne, un lieu, une organisation.
- La tâche de la **reconnaissance d'entité nommée** (NER) est de trouver des parties de texte qui constituent des noms propres et d'étiqueter le type d'entité nommée.

# Named Entity Recognition

- Quatre balises d'entité sont les plus courantes: **PER** (personne), **LOC** (emplacement), **ORG** (organisation) ou **GPE** (entité géopolitique).
- Le terme entité nommée est aussi étendu pour inclure des éléments qui ne sont pas des entités en soi, y compris les **dates**, **temps**, et d'autres types **d'expressions temporelles**, et même des expressions **numériques** comme des prix, etc.
- Online test : <https://explosion.ai/demos/displacy-ent>

In fact, the **Chinese** **NORP** market has the **three** **CARDINAL** most influential names of the retail and tech space – **Alibaba** **GPE**, **Baidu** **ORG**, and **Tencent** **PERSON** (collectively touted as **BAT** **ORG**), and is betting big in the global **AI** **GPE** in retail industry space. The **three** **CARDINAL** giants which are claimed to have a cut-throat competition with the **U.S.** **GPE** (in terms of resources and capital) are positioning themselves to become the 'future **AI** **PERSON** platforms'. The trio is also expanding in other **Asian** **NORP** countries and investing heavily in the **U.S.** **GPE** based **AI** **GPE** startups to leverage the power of **AI** **GPE**. Backed by such powerful initiatives and presence of these conglomerates, the market in APAC AI is forecast to be the fastest-growing **one** **CARDINAL**, with an anticipated **CAGR** **PERSON** of **45%** **PERCENT** over **2018 - 2024** **DATE**.

To further elaborate on the geographical trends, **North America** **LOC** has procured **more than 50%** **PERCENT** of the global share in **2017** **DATE** and has been leading the regional landscape of **AI** **GPE** in the retail market. The **U.S.** **GPE** has a significant credit in the regional trends with **over 65%** **PERCENT** of investments (including M&As, private equity, and venture capital) in artificial intelligence technology. Additionally, the region is a huge hub for startups in tandem with the presence of tech titans, such as **Google** **ORG**, **IBM** **ORG**, and **Microsoft** **ORG**.

## Entity labels (select all)

<input checked="" type="checkbox"/> <b>PERSON</b>	<input checked="" type="checkbox"/> <b>NORP</b>	<input checked="" type="checkbox"/> <b>ORG</b>	<input checked="" type="checkbox"/> <b>GPE</b>
<input checked="" type="checkbox"/> <b>LOC</b>	<input checked="" type="checkbox"/> <b>PRODUCT</b>	<input type="checkbox"/> <b>EVENT</b>	<input type="checkbox"/> <b>WORK OF ART</b>
<input type="checkbox"/> <b>LANGUAGE</b>	<input checked="" type="checkbox"/> <b>DATE</b>	<input type="checkbox"/> <b>TIME</b>	<input type="checkbox"/> <b>PERCENT</b>
<input type="checkbox"/> <b>MONEY</b>	<input type="checkbox"/> <b>QUANTITY</b>	<input type="checkbox"/> <b>ORDINAL</b>	
<input type="checkbox"/> <b>CARDINAL</b>			

# Références

Speech and Language Processing - Livre de Dan Jurafsk -

<https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf>

Cours - *François Yvon* – Une petite introduction au Traitement Automatique des Langues Naturelles,

<https://perso.limsi.fr/anne/coursM2R/intro.pdf>

Article – Marcel Cori - Des méthodes de traitement automatique aux linguistiques fondées sur les corpus

- <https://www.cairn.info/revue-langages-2008-3-page-95.htm>

Article - Pascale Sébillot - Le traitement automatique des langues face aux données textuelles volumineuses et potentiellement dégradées : qu'est-ce que cela change ?

- <https://hal.archives-ouvertes.fr/hal-01056396/document>

Cours – HUGO LAROCHELLE - Analyse Syntaxique

[http://info.usherbrooke.ca/hlarochelle/ift607/o6\\_analyse\\_syntaxique.pdf](http://info.usherbrooke.ca/hlarochelle/ift607/o6_analyse_syntaxique.pdf)

[http://faculty.washington.edu/fxia/lisa2011/slides/intro\\_to\\_treebanks.pdf](http://faculty.washington.edu/fxia/lisa2011/slides/intro_to_treebanks.pdf)

Cours - ARIES Abdelkrime - Le traitement automatique du langage naturel.

[https://github.com/projeduc/ESI\\_2CS\\_TALN](https://github.com/projeduc/ESI_2CS_TALN)