# Introduction à l'analyse syntaxique

felipe@iro.umontreal.ca

**RALI**
Dept. Informatique et Recherche Opérationnelle
Université de **Montréal**


V1.0

Last compiled: 20 octobre 2019

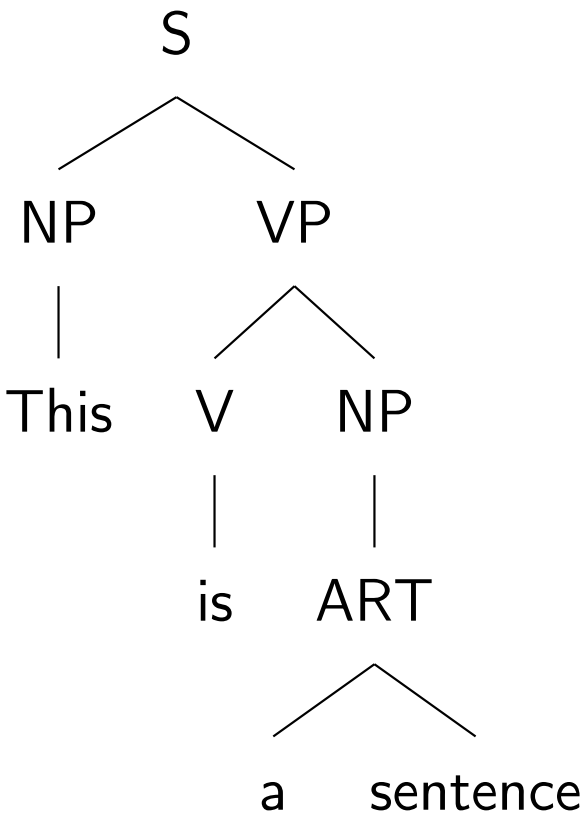
# Quelques sources à l'origine de ces transparents

- Les chapitres concernés dans le livre de Jurafsky & Martin : Speech and Language processing.
  Disponible en ligne :
  https://web.stanford.edu/~jurafsky/slp3/
- Le livre de Charniak (Statistical Language Learning)
- Les articles mentionnés dans les slides

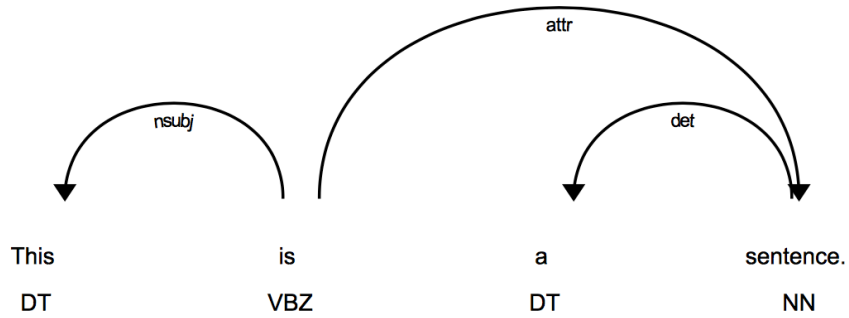
rali

# Deux grandes familles d'analyseurs

- Analyseurs en constituants




- Analyseurs en dépendances



# Plan



rali

# Plan

# Quelques éléments de base

- Rappels de quelques notions de théorie des langages formels
- Technique d'analyse descendante (top-down)
  - analyse descendante récursive
  - analyse descendante prédictive
- Technique d'analyse montante (bottom-up)
  - Shift-Reduce parsing
- Techniques d'analyse génériques
  - CYK
  - Earley

rali

# Plan



rali

# Éléments de vocabulaire

Définition : Une **grammaire** est définie par un quadruplet $< N, T, R, S >$ où :

- $N$ est l'ensemble des symboles non-terminaux
- $T$ est l'ensemble des symboles terminaux
- $R$ est l'ensemble des règles de production (de la forme : $\alpha \rightarrow \beta$)
- $S$ est l'axiome de départ

Conventions de notation :

- $V$ est le vocabulaire de la grammaire $V = N \cup T$.
- Les lettres minuscules désignent dans ce document des symboles terminaux ($a, b, \ldots$).
- Les lettres majuscules désignent des symboles non-terminaux ($A, B, \ldots$).
- Les symboles grecs désignent n'importe quelle suite (éventuellement vide) de symboles terminaux et non-terminaux ($\alpha, \beta, \ldots$).
- $\epsilon$ désigne une chaîne vide.

# Éléments de vocabulaire

- Soit une chaîne $\alpha A \beta$ et la règle de grammaire $A \rightarrow \gamma \in R$
- On note par $\alpha A \beta \Rightarrow \alpha \gamma \beta$ le fait que la première chaîne produit la seconde par substitution de $A$ par $\gamma$.
- Une séquence de zéro ou plus de ces substitutions est appelée une dérivation et est indiquée par $\stackrel{\star}{\Rightarrow}$.

  **Exemple :** Soit la grammaire : $S \rightarrow (S)S|\epsilon$

  Alors $S \stackrel{\star}{\Rightarrow} (()())()$ car :

$$S \underset{1}{\Rightarrow} (S)\underline{S} \underset{1}{\Rightarrow} (S)(S)\underline{S} \underset{2}{\Rightarrow} (S)(\underline{S})$$

$$\underset{2}{\Rightarrow} (\underline{S})() \underset{1}{\Rightarrow} ((\underline{S})S)() \underset{2}{\Rightarrow} (()\underline{S})()$$

$$\underset{1}{\Rightarrow} (()(S)\underline{S})() \underset{2}{\Rightarrow} (()(\underline{S}))() \underset{2}{\Rightarrow} (()())()$$

# Éléments de vocabulaire

- Il existe (souvent) plusieurs façons de réduire une chaîne.
- On appelle la dérivation gauche (*left-most derivation*), la dérivation obtenue en substituant le non-terminal le plus à gauche de la chaîne à réduire.

$$\underline{S} \underset{1}{\Rightarrow} (\underline{S})S \underset{1}{\Rightarrow} ((\underline{S})S)S \underset{2}{\Rightarrow} (()\underline{S})S$$

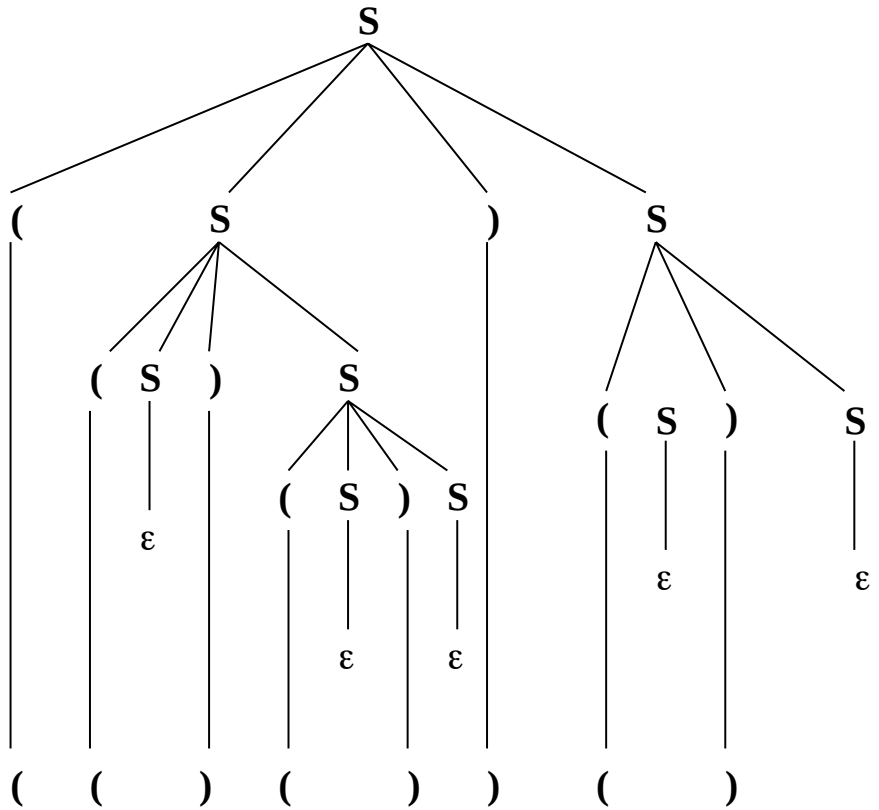
$$\underset{1}{\Rightarrow} (()(\underline{S})S)S \underset{2}{\Rightarrow} (()()\underline{S})S \underset{2}{\Rightarrow} (()())\underline{S}$$

$$\underset{1}{\Rightarrow} (()())(\underline{S})S \underset{2}{\Rightarrow} (()())()\underline{S} \underset{2}{\Rightarrow} (()())()$$

- Un arbre d'analyse (*parse tree*), ou arbre de dérivation) est une représentation graphique d'une dérivation dans laquelle on fait abstraction de l'ordre dans lequel les non-terminaux sont dérivés.

rali

# Éléments de vocabulaire




Un parcours en profondeur de l'arbre permet d'obtenir la chaîne analysée

# Éléments de vocabulaire

- Une grammaire qui produit plus d'un arbre syntaxique pour une chaîne donnée est dite ambiguë (cad qu'elle produit plus d'une dérivation gauche (ou droite) d'une même chaîne).

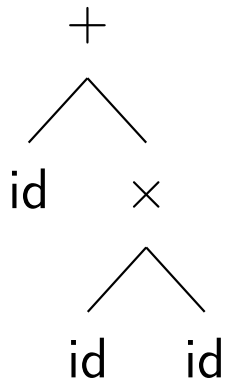
  Ex : $E \rightarrow E + E | E \times E | (E) | \text{id}$
  - Il existe deux dérivations de la chaîne : $\text{id} + \text{id} \times \text{id}$ :

$$\underline{E} \underset{1}{\Rightarrow} \underline{E} + E \underset{4}{\Rightarrow} \text{id} + \underline{E} \underset{2}{\Rightarrow} \text{id} + \underline{E} \times E \underset{4}{\Rightarrow} \text{id} + \text{id} \times \underline{E} \underset{4}{\Rightarrow} \text{id} + \text{id} \times \text{id}$$

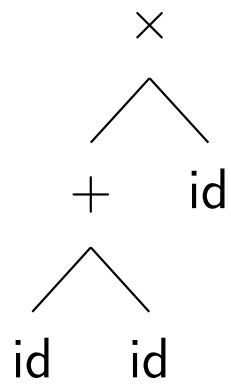
$$\underline{E} \underset{2}{\Rightarrow} \underline{E} \times E \underset{1}{\Rightarrow} \underline{E} + E \times E \underset{4}{\Rightarrow} \text{id} + \underline{E} \times E \underset{4}{\Rightarrow} \text{id} + \text{id} \times \underline{E} \underset{4}{\Rightarrow} \text{id} + \text{id} \times \text{id}$$

  - Chacune de ces dérivations correspond à un arbre syntaxique différent.

# Éléments de vocabulaire




$$\begin{aligned}
\underline{E} &\underset{1}{\Rightarrow} \underline{E} + E \\
&\underset{4}{\Rightarrow} \mathrm{id} + \underline{E} \\
&\underset{2}{\Rightarrow} \mathrm{id} + \underline{E} \times E \\
&\underset{4}{\Rightarrow} \mathrm{id} + \mathrm{id} \times \underline{E} \\
&\underset{4}{\Rightarrow} \mathrm{id} + \mathrm{id} \times \mathrm{id}
\end{aligned}$$




$$\begin{aligned}
\underline{E} &\underset{2}{\Rightarrow} \underline{E} \times E \\
&\underset{1}{\Rightarrow} \underline{E} + E \times E \\
&\underset{4}{\Rightarrow} \mathrm{id} + \underline{E} \times E \\
&\underset{4}{\Rightarrow} \mathrm{id} + \mathrm{id} \times \underline{E} \\
&\underset{4}{\Rightarrow} \mathrm{id} + \mathrm{id} \times \mathrm{id}
\end{aligned}$$


rali

# Lever une ambiguïté (intuition)

- on introduit les opérateurs les moins prioritaires en premier dans la grammaire (par ajout de non-terminaux),
- les opérateurs associatifs à gauche (ex : a + b + c = ((a + b) + c)) sont traduits par des règles récursives à gauche ($A \rightarrow A\alpha$).

$$\begin{array}{lll} E \rightarrow E+T & E \rightarrow E\text{-}T & E \rightarrow T \\ T \rightarrow T*F & T \rightarrow T/F & T \rightarrow F \\ F \rightarrow id & F \rightarrow (E) & \end{array}$$

- seule la première dérivation est possible avec cette nouvelle grammaire.

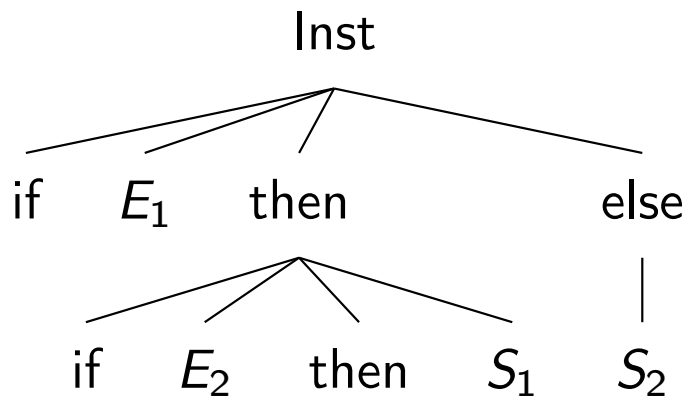

rali

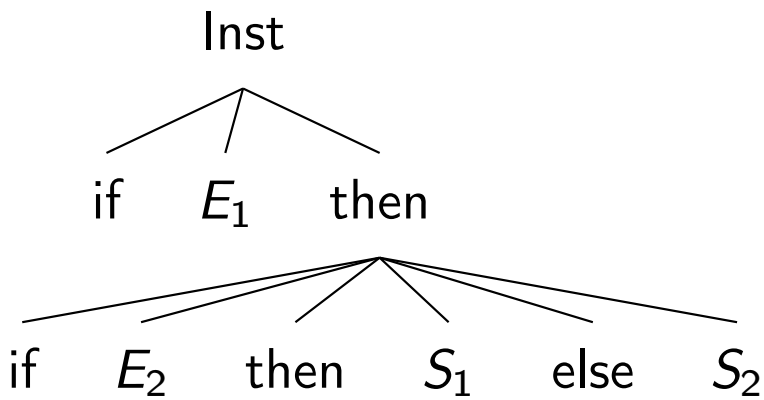
# La construction *if-then-else*

- La grammaire suivante est ambiguë :

$$\begin{array}{lcl} \text{Inst} & \rightarrow & \textbf{if}\ \text{Expr}\ \textbf{then}\ \text{Inst} \\ \text{Inst} & \rightarrow & \textbf{if}\ \text{Expr}\ \textbf{then}\ \text{Inst}\ \textbf{else}\ \text{Inst} \\ \text{Inst} & \rightarrow & \textbf{other} \end{array}$$

- preuve : **if** $E_1$ **then if** $E_2$ **then** $S_1$ **else** $S_2$ a deux arbres syntaxiques.



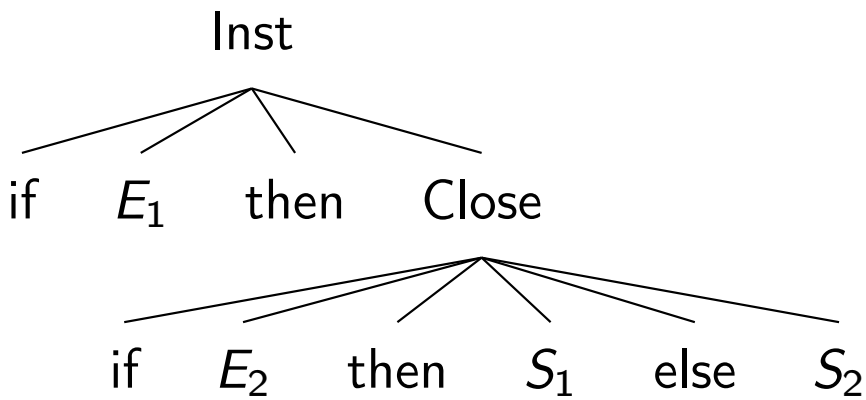




rali

# la construction *if-then-else*

- une version non ambiguë :

$$
\begin{array}{lcl}
\text{Inst} & \rightarrow & \textbf{if}\ \text{Expr}\ \textbf{then}\ \text{close}\ \textbf{else}\ \text{Inst} \\
 & \rightarrow & \textbf{if}\ \text{Expr}\ \textbf{then}\ \text{Inst} \\
 & \rightarrow & \textbf{Close} \\
\text{Close} & \rightarrow & \textbf{if}\ \text{Expr}\ \textbf{then}\ \text{close}\ \textbf{else}\ \text{close} \\
 & \rightarrow & \textbf{other}
\end{array}
$$




- **Note :** il n'existe **pas** d'algorithme capable de déterminer si une grammaire est ambiguë

# Classification chomskienne

| type | contraintes sur $R$ | langages | analyse |
|---|---|---|---|
| 3 | $A \to aB$ *ou* $A \to a$ | réguliers | $\mathcal{O}(n)$ |
| 2 | $A \to \alpha$ | hors-contexte | $\mathcal{O}(n^3)$ |
| 1 | $\alpha \to \beta$ *avec* $\lvert\alpha\rvert \leq \lvert\beta\rvert$ | contextuels | $\mathcal{O}(n^6)$ |
| 0 | $\alpha \to \beta$ | récursivement énumérable | — |

# Les langages réguliers

- Représentables par une expression régulière (ou un automate ou une grammaire régulière)
- **Définition :** d'une expression régulière (ER)
  - les éléments terminaux sont des ER,
  - si $a$ et $b$ sont des ER, alors $ab$ est une ER (concaténation),
  - si $a$ est une ER, alors $a^\star$ est une ER (nb. quelconque de fois de $a$),
  - si $a$ est une ER, alors $(a)$ est une ER
  - si $a$ et $b$ sont des ER, alors $a|b$ est une ER ($a$ OU $b$)
- **Note :** on utilise souvent des ER étendues :

| | | |
|---|---|---|
| $(a\|b)+$ | une fois ou plus | $(a\|b)(a\|b)^\star$ |
| $[a-z]$ | de $a$ à $z$ | $a\|b\|c\|d\|\ldots\|z$ |

# Les langages réguliers

**Ex 1 :** $\{a^\star b^\star\}$ représente l'ensemble des chaînes constituées d'un nombre quelconque (éventuellement nul) de $a$ suivi d'un nombre quelconque (éventuellement nul) de $b$.

$abb, a, b, bbb$ sont des exemples de telles chaînes.

$$\text{grammaire équivalente} \begin{cases} A \rightarrow aA|bB|\epsilon \\ B \rightarrow bB|\epsilon \end{cases}$$

**Ex 2 :** $\{0\,|\,(0|1)^\star 0\}$ représente l'ensemble des chaînes binaires paires (simplifiable !).

**Ex 3 :** un exemple de (mauvaise) ER capable de reconnaître les adresses emails dans un document (en vue par exemple d'envoyer du spam) :
$[A-Z, a-z, 0-9, -, , .]+@[A-Z, a-z, 0-9, -, , .]^\star.[a-z, A-Z]^\star$

# Langages réguliers et Unix

```
>cat fic
chaffars:Chaffar, Soumaya: XXXXX:
drouinta:Drouin-Trempe, Antoine: XXXXX:
leberrej:Le Berre, Jean-François: XXXXX:
leflocam:Le Floch, Amélie: XXXXX:
leplusth:Leplus, Thomas: XXXXX:
mabroukm:Mabrouk, Moez: XXXXX:
merdaoub:Merdaoui, Badis: XXXXX:
morissm:Morissette, Marc-André: XXXXX:
talebikb:Taleb, Ikbal: XXXXX:
tawbebil:Tawbe, Bilal: XXXXX:
ulrichal:Ulrich, Alexis: XXXXX:

>sed -e "s/\([^:]*\):\([^:]*\),\(.*\):\([^:]*\):/\3 \2/g" f
```

# Langages réguliers et Unix

```
>sed -e "s/\([^:]*\):\([^:]*\),\(.*\):\([^:]*\):/\3 \2/g" f
```

```
Soumaya Chaffar
Antoine Drouin-Trempe
Jean-François Le Berre
Amélie Le Floch
Thomas Leplus
Moez Mabrouk
Badis Merdaoui
Marc-André Morissette
Ikbal Taleb
Alexis Ulrich
```

**Note** : simplifiable :

```
>sed -e "s/[^:]*:\([^:]*\),\(.*\):[^:]*:/\2 \1/g" fic
```

rali

# Considérations concernant les langages

- les languages suivants ne sont pas réguliers :
  - $\{a^n b^n / n \geq 1\}$
  - le langage des expressions correctement parenthésées
  - le langage de l'exemple suivant
- un language est non régulier s'il n'existe pas de grammaire de type 3 le décrivant
  - **note :** le fait de trouver une grammaire de type 2 décrivant un language ne signifie pas que le language est de type 2 (on peut peut-être trouver une grammaire de type 3 le décrivant)
- le langage dont les chaînes sur l'alphabet $\{a, b\}$ contiennent un nombre égal (éventuellement nul) de $a$ et de $b$ est un language de type 2 :

$$\begin{aligned} S &\rightarrow aSbS \\ S &\rightarrow bSaS \\ S &\rightarrow \epsilon \end{aligned}$$

# Considérations concernant les langages

- Il existe des langages qui ne peuvent être décrits par une grammaire hors-contexte :

  Ex1 : $\{wcw/w \in (a|b)^*\}$ (ex : $aabcaab$)
  $\hookrightarrow$ mais le langage $\{wcw^R\}$ (ex : $aabcbaa$) est hors-contexte, car ($S \rightarrow aSa|bSb|c$)

  Ex 2 : $\{a^n b^m c^n d^m / n \geq 1, m \geq 1\}$ (ex : $aabbbccddd$)
  $\hookrightarrow$ mais le langage $\{a^n b^m c^m d^n / n \geq 1, m \geq 1\}$ (ex : $aabcdd$) est hors-contexte car :

  $$S \rightarrow aSd|aAd$$
  $$A \rightarrow bAc|bc$$

  Ex 3 : $\{a^n b^n c^n / n \geq 0\}$ (ex : $aabbcc$)
  $\hookrightarrow$ mais le langage $\{a^n b^n / n \geq 1\}$ (ex : $aabb$) est hors-contexte ($S \rightarrow aSb|ab$)
  $\hookrightarrow$ mais le complément de ce langage est hors-contexte


rali

# Considérations concernant les langages

- Exemple de grammaire contextuelle pour le langage $a^n b^n c^n$ :

| | |
|---|---|
| $S \rightarrow aBC$ | $bC \rightarrow bc$ |
| $S \rightarrow SABC$ | $aA \rightarrow aa$ |
| $CA \rightarrow AC$ | $aB \rightarrow ab$ |
| $BA \rightarrow AB$ | $bB \rightarrow bb$ |
| $CB \rightarrow BC$ | $cC \rightarrow cc$ |

- Exemple de dérivation

  $S \Rightarrow SABC \Rightarrow aBCABC \Rightarrow aBACBC \Rightarrow aBABCC \Rightarrow aABBCC \Rightarrow aaBBCC \Rightarrow aabBCC \Rightarrow aabbCC \Rightarrow aabbcC \Rightarrow aabbcc$


rali

# Considérations concernant les langages

**Question :** peut-on représenter une langue naturelle par une grammaire hors-contexte ?

- $S \rightarrow NP\ VP$ ne capture pas l'accord en genre et nombre entre le sujet et le groupe verbal.
  - fonctionne avec des règles comme : $S_{plur} \rightarrow NP_{plur}\, VP_{plur}$
- Dépendances arbitrairement longues [1], où le groupe prépositionnel n'a pas d'objet (implicite) :

  *Whom did Fred give the ball to ?*
  *Whom does Alice believes Fred wants to give the ball to ?*
  - fonctionne en introduisant de nouveaux symboles non terminaux : $S \rightarrow \text{whom}\ S/NP$.
- Deux preuves (non réfutées) semblent infirmer l'hypothèse pour le suisse allemand et le bambara

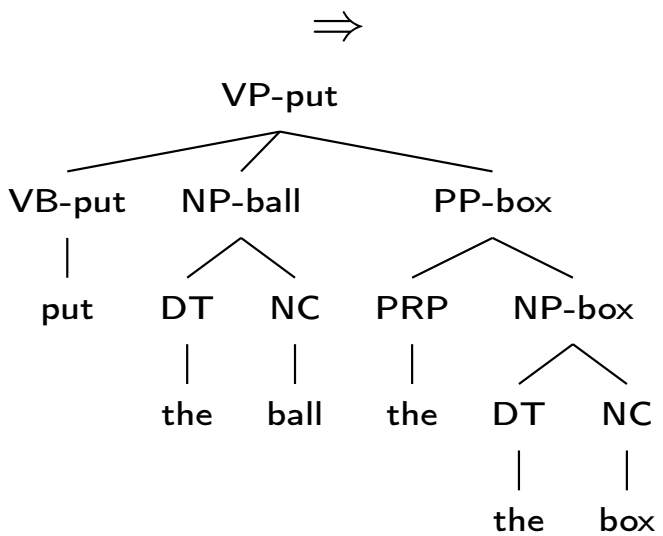
1. Pris dans [Charniak, 1993], p.9

# Plan

# Analyse descendante récursive

- intuitif, simple à mettre en œuvre,
- construit un arbre du haut vers le bas (en pré-ordre)
- le plus simple de la famille des analyseurs LL (lecture de gauche à droite (left-to-right) de l'input à analyser, produit une dérivation gauche (leftmost-derivation))
- Problème (classique) : conversion de formats

```
VP(VERB("put","put").
   NP(DET("the").
      NOUN("ball"),
      "ball").
   PP( PREP("in","in").
       NP(DET("the").
       NOUN("box"),
       "box"),"box") ,"put");
```



# Analyse descendante récursive

input :

```
VP(VERB("put","put").
   NP(DET("the").
      NOUN("ball"),
      "ball").
   PP( PREP("in","in").
       NP(DET("the").
       NOUN("box"),
       "box"),"box")
   ,"put");
```

LaTeX :

```
\begin{forest}
[VP-put
     [VB-put
        [put]]
        [NP-ball
             [DT  [the]]
             [NC [ball]] ]
        [PP-box
             [PRP [the]]
                 [NP-box
                      [DT [the]]
                      [NC [box]]]]]
\end{forest}
```

# Analyse descendante récursive

- Écrire une grammaire de l'input :

$$
\begin{aligned}
&\text{Sent} \rightarrow \text{PH}(\text{Node}) \\
&\text{Node} \rightarrow \text{Cat}(\text{Fils Suite}) \ \text{Frere} \\
&\text{Frere} \rightarrow \epsilon \mid \ . \ \text{Node} \\
&\text{Fils} \rightarrow \text{Mot} \mid \text{Node} \\
&\text{Suite} \rightarrow \epsilon \mid \ , \ \text{Mot} \\
&\text{Cat} \rightarrow \textit{tout sauf un separateur} \\
&\text{Mot} \rightarrow \text{"}\textit{suite de lettres}\text{"}
\end{aligned}
$$

- Écrire une méthode par symbole non-terminal
  - on entre dans une méthode avec le premier symbole de l'input à analyser
  - on sort d'une méthode avec le premier symbole que la méthode n'a pas analysé

# Analyse descendante récursive

- Plus facile si on implémente un analyseur lexical
  - `lex` le prochain **lexème** à analyser (variable globale)
  - `next()` une fonction qui récupère le prochain lexème (réaffecte `lex`)
- Exemple pour la règle : Sent $\rightarrow$ PH(Node) :

```
methode sent() {
  if (lex != "PH") Error("PH attendu");
   next();
   if (lex != "(") Error("( attendue");
   next();
   node();
   if (lex != ")") Error(") attendue");
}
```

- **Note :** l'émission d'une erreur systématique enlève la possibilité de backtrack

# Analyse descendante récursive : problèmes

- Ne fonctionne qui si on est capable de décider à tout moment de la bonne action à prendre au cours de l'analyse (la grammaire doit être $LL_1$).
- Une alternative avec retour-arrière (back-tracking) est possible mais inefficace.

$$\begin{aligned} S &\rightarrow c\ A\ d \\ A &\rightarrow a\ b \\ A &\rightarrow a \end{aligned}$$

  - chaîne à analyser $w = cad$, alors :

$$S \underset{1}{\Rightarrow} cAd \underset{2}{\Rightarrow} cabd \text{ (\textbf{blocage})}$$
$$\underset{3}{\Rightarrow} cad$$

- La factorisation gauche peut aider
  - **Idée :** si $A \rightarrow \alpha\beta_1|\alpha\beta_2$, alors réécrire ces deux règles avec : $A \rightarrow \alpha A'$ et $A' \rightarrow \beta_1|\beta_2$

# Analyse descendante récursive : problèmes

- **Définition :** Une grammaire est récursive gauche s'il existe (au moins) un $A$ tel que $A \stackrel{\star}{\Rightarrow} A\alpha$.
  - Exemple de récursion (gauche) directe :

    $$A \rightarrow A + B$$

  - Exemple de récursion (gauche) indirecte :

    $$A \rightarrow B\ a\ C$$
    $$B \rightarrow A\ b$$

- Un analyseur descendant boucle (ou ne peut gérer) une grammaire récursive gauche
- Il existe un algorithme pour rendre une grammaire non récursive à gauche.

  **intuition :** On remplace les règles $A \rightarrow A\alpha|\beta$ par :

$$\begin{array}{lcl} A & \rightarrow & \beta A' \\ A' & \rightarrow & \alpha A'|\epsilon \end{array}$$

# Analyseur prédictif non récursif

**Input :** $w$ : chaîne à analyser, $M$ : table d'analyse, $S$ : axiome
**Output :** dériv. gauche si $w$ appartient au langage, erreur sinon

push \$ ; push S ; $I = w\$$ ; $ip \leftarrow 1$
**repeat**
  $X \leftarrow top()$ et $a \leftarrow I[ip]$
  **if** $X$ est un terminal ou \$ **then**
    **if** $X = a$ **then** POP(X) ; ip++ **else** error()
  **else**
    **if** $M[X, a] = X \rightarrow Y_1 Y_2 \dots Y_k$ **then**
      pop $X$
      push $Y_k Y_{k-1} \dots Y_1$
      output $X \rightarrow Y_1 Y_2 \dots Y_k$
    **else**
      error()
**until** X = \$ /* pile vide */

# Analyseur prédictif non récursif

- Grammaire (non récursive gauche) :

$$E \to TE' \qquad E' \to +TE' \mid \epsilon \qquad T \to FT'$$
$$T' \to \times FT' \mid \epsilon \qquad F \to (E) \mid id$$

- Table d'analyse associée :

| | Input symbols | | | | | |
|---|---|---|---|---|---|---|
| N | id | + | × | ( | ) | $ |
| E | $E \to TE'$ | | | $E \to TE'$ | | |
| E' | | $E' \to +TE'$ | | | $E' \to \epsilon$ | $E' \to \epsilon$ |
| T | $T \to FT'$ | | | $T \to FT'$ | | |
| T' | | $T' \to \epsilon$ | $T' \to \times FT'$ | | $T' \to \epsilon$ | $T' \to \epsilon$ |
| F | $F \to id$ | | | $F \to (E)$ | | |

- Chaîne à analyser : $id + id \times id$

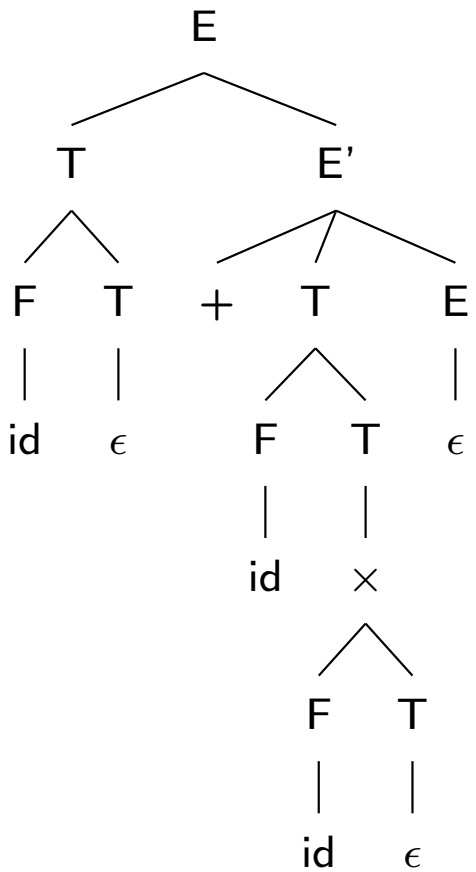

rali

# Analyseur prédictif non récursif

| pile | input | output |
|---|---|---|
| \$E | id + id × id \$ | |
| \$E'T | id + id × id \$ | $E \rightarrow TE'$ |
| \$E'T'F | id + id × id \$ | $T \rightarrow FT'$ |
| \$E'T'id | id + id × id \$ | $F \rightarrow id$ |
| \$E'T' | + id × id \$ | |
| \$E' | + id × id \$ | $T' \rightarrow \epsilon$ |
| \$E'T+ | + id × id \$ | $E' \rightarrow +TE'$ |
| \$E'T | id × id \$ | |
| \$E'T'F | id × id \$ | $T \rightarrow FT'$ |
| \$E'T'id | id × id \$ | $F \rightarrow id$ |
| \$E'T' | × id \$ | |
| \$E'T'F× | × id \$ | $T' \rightarrow \times FT'$ |
| \$E'T'F | id \$ | |
| \$E'T'id | id \$ | $F \rightarrow id$ |
| \$E'T' | \$ | |
| \$E' | \$ | $T' \rightarrow \epsilon$ |
| \$ | \$ | $E' \rightarrow \epsilon$ |
| succès | | |

# Analyseur prédictif non récursif

La liste des règles produites indique une dérivation gauche produisant la chaine




$\Rightarrow id + (id \times id)$

# Construction de la table d'analyse

Deux opérateurs définissant des ensembles à valeur dans $T$ (les symboles terminaux) :

FIRST($\alpha$) : pour toute chaîne $\alpha$ de terminaux et non-terminaux
C'est l'ensemble des terminaux qui commencent les chaînes dérivées de $\alpha$.
cad : $\{a \in T / \alpha \stackrel{\star}{\Rightarrow} a\beta\}$
Si $\alpha \stackrel{\star}{\Rightarrow} \epsilon$ alors $\epsilon$ est ajouté à FIRST($\alpha$).

FOLLOW($A$) : pour tout non-terminal
C'est l'ensemble des terminaux qui peuvent apparaître dans les $S$-dérivations directement à droite de $A$.
cad : $\{a \in T / \exists\, S \stackrel{\star}{\Rightarrow} \alpha A a \beta\}$
Si $A$ peut être le symbole le plus à droite dans une $S$-dérivation, alors ajouter \$ à FOLLOW($A$).

rali

# Construction de la table d'analyse

**Idée :** S'il existe $A \rightarrow \alpha$ et que $a$ est dans FIRST($\alpha$) alors appliquer $A \rightarrow \alpha$ à la lecture de $a$.

**Input :** Grammaire $G$
**Output :** Table d'analyse M

1. Pour chaque production $A \rightarrow \alpha$ appliquer 2 et 3
2. Pour chaque terminal $a$ dans FIRST($\alpha$), $M[A, a]$ += $A \rightarrow \alpha$
3. Si $\epsilon$ est dans FIRST($\alpha$), alors $M[A, b]$ += $A \rightarrow \alpha$ pour tout terminal $b$ dans FOLLOW($A$).

   Si $\epsilon$ est dans FIRST($\alpha$) et que \$ est dans FOLLOW($A$) alors $M[A, \$]$ += $A \rightarrow \alpha$
4. Les entrées vides sont des erreurs.

# Calcul de FIRST

1. Appliquer en boucle les règles suivantes, tant que l'on effectue des ajouts :
   - si $X \in T$, alors $\text{FIRST}(X) = \{X\}$
   - si $X \to \epsilon$ alors ajouter $\epsilon$ à $\text{FIRST}(X)$
   - si $X \to Y_1 Y_2 \dots Y_k$ alors ajouter $\text{FIRST}(Y_1)$ (sauf $\epsilon$) à $\text{FIRST}(X)$.
     - si $\epsilon$ est dans $\text{FIRST}(Y_1)$ alors ajouter $\text{FIRST}(Y_2)$ à $\text{FIRST}(X)$ ; etc.
     - si $\epsilon$ est dans tous les $Y_i$ alors ajouter $\epsilon$ à $\text{FIRST}(X)$.
2. Pour toute séquence $X = X_1 X_2 \dots X_k$ : ajouter tous les symboles (sauf $\epsilon$) de $\text{FIRST}(X_1)$ à $\text{FIRST}(X)$.
   - si $\epsilon$ est dans $\text{FIRST}(X_1)$, alors ajouter les non-$\epsilon$ symboles de $\text{FIRST}(X_2)$ à $\text{FIRST}(X)$ ; etc.
   - si $\epsilon \in \text{FIRST}(X_i)$, $\forall i \in [1, k]$ alors ajouter $\epsilon$ à $\text{FIRST}(X)$

rali

# Calcul de FIRST : exemple

$$\begin{array}{lll} E \to T\ E' & E' \to +\ T\ E' & E' \to \epsilon \\ T \to F\ T' & T' \to \times\ F\ T' & T' \to \epsilon \\ F \to (E) & F \to \mathrm{id} & \end{array}$$

On a (par exemple) :

$$\begin{aligned} \mathrm{FIRST}(E) &= \mathrm{FIRST}(T) = \mathrm{FIRST}(F) = \{(, id\} \\ \mathrm{FIRST}(E') &= \{+, \epsilon\} \\ \mathrm{FIRST}(T') &= \{\times, \epsilon\} \\ \mathrm{FIRST}(E'T') &= \{+, \times, \epsilon\} \end{aligned}$$

# Calcul de FOLLOW

Appliquer tant que l'on peut ajouter des symboles :

1. Mettre \$ dans FOLLOW($S$) où $S$ est l'axiome de départ
2. si $A \rightarrow \alpha B \beta$, alors mettre tous les non-$\epsilon$ symboles de FIRST($\beta$) dans FOLLOW(B)
3. si il existe $A \rightarrow \alpha B$ ou $A \rightarrow \alpha B \beta$ avec $\epsilon \in$ FIRST($\beta$), alors tout ce qui est dans FOLLOW($A$) est dans FOLLOW($B$).

Exemple :

$$\begin{aligned} \text{FOLLOW(E)} &= \text{FOLLOW(E')} = \{), \$\} \\ \text{FOLLOW(T)} &= \text{FOLLOW(T')} = \{+, ), \$\} \\ \text{FOLLOW(F)} &= \{+, \times, ), \$\} \end{aligned}$$


rali

# Grammaire LL1

- **Définition :** Une grammaire pour laquelle on peut calculer une table d'analyse de telle manière qu'il n'existe au plus qu'une seule règle dans chaque case de $M$ est dite LL(1).
  - premier L est pour left-to-right,
  - second L car on produit une dérivation gauche
  - le chiffre entre parenthèse est ce qu'on appelle le look ahead.
- **Propriété :** Une grammaire LL1 ne peut pas être ambiguë ou récursive à gauche.
- **Propriété :** G est LL1 **ssi** pour toute paire de productions $A \rightarrow \alpha|\beta$ :
  1. Il n'existe pas de termimal $a$ tel que $\alpha \stackrel{\star}{\Rightarrow} a\alpha_1$ **et** $\beta \stackrel{\star}{\Rightarrow} a\beta_1$
  2. Au plus $\alpha$ ou $\beta$ peut dériver $\epsilon$
  3. Si $\beta \stackrel{\star}{\Rightarrow} \epsilon$, alors $\alpha$ ne peut pas dériver une chaîne qui débute par un terminal dans FOLLOW($A$).

rali

# Plan



rali

# Analyse par décalage réduction (Shift-Reduce Parsing)

- Technique d'analyse montante (bottom-up) qui n'a pas de problème avec les grammaires récursives à gauche.
- Utilise une file de lecture $I$, une pile $P$ et une table d'analyse comportant l'une des 4 actions :
  - shift du symbole en tête de $I$ sur le sommet de la pile $P$.
  - reduction de la séquence $s$ en sommet de pile par une partie gauche de règle qui a $s$ pour partie droite
  - accept si $I$ est vide et $P$ contient l'axiome
  - erreur

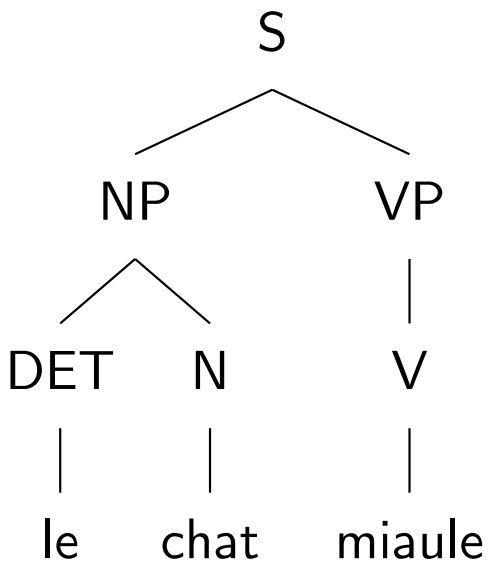

rali

# Exemple

| $P$ | $I$ | action |
|---|---|---|
| | le chat miaule | shift |
| le | chat miaule | reduce DET $\rightarrow$ le |
| DET | chat miaule | shift |
| DET,chat | miaule | reduce N $\rightarrow$ chat |
| DET,N | miaule | reduce NP $\rightarrow$ DET N |
| NP | miaule | shift |
| NP,miaule | | reduce V $\rightarrow$ miaule |
| NP,V | | reduce VP $\rightarrow$ V |
| NP,VP | | reduce S $\rightarrow$ NP VP |
| S | | accept |

# Exemple

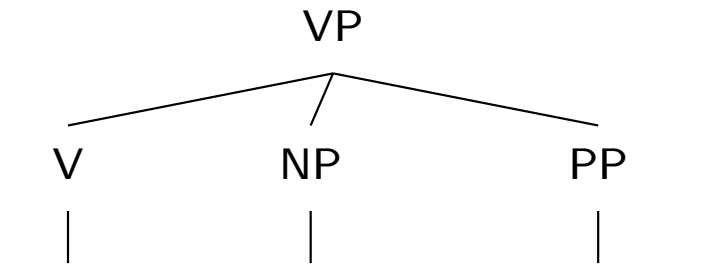
- Arbre associé :
  - on lit les réductions en commençant par le bas
  - cela indique une dérivation droite de la chaîne analysée

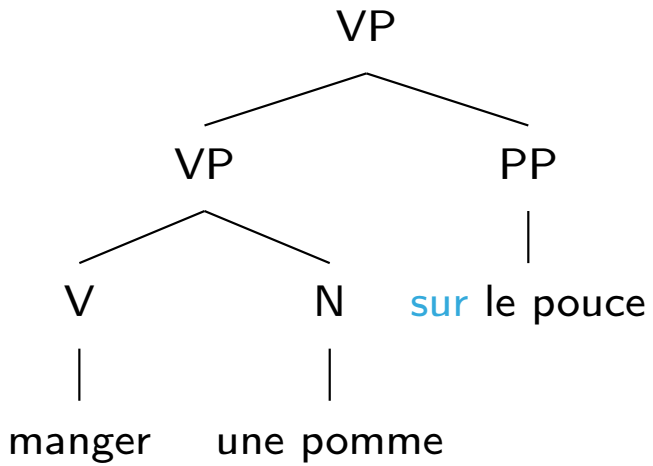


# Conflit Shift-Reduce : réduire ou décaler ?

$$VP \rightarrow V\ NP\ PP \mid VP \rightarrow VP\ PP \mid VP \rightarrow V\ NP$$

| *P* | *I* | action |
|---|---|---|
| V,NP | sur ... | shift ou reduce VP → V NP ? |

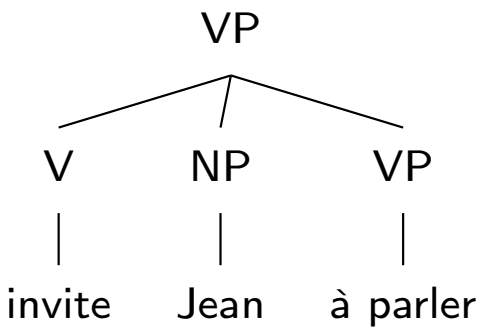


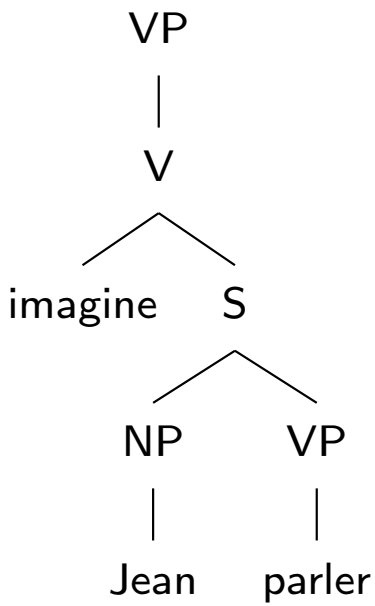



# Conflit Reduce-Reduce : quelle réduction ?

$$VP \rightarrow V\ NP\ VP \mid VP \rightarrow V\ S \mid S \rightarrow NP\ VP$$

| *P* | *I* | action |
|---|---|---|
| V,NP,VP | | reduce $VP \rightarrow V\ NP\ VP$ ou $S \rightarrow NP\ VP$ ? |



# Analyse LR(k)

- Famille d'analyseurs utilisée pour compiler les languages de programmation. Voir des outils comme Yacc.
- **Principe :** pré-calculer (sous forme d'un automate) toutes les substitutions possibles d'une forme S-dérivée que l'on pourra rencontrer lors de l'analyse d'une chaîne étant donnée une grammaire. Cet automate fournit toutes les informations dont on a besoin pour compiler une table d'analyse LR.
- **Note :** pas une technique générique. La construction de la table peut mener à des conflits qu'il faut gérer (ce qui est "facile" dans le cas d'un language de programmation, mais pas pour le language naturel)

rali

# Analyse LR(k) : construction de la table

- La construction de l'automate consiste à construire des ensembles d'items.
- L'opération de base permettant de construire un ensemble d'items est la fermeture.
  - La fermeture d'un item $I \equiv \alpha \bullet A\beta$ est définie par l'ensemble des items $A \rightarrow \bullet\gamma$ que l'on peut créer à partir des règles de G de la forme : $A \rightarrow \gamma$
  - La fermeture d'un ensemble d'items est constituée de l'union de la fermeture de chacun de ses items.
- Les états de l'automate sont les ensembles d'items construits. Les transitions correspondent à tout symbole (terminal ou non) se situant directement à droite de $\bullet$ dans un ensemble d'items.

# Analyse LR(k) : construction de la table

$$\{\{E, T\}, \{id, (, ), +)\}, E, \{\mathrm{E} \to \mathrm{E} + \mathrm{T}, \mathrm{E} \to \mathrm{T}, \mathrm{T} \to \mathrm{id}, \mathrm{T} \to (\mathrm{E})\}\}$$

- Le premier ensemble ($I_0$) contient l'item S$\to \bullet E\$$ où $S$ est un méta-axiome et sa fermeture.
- Sur la lecture d'une parenthèse ouvrante (par exemple), on obtient un nouvel ensemble d'items ($I_6$).

$I_0$

| |
|---|
| $\mathrm{S} \to \bullet\ \mathrm{E}$ |
| $\mathrm{E} \to \bullet\ \mathrm{E} + \mathrm{T}$ |
| $\mathrm{E} \to \bullet\ \mathrm{T}$ |
| $\mathrm{T} \to \bullet\ \mathrm{id}$ |
| $\mathrm{T} \to \bullet\ (\mathrm{E})$ |

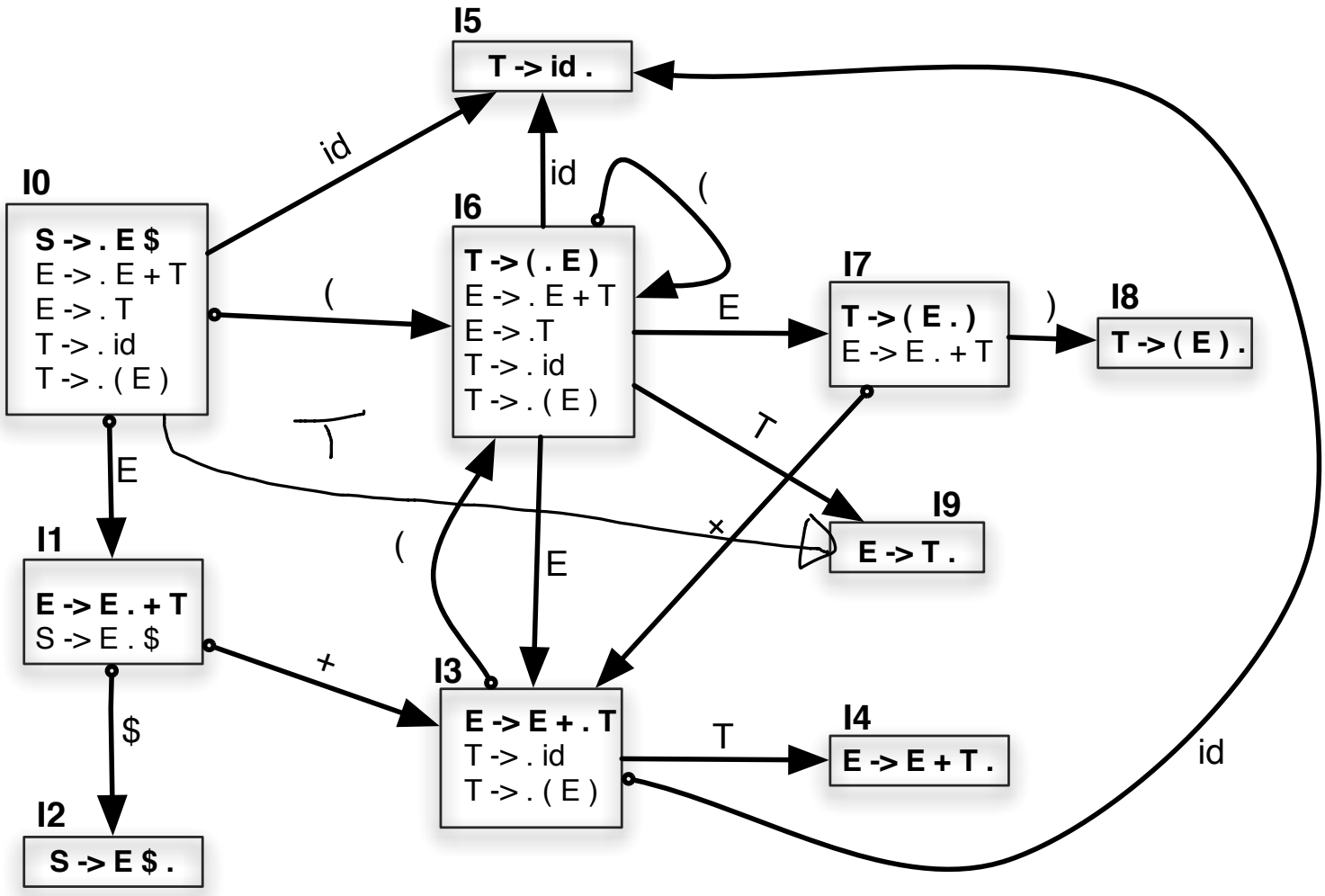
$I_6$

| |
|---|
| $\mathrm{T} \to (\bullet\ \mathrm{E})$ |
| $\mathrm{E} \to \bullet\ \mathrm{E} + \mathrm{T}$ |
| $\mathrm{E} \to \bullet\ \mathrm{T}$ |
| $\mathrm{T} \to \bullet\ \mathrm{id}$ |
| $\mathrm{T} \to \bullet\ (\mathrm{E})$ |

rali

# Analyse LR(k) : construction de la table

En continuant ce processus, on obtient un automate :



# Analyse LR(k) : construction de la table SLR(0)

- Une transition de l'état $I$ vers l'état $I'$ sur un symbole terminal $a$ correspond à une action : action[I,a] = shift-I'
- Une transition de l'état $I$ vers l'état $I'$ sur un symbole non-terminal $A$ correspond à un goto : goto[I,A] = I'
- Pour tous les états $I$ contenant un item $r \equiv A \rightarrow \alpha \bullet$, ajouter l'action :
  action[I,a] = Réduire-$r$ pour tout $a \in T$
- Pour tout état $I$ contenant l'item $S \rightarrow \alpha \bullet$ ajouter l'action :
  action[I,a] = accept pour tout $a \in T$
- Les cases vides sont des erreurs d'analyse

Une grammaire pour laquelle une telle table peut être construite sans qu'aucune case ne contienne deux actions est dite LR(0).

# Analyse LR(k) : construction de la table SLR(0)

2 $E \rightarrow E + T$
3 $E \rightarrow T$
4 $T \rightarrow id$
5 $T \rightarrow (E)$

| | id | ( | ) | + | $ | S | E | T |
|---|---|---|---|---|---|---|---|---|
| 0 | S5 | S6 | | | | | 1 | 9 |
| 1 | | | | S3 | S2 | | | |
| 2 | accept | | | | | | | |
| 3 | S5 | S6 | | | | | | 4 |
| 4 | R2 | R2 | R2 | R2 | R2 | | | |
| 5 | R4 | R4 | R4 | R4 | R4 | | | |
| 6 | S5 | S6 | | | | | 7 | 9 |
| 7 | | | S8 | S3 | | | | |
| 8 | R5 | R5 | R5 | R5 | R5 | | | |
| 9 | R3 | R3 | R3 | R3 | R3 | | | |


rali

# Analyse LR(k)

push(0) // état initial

**loop**
  s ← top()
  **if** (action[s,first(w)] == Shift-n) **then**
    push(n) ; remove-first(w)
  **else if** (action[s,first(w)] == Reduce-rule) **then**
    soit *rule* ≡ A → $A_1 \ldots A_k$
    k times **do** pop()
    push(goto[top(),A])
  **else if** (action[s,first(w)] == accept) **then**
    succes()
  **else**
    error()


rali

# Analyse LR(k)

| pile | input | actions | | |
|---|---|---|---|---|
| 0/S | id + (id + id) | S5 | | |
| 0/S 5/id | + (id + id) | R4 | T → id | goto 9 |
| 0/S 9/T | + (id + id) | R3 | E → T | goto 1 |
| 0/S 1/E | + (id + id) | S3 | | |
| 0/S 1/E 3/+ | (id + id) | S6 | | |
| 0/S 1/E 3/+ 6/( | id + id) | S5 | | |
| 0/S 1/E 3/+ 6/( 5/id | + id) | R4 | T → id | goto 9 |
| 0/S 1/E 3/+ 6/( 9/T | + id) | R3 | E → T | goto 1 |
| 0/S 1/E 3/+ 6/( 1/E | + id) | S3 | | |
| 0/S 1/E 3/+ 6/( 1/E 3/+ | id) | S5 | | |
| 0/S 1/E 3/+ 6/( 1/E 3/+ 5/id | ) | R4 | T → id | goto 4 |
| 0/S 1/E 3/+ 6/( 1/E 3/+ 4/T | ) | R2 | E → E+T | goto 7 |
| 0/S 1/E 3/+ 6/( 7/E | ) | S8 | | |
| 0/S 1/E 3/+ 6/( 7/E 8/) | $ | R5 | T → (E) | goto 4 |
| 0/S 1/E 3/+ 4/T | $ | R2 | E → E+T | goto 1 |
| 0/S 1/E | $ | S2 | | |
| 0/S 1/E 2/$ | $ | accept | | |

# Analyse LR(k) : illustration d'un conflit reduce-reduce

- Si dans un ensemble d'items, deux items sont de la forme A $\rightarrow \alpha \bullet$ alors, nous avons un conflit reduce-reduce.
- On peut tenter d'éviter le conflit en changeant la façon de construire la table $\rightarrow$ tables SLR(1) :
  - pour un item A $\rightarrow \alpha \bullet$ dans un ensemble I, ajouter pour tout *a* dans FOLLOW(A) :
    action[I,a] = reduce-A $\rightarrow \alpha$
- Si le conflit n'est toujours pas résolu, alors il existe d'autres heuristiques pour construire la table (LALR, LR(k)). Si le conflit persiste, l'expert doit trancher.

rali


rali

# Plan



rali

# CYK : un parseur "universel"

## *[Cocke, 1965, Younger, 1967, Kasami, 1965]*

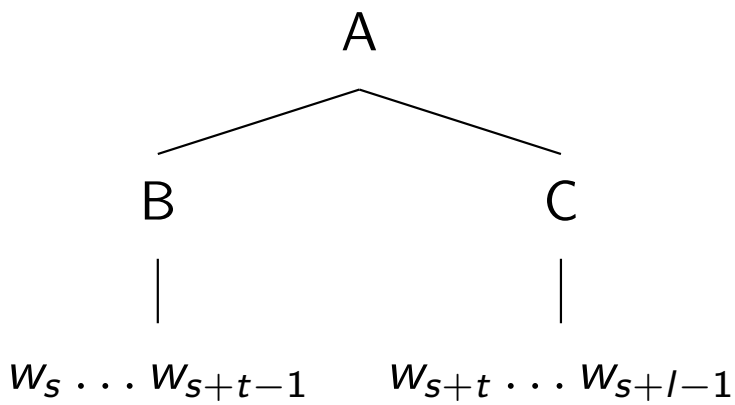
- Dans sa forme la plus populaire, les grammaires analysées sont exprimées en forme normale de Chomsky (CNF)[2] : $A \rightarrow BC|a$
- **Structure centrale :** un tableau de booléens à trois dimensions dont chaque élément $\langle i, A, j \rangle$ est vrai *ssi* $A \stackrel{\star}{\Rightarrow} w_i \dots w_{j-1}$.
- On suppose dans l'énoncé de l'algorithme qui suit qu'il y a $N$ non terminaux et que la chaîne à analyser contient $n$ mots : $w_1 \dots w_n$.
- On cherche à vérifier si : $\langle 1, S, n+1 \rangle$ est vrai (analyse réussie) ou pas (analyse ratée).
- **Note :** C'est un algorithme bottom-up.

2. Toute grammaire CF peut-être exprimée par une grammaire CNF qui lui est faiblement équivalente : même langage engendré, dérivations différentes.


rali

# CYK : Idée générale

- **Définition :** Un span est un ensemble de positions adjacentes dans la chaîne à analyser qui sont dominées par un symbole non-terminal ($A \stackrel{\star}{\Rightarrow} w_i \dots w_j$ est un span de longueur $j - i + 1$)
- CYK considère tous les regroupements de spans adjacents en commençant par les plus petits :




- Dans l'algorithme suivant, un span ($A \stackrel{\star}{\Rightarrow} w_i \dots w_j$) est dénoté par : $\langle i, A, j+1 \rangle$


rali

# CYK [3]

// init
$\langle 1..n, 1..N, 1..n+1 \rangle \leftarrow$ false
// règles lexicales
**for** $s \leftarrow 1$ à $n$ **do**
  **for all** rule $A \rightarrow w_s$ **do**
    $\langle s, A, s+1 \rangle \leftarrow$ true
// règles internes
**for all** length $l$, shortest (2) to longest ($n$) **do**
  **for all** valid start $s \in [1, n]$ **do**
    **for all** split length $t$ **do**
      **for all** rule $A \rightarrow BC$ **do**
        $\langle s, A, s+l \rangle \leftarrow \langle s, A, s+l \rangle \vee$
          $(\langle s, B, s+t \rangle \wedge \langle s+t, C, s+l \rangle)$

**return** $\langle 1, S, n+1 \rangle$

3. Pour grammaire CNF. Je reprends ici l'algorithme tel que décrit dans [Goodman, 1998]

# CYK : Exemple

| | |
|---|---|
| $S \rightarrow NP\ VP$ | $Det \rightarrow a$ |
| $NP \rightarrow Det\ N$ | $N \rightarrow circle \mid square \mid triangle$ |
| $VP \rightarrow VT\ NP$ | $VT \rightarrow touches$ |
| $VP \rightarrow VI\ PP$ | $VI \rightarrow is$ |
| $PP \rightarrow P\ NP$ | $P \rightarrow above \mid below$ |

Chaîne : *a circle touches a triangle*

$l = 1$ initialiser à vrai les cinq cellules :
$\langle 1, Det, 2\rangle, \langle 2, N, 3\rangle, \langle 3, VT, 4\rangle, \langle 4, Det, 5\rangle, \langle 5, N, 6\rangle$

$l = 2$ $\langle 1, NP, 3\rangle, \langle 4, NP, 6\rangle$.

$l = 3$ $\langle 3, VP, 6\rangle$

$l = 4$ $\langle 1, S, 6\rangle \leftarrow$ analyse réussie


rali

# CYK : Table d'analyse

On peut ranger les spans dans une table d'analyse où la case $(i, j)$ contient le non-terminal $A$ ssi $A \stackrel{\star}{\Rightarrow} w_i \ldots w_j$

| | | | | | |
|---|---|---|---|---|---|
| triangle | S | | VP | NP | N |
| a | | | | Det | |
| touches | | | VT | | |
| circle | NP | N | | | |
| a | Det | | | | |
| | a | circle | touches | a | triangle |


rali

# Earley *[Earley, 1968]*

- un item $[i, A \rightarrow \alpha \bullet \beta, j]$, avec la sémantique suivante :
  - la position courante dans la chaîne d'entrée est $i$ : $w_1 \ldots w_{i-1}$ a déjà été analysé.
  - la règle $A \rightarrow \alpha\beta$ est considérée. Elle a été appliquée en position $j$ et $\alpha \stackrel{\star}{\Rightarrow} w_j \ldots w_{i-1}$.
- un état $S_i$ qui constitue l'ensemble des tentatives viables pour une position de l'entrée donnée $i \in [0, n]$ :

$$S_i = \{[i, A \rightarrow \alpha \bullet \beta, j], \forall A, \alpha, \beta, j\}$$

**Idée :** Construire les états $S_i$. L'analyse est positive si $S_n$ contient l'item $[n, S' \rightarrow S\bullet, 0]$, négative sinon. $S'$ est un non terminal ajouté à la grammaire (nouvel axiome).


rali

# Earley : Trois opérations

prédiction : pour tout item $[i, X \to \lambda \bullet Y\mu, k]$ et pour toute règle $Y \to \alpha$ ajouter $[i, Y \to \bullet\alpha, i]$

lecture : pour tout item $[i, X \to \lambda \bullet a\mu, k]$ où $a$ est le terminal $w_i$
ajouter $[i+1, X \to \lambda a \bullet \mu, k]$

complétion : pour tout item $[i, Y \to \alpha\bullet, j]$ et pour tout item qui a $Y$ à droite du $\bullet$ dans l'état $j$ $(\leq i)$ :
$[j, X \to \lambda \bullet Y\mu, k]$
ajouter $[i, X \to \lambda Y \bullet \mu, k]$

**Départ :** $[0, S' \to \bullet S, 0]$
**But :** $[n, S' \to S\bullet, 0]$

# Earley : Exemple (input : *a circle touches a triangle*)

| a | circle | touches | a |
|---|---|---|---|
| start : | lecture : | lecture : | lecture : |
| $[S' \rightarrow \bullet S, 0]$ | [Det $\rightarrow a\bullet$, 0] | $[N \rightarrow$ circle$\bullet$, 1] | $[VT \rightarrow$ touches$\bullet$, 2] |
| prédiction : | complétion : | complétion : | complétion : |
| [S $\rightarrow \bullet$NP VP, 0] | $[NP \rightarrow$ Det $\bullet$ $N$, 0] | [NP $\rightarrow$ Det N$\bullet$, 0] | $[VP \rightarrow VT \bullet NP, 2]$ |
| [NP $\rightarrow \bullet$Det N, 0] | prédiction : | $[S \rightarrow NP \bullet VP, 0]$ | prédiction : |
| [Det $\rightarrow \bullet$a, 0] | $[N \rightarrow \bullet$circle, 1] | prédiction : | $[NP \rightarrow \bullet$Det N, 3] |
| | $[N \rightarrow \bullet$square, 1] | $[VP \rightarrow \bullet$VT NP, 2] | [Det $\rightarrow \bullet a$, 3] |
| | $[N \rightarrow \bullet$triangle, 1] | $[VP \rightarrow \bullet$VI PP, 2] | |
| | | $[VT \rightarrow \bullet$touches, 2] | |
| | | $[VI \rightarrow \bullet$is, 2] | |
| $S_0$ | $S_1$ | $S_2$ | $S_3$ |

| triangle | |
|---|---|
| lecture : | lecture : |
| [Det $\rightarrow a\bullet$, 3] | $[N \rightarrow$ triangle$\bullet$, 4] |
| complétion : | complétion : |
| $[NP \rightarrow$ Det $\bullet$ $N$, 3] | [NP $\rightarrow$ Det N$\bullet$, 3] |
| prédiction : | $[VP \rightarrow$ VT NP$\bullet$, 2] |
| $[N \rightarrow \bullet$circle, 4] | [S $\rightarrow$ NP VP$\bullet$, 0] |
| $[N \rightarrow \bullet$square, 4] | $[S' \rightarrow S\bullet, 0]$ |
| $[N \rightarrow \bullet$triangle, 4] | |
| $S_4$ | $S_5$ |

| | |
|---|---|
| S $\rightarrow$ NP VP | Det $\rightarrow$ a |
| NP $\rightarrow$ Det N | VT $\rightarrow$ touches |
| VP $\rightarrow$ VT NP | VI $\rightarrow$ is |
| VP $\rightarrow$ VI PP | N $\rightarrow$ circle |
| PP $\rightarrow$ P NP | N $\rightarrow$ square |
| P $\rightarrow$ above | N $\rightarrow$ triangle |
| P $\rightarrow$ below | |

rali

# Plan

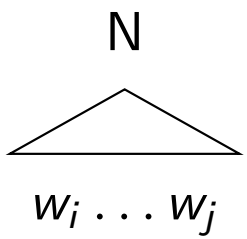
# Définition

- Une grammaire probabiliste hors-contexte (PCFG) est définie par un 5-uplet $< N, T, R, S, P >$ où :
  - $N$ est l'ensemble des symboles *non-terminaux*
  - $T$ est l'ensemble des symboles *terminaux*
  - $R$ est l'ensemble des règles $r_i$ de la forme : $A \rightarrow \beta$
  - $S$ est l'axiome de départ
  - $P$ est l'ensemble des probabilités $p_i$ associées aux règles $r_i$ telles que : $\sum_\beta p(A \rightarrow \beta) = 1, \quad \forall A \in N$

- Rappel notationnel :
  - $\beta, \alpha, \ldots \in (N \vee T)*$
  - $A, B, \ldots \in N$
  - $a, b, \ldots \in T$
  - $\{N^1, N^2, \ldots, N^N\} = N$
  - $w_1^n = w_1 w_2 \ldots w_n$ est la chaîne à analyser (input)

# Vocabulaire

- On dit que $N$ **domine** la chaîne $w_i \ldots w_j$ si $N \stackrel{\star}{\Rightarrow} w_i \ldots w_j$, ce que l'on peut représenter graphiquement par :




- Pour spécifier que $N$ **s'étend** sur les mots d'indices $i$ à $j$ dans une chaîne, sans pour autant faire mention des mots eux-mêmes, on utilise la notation : $N_{ij}$. On parle également de span.
- $N^k_{ij}$ Signifie de même que le non terminal $N^k$ s'étend sur les symboles de $i$ à $j$.

# Quelques propriétés des PCFGs

invariance positionnelle : la probabilité d'un sous-arbre dominant des mots consécutifs dans une chaîne ne dépend pas de la position de ces mots dans la chaîne.

hors-contexte : la probabilité d'un sous arbre ne dépend pas de mots qui ne sont pas dominés par ce sous-arbre.

ingratitude parentale : la probabilité d'un sous-arbre ne dépend d'aucun nœud à l'extérieur du sous-arbre.

La probabilité d'un arbre syntaxique est obtenue en multipliant la probabilité de chaque règle utilisée à chaque nœud de l'arbre.

rali

# Exemple de CFG probabilisée

| | | |
|---|---|---|
| S → NP VP 1.0 | P → *with* 1.0 | NP → *ears* 0.18 |
| PP → P NP 1.0 | V → *saw* 1.0 | NP → *saw* 0.04 |
| VP → V NP 0.7 | NP → NP PP 0.4 | NP → *stars* 0.18 |
| VP → VP PP 0.3 | NP → *astronomers* 0.1 | NP → *telescopes* 0.1 |

$S = w_1^5 =$ astronomers saw stars with ears.

- Cette phrase contient deux analyses possibles dont les interprétations sont :

  $t_1$ : les astronomes ont vu des étoiles qui avaient des oreilles

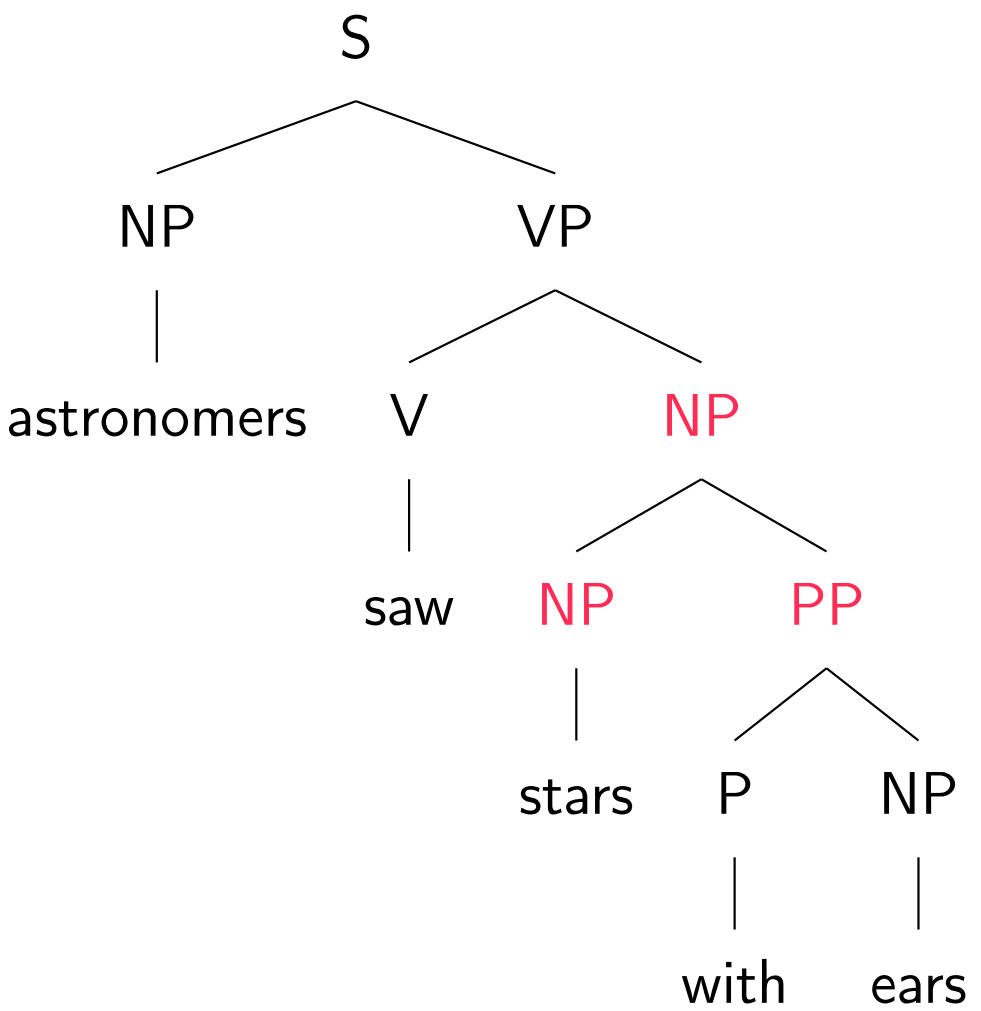
  $t_2$ : les astronomes ont vu des étoiles en utilisant leurs oreilles

- Quelle interprétation préférez-vous ?

rali

# Les étoiles ont-elles des oreilles ?

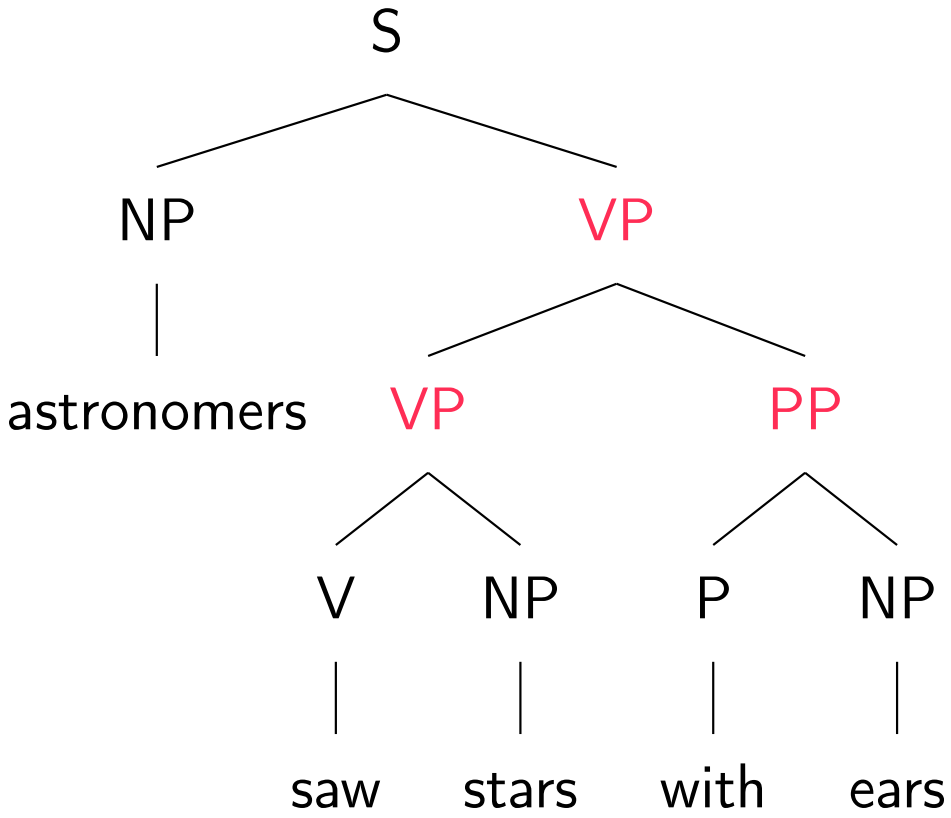
$t_1 =$




$$
\begin{aligned}
p(t_1) &= 1.0 \times \\
&\quad 0.1 \times \\
&\quad 0.7 \times \\
&\quad 1.0 \times \\
&\quad 0.4 \times \\
&\quad 0.18 \times \\
&\quad 1.0 \times \\
&\quad 1.0 \times \\
&\quad 0.18 \\
&= 0.0009072
\end{aligned}
$$

# Les astronomes se servent-ils de leurs oreilles pour mieux voir ?

$t_2 =$




$$
\begin{aligned}
p(t_2) &= 1.0 \times \\
&\quad 0.1 \times \\
&\quad 0.3 \times \\
&\quad 0.7 \times \\
&\quad 1.0 \times \\
&\quad 0.18 \times \\
&\quad 1.0 \times \\
&\quad 1.0 \times \\
&\quad 0.18 \\
&= 0.0006804
\end{aligned}
$$

**Note :** les probabilités de ces deux arbres ne diffèrent que par la probabilité des règles NP $\rightarrow$ NP PP et VP $\rightarrow$ VP PP

# Avantage des PCFG sur les CFG

- potentiel supérieur des PCFG pour la désambiguisation,
- tolérance plus grande aux phrases non grammaticales (mais compréhensibles)
- apprentissage avec des algorithmes à partir d'exemples positifs
- utilisation possible comme modèle de langue (en combinaison possible avec un modèle n-gramme).

rali

# Désambiguïser

Notre grammaire "préfère" la première interprétation si nous acceptons l'idée suivante pour désambiguïser :

$$
\begin{aligned}
\hat{t} &= \text{argmax}_{t \in \tau(S)}\, p(t|S) \\
&= \text{argmax}_{t \in \tau(S)}\, \frac{p(t,S)}{p(S)} \\
&= \text{argmax}_{t \in \tau(S)}\, p(t,S) \\
&= \text{argmax}_{t \in \tau(S)}\, p(t) \times \underbrace{P(S|t)}_{1} = \text{argmax}_{t \in \tau(S)}\, p(t)
\end{aligned}
$$

où $\tau(S)$ désigne l'ensemble des arbres pouvant générer $S$.
**Cependant :** les PCFGs ne peuvent que s'appuyer sur le fait qu'une construction est plus importante qu'une autre dans le corpus.

# Induction

- On peut apprendre automatiquement les paramètres (probabilités des règles et/ou les règles elles-mêmes) d'une PCFG à partir d'exemple positifs (*ie* un corpus de phrases correctement constituées).
- Il a été démontré que cela n'était pas possible pour l'induction de grammaires hors contexte non probabilisées : il faut des exemples négatifs (voir [Charniak, 1993], p.80-81)

$$\begin{aligned} S &\rightarrow wS|w \\ w &\rightarrow \text{abaca}|\text{abacule}|\text{abaissé}|\text{abaissa}|\ldots|\text{zézayant}|\text{zézayer} \end{aligned}$$

(cette grammaire génère bien tous les énoncés possibles du français)

- Il existe des indices qui nous font penser que les enfants apprennent une grammaire sans exemple négatif.

# Modélisation de la langue

**Définition :** $p(S) = \sum_{t \in \tau(S)} p(t)$

- Il existe des algorithmes pour calculer la probabilité d'un préfixe d'une chaîne d'un langage
  [Jelinek and Lafferty, 1991, Stolcke, 1995].
- il est donc possible d'en faire un modèle prédicitif :

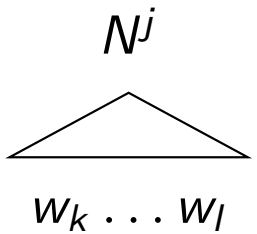
$$p(w_n | w_1^{n-1}) = \frac{p(w_1^n)}{p(w_1^{n-1})}$$

- Lire aussi [Chelba et al., 1997]

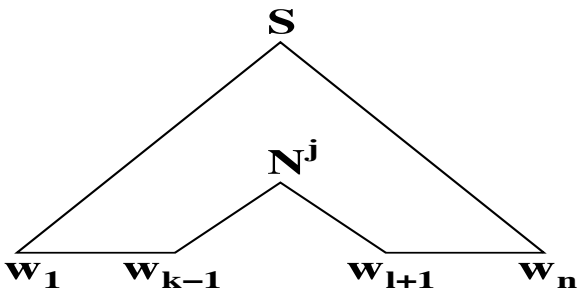
# Plan

# Calculer la probabilité d'une chaîne
# Cas d'une grammaire CNF

- idem aux HMMs : sommer sur tous les arbres est trop coûteux :
- inside probabilities : $\alpha_j(k,l) \stackrel{def}{=} p(w_k^l | N_{k,l}^j, G)$




- outside probabilities : $\beta_j(k,l) \stackrel{def}{=} p(w_1^{k-1}, N_{k,l}^j, w_{l+1}^n | G)$



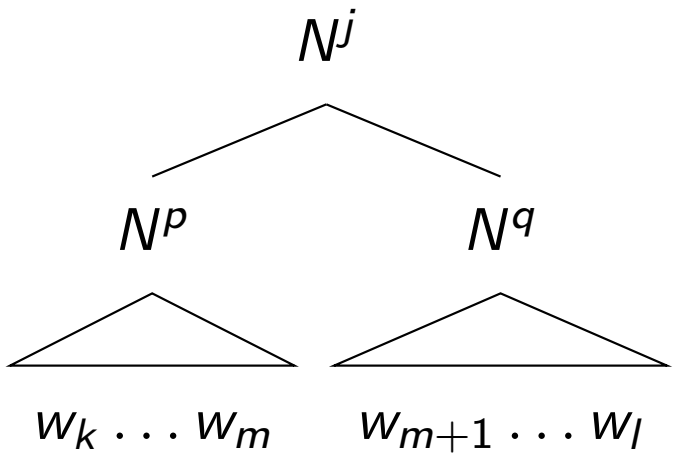
# Calculer la probabilité d'une chaîne (via inside)

**cas terminal :** $\alpha_j(k,k) = p(w_k|N^j_{k,k}) = p(N^j \rightarrow w_k)$
**récursion :** pour le cas $l > k$, $(m \in [k,l[, p,q \in [1,N])$, alors :

$$\begin{aligned}
\alpha_j(k,l) &\overset{def}{=} p(w^l_k|N^j_{k,l}) \\
&= \sum_{p,q,m} p(w^m_k, w^l_{m+1}, N^p_{k,m}, N^q_{m+1,l}|N^j_{k,l}) \\
&= \sum_{p,q,m} p(N^p_{k,m}, N^q_{m+1,l}|N^j_{k,l}) \\
&\quad \times p(w^m_k|N^p_{k,m}, N^q_{m+1,l}, N^j_{k,l}) \\
&\quad \times p(w^l_{m+1}|w^m_k, N^p_{k,m}, N^q_{m+1,l}, N^j_{k,l}) \\
&= \sum_{p,q,m} p(N^p_{k,m}, N^q_{m+1,l}|N^j_{k,l}) \\
&\quad \times p(w^m_k|N^p_{k,m}) \times p(w^l_{m+1}|N^q_{m+1,l}) \\
&= \sum_{p,q,m} p(N^j \rightarrow N^p N^q) \times \alpha_p(k,m) \times \alpha_q(m+1,l)
\end{aligned}$$

**calcul :** $\alpha_1(1,n) \overset{def}{=} p(w^n_1|N^1_{1,n}) \overset{def}{=} p(w^n_1)$

# Calculer la probabilité d'une chaîne (via inside)
## Un dessin vaut mieux qu'une formule



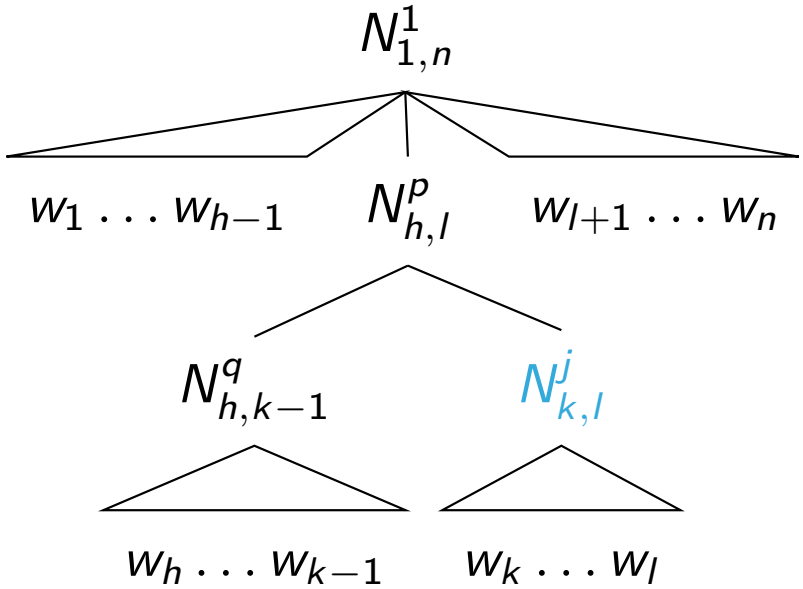

- On ne fait qu'envisager les découpages possibles (binaires car nous supposons ici une forme normale de Chomsky).
- **Note :** une variante de CYK

# Calculer la probabilité d'une chaîne (via inside)

inside$[1..n, 1..N, 1..n] := 0$
**for all** $k \in [1, n]$ **do**
  **for all** rule $A \rightarrow w_k$ **do**
    inside$[k, A, k] := p(A \rightarrow w_k)$
**for all** $l \in [2, n]$ **do**
  **for all** $s \in [1, n - l + 1]$ **do**
    **for all** rule $A \rightarrow BC \in R$ **do**
      **for all** $k \in [s, s + l - 2]$ **do**
        inside$[s, A, s + l - 1] += ( p(A \rightarrow BC) \times$
          inside$[s, B, k] \times$
          inside$[k + 1, C, s + l - 1] )$
**return** inside$[1, S, n]$

# Calculer la probabilité d'une chaîne (via outside)

- De deux choses l'une, ou bien on a appliqué $N^p \rightarrow N^q N^j$ :

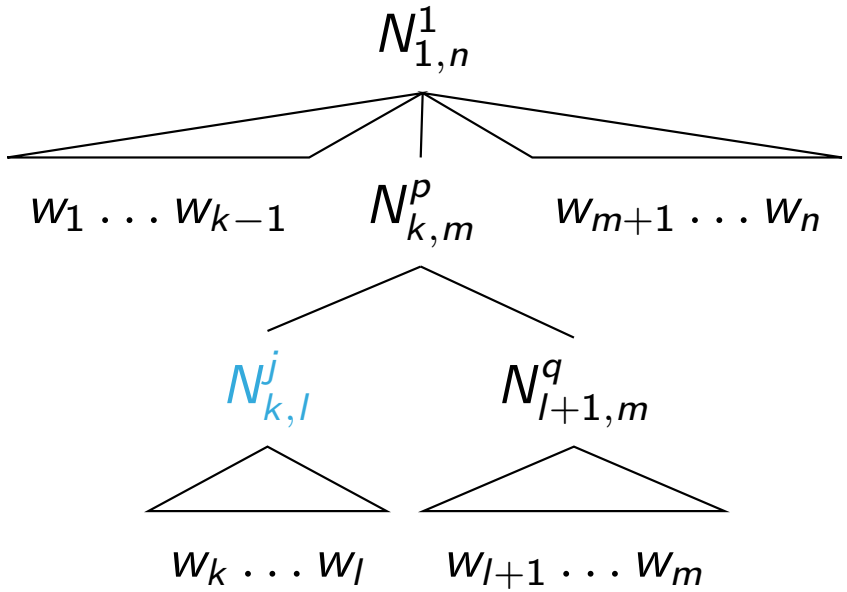



- $\beta_j(k,l) = \sum_{h,p,q} p(w_1^{h-1}, w_h^{k-1}, w_{l+1}^n, N^j_{k,l}, N^p_{h,l}, N^q_{h,k-1}) =$

$$\sum_{h,p,q} \quad p(w_1^{h-1}, w_{l+1}^n, N^p_{h,l}) \times$$
$$p(N^q_{h,k-1}, N^j_{k,l} | w_1^{h-1}, w_{l+1}^n, N^p_{h,l}) \times$$
$$p(w_h^{k-1} | N^q_{h,k-1}, N^j_{k,l}, w_1^{h-1}, w_{l+1}^n, N^p_{h,l})$$

# Calculer la probabilité d'une chaîne (via outside)

- ou bien on a appliqué $N^p \rightarrow N^j N^q$ :




- $\beta_j(k,l) = \sum_{m,p,q} p(w_1^{k-1}, w_{l+1}^m, w_{m+1}^n, N^j_{k,l}, N^p_{k,m}, N^q_{l+1,m}) =$

$$\sum_{m,p,q} \quad p(w_1^{k-1}, w_{m+1}^n, N^p_{k,m}) \times$$
$$p(N^j_{k,l}, N^q_{l+1,m} | w_1^{k-1}, w_{m+1}^n, N^p_{k,m}) \times$$
$$p(w_{l+1}^m | N^j_{k,l}, N^q_{l+1,m}, w_1^{k-1}, w_{m+1}^n, N^p_{k,m})$$

# Calculer la probabilité d'une chaîne (via outside)

- En mettant les deux possibilités ensembles (et en faisant attention à ne pas compter deux fois le cas $N^p \rightarrow N^j N^j$), on obtient :
- récurrence :

$$\begin{aligned}\beta_j(k,l) = \quad & \sum_{h,p,q} p(w_1^{h-1}, w_{l+1}^n, N_{h,l}^p) \times \\ & p(N_{h,k-1}^q, N_{k,l}^j | N_{h,l}^p) \times p(w_h^{k-1} | N_{h,k-1}^q) \\ & + \\ & \sum_{m,p,q \neq j} p(w_1^{k-1}, w_{m+1}^n, N_{k,m}^p) \times \\ & p(N_{k,l}^j, N_{l+1,m}^q | N_{k,m}^p) \times p(w_{l+1}^m | N_{l+1,m}^q)\end{aligned}$$

$$\begin{aligned}\beta_j(k,l) = \quad & \sum_{h,p,q} \beta_p(h,l) \times p(N^p \rightarrow N^q N^j) \times \alpha_q(h,k-1) + \\ & \sum_{m,p,q \neq j} \beta_p(k,m) \times p(N^p \rightarrow N^j N^q) \times \alpha_q(l+1,m)\end{aligned}$$

- **cas terminal :** $\beta_1(1,n) = 1$

# Calculer la probabilité d'une chaîne (via outside)

- La probabilité que notre grammaire génère $w_1^n$ et que $w_p^q$ soit un constituant est : $p(w_1^n, N_{p,q}) = \sum_{j \in [1,N]} \beta_j(p,q)\alpha_j(p,q)$
- En particulier, si on considère les symboles *pré-terminaux* (cad les symboles non terminaux qui possèdent juste un fils qui est un terminal), alors :

$$\begin{aligned} p(w_1^n, N_{k,k}) &= \sum_{j \in [1,N]} \beta_j(k,k)\alpha_j(k,k) \\ &= \sum_{j \in [1,N]} p(N^j \rightarrow w_k)\beta_j(k,k) \end{aligned}$$

- or il existe forcément pour une chaîne du langage un pré-terminal pour chaque mot de la phrase, donc $p(w_1^n, N_{k,k}) = p(w_1^n)$ ; donc :

$$p(w_1^n) = \sum_{j \in [1,N]} \beta_j(k,k)p(N^j \rightarrow w_k)$$

# Chercher l'arbre le plus probable ($\mathcal{O}(n^3|N|^3)$)

- On peut modifier notre algorithme de calcul des probabilités *inside* (qui lui même est une version à peine modifiée de CYK) pour obtenir un algorithme qui ressemble beaucoup à viterbi : on remplace le $+$ par un max, et on mémorise le chaînage des meilleurs constituants :

| | |
|---|---|
| best$[i, A, j]$ | est la probabilité maximale (ou son log) qu'un non-terminal $A$ dérive $w_i^j$, |
| back$[i, A, j]$ | est le *back-pointeur* qui nous permet de retrouver la meilleure dérivation. Ce pointeur est un doublet $\langle r, k \rangle$ qui indique que la $r$-ième règle de $R$ a été appliquée et que le split a eu lieu en $k$. |

# Chercher l'arbre le plus probable ($\mathcal{O}(n^3|N|^3)$)

best$[1..n, 1..N, 1..n] := 0$; back$[1.., 1..N, 1..n] :=$ NULL
**for all** $k \in [1, n]$ **do**
  **for all** rule $A \rightarrow w_k$ (soit $r$ l'indice de cette règle) **do**
    best$[k, A, k] := p(A \rightarrow w_k)$
    back$[k, A, k] := \langle r, 0 \rangle$
**for all** $l \in [2, n]$ **do**
  **for all** $s \in [1, n - l + 1]$ **do**
    **for all** $r \in [1, |R|]$ **do**
      *// soit $A \rightarrow BC$ la r-ième règle de R*
      **for all** $k \in [s, s + l - 2]$ **do**
        $score = p(A \rightarrow BC) \times$ best$[s, B, k] \times$ best$[k + 1, C, s + l - 1]$
        **if** ($score >$ best$[s, A, s + l - 1]$) **then**
          best$[s, A, s + l - 1] = score$
          back$[s, A, s + l - 1] = \langle r, k \rangle$

best$[1, S, n]$ est la probabilité de la meilleure analyse
back$[1, S, n]$ permet d'obtenir l'arbre le plus probable

# Chercher l'arbre le plus probable (CYK $l = 1$)

| | | | | | |
|---|---|---|---|---|---|
| ears | | | | | 0.18 NP [9,0] |
| with | | | | 1.0 P [5,0] | |
| stars | | | 0.18 NP [11,0] | | |
| saw | | 1.0 *V* [6, 0]<br>0.04 *NP* [10, 0] | | | |
| astro | 0.1 NP [8,0] | | | | |
| | astro | saw | stars | with | ears |

1- S → NP VP 1.0
2- PP → P NP 1.0
3- VP → V NP 0.7
4- VP → VP PP 0.3
5- P → *with* 1.0
6- V → *saw* 1.0
7- NP → NP PP 0.4
8- NP → *astro* 0.1
9- NP → *ears* 0.18
10- NP → *saw* 0.04
11- NP → *stars* 0.18
12- NP → *telescopes* 0.1

# Chercher l'arbre le plus probable (CYK $I = 2$)

| | | | | | |
|---|---|---|---|---|---|
| ears | | | | 0.18 PP [2,4] | 0.18 NP [9,0] |
| with | | | | 1.0 P [5,0] | |
| stars | | 0.126 VP [3,2] | 0.18 NP [11,0] | | |
| saw | | 1.0 *V* [6, 0]<br>0.04 *NP* [10, 0] | | | |
| astro | 0.1 NP [8,0] | | | | |
| | astro | saw | stars | with | ears |

1- S → NP VP 1.0
2- PP → P NP 1.0
3- VP → V NP 0.7
4- VP → VP PP 0.3
5- P → *with* 1.0
6- V → *saw* 1.0
7- NP → NP PP 0.4
8- NP → *astro* 0.1
9- NP → *ears* 0.18
10- NP → *saw* 0.04
11- NP → *stars* 0.18
12- NP → *telescopes* 0.1

- VP ⇒ saw stars $0.126 = 0.7 \times 1.0 \times 0.18$
- PP ⇒ with ears $0.18 = 1.0 \times 0.18 \times 1.0$

# Chercher l'arbre le plus probable (CYK $I = 3$)

| | | | | | |
|---|---|---|---|---|---|
| ears | | | 0.01296 NP [7,3] | 0.18 PP [2,4] | 0.18 NP [9,0] |
| with | | | | 1.0 P [5,0] | |
| stars | 0.0126 S [1,1] | 0.126 VP [3,2] | 0.18 NP [11,0] | | |
| saw | | 1.0 *V* [6, 0]<br>0.04 *NP* [10, 0] | | | |
| astro | 0.1 NP [8,0] | | | | |
| | astro | saw | stars | with | ears |

1- S → NP VP 1.0
2- PP → P NP 1.0
3- VP → V NP 0.7
4- VP → VP PP 0.3
5- P → *with* 1.0
6- V → *saw* 1.0
7- NP → NP PP 0.4
8- NP → *astro* 0.1
9- NP → *ears* 0.18
10- NP → *saw* 0.04
11- NP → *stars* 0.18
12- NP → *telescopes* 0.1

- S ⇒ astro saw stars $\quad 0.0126 = 1.0 \times 0.1 \times 0.126$
- NP ⇒ stars with ears $\quad 0.1296 = 0.4 \times 0.18 \times 0.18$

# Chercher l'arbre le plus probable (CYK $I = 4$)

| | | | | |
|---|---|---|---|---|
| ears | | 0.009072 *VP* [3, 2]<br>0.006804 *VP* [4, 3] | 0.01296 NP [7,3] | 0.18 PP [2,4] | 0.18 NP [9,0 |
| with | | | | 1.0 P [5,0] | |
| stars | 0.0126 S [1,1] | 0.126 VP [3,2] | 0.18 NP [11,0] | | |
| saw | | 1.0 *V* [6, 0]<br>0.04 *NP* [10, 0] | | | |
| astro | 0.1 NP [8,0] | | | | |
| | astro | saw | stars | with | ears |

1- S → NP VP 1.0
2- PP → P NP 1.0
3- VP → V NP 0.7
4- VP → VP PP 0.3
5- P → *with* 1.0
6- V → *saw* 1.0
7- NP → NP PP 0.4
8- NP → *astro* 0.1
9- NP → *ears* 0.18
10- NP → *saw* 0.04
11- NP → *stars* 0.18
12- NP → *telescopes* 0.1

VP ⇒ saw stars with ears

- VP → VP PP $0.006804 = 0.3 \times 0.126 \times 0.18$
- NP → V NP $0.009072 = 0.7 \times 1.0 \times 0.01296$

# Chercher l'arbre le plus probable (CYK $I = 5$)

| | | | | | |
|---|---|---|---|---|---|
| ears | 0.0009072 S [1,1] | 0.009072 VP [3,2] | 0.01296 NP [7,3] | 0.18 PP [2,4] | 0.18 NP [9,0] |
| with | | | | 1.0 P [5,0] | |
| stars | 0.0126 S [1,1] | 0.126 VP [3,2] | 0.18 NP [11,0] | | |
| saw | | 1.0 *V* [6, 0]<br>0.04 *NP* [10, 0] | | | |
| astro | 0.1 NP [8,0] | | | | |
| | astro | saw | stars | with | ears |

1- S → NP VP 1.0
2- PP → P NP 1.0
3- VP → V NP 0.7
4- VP → VP PP 0.3
5- P → *with* 1.0
6- V → *saw* 1.0
7- NP → NP PP 0.4
8- NP → *astro* 0.1
9- NP → *ears* 0.18
10- NP → *saw* 0.04
11- NP → *stars* 0.18
12- NP → *telescopes* 0.1

- S ⇒ astro saw stars with ears $0.0009072 = 1.0 \times 0.1 \times 0.009072$
- C'est l'arbre de la page 66

# Chercher l'arbre le plus probable cas d'une grammaire non CNF

- envisager tous les découpages faisant intervenir les $r$ symboles de la partie droite de la règle considérée.
- Ex : Soit la règle VP $\rightarrow$ ADVP VBD NP , NP et supposons que l'on cherche toutes les façons que *VP* a de dominer les 7 premiers mots de la chaîne à analyser. Les découpages suivants doivent *a priori* être considérés :

| ADVP | VBD | NP | , | NP |
|---|---|---|---|---|
| 1-1 | 2-2 | 3-3 | 4-4 | 5-7 |
| 1-1 | 2-2 | 3-4 | 5-5 | 6-7 |
| 1-1 | 2-2 | 3-5 | 6-6 | 7-7 |
| 1-1 | 2-3 | 4-4 | 5-5 | 6-7 |
| 1-1 | 2-3 | 4-5 | 6-6 | 7-7 |
| 1-1 | 2-4 | 5-5 | 6-6 | 7-7 |
| 1-2 | 3-3 | 4-4 | 5-5 | 6-7 |
| 1-2 | 3-4 | 5-5 | 6-6 | 7-7 |
| 1-3 | 4-4 | 5-5 | 6-6 | 7-7 |

# Chercher l'arbre le plus probable cas d'une grammaire non CNF

Soit *rule* une règle. *rhs*(*rule*) désigne sa partie droite, *rhs*(*rule*, *n*) désigne le n-ième symbole (0-indicé) de la partie droite, et *lhs*(*rule*) désigne sa partie gauche.

**function** split(rule, n, memo, start, stop, cuts, score) :

**if** (n == $|rhs(rule)|$) **then**
  **if** (start > stop) **then**
    **if** ($score >$ best[$memo, lhs(rule), stop$]) **then**
      best[$memo, lhs(rule), stop$] = $score$
      back[$memo, lhs(rule), stop$] = $\langle rule, cuts \rangle$
**else**
  **for all** $j \in [start, stop - |rhs(rule)| + n + 1]$ **do**
    **if** ($rhs(rule, n)$ est dans la table avec un score $sc$) **then**
      **if** ($score + sc >$ best[$start, lhs(rule), j$]) **then**
        split(rule,n+1,memo,j+1,stop,$cuts \cup \{j\}$,score + sc)

ex : split(VP → ADVP VBD NP , NP ; 0 ; 1 ; 1 ; 7 ; {} ; log(prob(rule)))

rule

# Chercher l'arbre le plus probable

best$[1..n, 1..N, 1..n] := 0$
back$[1.., 1..N, 1..n] := \{\}$
**for all** $k \in [1, n]$ **do**
  **for all** rule $A \rightarrow w_k$ (soit $r$ l'indice de cette règle) **do**
    best$[k, A, k] := \log p(A \rightarrow w_k)$
    back$[k, A, k] := \langle r, \{\} \rangle$
**for all** $l \in [1, n]$ **do**
  **for all** $s \in [1, n - l + 1]$ **do**
    **for all** $r \in [1, |R|]$ **do**
      split(r,0,s,s,s+l-1,{},log($prob(r)$))

best$[1, S, n]$ est la probabilité de la meilleure analyse
back$[1, S, n]$ permet d'obtenir l'arbre le plus probable

**Note :** il existe un problème dans le cas où la grammaire possède des règles unitaires (ex : A $\rightarrow$ B)...

# Exemple de table d'analyse

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| . | [1] | −33.1 | (1, 2, 4) | | | | | | |
| | [2] | −43.3 | (1, 2, 4) | | | | | | |
| possible | [3] | −50.4 | (3) | | | | [3] | −13.2 | (3) |
| | [4] | −46.7 | (2, 3) | [7] | −17.7 | (2, 3) | [9] | −9.5 | (3) |
| | [5] | −33.5 | (1, 3) | [8] | −27.1 | (2, 3) | [10] | −7.5 | (3) |
| | [6] | −39.4 | (1, 3) | | | | [11] | −16.7 | (3) |
| | | | | | | | [12] | −15.1 | (3) |
| | | | | | | | [13] | −11.2 | (3) |
| are | [5] | −26.2 | (1, 2) | [14] | −1.6 | (2) | | | |
| | [6] | −32.0 | (1, 2) | [15] | −10.4 | (2) | | | |
| compromises | [16] | −14.0 | (1) | | | | | | |
| | [17] | −11.2 | (1) | | | | | | |
| | [18] | −9.2 | (1) | | | | | | |
| | compromises | | | are | | | possible | | . |

1 S → NP VP . [-1.4]
2 NP → NP VP . [-11.5]
3 FRAG → ADJP [-3.7]
4 ADJP → NP JJ [-7.1]
5 S → NP VP [-1.8]
6 NP → NP VP [-7.6]
7 VP → VBP ADJP [-6.6]
8 SINV → VP NP [0.0]
9 ADJP → JJ [-2.0]
10 JJ → possible [-7.5]
11 NP → JJ [-9.2]
12 ADVP → JJ [-7.6]
13 PRT → JJ [-3.7]
14 VBP → are [-1.6]
15 VP → VBP [-8.7]
16 NP → NNS [-4.8]
17 NX → NNS [-2.0]
18 NNS → compromises [-9.2]

item = règle, logprob, découpage de chaque RHS-symbole

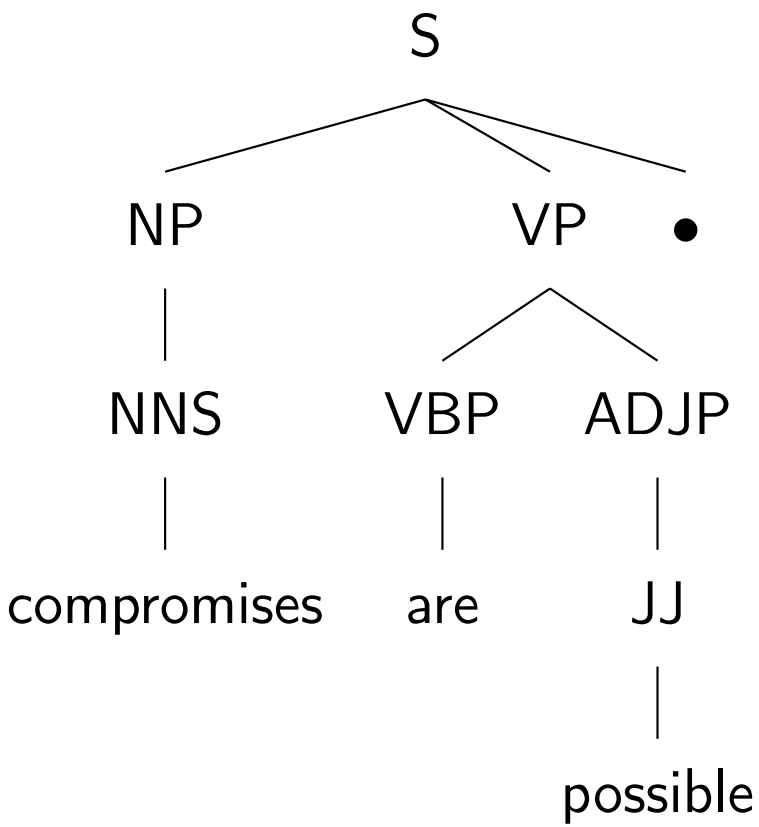
Ex : $(1, 3) \equiv$ 1er symb domine de 1 à 2, le 2e domine à partir de 3

# Retour arrière

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| . | [1] | −33.1 | (1, 2, 4) | | | | | | |
| | [2] | −43.3 | (1, 2, 4) | | | | | | |
| possible | [3] | −50.4 | (3) | [7] | −17.7 | (2, 3) | [3] | −13.2 | (3) |
| | [4] | −46.7 | (2, 3) | [8] | −27.1 | (2, 3) | [9] | −9.5 | (3) |
| | [5] | −33.5 | (1, 3) | | | | [10] | −7.5 | (3) |
| | [6] | −39.4 | (1, 3) | | | | [11] | −16.7 | (3) |
| | | | | | | | [12] | −15.1 | (3) |
| | | | | | | | [13] | −11.2 | (3) |
| are | [5] | −26.2 | (1, 2) | [14] | −1.6 | (2) | | | |
| | [6] | −32.0 | (1, 2) | [15] | −10.4 | (2) | | | |
| compromises | [16] | −14.0 | (1) | | | | | | |
| | [17] | −11.2 | (1) | | | | | | |
| | [18] | −9.2 | (1) | | | | | | |
| | compromises | | | are | | | possible | | . |

1 S → NP VP . [-1.4]
2 NP → NP VP . [-11.5]
3 FRAG → ADJP [-3.7]
4 ADJP → NP JJ [-7.1]
5 S → NP VP [-1.8]
6 NP → NP VP [-7.6]
7 VP → VBP ADJP [-6.6]
8 SINV → VP NP [0.0]
9 ADJP → JJ [-2.0]
10 JJ → possible [-7.5]
11 NP → JJ [-9.2]
12 ADVP → JJ [-7.6]
13 PRT → JJ [-3.7]
14 VBP → are [-1.6]
15 VP → VBP [-8.7]
16 NP → NNS [-4.8]
17 NX → NNS [-2.0]
18 NNS → compromises [-9.2]

# Analyse correspondant à la table précédente



# CNF ou pas ?

| | | | | | |
|---|---|---|---|---|---|
| 1 | S → NP VP . [-1.4] | 7 | VP → VBP ADJP [-6.6] | 13 | PRT → JJ [-3.7] |
| 2 | NP → NP VP . [-11.5] | 8 | SINV → VP NP [0.0] | 14 | VBP → are [-1.6] |
| 3 | FRAG → ADJP [-3.7] | 9 | ADJP → JJ [-2.0] | 15 | VP → VBP [-8.7] |
| 4 | ADJP → NP JJ [-7.1] | 10 | JJ → possible [-7.5] | 16 | NP → NNS [-4.8] |
| 5 | S → NP VP [-1.8] | 11 | NP → JJ [-9.2] | 17 | NX → NNS [-2.0] |
| 6 | NP → NP VP [-7.6] | 12 | ADVP → JJ [-7.6] | 18 | NNS → compromises [-9.2] |

## La même grammaire au format CNF :

| | | | | | |
|---|---|---|---|---|---|
| 1 | S → NP VP [-1.8] | 9 | STOP → . [0] | 17 | NP → possible [-16.7] |
| 2 | S → NP #0 [-1.4] | 10 | VP → VBP ADJP [-6.6] | 18 | ADJP → possible [-9.5] |
| 3 | NP → NP #1 [-11.5] | 11 | SINV → VP NP [0] | 19 | FRAG → possible [-13.2] |
| 4 | NP → NP VP [-7.6] | 12 | NP → compromises [-14] | 20 | ADVP → possible [-15.1] |
| 5 | FRAG → NP JJ [-10.8] | 13 | NNS → compromises [-9.2] | 21 | PRT → possible [-11.2] |
| 6 | ADJP → NP JJ [-7.1] | 14 | NX → compromises [-11.2] | 22 | JJ → possible [-7.5] |
| 7 | #0 → VP STOP [0] | 15 | VP → are [-10.3] | | |
| 8 | #1 → VP STOP [0] | 16 | VBP → are [-1.6] | | |

# Problème III : Apprendre une PCFG

- **Question :** peut-on apprendre les paramètres d'une PCFG avec simplement un texte suffisamment grand d'une langue donnée ?
  - paramètres = règles + probabilités
  - c'est un problème difficile.
- **Soyons plus modestes :**

  Peut-on apprendre automatiquement les probabilités à associer aux règles d'une grammaire donnée ?
  - algorithme inside-outside [Baker, 1979]

rali

# Apprendre une PCFG : par *inside-outside*

- **Donnée incomplète :** les phrases $W_1, \ldots W_N$
- **Donnée manquante :** arbre syntaxique associé à chaque phrase : $t_1, \ldots, t_N$
- une instance de EM :

$$
\begin{aligned}
Q(\phi, \phi') &= \sum_W \tilde{p}(W) \underbrace{\sum_{t \in \tau(W)} p_{\phi'}(t|W) \log\left(p_\phi(W,t)\right)}_{E_t[\log p_\phi(W,t)|W,\phi']} \\
&= \sum_W \tilde{p}(W) \sum_{t \in \tau(W)} p_{\phi'}(t|W) \log\left(\prod_{r \in t} p_\phi(r)^{c(r;W,t)}\right) \\
&= \sum_W \tilde{p}(W) \sum_{t \in \tau(W)} p_{\phi'}(t|W) \sum_{r \in t} \log\left(p_\phi(r)^{c(r;W,t)}\right) \\
&= \sum_W \tilde{p}(W) \sum_{t \in \tau(W)} p_{\phi'}(t|W) \sum_{r \in t} c(r;W,t) \log\left(p_\phi(r)\right)
\end{aligned}
$$

- où $c(r; W, t)$ est le compte du nombre de fois où la règle $r$ est utilisée dans l'arbre $t$ de la phrase $W$

# Apprendre une PCFG : par *inside-outside*

$$\frac{\delta Q(\phi, \phi')}{\delta p_\phi(r)} = \sum_W \tilde{p}(W) \frac{\sum_{t \in \tau(W)} p_{\phi'}(t|W) c(r; W, t)}{p_\phi(r)} - \gamma = 0$$

D'où :

$$p_\phi(r) = \sum_W \tilde{p}(W) \frac{\sum_{t \in \tau(W)} p_{\phi'}(t|W) c(r; W, t)}{\gamma}$$

- $\gamma \equiv \sum_{t \in \tau(W)} p_{\phi'}(t|W) c(LHS(r); W, t)$
- Il faut donc calculer les comptes estimés $\hat{c}_W(r) \equiv \sum_{t \in \tau(W)} p_{\phi'}(t|W) c(r; W, t)$ qui impliquent une somme (potentiellement) exponentielle.

# Apprendre une PCFG : par *inside-outside*

- On peut réduire la complexité de la réestimation par programmation dynamique.

$$\begin{aligned}\beta_j(p,q) \times \alpha_j(p,q) &= p(w_1^{p-1}, N^j_{p,q}, w^n_{q+1}|G) \times p(w_p^q|N^j_{p,q}, G)\\ &= p(w_1^{p-1}, N^j_{p,q}, w^n_{q+1}, w_p^q|G)\\ &= p(w_1^n, N^j_{p,q}|G)\\ &= \underbrace{p(w_1^n|G)}_{\pi} \times p(N^j_{p,q}|w_1^n, G)\end{aligned}$$

- Donc : $p(N^j_{p,q}|w_1^n, G) = \frac{\beta_j(p,q)\times\alpha_j(p,q)}{\pi}$ (on vient de voir comment calculer $\pi$)
- C'est la probabilité que $N^j$ domine les positions $p, q$, sachant que la chaîne $w_1^n$ est une phrase du langage décrit par la grammaire.

# Apprendre une PCFG : par *inside-outside*

- Si l'on veut l'estimée d'utilisation de $N^j$, alors il suffit de considérer toutes les dominations possibles, notre estimée est donc :

$$E(N^j \text{ soit utilisé dans une dérivation}) = \sum_{p,q \,/\, p \leq q} \frac{\beta_j(p,q) \times \alpha_j(p,q)}{\pi}$$

- C'est notre $\gamma$ dans notre équation de réestimation (ça se démontre)
- Deux cas possibles (dans le cas CNF) : $r \equiv N^j \rightarrow N^r N^s$ ou $r \equiv N^j \rightarrow w_k$.

# Apprendre une PCFG : par *inside-outside*

$$r \equiv N^j \rightarrow N^u N^v$$

- Pour $(p, q)$ fixé, on peut introduire notre dérivation depuis $N^j$ qui ne concerne (par définition) que la probabilité interne $\alpha_j(p, q)$ :

  $c_W(r) \equiv E(N^j \text{ soit utilisé}, N^j \rightarrow N^u N^v) =$

$$\frac{\sum_{p,q} \beta_j(p, q) \times \sum_{d=p}^{q-1} p(N^j \rightarrow N^u N^v) \times \alpha_u(p, d) \times \alpha_v(d+1, q)}{\pi}$$

- donc : $p_\phi(r) =$

$$\sum_{i=1}^{N} \hat{p}(W_i) \times \frac{\sum_{p,q} \beta_j(p, q) \times \sum_{d=p}^{q-1} p(r) \times \alpha_u(p, d) \times \alpha_v(d+1, q)}{\sum_{p,q} \beta_j(p, q) \times \alpha_j(p, q)}$$

# Apprendre une PCFG : par *inside-outside*

$$r \equiv N^j \rightarrow w_k$$

- De la même manière :

$$c_W(r) \equiv E(N^j \text{ utilisée et } N^j \rightarrow w_k) = \frac{\sum_{p=1|w_p \equiv w_k}^{n} \beta_j(p,p) \times p(N^j \rightarrow w_k)}{\pi}$$

- donc :

$$\hat{p}(r) = \sum_{i=1}^{N} \hat{p}(W_i) \times \frac{\sum_{p=1|w_p \equiv w_k}^{n} \beta_j(p,p) \times p(r)}{\sum_{p,q} \beta_j(p,q) \times \alpha_j(p,q)}$$

# *inside-outside* : Quelques faits

lenteur : complexité d'une itération pour une phrase de $|w|$ mots : $\mathcal{O}(|N|^3 \times |w|^3)$. Une itération d'un HMM à $s$ états, pour une phrase de $|w|$ mots est : $\mathcal{O}(s^2 \times |w|)$.

maximum locaux : problème encore plus important ici que dans le cas des HMMs $\rightarrow$ ça n'est peut-être pas la bonne méthode d'apprentissage pour ce genre de problème.

facilement parallélisable : mais ça ne change pas la complexité de l'algorithme.

Pour plus d'information, consulter : [Lari and Young, 1990].

# Inside-outside et apprentissage des règles

- **Idée :** on part d'une grammaire complète (toutes les règles possibles sont représentées pour un choix fixé de non-terminaux et terminaux). On applique *inside-outside* et on filtre éventuellement les règles dont la probabilité est trop faible (en absolu ou en relatif).
- **Problème :** temps prohibitifs d'entraînement, trop de paramètres. Pour $|N|$ symboles non-terminaux et $|T|$ symboles terminaux, on a $|N|^3 + |N| \times |T|$ règles (cas CNF).

  Pour $|N| = 100$ et $|T| = 30000$ ça fait 4 millions de paramètres à estimer !
- **Solutions :**
  - utiliser un corpus parenthésé [Pereira and Schabes, 1992]
  - augmenter progressivement la complexité de la grammaire [Hogenhout and Matsumoto, 1998]
  - utiliser un corpus arboré [Collins, 1996, Charniak, 1996]

# IO et corpus parenthésé

- **Idée :** on dispose d'un corpus partiellement parenthésé. On ne tient pas compte des arbres qui contredisent le parenthésage donné :

| | | |
|---|---|---|
| **référence** | ( ) | ( ) |
| **grammaire** | ( ) | ( ) |
| | compatible | incompatible |

- Les récurrences permettant le calcul des $\beta$ et $\alpha$ sont modifiées en introduisant des fonctions indicatrices $\overline{c}(i,j)$ qui valent 1 si $w_i^j$ est compatible avec le parenthésage du corpus d'entraînement, 0 sinon :

$$\begin{aligned} \beta_p(i,j)' &= \overline{c}(i,j)\beta_p(i,j) \\ \alpha_p(i,j)' &= \overline{c}(i,j)\alpha_p(i,j) \end{aligned}$$

- Les formules de réestimation ne sont pas modifiées si ce n'est qu'elles utilisent ces nouvelles versions des probabilités inside et outside.

## IO et corpus parenthésé [Pereira and Schabes, 1992]

- **Expérience 1 :** les palindromes $\mathcal{L} = \{ww^R | w \in \{a, b\}^*\}$.
  100 phrases générées aléatoirement avec la grammaire :

| | | | |
|---|---|---|---|
| S → A C 0.4 | S → B D 0.4 | S → A A 0.1 | S → B B 0.1 |
| C → S A 1.0 | D → S B 1.0 | A → a 1.0 | B → b 1.0 |

- Au départ une grammaire de 135 règles (5 non-terminaux, 2 terminaux) dont les probabilités sont initialisées aléatoirement (avec respect des contraintes stochastiques).
- Inside-outside classique converge doucement vers un optimal local qui ne modélise pas du tout $\mathcal{L}$. Le inside-outside sur le même corpus mais parenthésé converge beaucoup plus rapidement vers une grammaire correcte (après filtrage des règles peu probables) :

| | | | |
|---|---|---|---|
| S → A D | S → C B | B → S C | D → S A |
| A → b | B → a | C → a | D → b |

rali

## IO et corpus parenthésé [Pereira and Schabes, 1992]

- **Expérience 2 :** Sur les POS uniquement
  700 phrases pour l'entraînement, 70 pour les tests.

```
((( VB ( DT NNS (IN ((NN) (NN CD)))))) .)
```

- Grammaire initiale : 4095 règles (15 non-terminaux, 48 symboles terminaux (POS)).
  - un *inside-outside* sur le corpus non parenthésé ($G_R$),
  - un autre sur le corpus parenthésé ($G_B$)
- Les deux entraînements convergent à peu près à la même vitesse. Cependant les deux grammaires se distinguent par leur performance (mesurée par le taux de bon parenthésage du corpus de test) : 37% pour $G_R$, 90% pour $G_B$.

# IO et augmentation progressive de la complexité de la grammaire [Hogenhout and Matsumoto, 1998]

**Idée :**

Sélection d'une grammaire $G$ restreinte (faible nombre de non-terminaux)

**while** (convergence non atteinte) **do**

- Entraîner $G$ (4 itérations)
- Retirer les règles $A \rightarrow BC$ de faible probabilité
- Sélection d'un non terminal $X_q$ (celui de plus grand compte estimé)
- Retirer les règles concernées par $X_q$ (à droite ou à gauche)
- Créer toutes les règles possibles avec les non-terminaux $X_q$ et $X_{q'}$
- Initialiser les probabilités de ces nouvelles règles aléatoirement (sous la contrainte de sommer à la masse de probabilité des règles retirées précédemment)

# [Hogenhout and Matsumoto, 1998]

- **Expérience :** grammaire CNF sur le PTB sur 31 POS
  - Train : 1000 phrases de 15 (resp. 20) mots ou moins dans exp .1 (resp. 2)
  - Test : 100 phrases
- init : une grammaire CNF complètement spécifiée avec 7 non-terminaux.
- L'algorithme précédent jusqu'à obtenir 15 (resp. 18) non terminaux sur des phrases d'au plus 15 (resp. 20) mots

rali

# [Hogenhout and Matsumoto, 1998]

- % de bon parenthésage

| long. | 0-10 | 0-15 | 10-19 | 20-30 |
|---|---|---|---|---|
| gram. inférée (15) | 92 | 91.7 | 83.8 | 72.0 |
| gram. inférée (18) | 94.1 | 91.5 | 86.9 | 81.8 |
| [Schabes et al., 1993] | 94.4 | 90.2 | 82.5 | 71.5 |

- **Note :** En principe, la convergence de l'algorithme au moment de l'expansion de règles n'est pas garantie. En pratique, l'algorithme se comporte bien (converge à l'entraînement).

# Plan

# Évaluation d'une analyse en constituants

- Intuitivement, un analyseur grammatical est bon s'il amène des améliorations à une tâche particulière qui en fait usage.
- En pratique, on souhaite évaluer une grammaire indépendamment de l'application visée, ce qui permet entre-autre de comparer des grammaires écrites dans des buts différents.
- on se sert d'une référence, cad, d'un corpus arboré manuellement qui représente le but à atteindre (par les concepteurs de grammaire).
- une première mesure consiste à créditer la grammaire d'un point lorsqu'une phrase est analysée de la même façon que dans le corpus de référence, et de 0 point sinon (tree accuracy, exact match).

rali

# Les mesures Parseval

- Les métriques PARSEVAL [Black, 1991] : trois mesures de base : précision (*precision*), rappel (*recall*) et parenthésage croisé (*crossing-brackets*).

$$\text{précision} = \frac{|\text{brackets candidats corrects}|}{|\text{brackets candidats}|}$$

$$\text{rappel} = \frac{|\text{brackets candidats corrects}|}{|\text{brackets dans la référence}|}$$

$$\text{crossing} = \text{moy. des parenthésages croisant ceux de la référence}$$

croisement : $\exists [i,j]$ et $[i',j']$ / $i < i' \leq j < j'$

- À l'origine ces mesures ne tenaient pas compte de l'étiquette attachée à un constituant. Lorsque l'on tient compte de cette étiquette, on parle alors de labeled precision et de labeled recall.

# Les mesures Parseval

Référence (They ((came) yesterday))
[1,3] [2,2] [2,3]

Candidat ((They (came)) (yesterday))
[1,3] [1,2] [2,2] [3,3]

- précision = 2/4 = 50%
- rappel= 2/3 = 66.7%
- crossing = 1 ([2,3] dans la référence et [1,2] dans le candidat)

$$\left(\text{F-}measure = \frac{2 \times Precision \times Recall}{Precision + Recall}\right) = 57.15\%$$

- **Note :** `(They came yesterday)` obtient une précision de 100%, et un taux de croisement de 0, mais un taux de rappel de 33.3%.
- toujours considérer ces trois mesures ensembles

# Les mesures Parseval

- **En pratique :** On ne tient pas compte en général des symboles pré-terminaux (car cela inclurait la performance d'une tâche de tagging, qu'il est préférable de mesurer séparément) et on ne tient pas compte non plus des symboles terminaux dont le parenthésage est toujours bon.
- Voir [Lin, 1995] pour une discussion plus détaillée des problèmes liés à ces métriques et pour une autre méthodologie d'évaluation. En particulier, une erreur faite en haut de la structure a plus d'impact sur les performances qu'une erreur faite en bas de l'arbre.
- La majorité des études mentionnent de toute façon les taux de précision et de rappel (étiquetés) ainsi que le taux de croisement.

rali

# Plan



rali

# Penn Tree Bank (PTB)

http://www.cis.upenn.edu/~treebank/home.html

- corpus arboré à la main pour l'anglais.
- deux versions :

  V1 (1992). 4.5 millions de mots taggés. Deux-tiers de ce corpus a été arboré avec des conventions simples.
  - phrases du *Dow-Jones News Service* (1.6 millions de mots parsés)
  - phrases du corpus équilibré : le Brown corpus (1 million de mots)

  V2 (1995). V1 améliorée +
  - 1M de mots du Wall Street Journal (1989)
  - 5k mots du corpus ATIS (5000 mots)

- Distribué via LDC (*Linguistic Data Consortium*) pour la *modique* somme d'environ 3000 $US

rali

# Phrase arborée du PTB (ATIS)

Étiqueté avec des diacritiques (SBJ, DIR, etc.)

```
( (S (NP-SBJ I)
     (VP need
         (NP (NP a flight)
             (PP-DIR to
                     (NP Seattle))
             (VP leaving
                 (PP-DIR from
                         (NP Baltimore)))
             (VP making
                 (NP (NP a stop)
                     (PP-LOC in
                             (NP Minneapolis)))))))
  )
```

*I need a flight to Seattle leaving from Baltimore making a stop in Minneapolis*

# Phrase arborée du PTB (ATIS)

```
( (S (NP-SBJ *)
     (VP List
         (NP (NP the flights)
             (PP-DIR from
                     (NP Baltimore))
             (PP-DIR to
                     (NP Seattle))
             (SBAR (WHNP-1 that)
                   (S (NP-SBJ *T*-1)
                      (VP stop
                          (PP-LOC in
                                  (NP Minneapolis)
  ))))))))
```

*List the flights from Baltimore to Seattle that stop in Minneapolis*

# Phrase arborée du PTB (WSJ)

## Annotation incomplète

```
( (S (NP-SBJ Rochester Telephone Corp.)
     (VP said
         (SBAR 0
               (S (NP-SBJ it)
                  (VP completed
                      (NP (NP its purchase)
                          (PP of
                              (NP (NP (NP Urban Telephone Corp. ,)
                                      (PP of
                                          (NP (NP Clintonville)
                                              ,
                                              (NP Wis.))))
                                  ,
                                  (NP (NP the second-largest unaffiliated
                                          independent telephone company)
                                      (PP-LOC in
                                              (NP that state)))))))))
     .))
```

# Un manuel pour les annotateurs

- Quelques 300 pages guident (dans la version II) l'annotation des phrases au niveau structurel (contre 100 pages dans la version I)
  - difficile et coûteux
- L'idée dans la version II est de permettre un codage facile en prédicat-argument.

```
(S (NP-SBJ-1 Chris)
   (VP wants
       (S (NP-SBJ *-1)
          (VP to
              (VP throw
                  (NP the ball))))))
```

prédicat/structure induite : *wants(Chris,throw(Chris,ball))*

# PTB et PCFG

- par fréquence relative !
- [Charniak, 1996] a "entraîné" sur 30 000 phrases arborées du Wall Street Journal (PTB) une grammaire de :
  - 10 605 règles
  - 3 943 sont apparues plus d'une fois dans le corpus d'entraînement
  - seulement sur les POS (aucune information sur les mots . . . ).
  - Les performances de cette grammaire (testée sur 30 000 autres phrases) sont étonnement "bonnes" :

| $\lvert sent. \rvert$ | $avr. \lvert sent \rvert$ | Pr | Rec | $\overline{cb}.$ |
|---|---|---|---|---|
| 2-12 | 8.7 | 88.6 | 91.7 | 97.9 |
| 2-16 | 11.4 | 85.0 | 87.7 | 94.5 |
| 2-20 | 13.8 | 83.5 | 86.2 | 92.8 |
| 2-30 | 18.7 | 80.6 | 82.5 | 89.5 |
| 2-40 | 21.9 | 78.8 | 80.4 | 87.7 |

# Plan

# Ignorance aux mots eux-mêmes

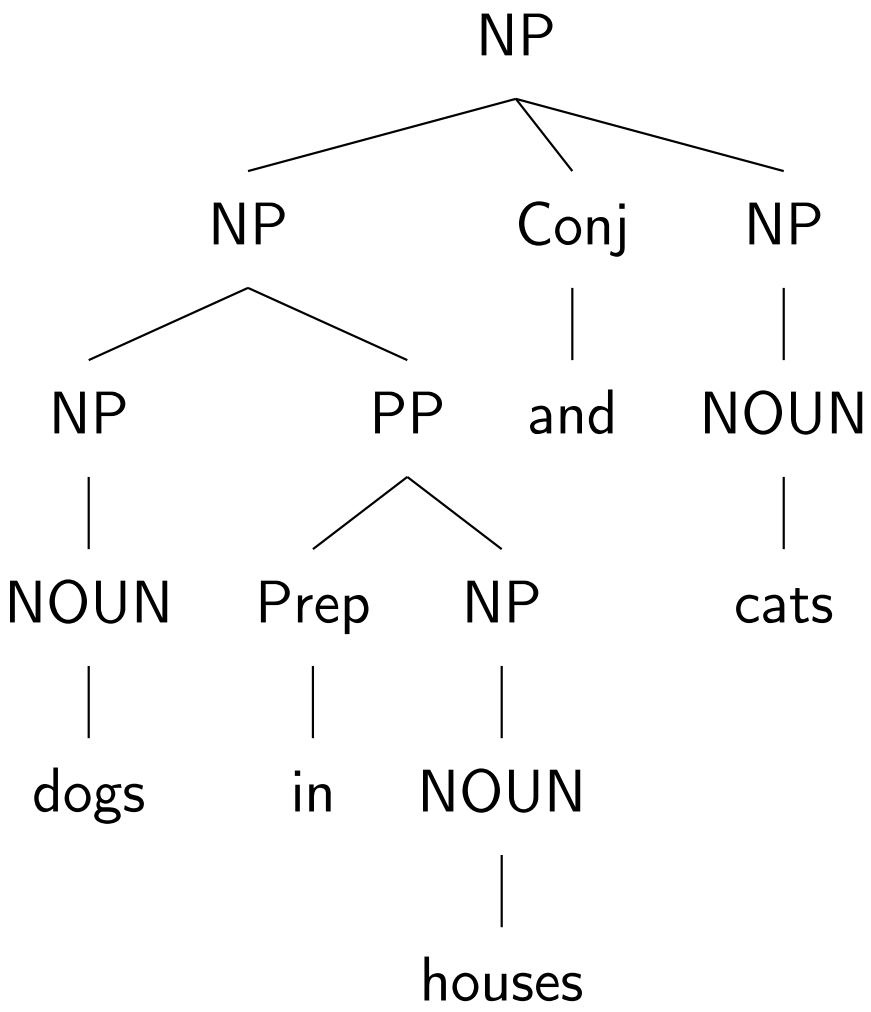
- L'information véhiculée par les mots se fait via les règles terminales (unigramme).
- Faible pouvoir de désambiguïsation [Hindle and Rooth, 1991] :

  *Moscow sent more than 100,000 soldiers into Afghanistan*

  **Question :** à quoi se rapporte le groupe prépositionnel *into Afghanistan* ?
- Dans une PCFG (classique), la différence se fait sur les probabilités associées à NP $\rightarrow$ NP PP et VP $\rightarrow$ NP PP. Hindle et al. rapportent que sur un corpus d'agence Press newswire, 67% des attachements prépositionnels sont sur le groupe nominal plutôt que sur le verbe.

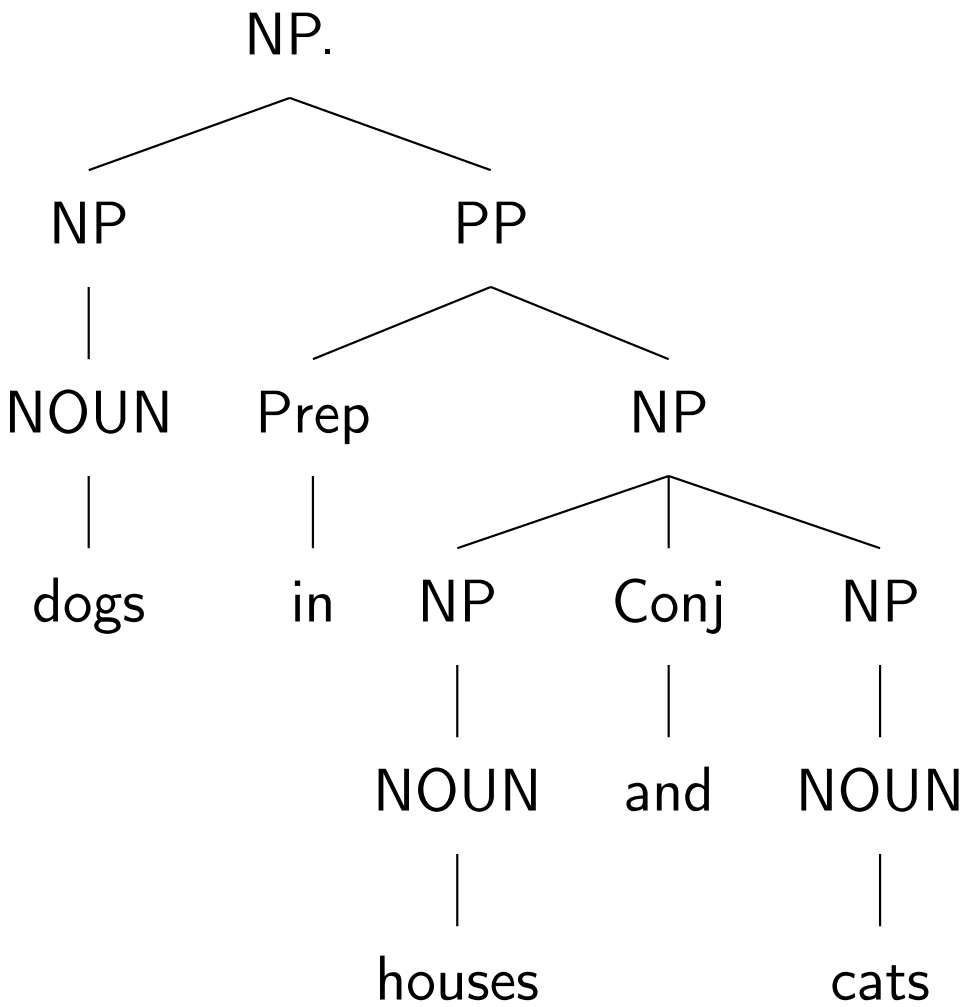
rali

# Un exemple pris de [Collins, 1999] (1/2)




$NP \rightarrow NP\ Conj\ NP$
$NP \rightarrow NP\ PP$
$Conj \rightarrow and$
$Prep \rightarrow in$
$PP \rightarrow Prep\ NP$
$NP \rightarrow NOUN\ [\times 3]$
$NOUN \rightarrow dogs$
$NOUN \rightarrow houses$
$NOUN \rightarrow cats$

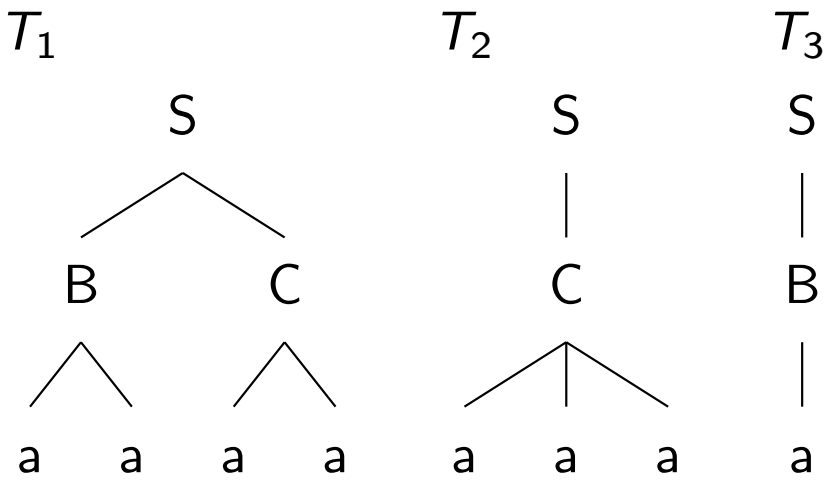
# Un exemple pris de [Collins, 1999] (2/2)




$NP \rightarrow NP\ Conj\ NP$
$NP \rightarrow NP\ PP$
$Conj \rightarrow and$
$Prep \rightarrow in$
$PP \rightarrow Prep\ NP$
$NP \rightarrow NOUN\ [\times 3]$
$NOUN \rightarrow dogs$
$NOUN \rightarrow houses$
$NOUN \rightarrow cats$

# Limites des PCFGs [Collins, 2003] (1/2)

- Soit un corpus généré à partir des 3 arbres selon une distribution $(p_1, p_2, p_3)$ :

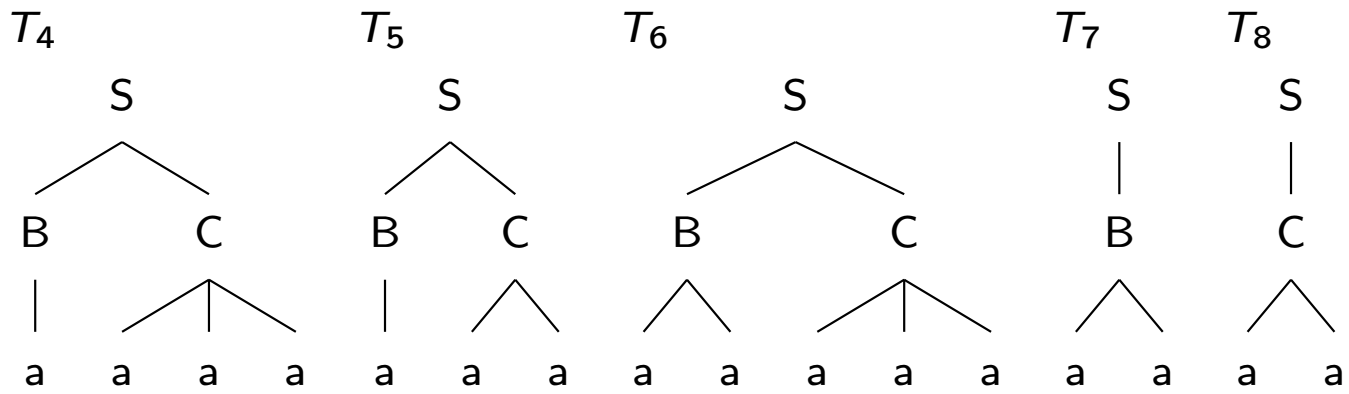



- Les probabilités des règles convergent vers :

| | | | |
|---|---|---|---|
| $S \to BC$ | $p_1$ | $B \to aa$ | $p_1/(p_1+p_3)$ |
| $S \to C$ | $p_2$ | $B \to a$ | $p_3/(p_1+p_3)$ |
| $S \to B$ | $p_3$ | $C \to aa$ | $p_1/(p_1+p_2)$ |
| | | $C \to aaa$ | $p_2/(p_1+p_2)$ |

# Un exemple pris de [Collins, 2003] (2/2)

- La grammaire génère également les arbres :




- probabilités de chaque arbre, si $p_1 = 0.2, p_2 = 0.1, p_3 = 0.7$ :
  $p(T_1^8) =$
  $\{0.0296, 0.0333, 0.544, 0.0519, 0.104, 0.0148, 0.156, 0.0667\}$
  - *aaaa* est générable par $T_1$ et $T_4$, mais $p(T_4) > p(T_1)$
  - *aaa* est générable par $T_2$ et $T_5$, mais $p(T_5) > p(T_2)$
  - *aaaa* et *aaa* sont mieux analysées par des arbres que ne font pas partie du corpus d'entraînement

# Plan

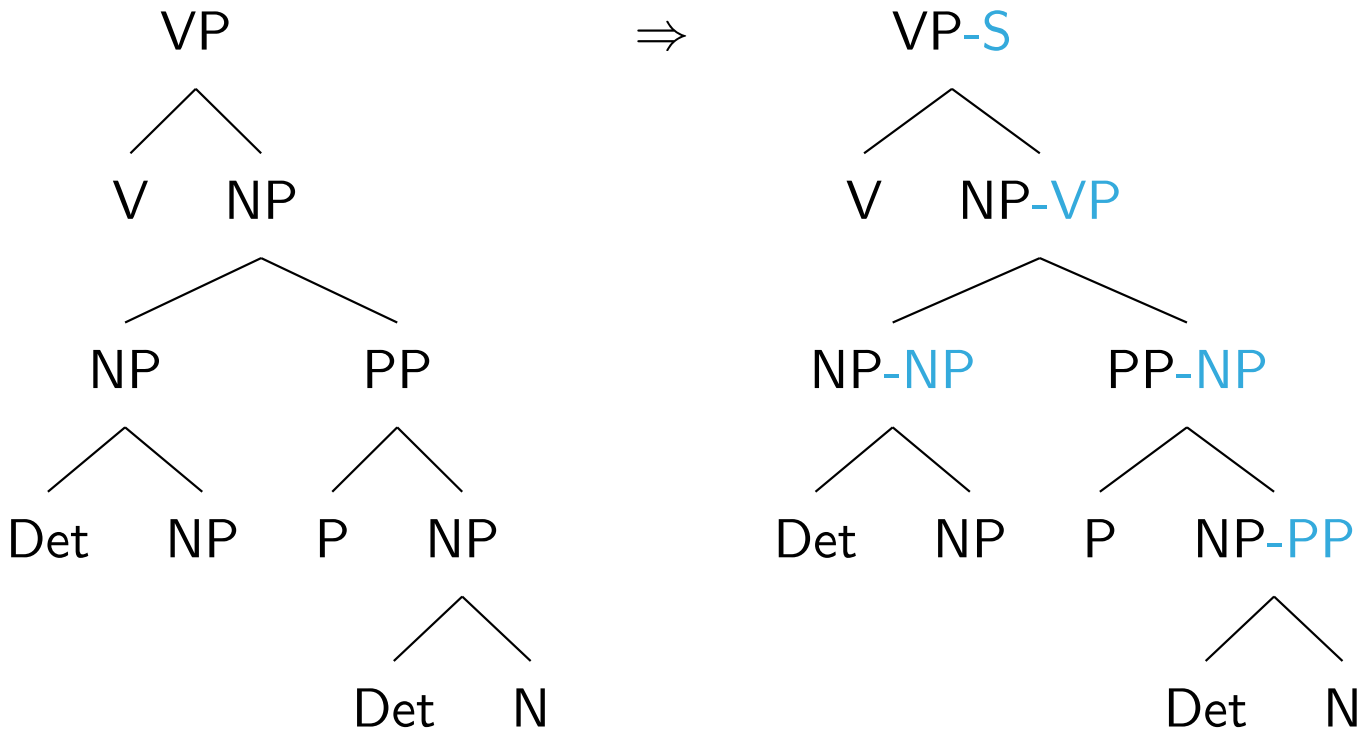
# Markovisation des règles : intuition

- on a peut-être vu en `train` des règles comme :
  - NP $\rightarrow$ ART NC ou
  - NP $\rightarrow$ ART ADJ NC
- mais pas une règle (légitime) comme :
  - NP $\rightarrow$ ART ADV ADJ NC.
- lissons à l'aide d'une hypothèse markovienne :
  - Ex (ordre 1) : $p(A \rightarrow A_1, \ldots, A_n) = \prod_i \ p_A(A_i|A_{i-1})$

rali

# Réduire les hypothèses d'indépendance des PCFGs

- [Johnson, 1998] étudie différentes transformations à caractère linguistique que l'on peut apporter au vocabulaire de la grammaire (les symboles non-terminaux) pour prendre en compte — dans un formalisme PCFG — un peu de contexte.
- Annoter le parent d'un nœud est une manière simple mais efficace d'augmenter les performances de la grammaire résultante.



# Sous-catégorisation

Expérience de Ford et al., 1982 décrite dans [Jurafsky and Martin, 2000]

- The women kept the dogs on the beach

# Sous-catégorisation

Expérience de Ford et al., 1982 décrite dans [Jurafsky and Martin, 2000]

- The women kept the dogs on the beach

  5% the women kept the dogs which were on the beach (dog,beach)

  95% the women kept them on the beach (kept,beach)

# Sous-catégorisation

Expérience de Ford et al., 1982 décrite dans [Jurafsky and Martin, 2000]

- The women kept the dogs on the beach

  5% the women kept the dogs which were on the beach (dog,beach)

  95% the women kept them on the beach (kept,beach)

- The women discussed the dogs on the beach

# Sous-catégorisation

Expérience de Ford et al., 1982 décrite dans [Jurafsky and Martin, 2000]

- The women kept the dogs on the beach
  - 5% the women kept the dogs which were on the beach (dog,beach)
  - 95% the women kept them on the beach (kept,beach)
- The women discussed the dogs on the beach
  - 90% the women discussed the dogs which were on the beach (dog,beach)
  - 10% the women discussed them while on the beach (discussed,beach)

# Sous-catégorisation

Expérience de Ford et al., 1982 décrite dans [Jurafsky and Martin, 2000]

- The women kept the dogs on the beach

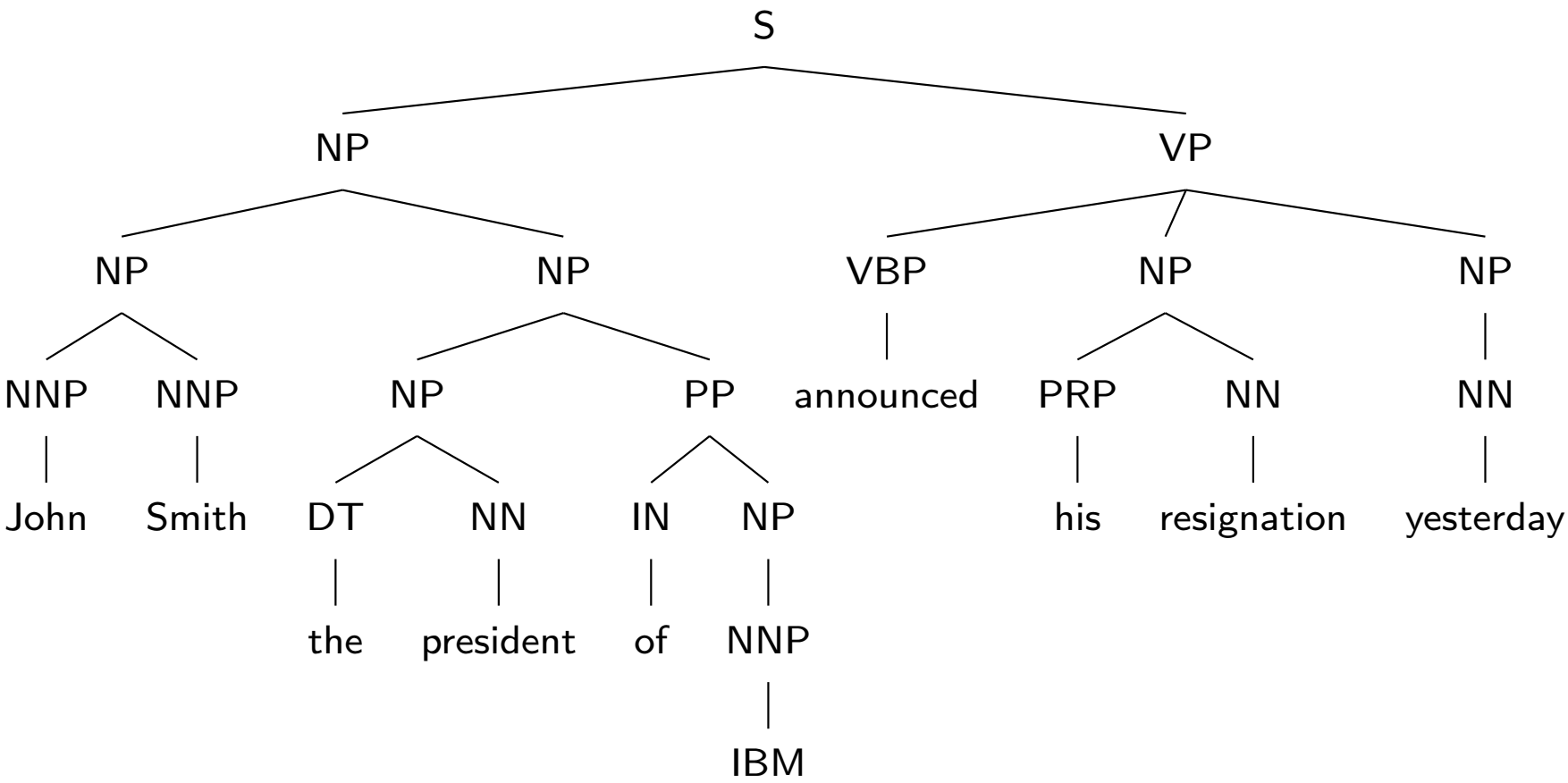
  5% the women kept the dogs which were on the beach (dog,beach)
  95% the women kept them on the beach (kept,beach)

- The women discussed the dogs on the beach

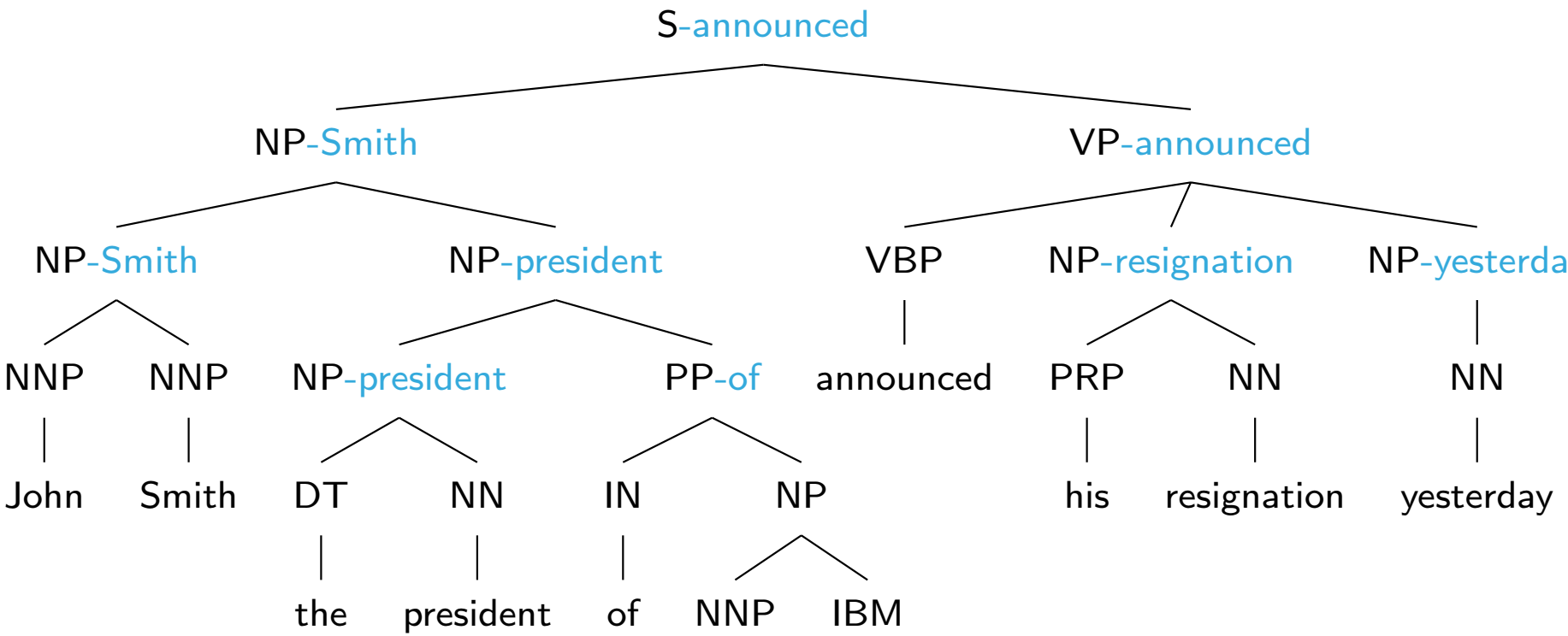
  90% the women discussed the dogs which were on the beach (dog,beach)
  10% the women discussed them while on the beach (discussed,beach)

- Les sujets préfèrent donc un attachement VP dans le cas de keep et un attachement NP dans le cas de discuss.

# Grammaires lexicalisées



# Grammaires lexicalisées



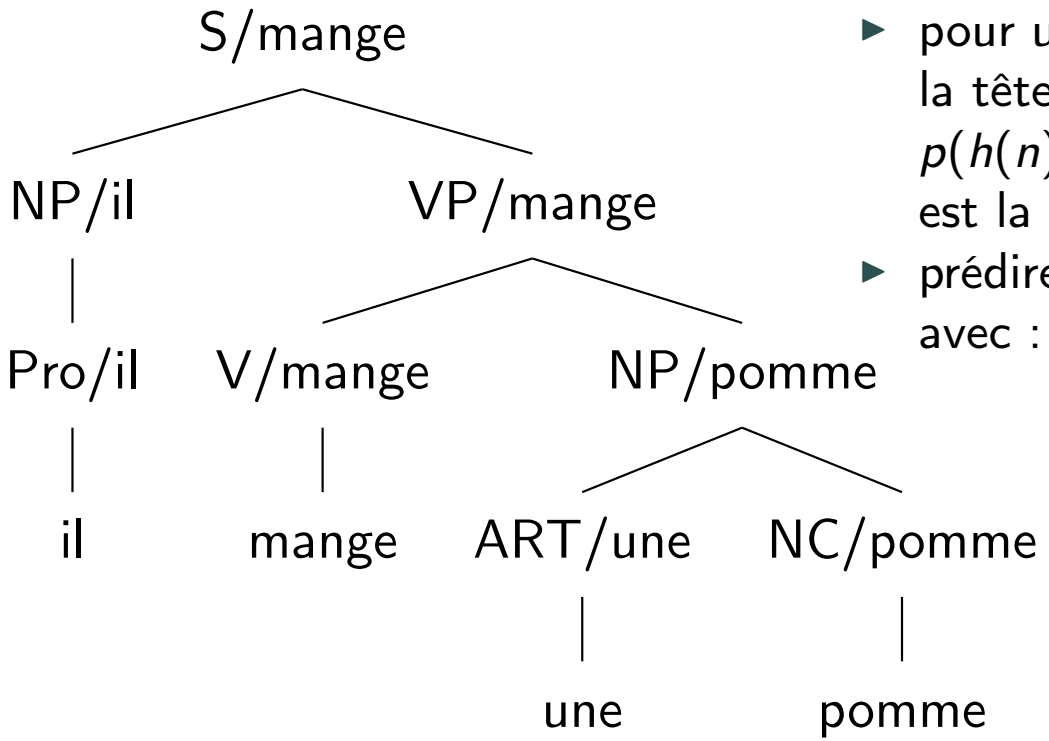
# Identification de la tête d'un constituant

- Processus déterministe (et bruité), à l'aide de règles [Magerman, 1995] déterminées empiriquement pour l'anglais et pour l'ensemble de tags du PTB.
- **Idée :** la tête d'un constituant est donnée par la tête de l'un de ses fils. Pour cela on consulte une table qui ressemble à ceci :

```
ADJP    right   % QP JJ VBN VBG ADJP $ JJR JJS DT FW **** RBR RBS RB
ADVP    left    RBR RB RBS FW ADVP CD **** JJR JJS JJ
CONJP   left    CC RB IN
FRAG    left    **
INTJ    right   **
LST     left    LS :
NAC     right   NN NNS NNP NNPS NP NAC EX $ CD QP PRP VBG JJ JJS JJR ADJP FW
NP      right   EX $ CD QP PRP VBG JJ JJS JJR ADJP DT FW RB SYM PRP$
NP$     right   NN NNS NNP NNPS NP NAC EX $ CD QP PRP VBG JJ JJS JJR ADJP FW SYM
...     ...     ...
```

# PCFG lexicalisée : [Charniak, 1997]

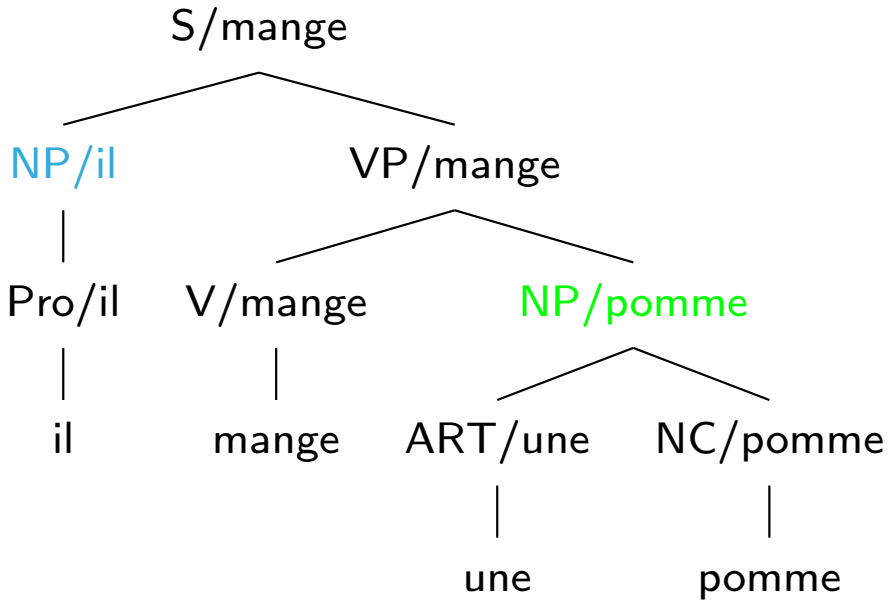
- **Idée** : on conditionne la probabilité d'une règle sur la tête lexicale du constituant impliqué par la dérivation.




- pour un constituant $n$, prédire la tête de $n$ avec $p(h(n)|n, h(m(n)))$, où $m(n)$ est la mère de $n$.
- prédire la règle à appliquer avec : $p(r(n)|n, h(n))$

$$P(T, S) = \prod_{n \in T} p(r(n)|n, h(n)) \times p(h(n)|n, h(m(n)))$$

# PCFG lexicalisée : [Charniak, 1997]



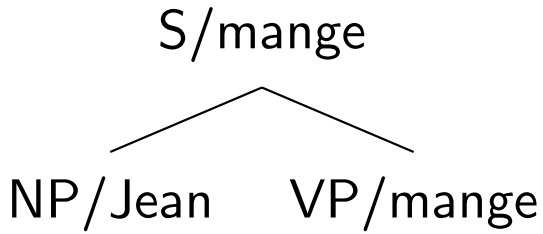

- $p(il|\text{NP}, mange)$
- $p(\text{Pro}|\text{NP}, il)$
- $p(pomme|\text{NP}, mange)$
- $p(\text{ART NC}|\text{NP}, pomme)$

$$P(T,S) = \prod_{n \in T} p(r(n)|n, h(n)) \times p(h(n)|n, h(m(n)))$$

Les choix lexicaux sont ici contrôlés par un modèle bigramme.

# PCFG + 3g > 3g [Charniak, 2001]

- Soit $c$ un constituant (un nœud) :




| | |
|---|---|
| $h(c)$ la tête du constituant | mange |
| $t(c)$ le tag de la tête du constituant | Verb |
| $e(c)$ l'expansion (markovienne) de $c$ | NP VP |
| $l(c)$ le label de $c$ | S |
| $H(c)$ l'historique de $c$ (information utile connue hors de $c$) | |

- $p(\pi) = \prod_{c \in \pi} p(t(c)|l(c), H(c)) \times p(h(c)|t(c), l(c), H(c)) \times p(e(c)|l(c), t(c), h(c), H(c))$

1. choisir un pré-terminal pour $c$
2. choisir la tête du constituant
3. choisir l'expansion de $c$

# La poutine [Charniak, 2001]

- L'auteur propose de garder de l'historique les informations suivantes :

$$H(c) = \begin{cases} m(c) & \equiv & label(mother(c)) \\ i(c) & \equiv & head(mother(c)) \\ u(c) & \equiv & POS(i(c)) \end{cases}$$

- $p(h(c)|t(c), l(c), m(c), i(c), u(c))$ est un modèle bigramme $p(manger|V, manger, S)$
- il existe une version du modèle où la tête de la grand-mère (ah ah) est considérée $\Rightarrow$ modèle trigramme

| | 3-gram | gramm. | interp. |
|---|---|---|---|
| [Chelba and Jelinek, 1998] | 167.14 | 158.28 | 148.90 |
| [Roark, 2001] | 167.02 | 152.26 | 137.26 |
| [Charniak, 2001] 2g | 167.89 | 144.98 | 133.15 |
| [Charniak, 2001] 3g | 167.89 | 130.20 | 126.07 |

# PCFG lexicalisées ou pas ? [Klein and Manning, 2003]

- Remet en cause certains gains attribués aux grammaires lexicalisées
- En étudiant des transformations d'arbres (non lexicalisés) et leur impact sur les performances
- parser : CYK, règles apprises par maximum de vraisemblance (sur le treebank transformé)

# [Klein and Manning, 2003]

*(tableau pris de l'article)*

| Vertical Order | | Horizontal Markov Order $h=0$ | $h=1$ | $h\leq 2$ | $h=2$ | $h=\infty$ |
|---|---|---|---|---|---|---|
| $v=1$ | No annotation | 71.27 (854) | 72.5 (3119) | 73.46 (3863) | 72.96 (6207) | 72.62 (9657) |
| $v\leq 2$ | Sel. Parents | 74.75 (2285) | 77.42 (6564) | 77.77 (7619) | 77.50 (11398) | 76.91 (14247) |
| $v=2$ | All Parents | 74.68 (2984) | 77.42 (7312) | 77.81 (8367) | 77.50 (12132) | 76.81 (14666) |
| $v\leq 3$ | Sel. GParents | 76.50 (4943) | 78.59 (12374) | 79.07 (13627) | 78.97 (19545) | 78.54 (20123) |
| $v=3$ | All GParents | 76.74 (7797) | 79.18 (15740) | 79.74 (16994) | 79.07 (22886) | 78.72 (22002) |

Figure 2: Markovizations: $F_1$ and grammar size.

vertical *v=1* veut dire qu'une règle A $\rightarrow \alpha$ ne dépend que de A
*v=2* veut dire que l'ancêtre de *A* est considéré, etc.

horizontal *h=2* modèle bigramme (sur les éléments de la partie droite)
*h=*$\infty$ veut dire : pas de décomposition

- les auteurs recommandent $v \leq 2$ et $h \leq 2$

# [Klein and Manning, 2003]

- leur meilleure configuration (*pris de l'article*) :

| Length ≤ 40 | LP | LR | $F_1$ | Exact | CB | 0 CB |
|---|---|---|---|---|---|---|
| Magerman (1995) | 84.9 | 84.6 | | | 1.26 | 56.6 |
| Collins (1996) | 86.3 | 85.8 | | | 1.14 | 59.9 |
| this paper | 86.9 | 85.7 | 86.3 | 30.9 | 1.10 | 60.3 |
| Charniak (1997) | 87.4 | 87.5 | | | 1.00 | 62.1 |
| Collins (1999) | 88.7 | 88.6 | | | 0.90 | 67.1 |

| Length ≤ 100 | LP | LR | $F_1$ | Exact | CB | 0 CB |
|---|---|---|---|---|---|---|
| this paper | 86.3 | 85.1 | 85.7 | 28.8 | 1.31 | 57.2 |

Figure 8: Results of the final model on the test set (section 23).

- comparable à certaines grammaires PCFG lexicalisées
- moins bon que l'état de l'art lexicalisé
- utilisé pour parser les phrases du benchmark SNLI *[Bowman et al., 2015]*

# PCFG-LA [Matsuzaki et al., 2005]

- **idée** : ajouter une annotation aux éléments non terminaux

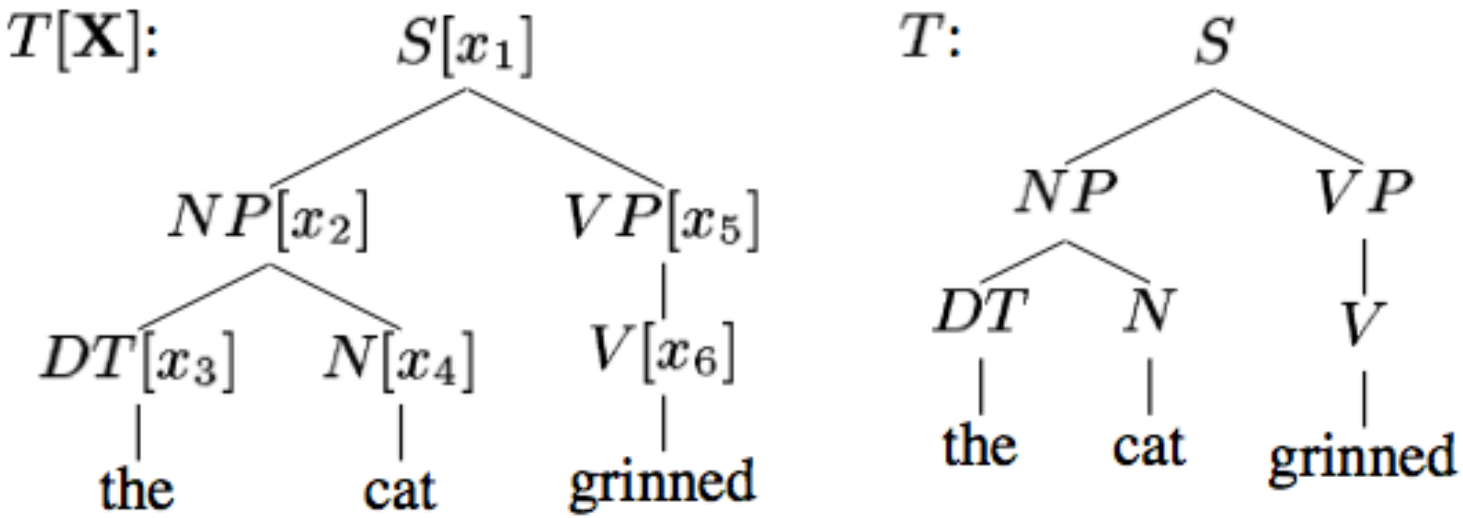



Figure 1: Tree with latent annotations $T[\mathbf{X}]$ (complete data) and observed tree $T$ (incomplete data).

- via EM (variante de inside-outside)

# PCFG-LA $= \langle N, T, H, R, \pi, \beta \rangle$

- $N$ non terminaux (observés)
- $T$ terminaux
- $H$ un ensemble de symboles latents $(x, y, \ldots)$
- $R$ un ensemble de règles CFG (observables)
- $\pi(\circ)$ prob. d'un NT annoté ($\circ = A[x]$ où $A \in N$ et $x \in H$) en racine
- $\beta(r)$ prob. d'une règle annotée ($r \in R[H]$)

# PCFG-LA $= \langle N, T, H, R, \pi, \beta \rangle$

- soit un ordre sur les nœuds d'un arbre $T$, $i = 1, \ldots, m$ (racine=1) et $x_i$ l'annotation (latente) du $i$ème nœud, et soit $X = (x_1, \ldots, x_m) \in H^m$ une annotation de l'arbre (observable) $T$

$$P(T[X]) = \pi(A_1[x_1]) \prod_r \beta(r)$$

  le produit concerne les règles annotées (ex : VP[x] $\rightarrow$ VB[y],NP[z])

- la probabilité d'un arbre observable $T$ est obtenue en marginalisant :

$$P(T) = \sum_{X \in H^m} P(T[X])$$

# Analyser avec un PCFG-LA : NP-difficile

- maximisation sur l'ensemble des arbres observables pouvant générer $w$ :

$$\hat{T} = \underset{T \in \mathcal{T}(w)}{\operatorname{argmax}} P(T|w) = \underset{T \in \mathcal{T}(w)}{\operatorname{argmax}} P(T)$$

- maximisation NP-complète $\rightarrow$ approximation
  - réduire le nombre d'arbres (observables) à l'aide d'une PCFG (observable), puis sélection à l'aide d'une PCFG-LA
  - prendre l'arbre (annoté) le plus probable (donc CYK) :

$$T[\hat{X}] = \underset{T \in \mathcal{T}(w), X \in H^{|X|}}{\operatorname{argmax}} P(T[X])$$

  - approche variationnelle (plus complexe)

# PCFG-LA

- comparable (meilleur) à [Klein and Manning, 2003] mais sans ingénierie des traits
- moins bon que les parseurs lexicalisés

| ≤ 40 words | LR | LP | CB | 0 CB |
|---|---|---|---|---|
| This paper | 86.7 | 86.6 | 1.19 | 61.1 |
| Klein and Manning (2003) | 85.7 | 86.9 | 1.10 | 60.3 |
| Collins (1999) | 88.5 | 88.7 | 0.92 | 66.7 |
| Charniak (1999) | 90.1 | 90.1 | 0.74 | 70.1 |
| ≤ 100 words | LR | LP | CB | 0 CB |
| This paper | 86.0 | 86.1 | 1.39 | 58.3 |
| Klein and Manning (2003) | 85.1 | 86.3 | 1.31 | 57.2 |
| Collins (1999) | 88.1 | 88.3 | 1.06 | 64.0 |
| Charniak (1999) | 89.6 | 89.5 | 0.88 | 67.6 |

Table 2: Comparison with other parsers.

- (environ 20 heures pour l'entraînement)

rali

# PCFG-LA [Petrov et al., 2006]

- comparable aux parseurs lexicalisés :

| ≤ 40 words | LP | LR | CB | 0CB |
|---|---|---|---|---|
| Klein and Manning (2003) | 86.9 | 85.7 | 1.10 | 60.3 |
| Matsuzaki et al. (2005) | 86.6 | 86.7 | 1.19 | 61.1 |
| Collins (1999) | 88.7 | 88.5 | 0.92 | 66.7 |
| Charniak and Johnson (2005) | 90.1 | **90.1** | **0.74** | **70.1** |
| This Paper | **90.3** | 90.0 | 0.78 | 68.5 |
| all sentences | LP | LR | CB | 0CB |
| Klein and Manning (2003) | 86.3 | 85.1 | 1.31 | 57.2 |
| Matsuzaki et al. (2005) | 86.1 | 86.0 | 1.39 | 58.3 |
| Collins (1999) | 88.3 | 88.1 | 1.06 | 64.0 |
| Charniak and Johnson (2005) | 89.5 | **89.6** | **0.88** | **67.6** |
| This Paper | **89.8** | **89.6** | 0.92 | 66.3 |

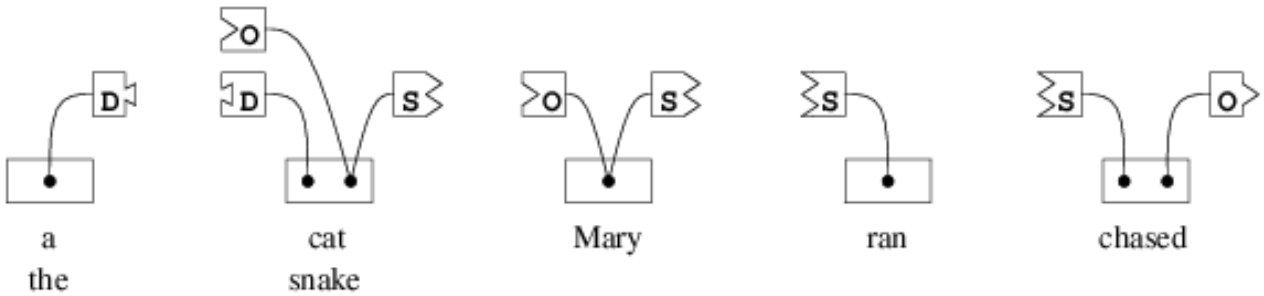
Table 4: Comparison of our results with those of others.

# Plan

# Plan

# Les link grammars [Sleator and Temperley, 1991]

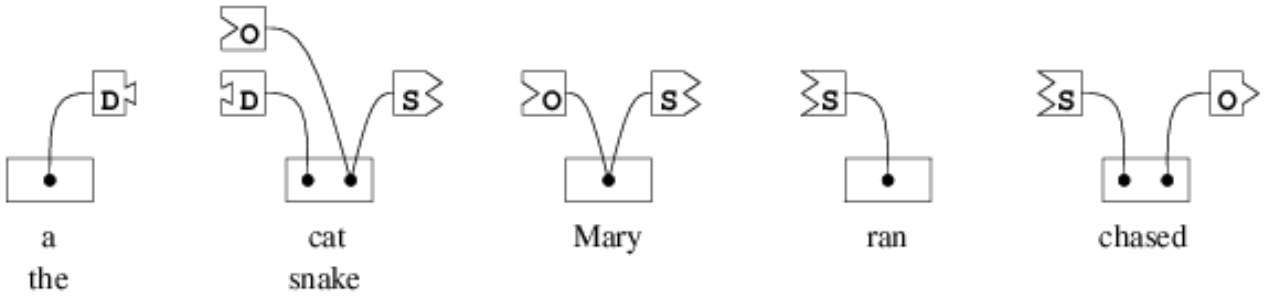
$$\text{Link grammar} = \begin{cases} \text{un ensemble de mots (les terminaux)} \\ \text{des contraintes de liage pour chaque mot} \end{cases}$$



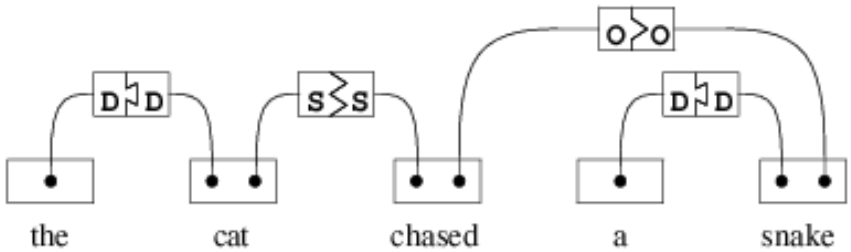

Une phrase fait partie du langage **ssi** il existe un moyen de tracer au dessus des mots des arcs (liens) tels que :

- les liens ne se croisent pas
- tous les mots sont connectés par ces liens
- les contraintes de liage de chaque mot sont satisfaites

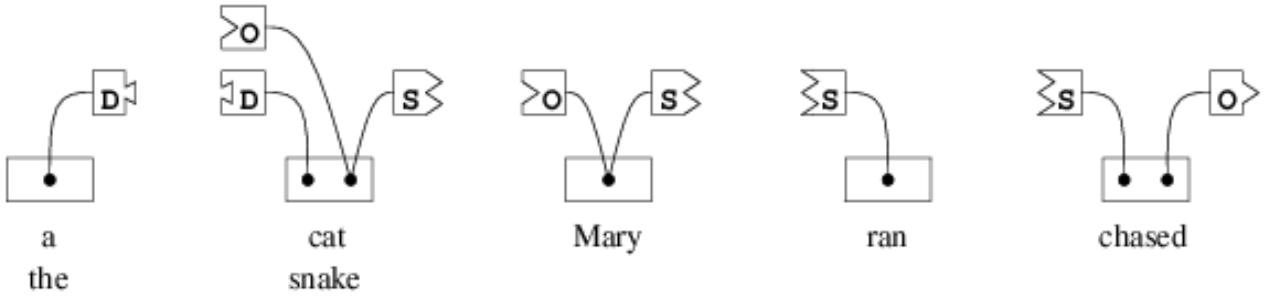
# Les link grammars



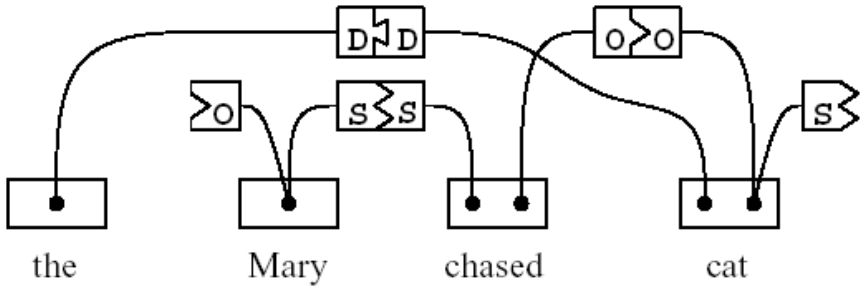

- les contraintes de liage des mots sont exprimées par des connecteurs à gauche et/ou à droite.
- un connecteur et un seul à gauche (resp. à droite) s'il en existe un, doit être satisfait (satisfait = emboîtement).



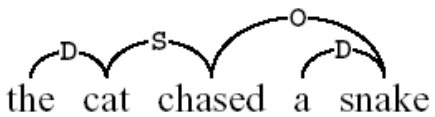
# Les link grammars



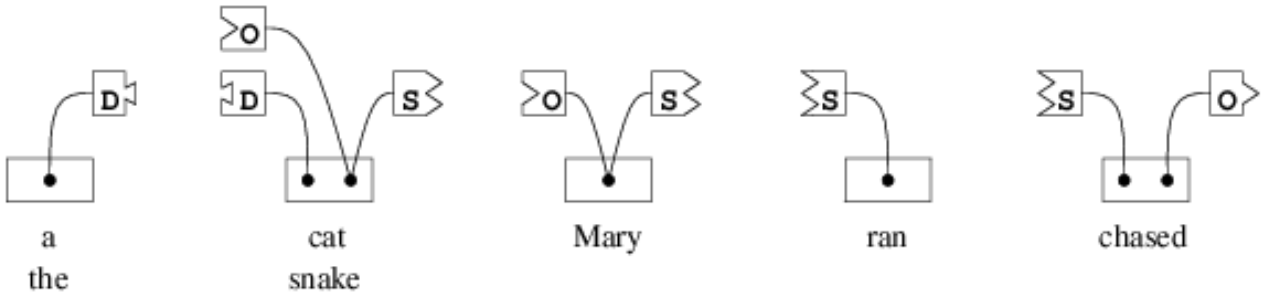

- phrase qui ne fait pas parti du langage :




- phrase en faisant partie :



# Les link grammars




| | |
|---|---|
| a the | D+ |
| snake cat | D- & (O- or S+) |
| Mary | O- or S+ |
| ran | S- |
| chased | S- & O+ |

(le ou est exclusif)

# Les link grammars

## Quelques faits

- un algorithme en $\mathcal{O}(n^3)$ (où $n$ est le nombre de mots dans la phrase) permet de trouver les liaisons d'une phrase (s'il en existe)
- dictionnaire pour l'anglais ($\sim$ 60 000 formes), l'algorithme de recherche ainsi qu'une API sont disponibles en ligne à `http://www.links.cs.cmu.edu/link`
- Il existe une version probabiliste des link grammars [Lafferty et al., 1992] (non disponible).

rali

# Plan

# Modèle de dépendance [Collins, 1996]

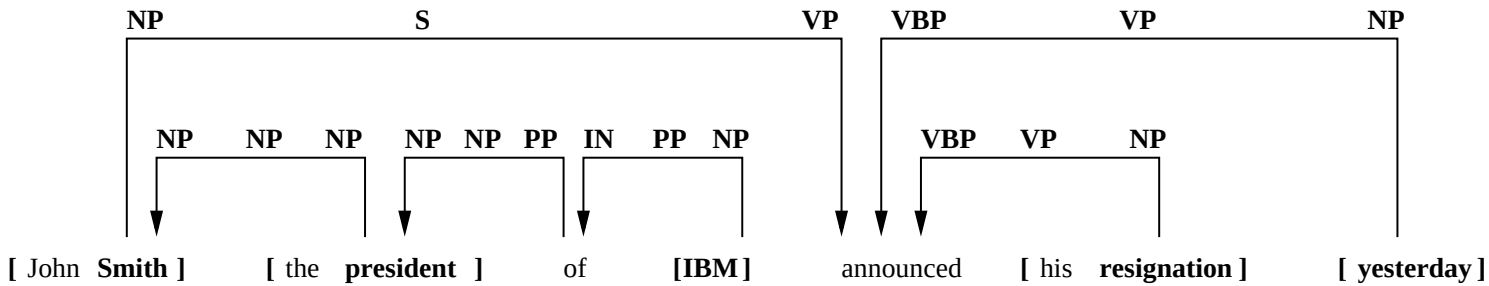
- Une phrase est représentée par une série de dépendances liant ses mots.
- Introduction d'un niveau de représentation intermédiaire simplifié d'une phrase, basé sur le découpage automatique de la phrase en groupes nominaux de base (*baseNP*) ; cad, des groupes nominaux non réentrants (pas *(NP le bruit (NP de l'eau))*)
- Versions disponibles (ça commence à dater) :
  - `www.cis.upenn.edu/~dbikel/#stat-parser`
  - `people.csail.mit.edu/mcollins/code.html`

# Modèle de dépendance [Collins, 1996]

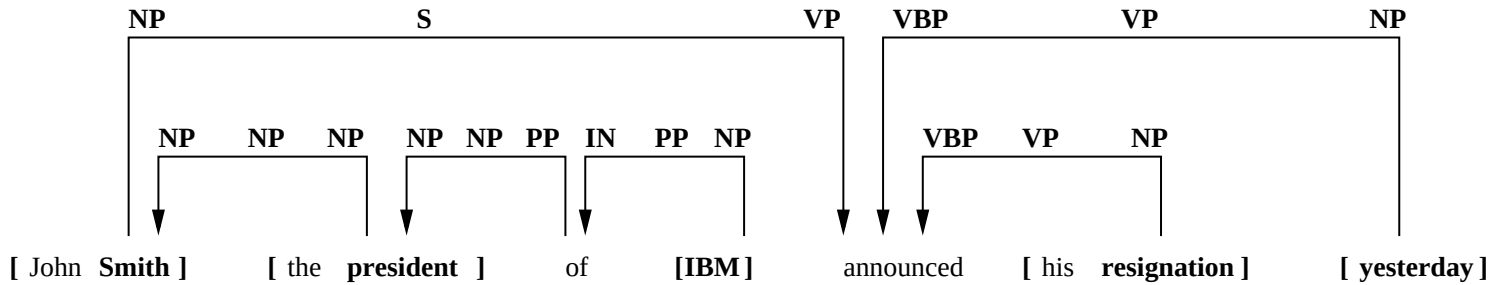
- **In :** une phrase étiquetée (*taggée*).

  (John,NNP), (Smith,NNP), (the,DT), (president,NN), (of,IN), (IBM,NNP), (announced,VBP), (his,PRP), (resignation,NN), (yesterday,NN)

- **Out :** un ensemble de dépendances $D$



# Modèle de dépendance [Collins, 1996]




- Les arcs identifient les relations syntaxiques entre les mots de tête (*head-words*) des groupes nominaux et les autres mots de la phrase réduite.
- Voici la liste des groupes nominaux (minimaux) de la phrase de départ, les mots en gras sont les têtes :

  B = { [John **Smith**] [the **president**] [**IBM**] [his **resignation**] [**yesterday**] }

# Modèle de dépendance [Collins, 1996]

- phrase réduite associée : $\overline{S} = \langle \langle Smith, NNP \rangle, \langle president, NN \rangle, \langle of, IN \rangle, \langle IBM, NNP \rangle, \langle announced, VBP \rangle, \langle resignation, NN \rangle, \langle yesterday, NN \rangle \rangle$.
- La représentation de l'arborescence sous forme de dépendances :

$$D = \left\{ \begin{array}{l} \langle Smith, announced, \langle NP, S, VP \rangle \rangle, \\ \langle president, Smith, \langle NP, NP, NP \rangle \rangle, \\ \langle of, president, \langle PP, NP, NP \rangle \rangle, \\ \langle IBM, of, \langle NP, PP, IN \rangle \rangle, \\ \langle resignation, announced, \langle NP, VP, VBP \rangle \rangle, \\ \langle yesterday, announced, \langle NP, VP, VBP \rangle \rangle \end{array} \right\}$$

- La première dépendance se lit : *Smith*, tête de groupe nominal, modifie *announced* qui est la tête d'un groupe verbal dans une relation S (relation sujet-verbe). On note cette dépendance par $AF(1) = (5, \langle NP, S, VP \rangle)$

# Modèle de dépendance [Collins, 1996]

- Analyse : $\hat{T} = \text{argmax}_T \, p(T|S)$
  avec $S = \langle (w_1, t_1), \ldots, (w_n, t_n) \rangle$ et $T = (B, D)$
  - $B$ ensemble de baseNPs
  - $D$ un ensemble de dépendances
- Modèle :

$$p(T|S) = P(B, D|S) = \underbrace{p(B|S)}_{\text{segmenteur en NP}} \times \underbrace{p(D|S, B)}_{\text{modèle de dépendance}}$$

  - $p(D|B, S) = \prod_{j=1}^{m} p(D_j|S, B)$
  - $p(B|S) = \prod_{i=2}^{n-1} \hat{p}(G_i|w_{i-1}, t_{i-1}, w_i, t_i, c_i)$
    - `John C Smith B the C president E of S IBM E has N announced S his C resignation B yesterday`
    - $c_i$ une fonction indicatrice qui indique s'il y a ou pas une virgule entre les mots $i$ et $i-1$
- 2 distributions (apprises par fréquence relative depuis le PTB)

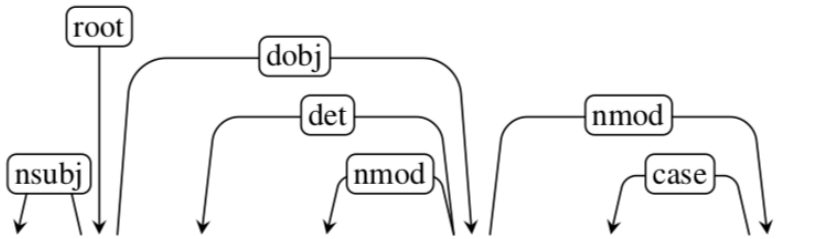
# Modèle de dépendance [Collins, 1996]

| test | LR % | LP % | CBs | 0 CBs | $\leq$ 2 CBs |
|---|---|---|---|---|---|
| $\leq$ 40 mots (2245 ph.) | 85.8 | 86.3 | 1.14 | 59.9% | 83.6% |
| $\leq$ 100 mots (2416 ph.) | 85.3 | 85.7 | 1.32 | 57.2% | 80.8% |
| [Charniak, 1996] | 80.4 | 78.8 | | 87.7% | |

- entraînement en 15 minutes ;
- analyse de 200 phrases à la minute (à l'époque)

# Plan

# Analyseur en dépendances (générique)



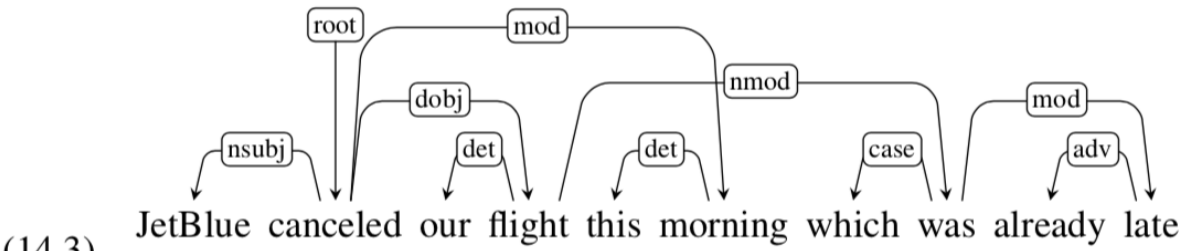

(14.1)

- chaque relation est typée et met en présence une tête (*head*) et un dépendant

| clausal arguments | | nominal modifiers | |
|---|---|---|---|
| nsubj | nominal subjet | nmod | nominal modifier |
| dobj | direct object | amod | adjectival modifier |
| iobj | indirect object | nummod | numeric modifier |
| ccomp | clausal complement | appos | appositional modifier |
| xcomp | open clausal complement | det | determiner |
| | | case | prepositions, etc. |
| nominal modifiers | | | |
| conj | conjunct | | |
| cc | coordinating conjunct | | |

# Analyseur en dépendances (générique)

- une analyse peut être projective (pas de croisement)
  - l'analyse est un arbre simplement enraciné et connecté
    - chaque mot (sauf la racine) a au plus un arc entrant (un mot peut cependant être la tête de plusieurs dépendants)
    - chemin unique de la racine à un quelconque mot
- ou pas




(14.3)

- **note :** le modèle de Colins et les link-grammars sont des analyseurs projectifs

# Analyseur par transitions

## arc-standard

- variante de shift-reduce (Aho et Ullman, 1972)
- état = configuration
  - {pile, buffer (input), D = ensemble de dépendances (parse)}
  - init : {[root], $[w_1, w_2, \ldots, w_n]$, [] }
  - final : {[root], [], D}
- actions [Covington, 2001]
  $s_1$ est le sommet de pile, $s_2$ le mot d'après :
  - leftarc ajout à D de $s_2 \leftarrow s_1$, **pop**($s_2$)
    - $s_2 \neq$root
  - rightarc ajout à D de $s_2 \rightarrow s_1$, **pop**()
  - shift empile le premier symbole du buffer (qui est éliminé du buffer)
- en pratique les actions de réductions sont étiquetées :
  leftarc(dobj), rightarc(dobj), etc.

# Comment trouver les actions ?

- les analyseurs état de l'art apprennent à prédire une action (étant donnée une configuration) de manière supervisée à l'aide d'un oracle
  - classification
- de sorte qu'au moment de l'analyse, une approche *greedy* s'avère "suffisante"
  - prendre l'action de plus fort score selon le classificateur
- d'où vient l'oracle ?
  - d'un corpus arboré (ou mieux en dépendances)

rali

# Oracle

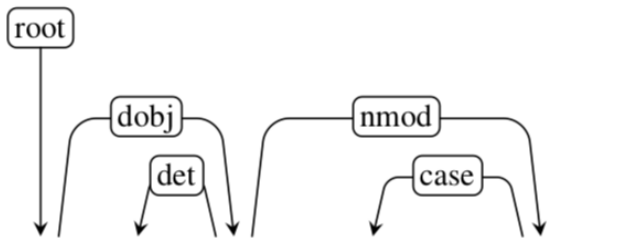
## arc-standard

- une configuration {pile,buffer,$R_c$}, $R_c$ les relations à date
- une analyse en dépendance de référence Ref
- choisir la prochaine action :
  - leftarc **si** $(s_1,r,s_2) \in$ Ref
  - rightarc **si** $(s_2,r,s_1) \in$ Ref **et** $\forall r' : (s_1, r', w) \in$ Ref **alors** $(s_1, r', w) \in R_c$
  - shift

# Oracle

## arc-standard

| | pile | input | action | $R_c$ |
|---|---|---|---|---|
| 0 | [root] | [book, the, flight, through, houston] | shift | |
| 1 | [root, book] | [the, flight, through, houston] | shift | |
| 2 | [root, book, the] | [flight, through, houston] | shift | |
| 3 | [root, book, the, flight] | [through, houston] | left | *the* ← *flight* |
| 4 | [root, book, flight] | [through, houston] | shift | |
| 5 | [root, book, flight, through] | [houston] | shift | |
| 6 | [root, book, flight, through, houston] | [] | left | *through* ← *hou* |
| 7 | [root, book, flight, houston] | [] | right | *flight* → *houst* |
| 8 | [root, book, flight] | [] | right | *book* → *flight* |
| 9 | [root, book] | [] | right | *root* → *book* |
| 10 | [root] | [] | | |




(14.7)

- On ne peut pas faire de rightarc car les dépendants de `flight` ne sont pas encore traités
- Ces paires (configuration, action) constituent un corpus d'entraînement

# Features

- le même genre de features que pour faire de l'étiquetage (tagging)
- en théorie : toute info extraite d'une configuration {pile, buffer, $R_C$}
- en pratique : on se concentre sur le haut de la pile et le préfixe du buffer de lecture à l'aide de patrons :

| | | | |
|---|---|---|---|
| word | $s_1.w$ | $s_1.t$ | $s_1.wt$ |
| | $s_2.w$ | $s_2.t$ | $s_2.wt$ |
| | $b_1.w$ | $b_1.t$ | $b_2.wt$ |
| 2-gram | $s_1.w \circ s_2.w$ | $s_1 \circ s_2.t$ | $s_1.t \circ b_1.w$ |
| | $s_1.t \circ s_2.wt$ | $s_1.w \circ s_2.w \circ s_2.t$ | $s_1.w \circ s_1.t \circ s_2.t$ |
| | ... | | |

- où $s_i$ désigne le $i$e sommet de pile, $b_i$ le $i$e symbole du buffer
- $.t$ désigne le tag, $.w$ le mot, $.wt$ la concaténation du mot et du tag

# Features

- configuration :

| | pile | input | action | $R_c$ |
|---|---|---|---|---|
| 3 | [root, book, the, flight] | [through, houston] | left | $+the \leftarrow flight$ |

# Features

- configuration :

| | pile | input | action | $R_c$ |
|---|---|---|---|---|
| 3 | [root, book, the, flight] | [through, houston] | left | $+the \leftarrow flight$ |

- patrons :

| | | | |
|---|---|---|---|
| word | $s_1.w$ | $s_1.t$ | $s_1.wt$ |
| | $s_2.w$ | $s_2.t$ | $s_2.wt$ |
| | $b_1.w$ | $b_1.t$ | $b_2.wt$ |
| 2-gram | $s_1.w \circ s_2.w$ | $s_1.t \circ s_2.t$ | $s_1.t \circ b_1.w$ |
| | $s_1.t \circ s_2.wt$ | $s_1.w \circ s_2.w \circ s_2.t$ | $s_1.w \circ s_1.t \circ s_2.t$ |
| | ... | | |

# Features

- configuration :

| | pile | input | action | $R_c$ |
|---|---|---|---|---|
| 3 | [root, book, the, flight] | [through, houston] | left | $+the \leftarrow flight$ |

- patrons :

| | | | |
|---|---|---|---|
| word | flight | NN | flight.NN |
| | the | DT | the.DT |
| | through | through | houston.NNP |
| 2-gram | flight.the | NN.DT | NN.through |
| | NN.the.DT | flight.the.DT | flight.NN.DT |
| | ... | | |

- voir chaque feature comme une fonction booléenne :
  - *est-ce que* $s_1.t \circ s_2.wt$ *vaut* $NN.the.DT$ *?*
- une configuration est donc représentée par un vecteur binaire de haute dimension avec des 1 là où le feature s'allume

# Vous avez dit feature ?

# [Zhang and Nivre, 2011]

| from single words |
|---|
| $S_0wp$; $S_0w$; $S_0p$; $N_0wp$; $N_0w$; $N_0p$; $N_1wp$; $N_1w$; $N_1p$; $N_2wp$; $N_2w$; $N_2p$; |
| **from word pairs** |
| $S_0wpN_0wp$; $S_0wpN_0w$; $S_0wN_0wp$; $S_0wpN_0p$; $S_0pN_0wp$; $S_0wN_0w$; $S_0pN_0p$ $N_0pN_1p$ |
| **from three words** |
| $N_0pN_1pN_2p$; $S_0pN_0pN_1p$; $S_{0h}pS_0pN_0p$; $S_0pS_{0l}pN_0p$; $S_0pS_{0r}pN_0p$; $S_0pN_0pN_{0l}p$ |

Table 1: Baseline feature templates.

- $S_o$ est $s_1$ dans les slides précédents, $N_0$ est $b_1$, $p$ (pos) correspond à $t$ (tag)
- $S_0wpN_0wp$ correspond donc à $s_1.wt \circ b_1.wt$

# Vous avez dit feature ?

| distance |
|---|
| $S_0wd$; $S_0pd$; $N_0wd$; $N_0pd$; $S_0wN_0wd$; $S_0pN_0pd$; |
| valency |
| $S_0wv_r$; $S_0pv_r$; $S_0wv_l$; $S_0pv_l$; $N_0wv_l$; $N_0pv_l$; |
| unigrams |
| $S_{0h}w$; $S_{0h}p$; $S_0l$; $S_{0l}w$; $S_{0l}p$; $S_{0l}l$; $S_{0r}w$; $S_{0r}p$; $S_{0r}l$;$N_{0l}w$; $N_{0l}p$; $N_{0l}l$; |
| third-order |
| $S_{0h2}w$; $S_{0h2}p$; $S_{0h}l$; $S_{0l2}w$; $S_{0l2}p$; $S_{0l2}l$; $S_{0r2}w$; $S_{0r2}p$; $S_{0r2}l$; $N_{0l2}w$; $N_{0l2}p$; $N_{0l2}l$; $S_0pS_{0l}pS_{0l2}p$; $S_0pS_{0r}pS_{0r2}p$; $S_0pS_{0h}pS_{0h2}p$; $N_0pN_{0l}pN_{0l2}p$; |
| label set |
| $S_0ws_r$; $S_0ps_r$; $S_0ws_l$; $S_0ps_l$; $N_0ws_l$; $N_0ps_l$; |

Table 2: New feature templates.

- $d$ la distance entre $s_1$ et $b_1$
- $v_{r/l}$ la valence

# [Zhang and Nivre, 2011]

(droite/gauche) : le nombre de modifiers (à droite/à gauche) d'un mot

- $h$ la tête d'un mot, $l$ et $r$ les modifiers immédiatement à gauche/droite d'un mot
  - $S_{0h}w$ signifie le mot associé à la tête du sommet de pile ($s_1$)
- $h2$ désigne la tête de la tête d'un mot, $l2/r2$ le deuxième modificateur (gauche/droite) d'un mot
- $s_l/s_r$ set des labels (ex : `nsubj`) gauche/droite d'un mot

# So what ?

# [Zhang and Nivre, 2011]

| feature | UAS | UEM |
|---|---|---|
| baseline | 92.18% | 45.76% |
| +distance | 92.25% | 46.24% |
| +valency | 92.49% | 47.65% |
| +unigrams | 92.89% | 48.47% |
| +third-order | 93.07% | 49.59% |
| +label set | 93.14% | 50.12% |

Table 3: The effect of new features on the development set for English. UAS = unlabeled attachment score; UEM = unlabeled exact match.

- les features non locaux aident
- améliore les perfs en test sur le PTB et sur un treebank chinois

rali

# Entraîner un classifieur

- chaque paire $(c_i, op_i)$ est un exemple d'apprentissage duquel des features sont extraits
- usual suspects :
  - Random Forest
  - SVM
  - Régression logistique (maxent)
  - Voted Perceptron (voir slides *aml*)
- note : les features sont
  - sparses (si lexicalisés) $\rightarrow$ faible pouvoir de généralisation
  - coûteux à manipuler ($>$ 90% du temps d'analyse)

rali

# Évaluation d'un analyseur en dépendance

En comparaison à une référence

- LAS Label Attachment Score
  - 1 point si on a une dépendance (arc) correcte (étiquette incluse)
  - % des mots de l'entrée qui sont correctement liés
  - voir aussi CLAS [Nivre and Fang, 2017] (LAS sur les mots de contenu)
- UAS Unlabeled Attachment Score
  - 1 point si on a une dépendance (arc) correcte sans regard à l'étiquette
- (U)EM (Unlabeled) Exact match
  - 1 point si tous les liens d'une phrase sont identiques à ceux de la référence (en considérant ou non le type d'arc)

rali

# Évaluation d'un analyseur en dépendance

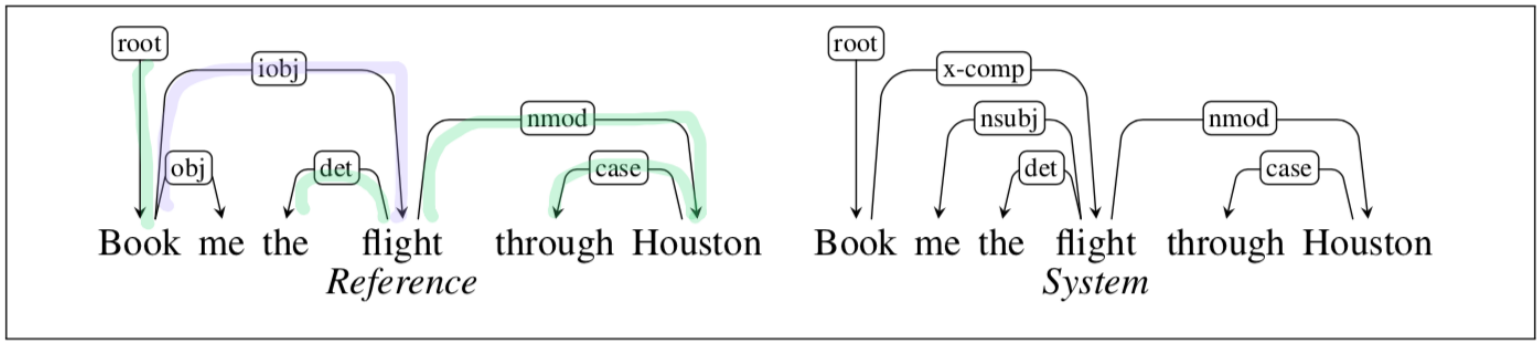



**Figure 14.15** Reference and system parses for *Book me the flight through Houston*, resulting in an LAS of 3/6 and an UAS of 4/6.

- EM et UEM = 0

# Variante : arc-eager

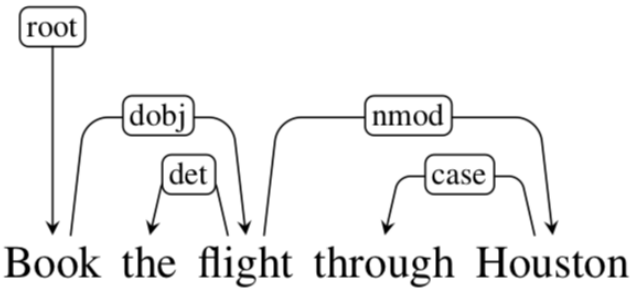
- leftarc **assert**($s_1 \leftarrow b_1$), **pop()**
- rightarc **assert**($s_1 \rightarrow b_1$) ; **shift**($b_1$)
- shift $b_1$ est mis sur la pile
- reduce **pop()**

- arc-standard on élimine de la pile le dépendant, ce qui nécessite que toutes ses dépendances soient déjà traitées (imposant des `shift`)
- arc-eager rightarc est fait même si les dépendants ne sont pas encore analysés car le dépendant est empilé ce qui permet de s'en servir
- opérations conditionnées sur $b_1$ et $s_1$ (et non $s_1$ et $s_2$)

rali

# arc-eager

| | pile | input | action | $R_c$ |
|---|---|---|---|---|
| 0 | [root] | [book, the, flight, through, houston] | right | *root* → *book* |
| 1 | [root, book] | [the, flight, through, houston] | shift | |
| 2 | [root, book, the] | [flight, through, houston] | left | *the* ← *flight* |
| 3 | [root, book] | [flight, through, houston] | right | *book* → *flight* |
| 4 | [root, book, flight] | [through, houston] | shift | |
| 5 | [root, book, flight, through] | [houston] | left | *through* ← *houston* |
| 6 | [root, book, flight] | [houston] | right | *flight* → *houston* |
| 7 | [root, book, flight, houston] | [] | reduce | |
| 8 | [root, book, flight] | [] | reduce | |
| 9 | [root, book] | [] | reduce | |
| 10 | [root] | [] | | |




(14.7)

- rightarc dès le début (fait à la fin dans arc-standard)
- leftarc : $b_1$ reste dans le buffer (peut encore gouverner)
- rightarc : le dépendant est empilé (peut encore gouverner) – voir état 3

# Recherche en faisceau

- Si le score d'une transition $t_i$ dépend de l'historique $(c_{j<\ i})$ :

$$\begin{aligned} score(c_o) &= 0.0 \\ score(c_i) &= score(c_{i-1}) + sc(t_i, c_{i-1}) \end{aligned}$$

- alors la recherche *greedy* n'est plus optimale, et il faut construire un espace de recherche qu'il faut (généralement) filtrer
- beam search consiste à maintenir les $b$ meilleurs hypothèses ($b$ est la taille du beam), les autres sont éliminées
  - politique de filtrage (souvent difficile de comparer des hypothèses)
  - on ne peut plus entrainer un classificateur local, on doit embarquer le search pour choisir la prochaine action (structured learning)
  - amène généralement des gains [Johansson and Nugues, 2007]
  - note : si $sc(t_i, c_{i-1}) \equiv sc(t_i)$ alors greedy est optimal

rali

# Plan

# Analyseurs basés sur les graphes

- Espace de recherche = ensemble des arbres d'analyse de $S$
- $\hat{t} = \text{argmax}_t \, score(t, S)$
- Le plus souvent : $score(t, s) = \sum_{e \in t} score(e)$ (comme les pcfgs)
- Parsing = maximum spanning tree
  - spanning tree = sous graphe sans cycle, avec un lien entrant sur chaque sommet (sauf la racine)
  - maximum spanning tree = spanning tree de score maximum
  - **init :** partir du graphe constitué de tous les arcs (dirigés) et de leurs poids (appris de manière supervisée)
- Motivation :
  - problème des dépendances longues
  - analyse non projective

rali

# MST

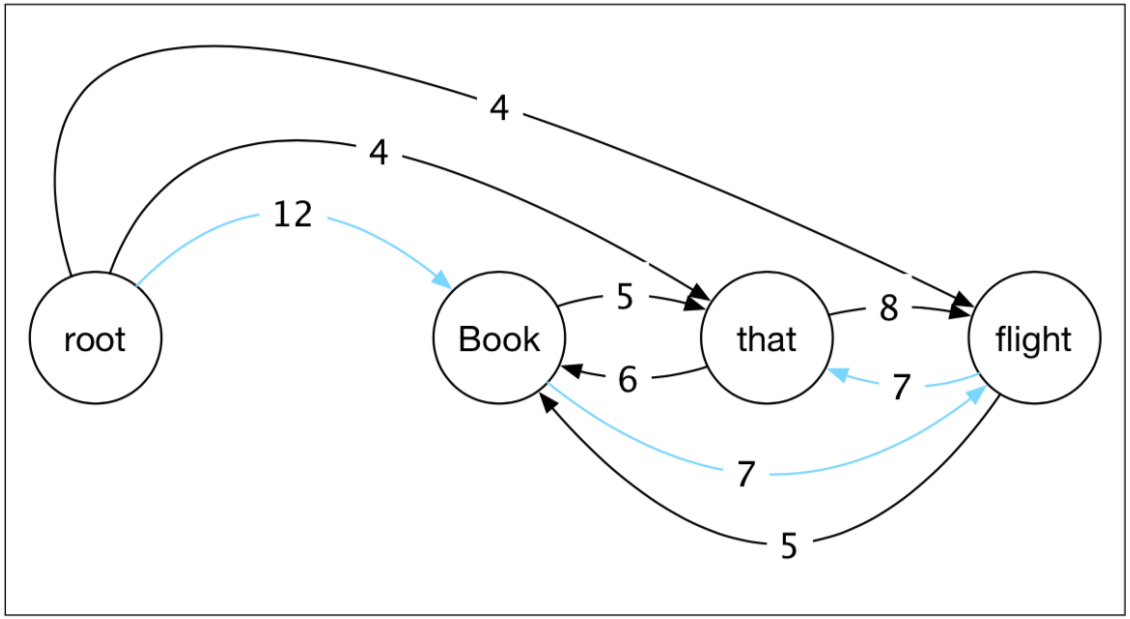



**Figure 14.12** Initial rooted, directed graph for *Book that flight*.

- 2 algos proposés $\sim$ en même temps en $O(n^3)$
  - [Chu and Liu, 1965] et https://en.wikipedia.org/wiki/Edmonds%27_algorithm
- 1 algo en $O(m + n \log n)$
  $n$ = nombre de sommets (mots) et $m$ le nombre d'arcs

# Features

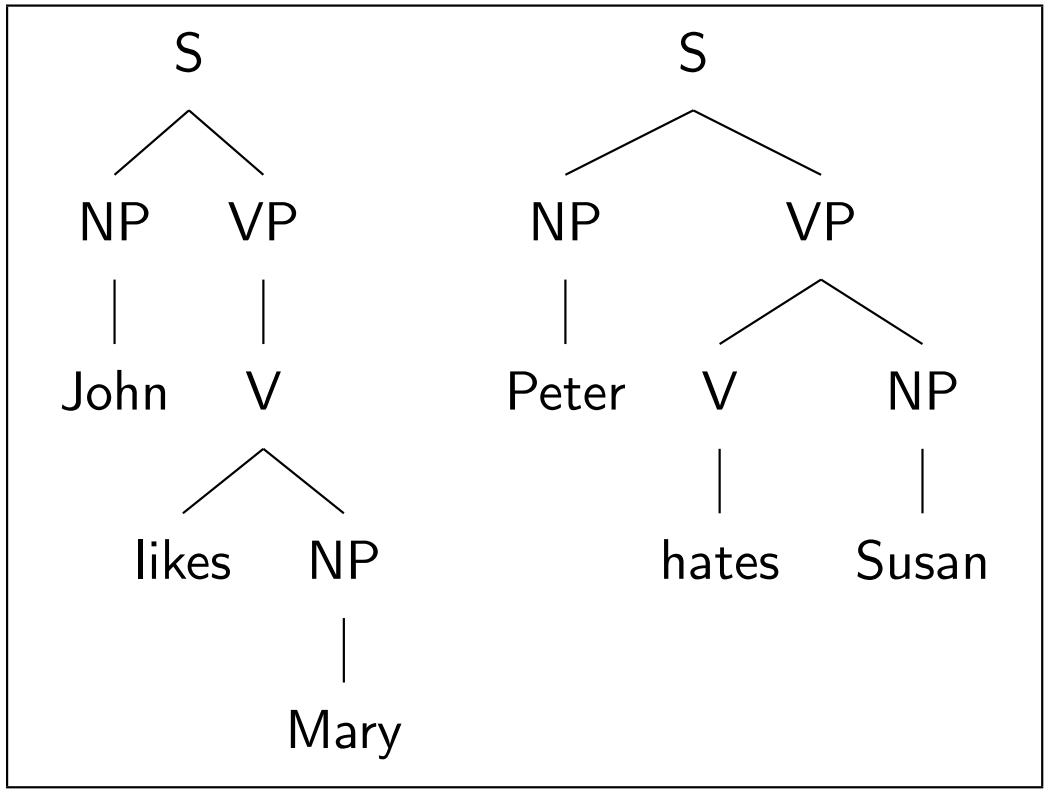
- plus ou moins les mêmes que pour les analyseurs en transition :
  - mots/lemmes/pos concernés par un lien
  - informations du contexte (mots voisins des mots reliés)
  - la nature de la relation, la direction du lien
  - etc.
- et les mêmes algos de prédiction structurée pour apprendre leur poids

rali

# Plan

# Data Oriented Parsing (DOP) [Bod, 1992]

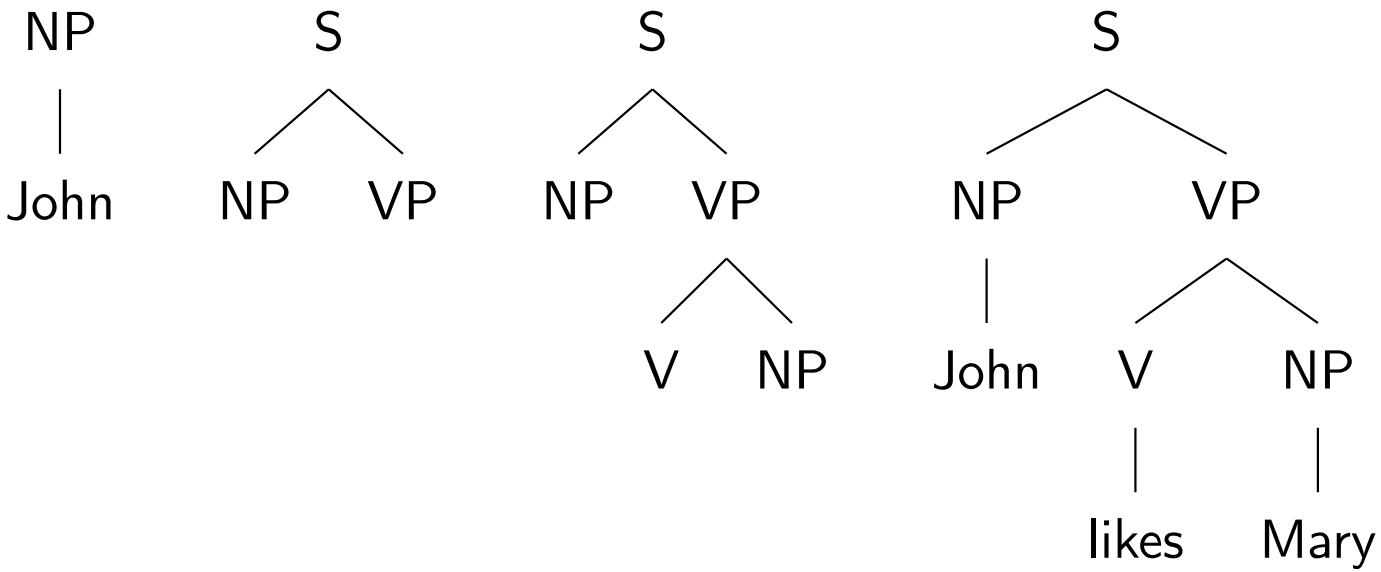
- utiliser les sous-arbres d'un treebank pour construire l'analyse d'une nouvelle phrase.




- construction d'une collection de sous-arbres élémentaires permettant — par composition — de construire de nouveaux arbres

# Sous-arbre valide d'un arbre (fragment)

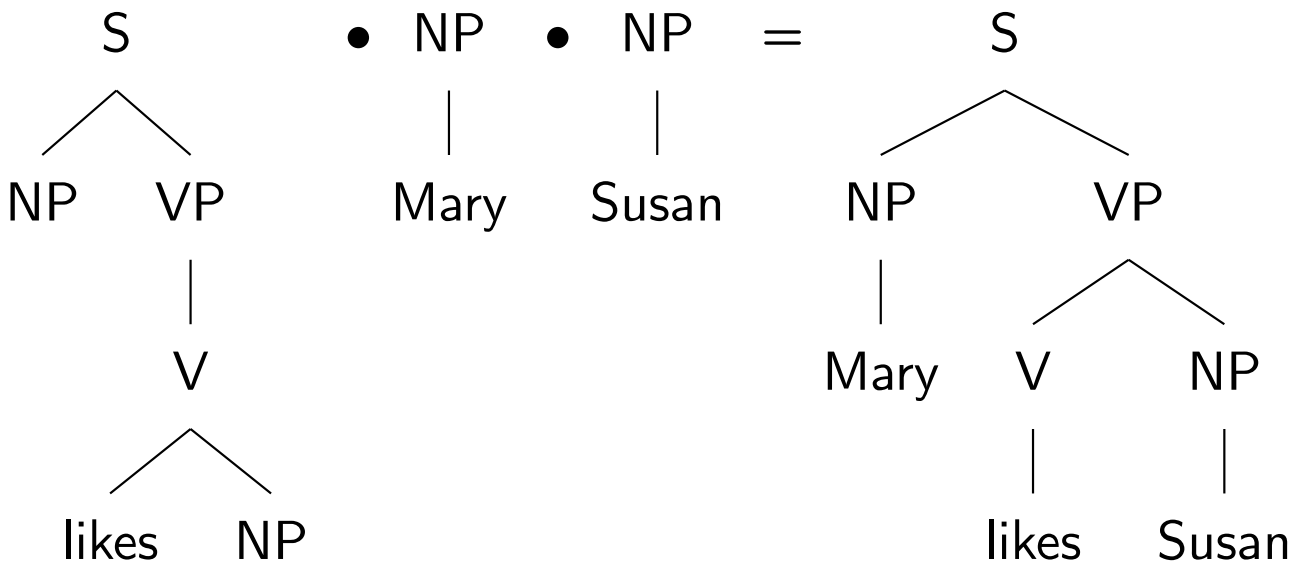
- Un fragment $t$ d'un arbre $T$ est un sous-graphe de $T$ qui vérifie :
  - $t$ possède au moins deux nœuds
  - $t$ est connecté
  - les nœuds non frontaliers de $t$ ont les mêmes nœuds fils que $T$.




- De notre treebank, on peut extraire 34 fragments

# Mary likes Susan

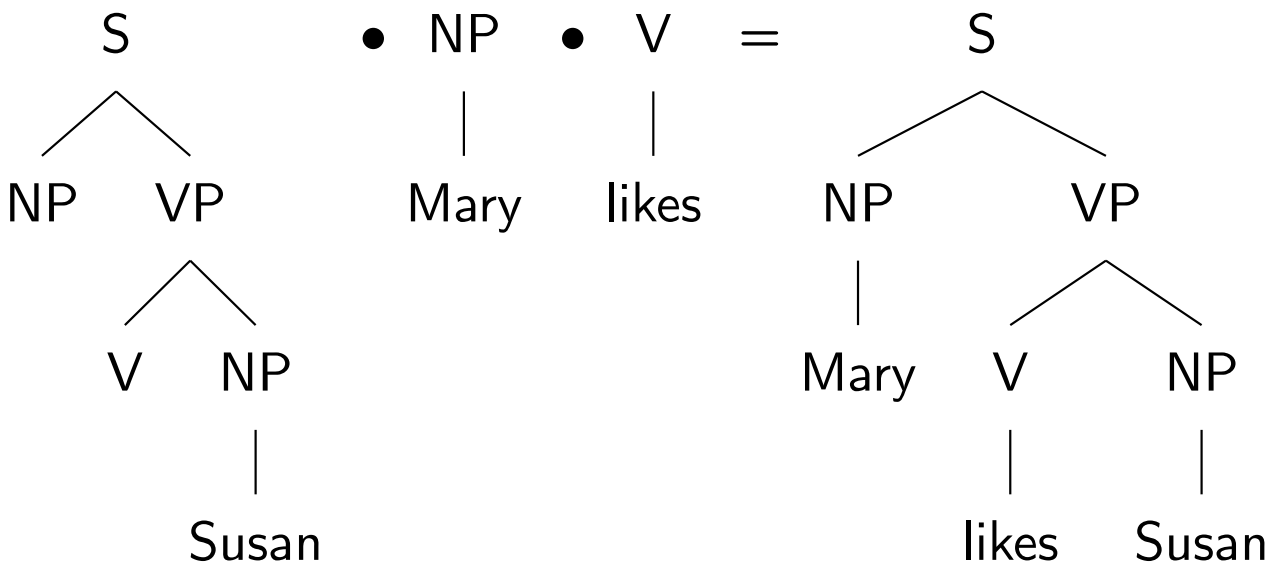
- • un opérateur compositionnel associatif à gauche
  - une composition possible :




  - $(t_1 \bullet t_2)$ s'applique sur le non terminal le plus à gauche de $t_1$
  - correspondance des labels des nœuds composés

# Mary likes Susan

- autre composition :



# DOP-I

- probabilité d'un sous-arbre $t$ :

$$p(t) = \frac{|t|}{\sum_{t':r(t)=r(t')} |t'|}$$

où $r(t)$ désigne la racine de $t$ et $|t|$ désigne le nombre de fois où $t$ apparaît dans la base.

- probabilité d'une dérivation $D = t_1 \bullet \ldots \bullet t_n$ :

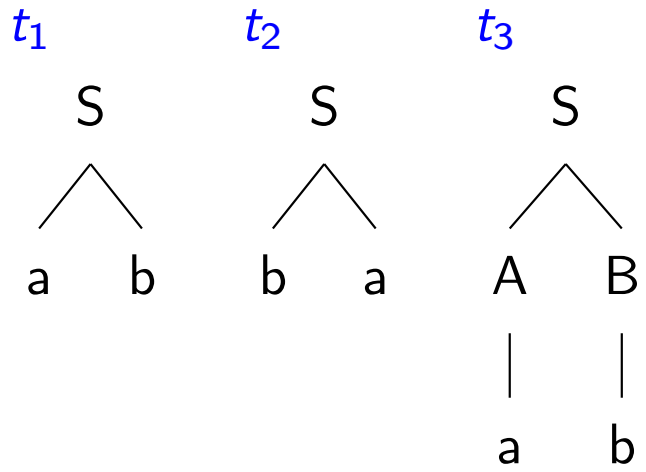
$$p(D) = \prod_{i=1}^{n} p(t_i)$$

- probabilité d'un arbre :

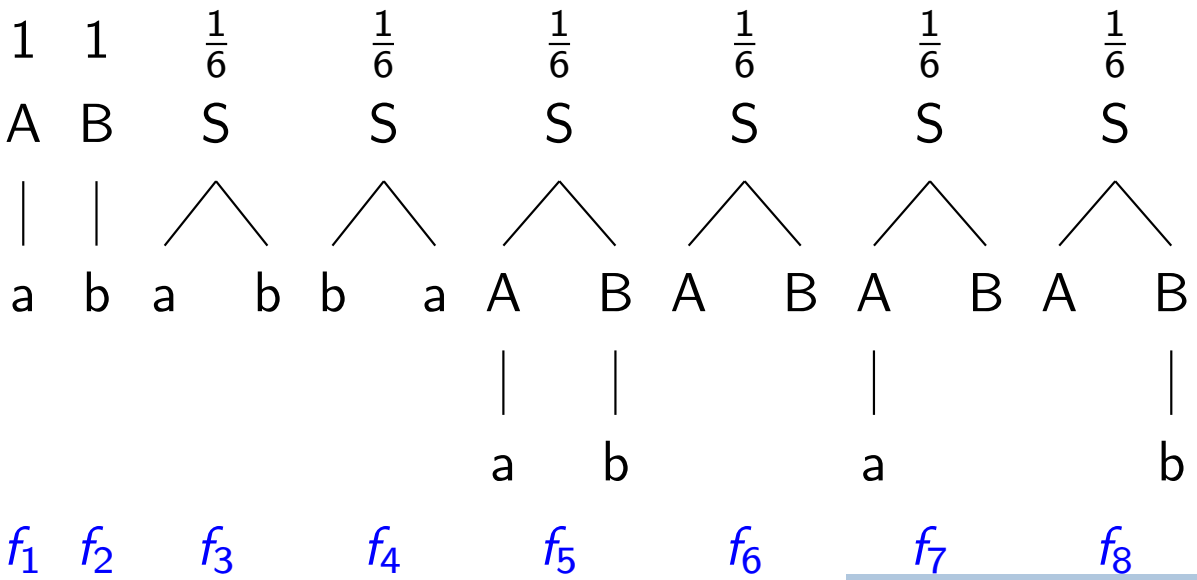
$$p_{\text{dop}}(t) = \sum_{d} \prod_{i} p(t_{id})$$

# DOP-I

- soit le treebank :




- et la collection associée est constituée des 8 fragments suivants :



# DOP-I

- 4 dérivations permettent de construire $t_3$ :
  - $p(d_1) \equiv p(f_5) = 1/6$
  - $p(d_2) \equiv p(f_6 \bullet f_1 \bullet f_2) = 1/6 \times 1 \times 1$
  - $p(d_3) \equiv p(f_7 \bullet f_2) = 1/6 \times 1$
  - $p(d_4) \equiv p(f_8 \bullet f_1) = 1/6 \times 1$
- donc :
  - $p_{dop}(t_3) = \sum_{i=1}^{4} p(d_i) = \frac{4}{6} = \frac{2}{3}$
  - $p_{dop}(t_1) = \frac{1}{6}$
- note : $p_{dop}(t_3) > p_{dop}(t_1)$
  - les arbres les plus profonds dominent la masse de probabilité.

# Limites

- Domination des arbres les plus profonds dans le treebank
- Nombre de fragments rapidement trop grand
- Trouver l'arbre le plus probable est une opération NP-complète qui implique de l'échantillonnage [Goodman, 1998, Bod, 2000].

rali

# Plan

# Plan

# SANCL 2012 ([Petrov and McDonald, 2012])

- Google Web Treebank comme alternative au PTB (pour l'anglais)
- mettre en avant les enjeux : adaptation au domaine et au type de textes (forum, etc.)

# Google Web Treebank

- 5 domaines (plus d'un million de phrases par domaine) :
  Yahoo ! Answers, emails, newsgroups, local business reviews, blogs
- un échantillon annoté par des linguistes (*pris de l'article*) :

| | Training | Development | | | Evaluation | | | |
|---|---|---|---|---|---|---|---|---|
| | WSJ-train | Emails | Weblogs | WSJ-dev | Answers | Newsgroups | Reviews | WSJ-eval |
| Sentences | 30,060 | 2,450 | 1,016 | 1,336 | 1,744 | 1,195 | 1,906 | 1,640 |
| Tokens | 731,678 | 29,131 | 24,025 | 32,092 | 28,823 | 20,651 | 28,086 | 35,590 |
| Types | 35,933 | 5,478 | 4,747 | 5,889 | 4,370 | 4,924 | 4,797 | 6,685 |
| OOV | 0.0% | 30.7% | 19.6% | 11.8% | 27.7% | 23.1% | 29.5% | 11.5% |

Table 1: Training, development and evaluation data statistics. Shown are the number sentences, tokens and unique types in the data. OOV is the percentage of tokens in the data set that are not observed in WSJ-train.

rali

# Tâche partagée : conditions

- 2 tracks : dépendances et constituants
- données disponibles :

  train :

  - sections 0-21 du PTB (30,060 phrases analysées)
  - (27k à 2M de) phrases non annotées par domaine

  dev : phrases annotées syntaxiquement :

  - 1 016 *weblog* + 2 450 *emails*
  - section 22 PTB (1 336 phrases)

  test :

  - 1k à 2k de phrases des domaines *answers*, *newsgroups* et *reviews*
  - section 23 du PTB (1.6k phrases) *contraste*

# Tâche partagée : Résultats (*tableau pris de l'article*)

- baseline : Berkeley parser ([Petrov et al., 2006], PCFG-LA)
- résultats des analyseurs (en constituants) :

| | Domain A (Answers) | | | | Domain B (Newsgroups) | | | | Domain C (Reviews) | | | | Domain D (WSJ) | | | | Average (A-C) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Team | LP | LR | F1 | POS | LP | LR | F1 | POS | LP | LR | F1 | POS | LP | LR | F1 | POS | LP | LR | F1 | POS |
| BerkeleyParser | 75.86 | 75.98 | 75.92 | 90.20 | 77.87 | 78.42 | 78.14 | 91.24 | 77.65 | 76.68 | 77.16 | 89.33 | 88.34 | 88.08 | 88.21 | 97.08 | 77.13 | 77.03 | 77.07 | 90.26 |
| OHSU | 73.21 | 74.60 | 73.90 | 90.15 | 73.22 | 75.48 | 74.33 | 91.14 | 76.31 | 76.22 | 76.27 | 90.05 | 83.17 | 83.79 | 83.48 | 96.84 | 74.25 | 75.43 | 74.83 | 90.45 |
| Vanderbilt | 75.09 | 76.78 | 75.93 | **91.76** | 78.10 | 79.05 | 78.57 | 92.91 | 77.74 | 78.18 | 77.96 | 91.94 | 87.82 | 88.00 | 87.91 | 97.49 | 76.98 | 78.00 | 77.49 | 92.20 |
| IMS | 79.46 | 78.10 | 78.78 | 90.22 | 80.85 | 80.12 | 80.48 | 91.09 | 81.31 | 78.61 | 79.94 | 89.93 | 89.83 | 88.96 | 89.39 | 97.31 | 80.54 | 78.94 | 79.73 | 90.41 |
| Stanford | 78.79 | 77.91 | 78.35 | 91.21 | 81.41 | 80.49 | 80.95 | 91.62 | 81.95 | 80.32 | 81.13 | 92.45 | 90.00 | 88.93 | 89.46 | 97.01 | 80.72 | 79.57 | 80.14 | 91.76 |
| DCU-Paris13-2 | 80.02 | 79.22 | 79.62 | 91.61 | 83.13 | 82.18 | 82.65 | **93.60** | 82.92 | 82.12 | 82.52 | **92.96** | 88.43 | 88.29 | 88.36 | 97.29 | 82.02 | 81.17 | 81.60 | **92.72** |
| Alpage-1 | 80.67 | 80.36 | 80.52 | 91.17 | 84.22 | 83.12 | 83.67 | 93.22 | 82.01 | 81.04 | 81.52 | 91.58 | 90.20 | 89.62 | 89.91 | 97.20 | 82.30 | 81.51 | 81.90 | 91.99 |
| Alpage-2 | 80.77 | 80.43 | 80.60 | 91.14 | 84.71 | 83.36 | 84.03 | 92.58 | 82.28 | 81.24 | 81.76 | 91.63 | 90.19 | 89.56 | 89.87 | 97.22 | 82.59 | 81.68 | 82.13 | 91.78 |
| DCU-Paris13-1 | **82.96** | **81.43** | **82.19** | 91.63 | **85.01** | **83.65** | **84.33** | 93.39 | **84.79** | **83.29** | **84.03** | 92.89 | **90.75** | **90.32** | **90.53** | **97.53** | **84.25** | **82.79** | **83.52** | 92.64 |

Table 3: Constituency parsing results. BerkeleyParser is the baseline model trained only on newswire. Numbers in bold are the highest score for each metric. Note that the average is computed only across the web domains.

# Plan

# http://universaldependencies.org/conll17/

- basé sur UD 2.0
- $>$ 50 langues
- 64 UD treebanks pour l'entraînement (56 avec un dev)
- fourni :
  - tokenization de référence, découpage en phrases, pos, +lemmes/analyse morphologique lorsque disponible
  - grandes quantités de textes
    - 9.4G mots en anglais, 28K en **Old Church Slavonic**
    - annotés automatiquement avec UDPipe http://ufal.mff.cuni.cz/udpipe
  - embeddings (dimension fixée à 100)
    - pré-entraînés avec Word2Vec [Mikolov et al., 2013]
    - http://hdl.handle.net/11234/1-1989

rali

# http://universaldependencies.org/conll17/

- test sets pour 63 des 64 treebanks, contenant au moins 10k mots
  - non distribué avec le matériel d'entraînement
  - +4 *surprise languages* (pas de train)
- Les participants ont remis un système (woh !)
  - minimum : 1CPU, 4G RAM (système gagnant : 4 CPUs, 16G RAM)
  - baseline : UDPipe
  - participants : 33, 12 dépassant le baseline

# http://universaldependencies.org/conll17/

| | Team | CLAS $F_1$ |
|---|---|---|
| 1. | Stanford (Stanford) | 72.57 |
| 2. | C2L2 (Ithaca) | 70.91 |
| 3. | IMS (Stuttgart) | 70.18 |
| 4. | HIT-SCIR (Harbin) | 67.63 |
| 5. | LATTICE (Paris) | 66.16 |
| 6. | NAIST SATO (Nara) | 65.15 |
| 7. | Koç University (İstanbul) | 64.61 |
| 8. | ÚFAL – UDPipe 1.2 (Praha) | 64.36 |
| 9. | Orange – Deskiñ (Lannion) | 64.15 |
| 10. | TurkuNLP (Turku) | 63.61 |
| 11. | UParse (Edinburgh) | 63.55 |
| 12. | darc (Tübingen) | 63.24 |
| 13. | BASELINE UDPipe 1.1 | 63.02 |

Table 3: Average CLAS $F_1$ score.

- Stanford : un modèle neuronal, graph-based
- http://universaldependencies.org/conll17/proceedings/pdf/K17-3002.pdf
- https://arxiv.org/pdf/1611.01734.pdf

rali

# Plan

# Outils

- transformation d'un arbre d'analyse en dépendances :
  - `Penn2Malt` `http://w3.msi.vxu.se/nivre/research/Penn2Malt.html`
  - [de Marneffe et al., 2006]
- analyseurs :
  - CoreNLP `https://nlp.stanford.edu/software/lex-parser.html`
  - SyntaxNet `https://www.tensorflow.org/versions/r0.12/tutorials/syntaxnet/`
  - puck `https://github.com/dlwh/puck`
  - MaltParser `http://www.maltparser.org`
  - MSTParser `http://www.seas.upenn.edu/~strctlrn/MSTParser/MSTParser.html`

rali

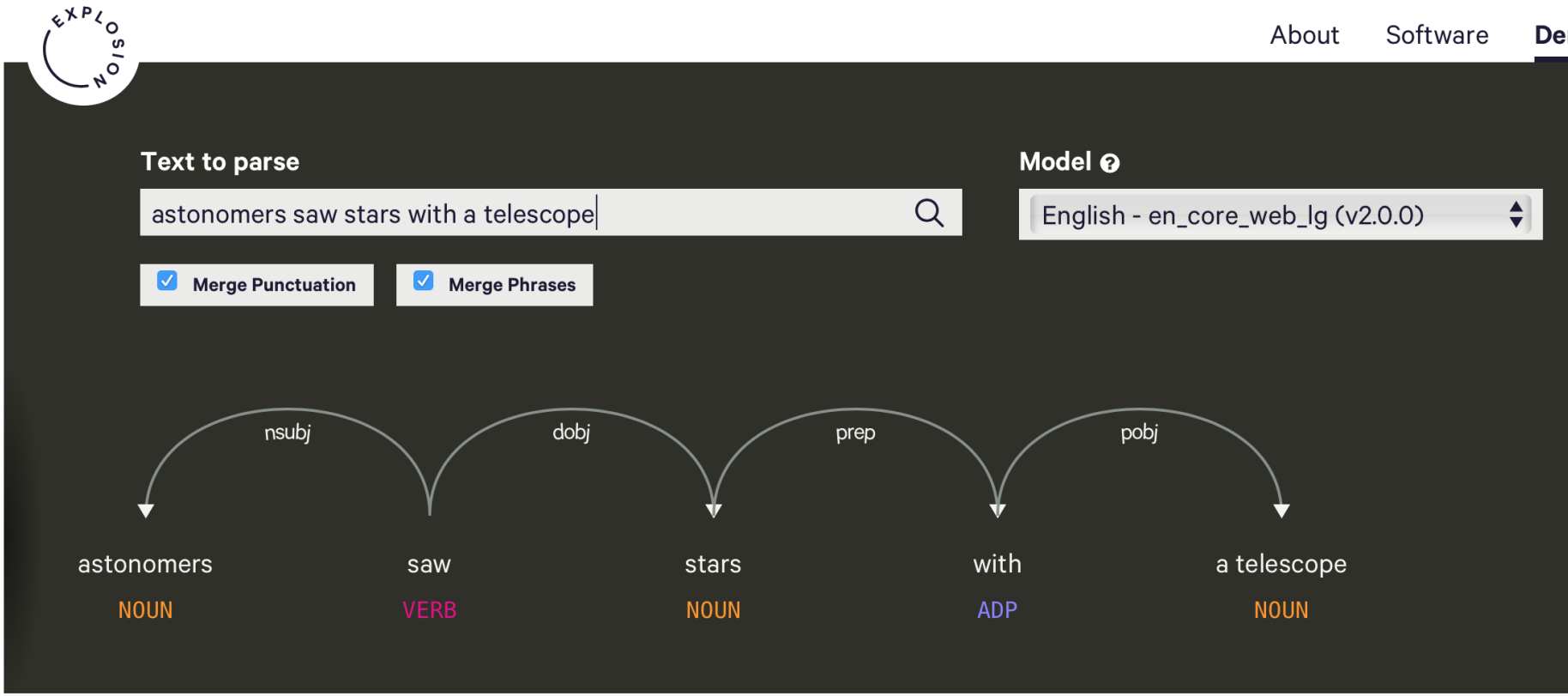
# Spacy

```
import spacy
from spacy import displacy

nlp = spacy.load('en')
doc = nlp(u'This is a sentence.')
displacy.serve(doc, style='dep')
```

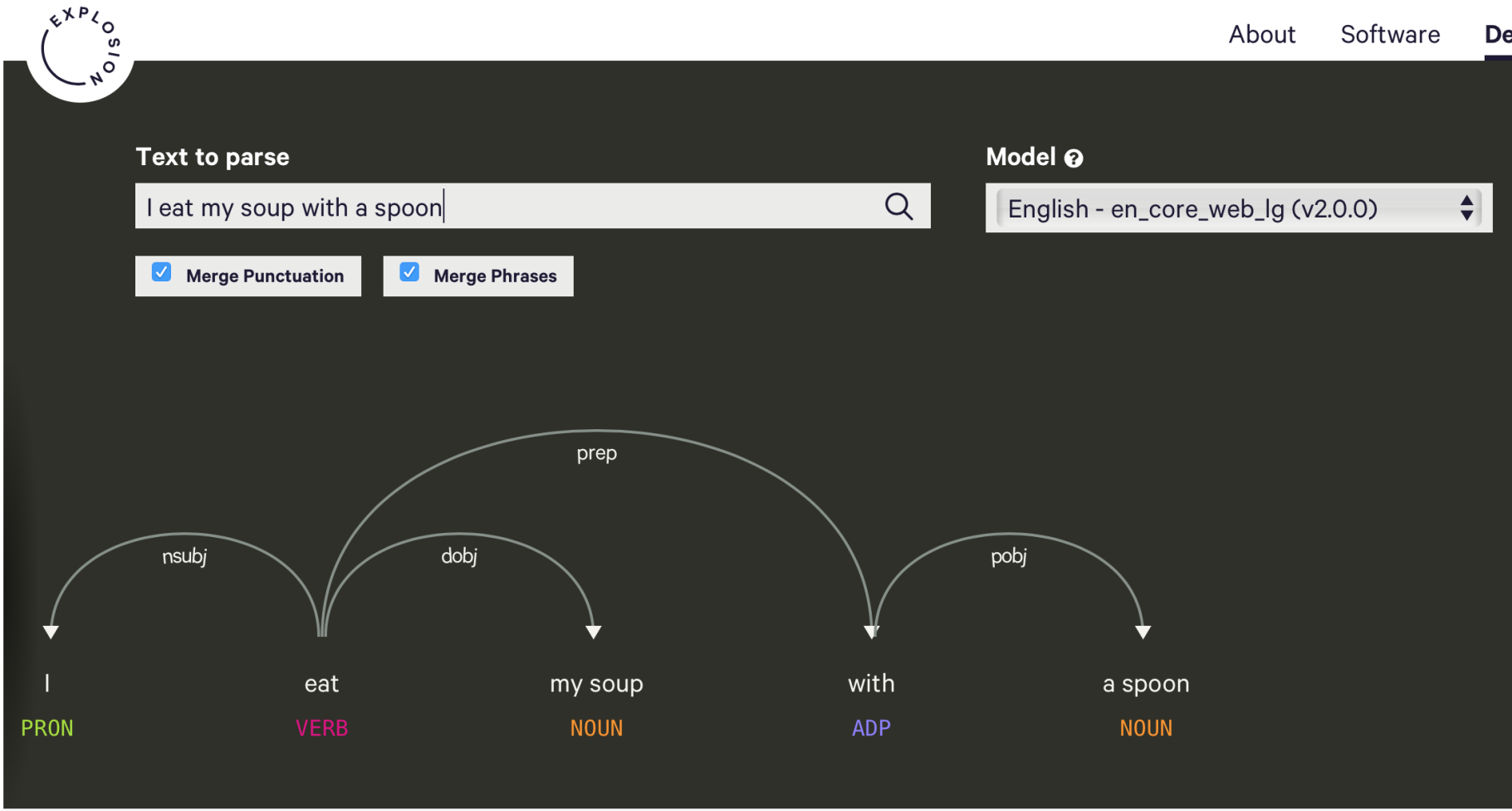
- on peut récupérer l'analyse avec un navigateur à l'adresse `http://localhost:5000`
- bien sûr, chaque élément visuel du graphe rendu est paramètrable.
- Il existe d'ailleurs une très belle démo de displacy : `https://explosion.ai/demos/displacy`

# Displacy



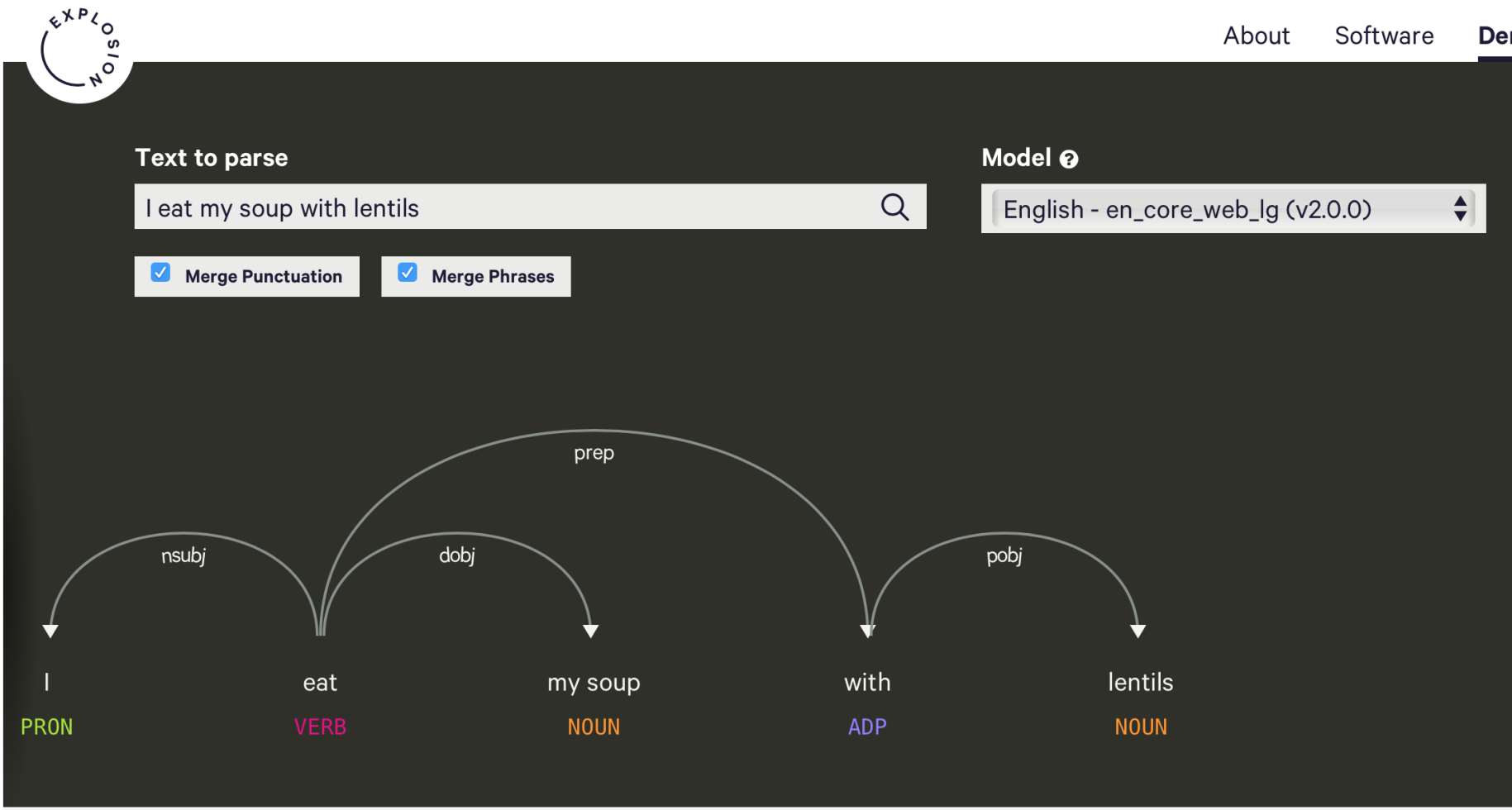

- Les étoiles ont des télescopes

# Displacy



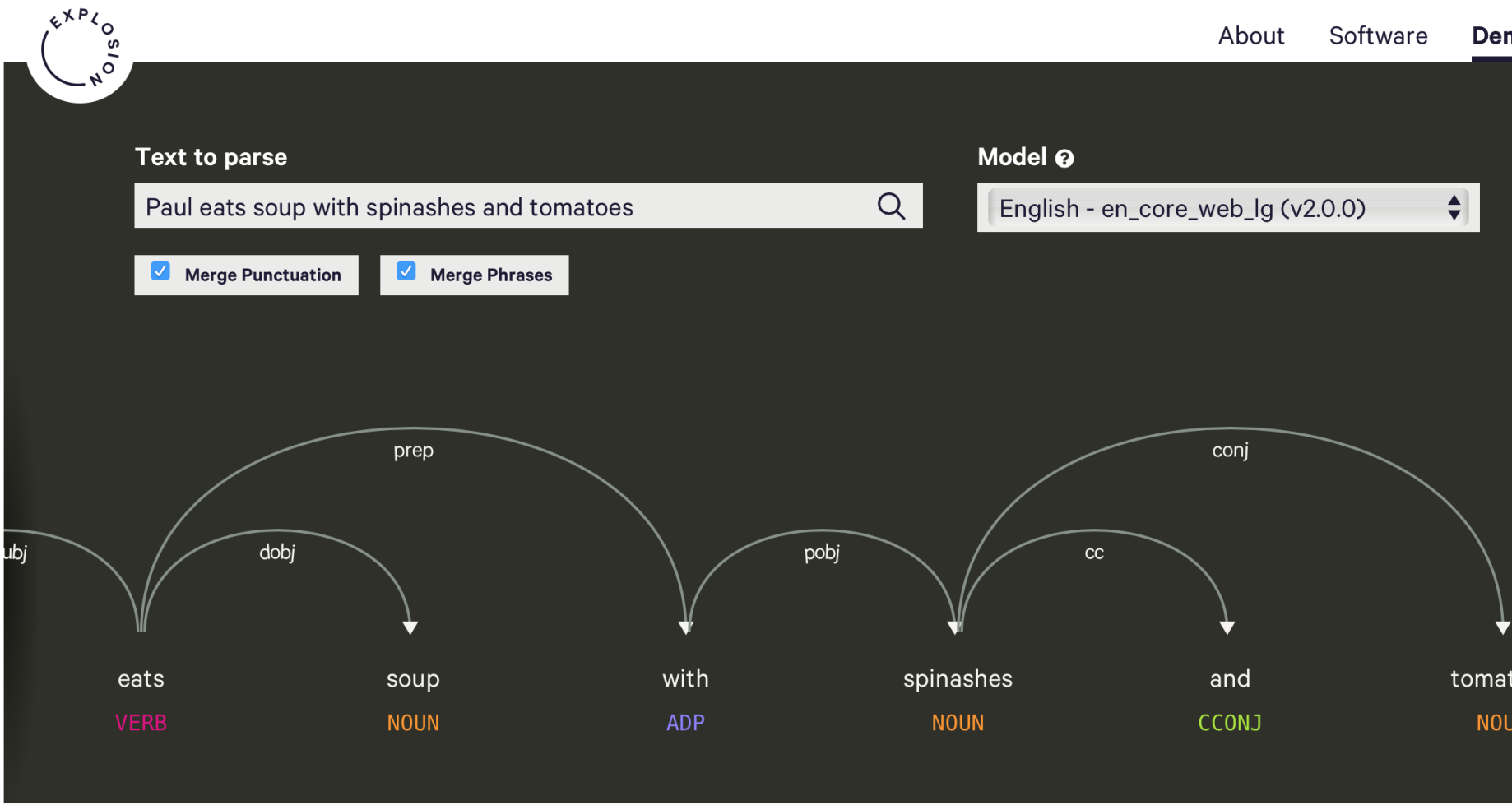

- C'est avec ma cuillère que je mange ma soupe (bien)

# Displacy



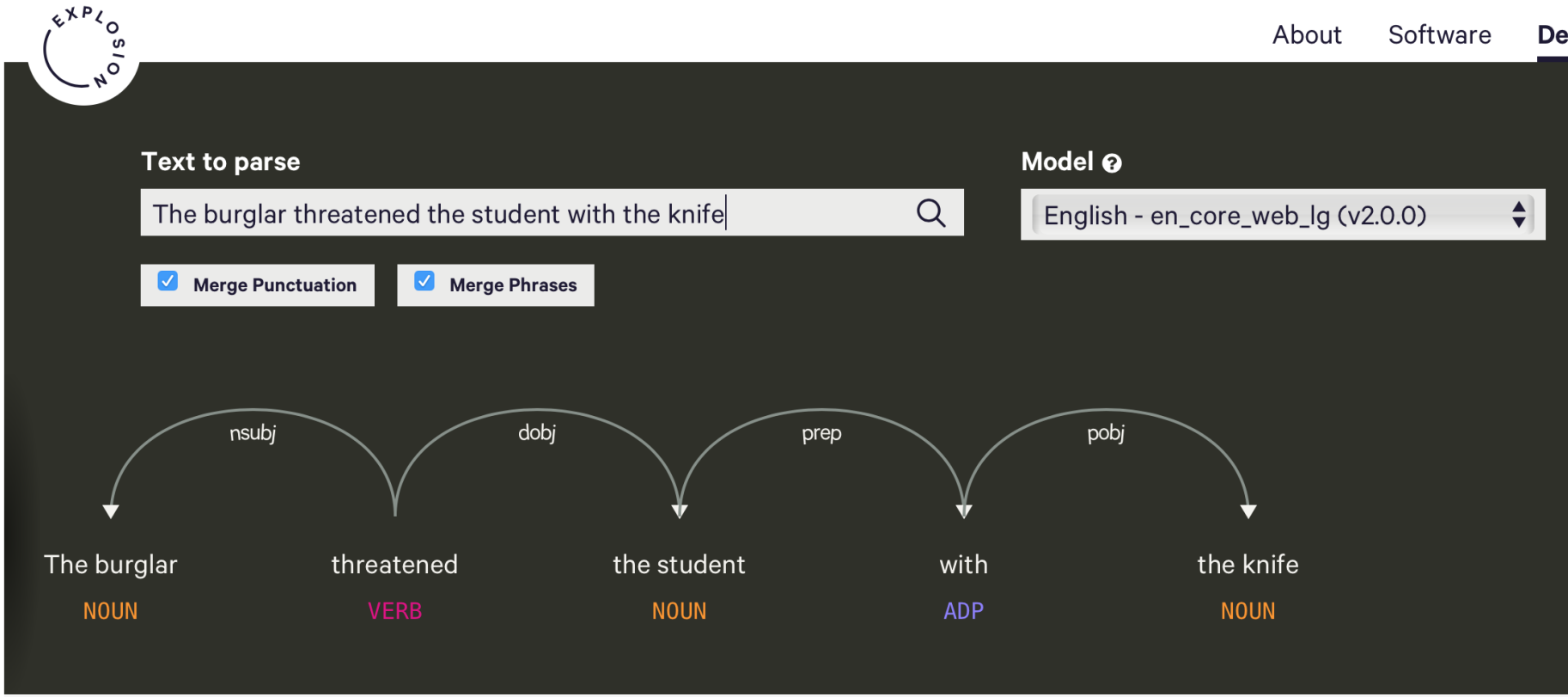

- C'est avec des lentilles que je mange ma soupe (moins bien . . . )

# Displacy



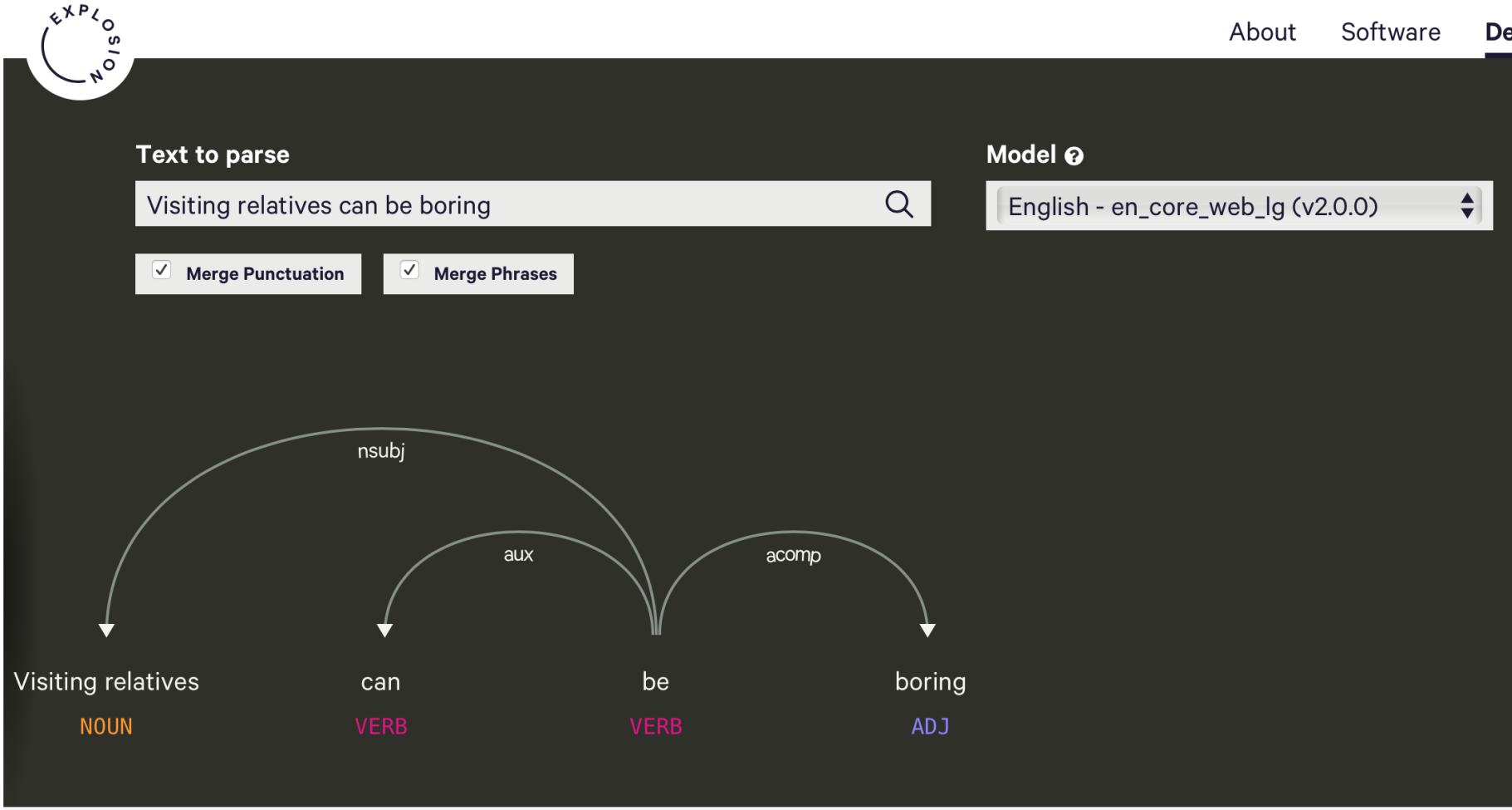

- Un seul lien fautif peut ruiner l'analyse (ici le lien prep)

# Displacy




- Le voleur prend des risques . . .

# Displacy




- Les liens sont bons, mais l'analyse fautive (Visiting relatives est un groupe verbal)

**Baker, J. K. (1979).**
Trainable grammars for speech recognition.
In *97th Meeting of the Acoustical Society of America*, pages 547–550.

**Black, E. (1991).**
A procedure for quantitatively comparing the syntactic coverage of english.

**Bod, R. (1992).**
Data oriented parsing (dop).
In *Proceedings of the International Conference on Computational Linguistics (COLING) 1992*, Nantes, France.

**Bod, R. (2000).**
Parsing with the shortest derivation.
In *Proceedings of the International Conference on Computational Linguistics (COLING) 2000*, Saarbrucken, Luxembourg, Nancy.

**Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015).**

rali

A large annotated corpus for learning natural language inference.
In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642.

**Charniak, E. (1993).**
*Statistical Language Learning*.
MIT Press.

**Charniak, E. (1996).**
Tree-bank grammars.
Technical report, Brown University.
CS-96-02.

**Charniak, E. (1997).**
Statistical parsing with a context-free grammar and word statitstics.

In *American Association for Artificial Intelligence (AAAI) Conference on Probabilistic Approaches to Natural Language*, pages 598–603.

**Charniak, E. (2001).**
Immediate-head parsing for language models.

rali

In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 116–124, Toulouse. Morgan Kaufmann Publishers.

**Chelba, C., Engle, D., Jelinek, F., Jimenez, V. M., Khudanpur, S., Mangu, L., Printz, H., Ristad, E., Rosenfeld, R., Stolcke, A., and Wu, D. (1997).**
Structure and performance of a dependency language model.
In *Proc. Eurospeech '97*, pages 2775–2778, Rhodes, Greece.

**Chelba, C. and Jelinek, F. (1998).**
Exploiting syntactic structure for language modeling.
In Boitet, C. and Whitelock, P., editors, *Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics*, pages 225–231, San Francisco, California. Morgan Kaufmann Publishers.

**Chu, Y. J. and Liu, T. H. (1965).**
On the shortest arborescence of a directed graph.
*Science Sinica*, 14.

rali

**Cocke (1965).**
peu probable que vous mettiez la main dessus(document à diffusion très restreinte).

**Collins, M. (1999).**
*Head-Driven Statistical Models for Natural Language Parsing*.
PhD thesis, University of Pennsylvania.

**Collins, M. (2003).**
Parameter estimation for statistical parsing models : Theory and practice of distribution-free methods.

**Collins, M. J. (1996).**
A new statistical parser based on bigram lexical dependencies.
In Joshi, A. and Palmer, M., editors, *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, pages 184–191, San Francisco. Morgan Kaufmann Publishers.

**Covington, M. A. (2001).**
A fundamental algorithm for dependency parsing.

rali

In *In Proceedings of the 39th Annual ACM Southeast Conference*, pages 95–102.

**de Marneffe, M.-C., MacCartney, B., and Manning, C. D. (2006).**
Generating typed dependency parses from phrase structure parses.
In *LREC*.

**Earley, J. (1968).**
*An efficient context-free parsing grammar*.
PhD thesis, Carnegie-Mellon University.

**Goodman, J. (1998).**
*Parsing inside-out*.
PhD thesis, Harvard University.

**Hindle, D. and Rooth, M. (1991).**
Structural ambiguity and lexical relations.
In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 229–236, Berkeley, California.

rali

**Hogenhout, W. R. and Matsumoto, Y. (1998).**
A fast method fro statistical grammar induction.
*Natural Language Engineering*, 4(3) :191–209.

**Jelinek, F. and Lafferty, J. D. (1991).**
Computation of the probability of initial substring generation by stochastic context-free grammars.
*Computational Linguistics*, 17 :315–324.

**Johansson, R. and Nugues, P. (2007).**
Incremental dependency parsing using online learning.
In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 1134–1138, Prague, Czech Republic.

**Johnson, M. (1998).**
PCFG models of linguistic tree representations.
*Computational Linguistics*, 24(4) :613–632.

**Jurafsky, D. and Martin, J. H. (2000).**
*Speech and Language Processing*.
Prentice Hall.

rali

**Kasami, J. (1965).**
An efficient recognition and syntax analysis algorithm for context-free languages.
Technical report, University of Hawaii.

**Klein, D. and Manning, C. D. (2003).**
Accurate unlexicalized parsing.
In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430, Stroudsburg, PA, USA. Association for Computational Linguistics.

**Lafferty, J., Sleator, D., and Temperley, D. (1992).**
Grammatical trigrams : A probabilistic model of link grammar.
In *American Association for Artificial Intelligence (AAAI) Conference on Probabilistic Approaches to Natural Language*.

**Lari, K. and Young, S. J. (1990).**
The estimation of stochastic context-free grammars using the inside-outside algorithm.
*Computer Speech and Language*, 4(1) :35–56.

rali

**Lin, D. (1995).**
A dependency-based method for evaluating broad-coverage parsers.
In *IJCAI*, pages 1420–1427.

**Magerman, D. (1995).**
Statistical decision-tree models for parsing.
In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 276–283, Cambridge, Massachusetts.

**Matsuzaki, T., Miyao, Y., and Tsujii, J. (2005).**
Probabilistic cfg with latent annotations.
In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 75–82, Stroudsburg, PA, USA. Association for Computational Linguistics.

**Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013).**
Efficient estimation of word representations in vector space.
*CoRR*, abs/1301.3781.

rali

**Nivre, J. and Fang, C.-T. (2017).**
Universal dependency evaluation.
In *NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, page 86–95, Gothenburg, Sweden.

**Pereira, F. and Schabes, Y. (1992).**
Inside-outside reestimation from partially bracketed corpora.
In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 128–135, Newark, Delaware.

**Petrov, S., Barrett, L., Thibaux, R., and Klein, D. (2006).**
Learning accurate, compact, and interpretable tree annotation.
In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 433–440, Stroudsburg, PA, USA. Association for Computational Linguistics.

**Petrov, S. and McDonald, R. (2012).**
Overview of the 2012 Shared Task on Parsing the Web.

rali

**Roark, B. (2001).**
Probabilistic top-down parsing and language modeling.
*Computational Linguistics*, 27(2) :249–276.

**Schabes, Y., Roth, M., and Osborne, R. (1993).**
Parsing the wall street journal with the inside-outside algorithm.
In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 341–347, Columbus, Ohio.

**Sleator, D. D. and Temperley, D. (1991).**
Parsing english with link grammar.
Technical report, Carnegie Mellon University, Pittsburgh.

**Stolcke, A. (1995).**
An efficient probabilistic context-free parsing algorithm that computes prefix probabilities.
*Computational Linguistics*, 21 :165–202.

**Younger, D. (1967).**
Recognition and parsing of context-free grammars in time $n^3$.
*Information and Control*, 10 :189–208.

rali

**Zhang, Y. and Nivre, J. (2011).**
Transition-based dependency parsing with rich non-local features.
In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics : Human Language Technologies : Short Papers - Volume 2*, HLT '11, pages 188–193.