Introduction aux grammaires probabilistes hors contexte (PCFG)

- Définition et propriétés de base d'une grammaire hors-contexte probabilisée (PCFG)
- Avantages sur une grammaire hors-contexte non probabilisée
- Apprentissage d'une PCFG
  - apprentissage des probabilités (inside-outside)
  - apprentissage des règles
- Évaluer un modèle PCFG (ParseEval)
- Le corpus PennTreeBank (PTB)
- Panorama de quelques modèles PCFG plus complexes
  - PCFG lexicalisées
  - Data oriented Parsing (DOP)
  - Link grammars

Chapitres 11 & 12 dans Manning and Schütze [1999]
Chapitre 12 dans Jurafsky and Martin [2000]

PCFG: définition formelle

Définition Une grammaire probabiliste hors-contexte est définie par un 5-uplet:
$< N, T, R, S, P >$ où:

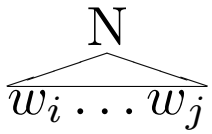
N est l'ensemble des symboles $non\text{-}terminaux$
T est l'ensemble des symboles $terminaux$
R est l'ensemble des règles r_i de la forme: $A \rightarrow \beta$
S est l'axiome de départ
P est l'ensemble des probabilités p_i associées aux règles r_i telles que:
$\sum_\beta p(A \rightarrow \beta) = 1, \quad \forall A \in N$

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)

Un peu de vocabulaire

- On dit que N **domine** la chaîne $w_i \dots w_j$ si $N \stackrel{\star}{\Rightarrow} w_i \dots w_j$, 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 l mots dans une chaîne ne dépend pas de la position de ces mots dans la chaîne (l mots contigus).

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.

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

Exemple de CFG probabilisée

Soit la grammaire G:

S $\longrightarrow$ NP VP [1.0]	P $\longrightarrow$ *with* [1.0]	NP $\longrightarrow$ *ears* [0.18]
PP $\longrightarrow$ P NP [1.0]	V $\longrightarrow$ *saw* [1.0]	NP $\longrightarrow$ *saw* [0.04]
VP $\longrightarrow$ V NP [0.7]	NP $\longrightarrow$ NP PP [0.4]	NP $\longrightarrow$ *stars* [0.18]
VP $\longrightarrow$ VP PP [0.3]	NP $\longrightarrow$ *astronomers* [0.1]	NP $\longrightarrow$ *telescopes* [0.1]

et la phrase à analyser: $S = w_1^5 =$ *astronomers saw stars with ears*.

Cette phrase contient deux analyses possibles avec G dont les interprétations sont les suivantes:

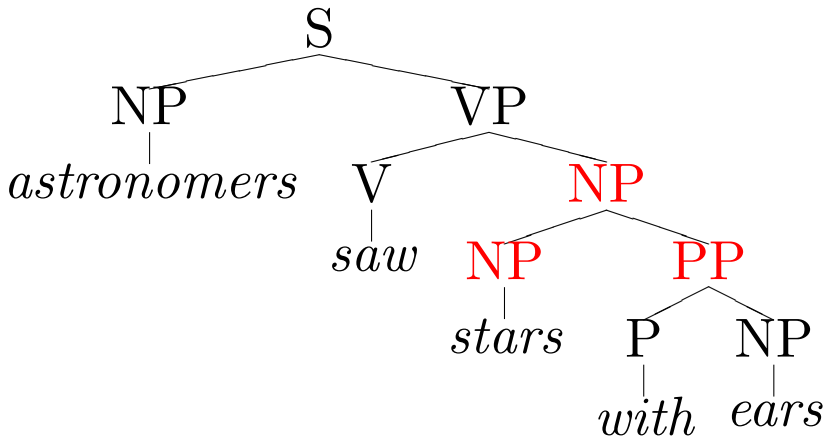
t_1: les astronomes ont vu des étoiles qui avaient des oreilles

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

Question: quelle interprétation préférez-vous ?

Les étoiles ont-elles des oreilles ?

$t_1 =$



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

Justification (rapide)

$$p(N \rightarrow XY \ldots Z) \stackrel{def}{=} p(X_{k,m}, Y_{m+1,n}, \ldots, Z_{q+1,l} | N_{k,l})$$
$$\forall k, l, m, n, \ldots, q \text{ tq. } k \leq m \leq n \ldots \leq q \leq l$$

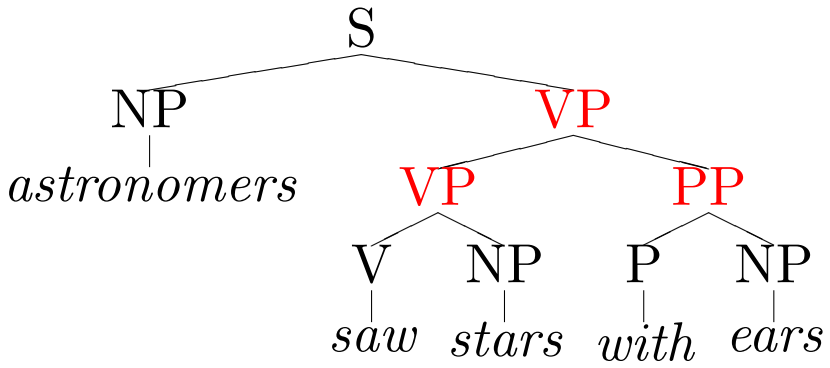
Idée: on conditionne pour faire apparaître chaque règle en utilisant la règle de la chaîne (*chain rule*), puis on simplifie chaque distribution en appliquant les simplifications décrites auparavant.

$$\begin{aligned}
p(t_1) &= p(S_{1,5}\, \mathrm{NP}_{1,1}\, \mathrm{VP}_{2,5}\, \mathrm{V}_{2,2}\, \mathrm{NP}_{3,5}\, \mathrm{NP}_{3,3}\, \mathrm{PP}_{4,5}\, \mathrm{P}_{4,4}\, \mathrm{NP}_{5,5}\; w_1^5 | \mathrm{S}_{1,5}) \\
&= p(\mathrm{S}_{1,5} \rightarrow \mathrm{NP}_{1,1}\mathrm{VP}_{2,5} | \mathrm{S}_{1,5}) \times \\
&\quad p(\mathrm{NP}_{11} \rightarrow \text{astronomers} | \mathrm{S}_{1,5},\ \mathrm{S}_{1,5} \rightarrow \mathrm{NP}_{1,1}\mathrm{VP}_{2,5}) \times \\
&\quad \ldots \\
&= p(\mathrm{S} \rightarrow \mathrm{NP}\,\mathrm{VP}) \times p(\mathrm{NP} \rightarrow \text{astronomers}) \times \ldots
\end{aligned}$$

En pratique: c'est totalement intuitif bien que pénible à écrire. . .

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

$t_2 =$



$$\begin{aligned} p(t_1) &= 1.0 \times 0.1 \times 0.3 \times 0.7 \times 1.0 \times 0.18 \times 1.0 \times 1.0 \times 0.18 \\ &= 0.0006804 \end{aligned}$$

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).

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.

Tolérance aux phrases agrammaticales

- De nombreuses applications doivent faire face à des énoncés agrammaticaux (dialogue, e-mail, etc). Un analyseur classique tente plutôt de modéliser ce qui est correct.

  On peut certes développer des grammaires non probabilistes pour ces "langages", mais encore faut-il être capable de décider ce qui est tolérable de ce qui ne l'est pas $\Rightarrow$ **décision catégorielle**

- Une grammaire probabiliste n'écarte rien $a\ priori$ et ne souffre donc pas de ce problème. Cette approche n'est cependant pas exempte de problèmes: une certaine masse de probabilité est distribuée à des analyses de phrases farfelues.

  Au mieux peut-on espérer que:
  $p(t|\text{colorless idea sleep furiously}) > p(t|\text{sleep idea furiously colorless})$.

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

De plus, il existe des algorithmes pour calculer la probabilité d'un préfixe d'une chaîne d'un langage Jelinek and Lafferty [1991], Stolcke [1995].

$\hookrightarrow$ il est donc possible d'en faire un modèle prédicitif:

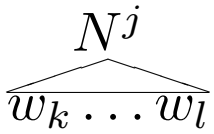
$$p(w_n|w_1^{n-1}) = \frac{p(w_1^n)}{p(w_1^{n-1})}$$

Lire aussi Chelba et al. [1997]

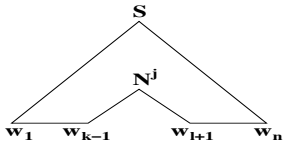
Calculer la probabilité d'une chaîne
Cas d'une grammaire CNF

Comme pour les HMMs, calculer la probabilité d'une observation (ici phrase) en sommant sur tous les chemins possibles (ici arbres possibles), est rapidement infaisable. On utilise à la place l'équivalent des probabilités $\beta_t(i), \alpha_t(i)$ et les récurrences associées pour calculer efficacement cette probabilité:

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)

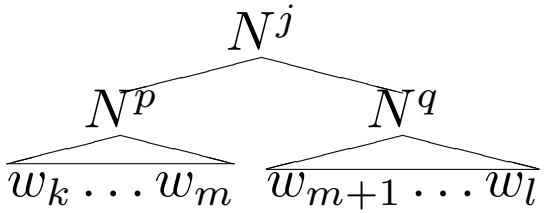
cas terminal: $\alpha_j(k,k) = p(w_k|N^j_{k,k}) = p(N^j \rightarrow w_k)$
récursion: pour le cas $l > k$, $(m \in [k,l[, p,q \in [1,N])$, alors:

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

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

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).

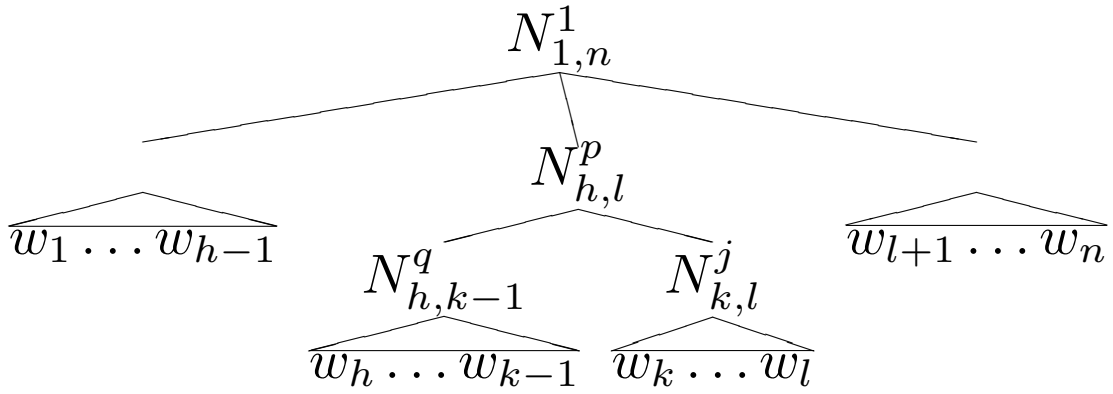
Note: Ca ressemble étrangement à l'algorithme CYK, d'ailleurs c'est le "même" algorithme que l'on peut utiliser.

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 $N^p \rightarrow N^q N^j$:

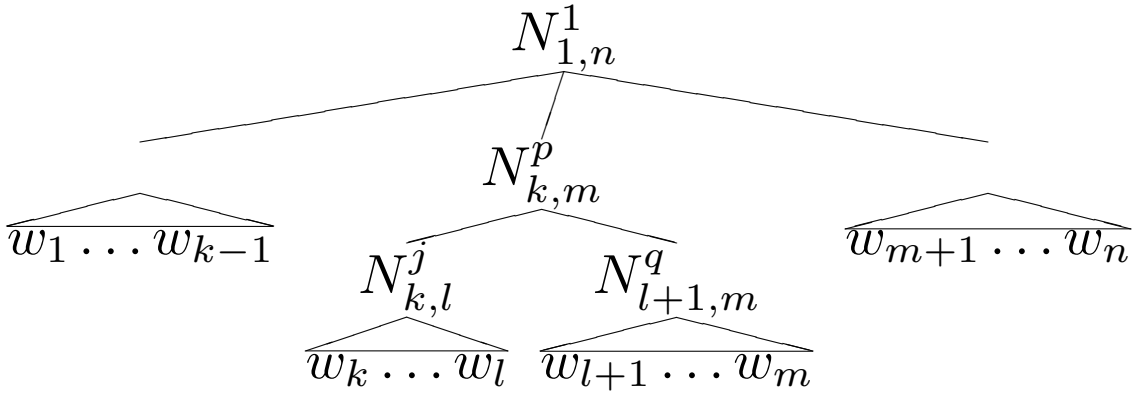


alors $\beta_j(k,l) = \sum_{h,p,q} p(w_1^{h-1}, w_h^{k-1}, w_{l+1}^n, N_{k,l}^j, N_{h,l}^p, N_{h,k-1}^q) =$

$$\sum_{h,p,q} \begin{array}{l} p(w_1^{h-1}, w_{l+1}^n, N_{h,l}^p) \times \\ p(N_{h,k-1}^q, N_{k,l}^j | w_1^{h-1}, w_{l+1}^n, N_{h,l}^p) \times \\ p(w_h^{k-1} | N_{h,k-1}^q, N_{k,l}^j, w_1^{h-1}, w_{l+1}^n, N_{h,l}^p) \end{array}$$

Calculer la probabilité d'une chaîne (via outside)

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



et $\beta_j(k,l) = \sum_{m,p,q} p(w_1^{k-1}, w_{l+1}^m, w_{m+1}^n, N_{k,l}^j, N_{k,m}^p, N_{l+1,m}^q) =$

$$\sum_{m,p,q} \quad p(w_1^{k-1}, w_{m+1}^n, N_{k,m}^p) \times$$
$$p(N_{k,l}^j, N_{l+1,m}^q | w_1^{k-1}, w_{m+1}^n, N_{k,m}^p) \times$$
$$p(w_{l+1}^m | N_{k,l}^j, N_{l+1,m}^q, w_1^{k-1}, w_{m+1}^n, N_{k,m}^p)$$

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:

récursion:

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

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

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 $pré\text{-}terminaux$ (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 ($\mathcal{O}(n^3|N|^3)$)

Une fois n'est pas coutume, on peut modifier légèrement 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 la prise d'un maximum, et on mémorise de plus 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 $back\text{-}pointeur$ 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 $[6,0]$ 0.04 NP $[10,0]$			
astro	0.1 NP [8,0]				
	astro	saw	stars	with	ears

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

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

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 $[6,0]$ 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 $l = 3$)

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] 0.04 NP [10, 0]			
astro	0.1 NP [8,0]				
	astro	saw	stars	with	ears

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

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

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

ears		0.009072 VP $[3,2]$ 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 $[6,0]$ 0.04 NP $[10,0]$			
astro	0.1 NP [8,0]				
	astro	saw	stars	with	ears

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

VP $\Rightarrow$ saw stars with ears

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

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

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 $[6,0]$ 0.04 NP $[10,0]$			
astro	0.1 NP [8,0]				
	astro	saw	stars	with	ears

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

- S $\Rightarrow$ astro saw stars with ears $\quad 0.0009072 = 1.0 \times 0.1 \times 0.009072$

$\hookrightarrow$ C'est l'arbre de la page

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

Il faut 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

Que se passe-t-il si la grammaire n'est pas CNF ?

On peut écrire cela de manière récursive. Pour une règle donnée $rule$ nous supposons la notation suivante: $rhs(rule)$ 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.

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$[memo, lhs(rule), stop]$) **then**
        split(rule,n+1,memo,j+1,stop,$cuts \cup \{j\}$,score + sc)

Appel: split(VP $\rightarrow$ ADVP VBD NP , NP; 0; 1; 1; 7; {}; log(prob(rule)))
(VP $\rightarrow$ ADVP VBD NP , NP = 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 encore un problème avec cet algorithme qui fait que l'arbre trouvé ne sera pas nécessairement le plus probable (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) [4] −46.7 (2, 3) [5] −33.5 (1, 3) [6] −39.4 (1, 3)	[7] −17.7 (2, 3) [8] −27.1 (2, 3)	[3] −13.2 (3) [9] −9.5 (3) [10] −7.5 (3) [11] −16.7 (3) [12] −15.1 (3) [13] −11.2 (3)
are	[5] −26.2 (1, 2) [6] −32.0 (1, 2)	[14] −1.6 (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]

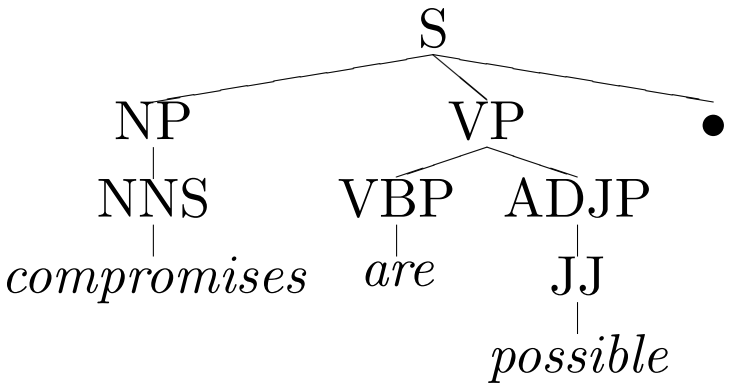
item = règle, logprob, découpage de chaque RHS-symbole. Ex: $(1, 3) \equiv$ 1er symb domine de 1 à 2, le 2e domine à partir de 3

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]		

Apprendre une PCFG

Question: peut-on apprendre les paramètres d'une PCFG avec simplement un texte suffisamment grand d'une langue donnée ?

(note: paramètres = règles + probabilités)

$\Rightarrow$ c'est un problème difficile.

Soyons plus modestes:

Peut-on apprendre automatiquement les probabilités à associer aux règles d'une grammaire donnée ?

$\Rightarrow$ algorithme *inside-outside* Baker [1979]

Apprendre une PCFG: par *inside-outside*

Donnée incomplète: les phrases $W_1, \ldots W_N$
Donnée complète: l'arbre syntaxique associé à chaque phrase: $t_1, \ldots, t_N$

$\Rightarrow$ une instance de EM.

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

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:

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

où γ est un coefficient de normalisation ($\gamma \equiv \sum_{t \in \tau(W)} p_{\phi'}(t|W) c(LHS(r); W, t)$).

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 simplifier la complexité des formules de réestimation par programmation dynamique.

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

Donc: $p(N_{p,q}^j | w_1^n, G) = \frac{\beta_j(p,q) \times \alpha_j(p,q)}{\pi}$ (on vient de voir comment calculer π)

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

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

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

Et donc:

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

Apprendre une PCFG: par *inside-outside*

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

De la même manière:

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

Et donc:

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

Apprendre une PCFG: par *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 $inside\text{-}outside$ 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]

Utilisation d'un 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 β et α 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.

Utilisation d'un 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

Utilisation d'un corpus parenthésé

Pereira and Schabes [1992]

Expérience 2: Sur les POS uniquement (les symboles pré-terminaux). 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).

$\hookrightarrow$ 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.

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)

Augmentation progressive de la grammaire

Hogenhout and Matsumoto [1998]

↪ Permet de réduire les temps d'entraînement, même si le nombre d'itérations est plus grand. En retirant les règles les moins probables à chaque étape, on contrôle également la taille de la grammaire.

Expérience: grammaire CNF sur le PTB sur 31 POS (dans Schabes et al. [1993] 47).

Train: 1000 phrases de 15 (resp. 20) mots ou moins dans exp .1 (resp. 2)
Test: 100 phrases

↪ Au départ de l'apprentissage, une grammaire CNF complètement spécifiée avec 7 non-terminaux. L'algorithme précédent s'applique jusqu'à obtenir 15 (resp. 18) non terminaux sur des phrases d'au plus 15 (resp. 20) mots

Augmentation progressive de la grammaire

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).

Intermède sur l'évaluation des grammaires

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 indépendemment de l'application visée, ce qui permet entre-autre de comparer des grammaires écrites dans des buts différents.

$\Rightarrow$ on se sert d'une référence, cad, d'un corpus arboré manuellement qui représente le but à atteindre (par les concepteurs de grammaire).

$\hookrightarrow$ 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*).

Intermède sur l'évaluation des grammaires

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*.

Intermède sur l'évaluation des grammaires

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%. $\Rightarrow$ toujours considérer ces trois mesures ensembles.

Intermède sur l'évaluation des grammaires

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.

Penn Tree Bank (PTB)

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

Un des rares corpora de phrases arborées à la main. Deux versions existantes pour l'anglais:

- La première, distribuée fin 92 contenait 4.5 millions de mots taggés. Deux-tiers de ce corpus a été arboré avec des conventions simples. Essentiellement des textes du *Dow-Jones News Service* (1.6 millions de mots parsés) ainsi que les phrases du corpus équilibré: le *Brown* corpus (1 million de mots).
- Une seconde version du Penn Treebank est distribuée depuis 1995 et contient outre une version améliorée du corpus Penn TreeBank I, 1 million de mots du Wall Street Journal (année 1989) arborés, ainsi qu'une sous-partie du corpus ATIS (5000 mots). Une nouvelle norme plus complète et systématique d'annotation a été suivie dans cette version.

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

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)

Étiquetage de groupes non présents (*)

```
( (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)

Le PTB est sous-parenthésé

```
( (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)
  $\Rightarrow$ Tâche difficile et coûteuse.

- 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: $wants(Chris, throw(Chris, ball))$

Apprendre une Penn Treebank PCFG

- Immédiat par fréquences relatives: on compte le nombre d'apparitions des différentes règles dans le corpus d'entraînement et on normalise !
- Charniak [1996] reporte qu'il 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). Entraînement sur les POS seulement (aucune information sur les mots !).
- Les performances de cette grammaire (testée sur 30 000 autres phrases) sont étonnement "bonnes" :

$\|sent.\|$	$avr.\|sent\|$	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

Apprendre une Penn Treebank PCFG

Plusieurs raisons expliquent les erreurs faites par ce genre de grammaire. En particulier, toutes les règles ne sont pas vues à l'entraînement. Pour cela, on peut non plus stocker toutes les règles et leur probabilités, mais modéliser l'expansion d'une règle à l'aide de paramètres de la forme:

$$p(r|l) = \prod_{t_i \in r} p(t_i|l, t_{i-1})$$

où r est la partie droite de la règle, l sa partie gauche (un non-terminal) et t_i l'ensemble des étiquettes de la partie droite.

$\hookrightarrow$ Vous reconnaissez les HMMs (ou les bigrammes)? Ici on a un HMM par non-terminal l. Les paramètres peuvent être appris par fréquence relative à partir du PTB (+ lissage).

Limites des PCFGs

Invariance positionnelle

- Les propriétés intrinsèques des PCFGs ne sont intuitivement pas satisfaisantes. En particulier, l'indépendance positionnelle est loin d'être vérifiée en pratique.
- Par exemple, il est très fréquent que le sujet d'une phrase en anglais soit un pronom. Francis et al. (1999) (`http://ucsu.Colorado.EDU/~francish/substart.html`) mesurent que 91% des sujets des phrases déclaratives du Switchboard sont des pronoms, alors que seulement 9% sont lexicalisés (groupe nominal). En revanche seulement 34% des objets directs dans ce même corpus sont des pronoms (66% étant lexicalisés).
- **Donc:** l'application de la règle NP $\longrightarrow$ Pronom (ou NP $\longrightarrow$ ART ADJ NC) devrait donc être conditionnelle à la nature du constituant que l'on dérive, ce qui n'est pas le cas dans une PCFG "classique".

Limites des PCFGs

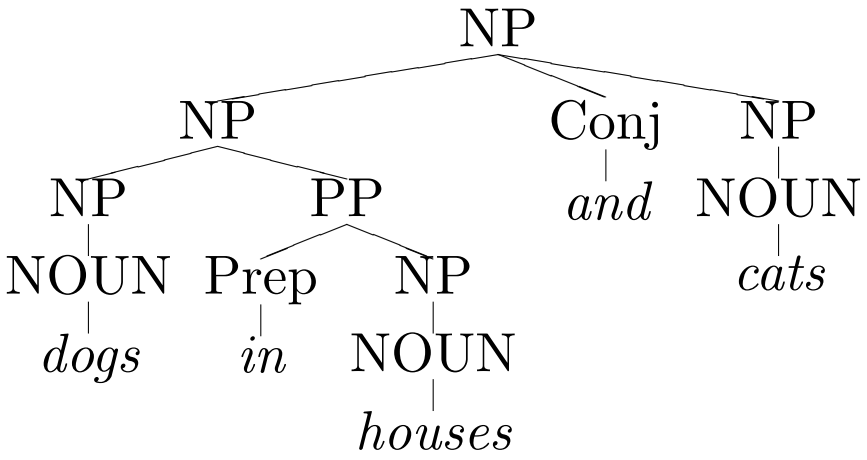
Ignorance aux mots eux-mêmes

- La seule manière de représenter l'information véhiculée par les mots dans une PCFG est via les symboles pré-terminaux (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 va se faire sur les probabilités associées à NP $\longrightarrow$ NP PP et VP $\longrightarrow$ NP PP. Hindle et al. reportent que sur un corpus d'agence Press newswire, 67% des attachements prépositionnels sont sur le groupe nominal plutôt que sur le verbe.
- Si le corpus d'entraînement maintient cette tendance, alors l'attachement précédant sera mal réalisé par la grammaire résultante.

$\hookrightarrow$ introduire – par exemple – la nature du verbe dans les statistiques.

Limites des PCFGs

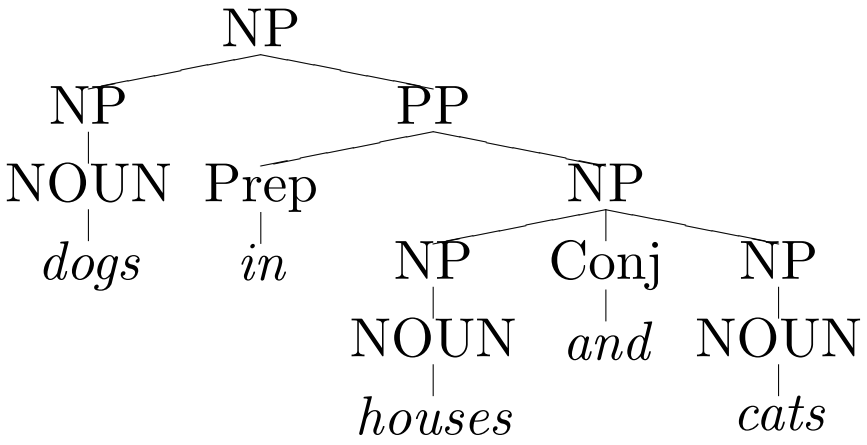
Un exemple pris de Collins [1999] (1/2)



NP → NP Conj NP
NP → NP PP
Conj → and
Prep → in
PP → Prep NP
NP → NOUN [×3]
NOUN → dogs
NOUN → houses
NOUN → cats

Limites des PCFGs

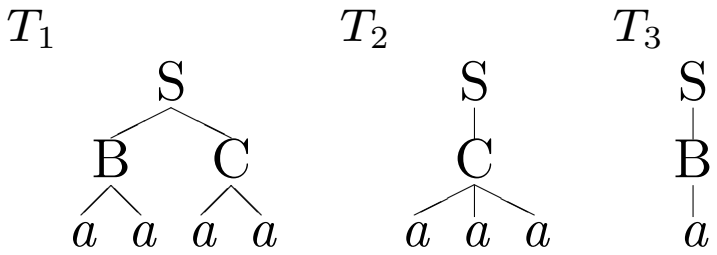
Un exemple pris de Collins [1999] (2/2)



NP → NP Conj NP
NP → NP PP
Conj → and
Prep → in
PP → Prep NP
NP → NOUN [×3]
NOUN → dogs
NOUN → houses
NOUN → cats

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

- Supposons un corpus d'entraînement généré à partir des 3 arbres suivants selon une distribution (p_1, p_2, p_3):



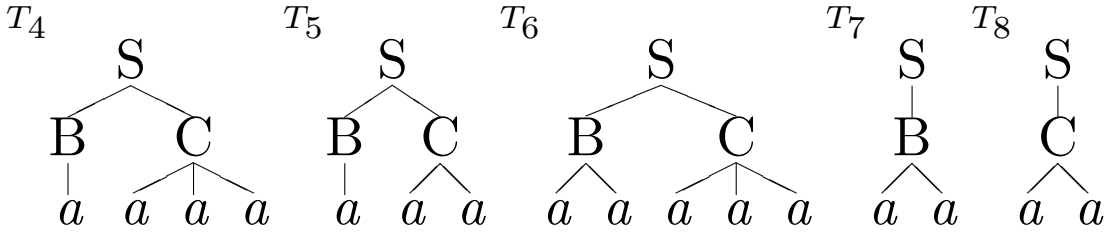
- Et soit le corpus de test $T = \{aaaa, aaa, a\}$.
- Si le corpus est infini, alors la fréquence relative des 3 arbres dans le corpus converge vers $\{p_1, p_2, p_3\}$, et donc:

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

Limites des PCFGs

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

- La grammaire génère également les arbres:



- et voici les probabilités (asymptotiques) associées à chaque arbre, si $p_1 = 0.2, p_2 = 0.1, p_3 = 0.7$:
$p(T_1^8) = \{0.0296, 0.0333, 0.544, 0.0519, 0.104, 0.0148, 0.156, 0.0667\}$

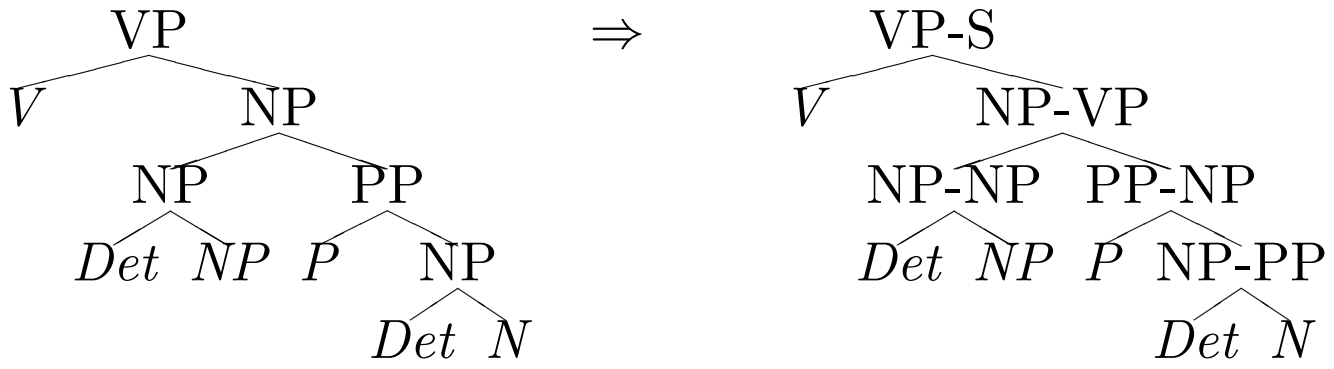
- $aaaa$ est générable par T_1 et T_4 (or $p(T_4) > p(T_1)$);
- aaa est générable par T_2 et T_5 (or $p(T_5) > p(T_2)$).

$\hookrightarrow$ Les chaînes $aaaa$ et aaa sont mieux analysées par des arbres que ne font pas partie du corpus d'entraînement

Réduire les hypothèses d'indépendance des PCFGs

Par exemple, 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.

En particulier, l'auteur montre qu'annoter le parent d'un nœud est une manière simple mais efficace d'augmenter les performances de la grammaire résultante.



Intermède sur la sous-catégorisation

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

- *The women kept the dogs on the beach*

Intermède sur la 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*

Intermède sur la 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*.

↪ lexicalisons les PCFGs

Place aux grammaires probabilistes lexicalisées

Une approche naïve consiste simplement à augmenter les symboles non-terminaux en introduisant de l'information lexicale.

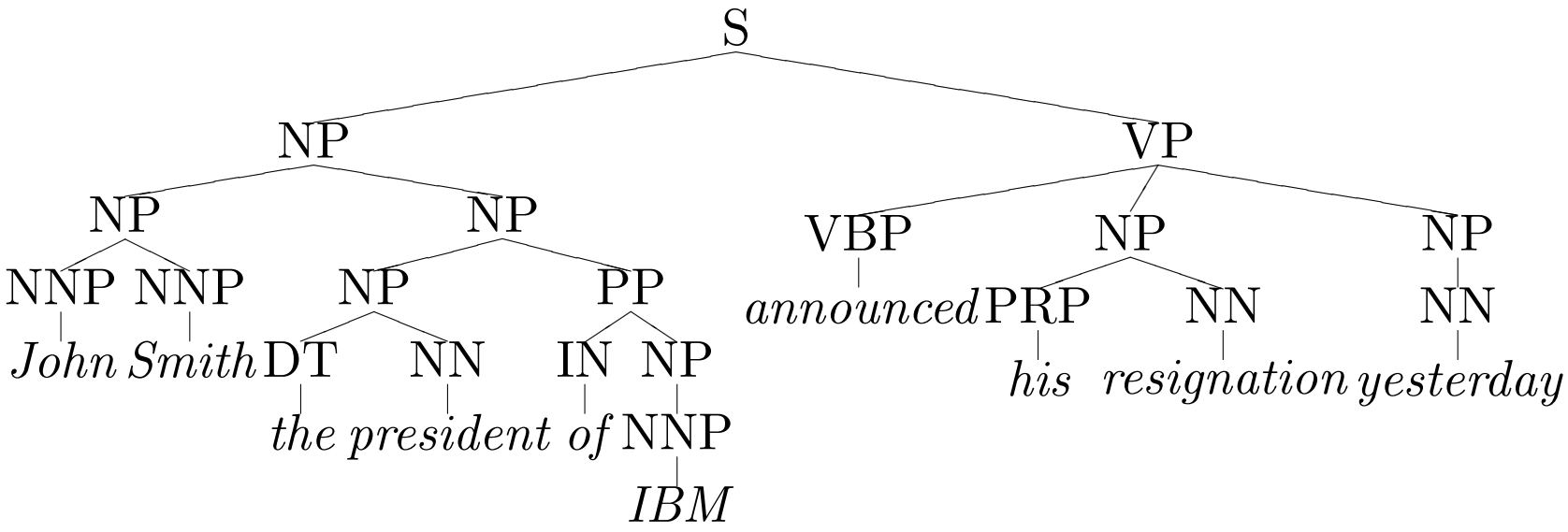
VP-partir $\longrightarrow$ Prep-de Noun-Paris $[3 \times 10^{-7}]$
VP-arriver $\longrightarrow$ Prep-à Noun-Marseille $[4 \times 10^{-6}]$
VP-arriver $\longrightarrow$ Prep-à Noun-Montréal $[6 \times 10^{-9}]$
...

Problème: Trop de non-terminaux $\Rightarrow$ amène des problèmes de sous-représentation des données $\Rightarrow$ estimation bruitée.

$\hookrightarrow$ Les différentes approches diffèrent essentiellement sur la manière de modéliser cette information lexicale et la nature des lissages introduits pour faire face aux problèmes de sous-représentation des données.

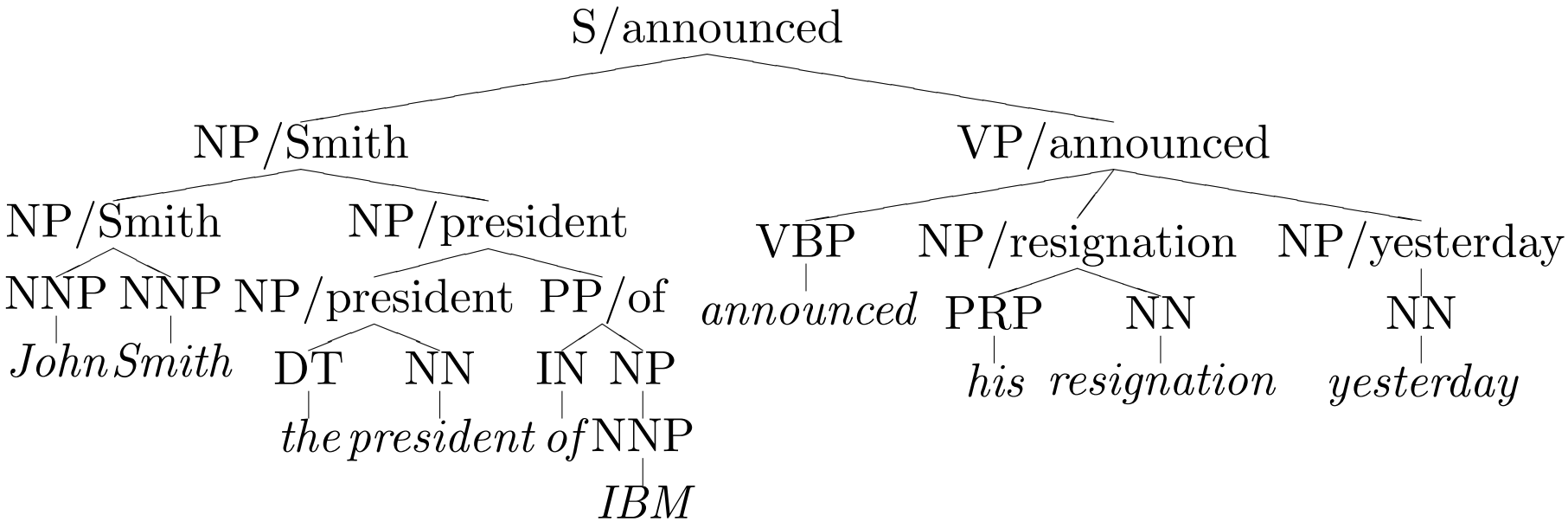
Grammaires lexicalisées

Représentation "classique"



Grammaires lexicalisées

Représentation lexicalisée



Identification de la tête d'un constituant

Processus déterministe (mais erroné), 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
...     ...     ...
```

Modèle de dépendance Collins [1996]

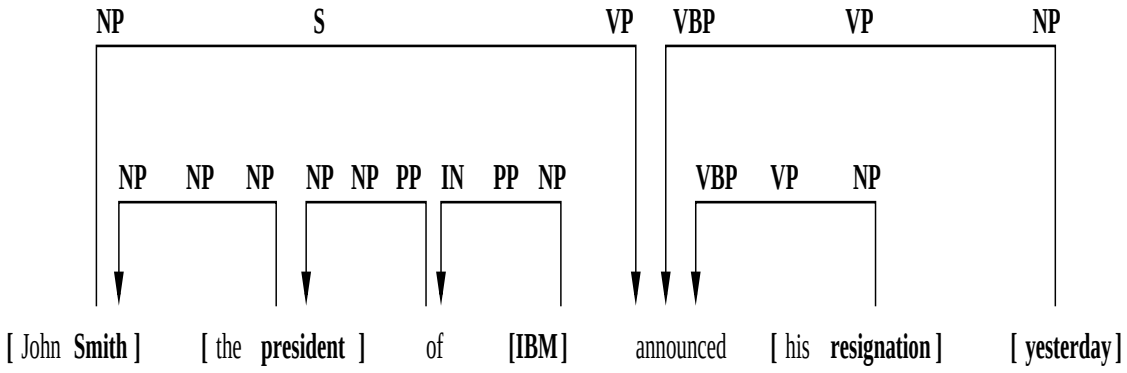
- Une phrase est représentée par une série de dépendances liant ses mots.
- Introduction d'un niveau de représentation intermédiaire simplifié d'une phrase, basé sur le découpage automatique de la phrase en groupes nominaux de base ($baseNP$); cad, des groupes nominaux non réentrants (pas de NP du genre: $(NP\ le\ bruit\ (NP\ de\ l'eau))$) (on retrouve ici les chunks)
- **Input:** Une phrase étiquetée ($taggée$).
  Output: Un ensemble de dépendances D codant une structure.
- Parser de référence, des versions disponibles:
  - `www.cis.upenn.edu/~dbikel/#stat-parser` (page de Daniel Bikel)
  - `people.csail.mit.edu/mcollins/code.html` (page de Michael Collins)

Modèle de dépendance Collins [1996]

- **Input:** une phrase étiquetée ($taggée$).

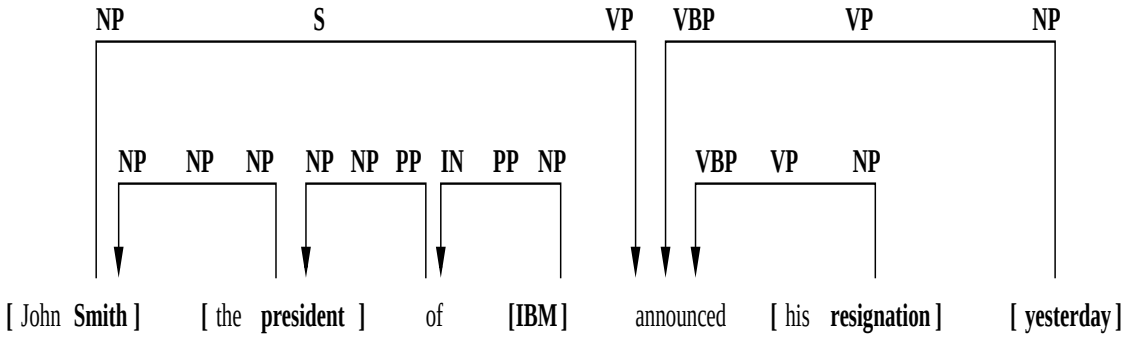
  (John,NNP), (Smith,NNP), (the,DT), (president,NN), (of,IN), (IBM,NNP), (announced,VBP), (his,PRP), (resignation,NN), (yesterday,NN)

- **Output:** un ensemble de dépendances D



Modèle de dépendance Collins [1996]

Représentation par dépendance



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]

Et voici la 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 est alors:

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

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 faite par:

$$\hat{T} = \underset{T}{\operatorname{argmax}}\, p(T|S)$$

avec $S = \langle (w_1, t_1), \ldots, (w_n, t_n) \rangle$ et $T = (B, D)$

où B est un ensemble de baseNPs, et D un ensemble de dépendances (déterminées à partir de la forme réduite de la phrase).

Le modèle devient alors:

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

$\hookrightarrow$ 2 distributions à modéliser.

Modèle de dépendance Collins [1996]

modèle de dépendance: $p(D|S,B)$

En faisant l'hypothèse d'indépendance des dépendances (!), on a (pour une phrase réduite de m mots):

$$p(D|B,S) = \prod_{j=1}^{m} p(D_j|S,B)$$

Soit les comptes suivants (où $\delta(x)$ vaut 1 si x est vrai, 0 sinon):

$$C(\langle w,t\rangle, \langle w',t'\rangle) = \sum_{\overline{S}\in\mathcal{T}, i,j\in[1,m], i\neq j} \delta(\overline{S}[i] = \langle w,t\rangle, \overline{S}[j] = \langle w',t'\rangle)$$

et

$$C(R, \langle w,t\rangle, \langle w',t'\rangle) = \sum_{\overline{S}, i\neq j} \delta \begin{pmatrix} \overline{S}[i] = \langle w,t\rangle \\ \overline{S}[j] = \langle w',t'\rangle \\ AF(i) = (j,R) \end{pmatrix}$$

Modèle de dépendance Collins [1996]

modèle de dépendance: $p(D|S,B)$

Alors la probabilité estimée que $\langle w,t\rangle$ modifie $\langle w',t'\rangle$ par une relation étiquetée R, sachant que $\langle w,t\rangle$ et $\langle w',t'\rangle$ sont dans la même phrase (réduite) est:

$$\hat{F}(R|\langle w,t\rangle,\langle w',t'\rangle) = \frac{C(R,\langle w,t\rangle,\langle w',t'\rangle)}{C(\langle w,t\rangle,\langle w',t'\rangle)}$$

et finalement:

$$p(AF(j)=(h_j,R_j)|S,B) \approx \frac{\hat{F}(R_j|\langle w_j,t_j\rangle,\langle w_{h_j},t_{h_j}\rangle)}{\sum_{k\neq j,r\in R}\hat{F}(r|\langle w_j,t_j\rangle,\langle w_k,t_k\rangle)}$$

Note: le dénominateur est constant pour l'opération de maximisation.

Modèle de dépendance Collins [1996]

modèle de dépendance: $p(D|S, B)$

- Le modèle de dépendance est ensuite compliqué de manière à prendre en compte la distance (comptée en mots) qui sépare deux mots en relation.

- Ceci afin de mieux prendre en compte les dépendances locales comme dans l'exemple suivant où les relations entre **sales** et **of** sont considérées:

  Shaw, based in Dalton, Ga., has annual **sales of** about $ 1.18 billion, and has economies **of** scale and lower raw-material costs that are expected to boost the profitability **of** Armstrong's brands, sold under the Armstrong ans Evans-Black names.

Modèle de dépendance Collins [1996]

modèle de segmentation en baseNPs: $p(B|S)$

Permet de limiter les problèmes de sous-représentation des données, et permet de se concentrer sur des dépendances potentiellement payantes.

Une version possible d'un chunker: ajouter un tag entre chaque mot parmi un ensemble restreint de tags: S(tart), C(ontinue), E(nd), B(etween) et N(ull). L'auteur parle de gap (G_i).

`John C Smith B the C president E of S IBM E has N announced S his C resignation B yesterday`

est équivalent au chunking: `[John Smith] [the president] of [IBM] has announced [his resignation] [yesterday]`

Modèle de dépendance Collins [1996]

modèle de segmentation en baseNPs: $p(B|S)$

$$p(B|S) = \prod_{i=2}^{n-1} \hat{p}(G_i|w_{i-1}, t_{i-1}, w_i, t_i, c_i)$$

où c_i est une fonction indicatrice (binaire) qui indique s'il y a ou pas une virgule entre les mots i et $i-1$.

+ lissage (en retirant de l'information contextuelle) pour l'estimation de ces distributions.

Modèle de dépendance Collins [1996]

performances

Pour la meilleure des variantes testées (entraînement sur $40\,000$ phrases):

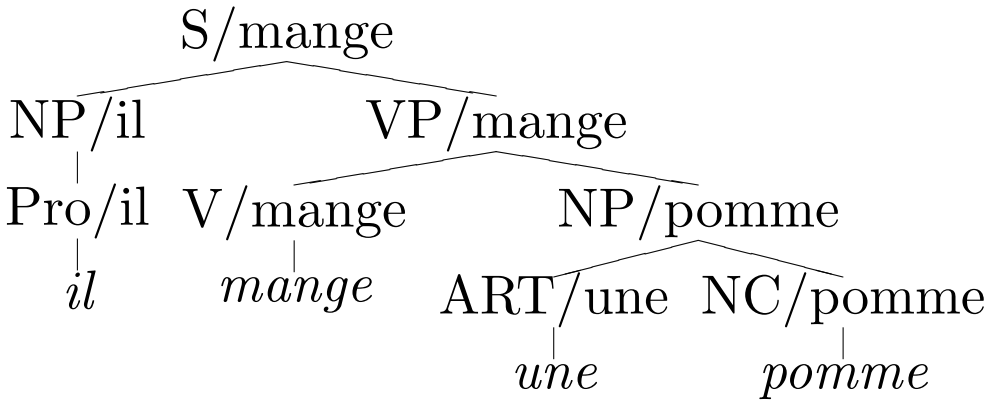
test	LR %	LP %	CBs	0 CBs	≤ 2 CBs
≤ 40 mots (2245 ph.)	85.8	86.3	1.14	59.9%	83.6%
≤ 100 mots (2416 ph.)	85.3	85.7	1.32	57.2%	80.8%
Charniak [1996]	80.4	78.8		87.7%	

Détails techniques: entraînement en 15 minutes; 8 minutes de chargement des tables de comptes; analyse de 200 phrases à la minute.

PCFG lexicalisée: Charniak [1997]

Un modèle plus simple à décrire. . .

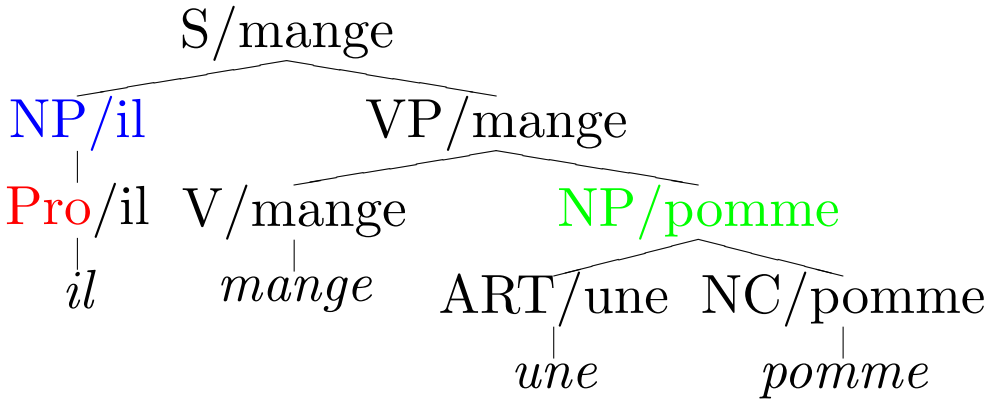
Idée: on conditionne la probabilité d'une règle sur la tête lexicale du constituant impliqué par la dérivation.



- 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.

Modèle Collins [1997]

Collins propose également une version générative de son modèle 96 qui intègre la notion d'expansion markovienne.

$$\hat{T} = \underset{T}{\operatorname{argmax}}\, p(T|S) = \underset{T}{\operatorname{argmax}}\, \frac{p(T,S)}{p(S)} = \underset{T}{\operatorname{argmax}}\, p(T,S)$$

Dans le modèle I (d'une série de 3):

Soit la séquence de dérivations pour obtenir une structure: $l_i \Rightarrow r_i$ pour $i \in [1,n]$ ($l_1 = S$), alors:

$$p(T,S) = \prod_{i \in [1,n]} p(r_i|l_i)$$

Modèle Collins [1997]

La probabilité d'une dérivation $p(r_i|l_i)$ est donnée par le modèle génératif suivant:

1. Choisir le constituant de tête des constituants fils d'un parent P ayant pour tête lexicale h selon une distribution $P_{\mathcal{H}}(H|P,h)$

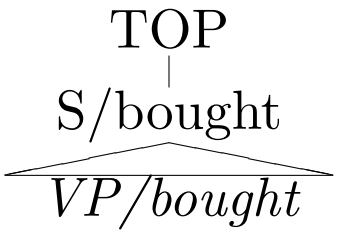
2. Générer les constituants à droite de H selon une distribution $P_{\mathcal{R}}(R_i(m_i)|P,h,H)$. $P_{\mathcal{R}}(STOP|P,h,H)$ indique la fin de la partie droite de la règle en cours d'expansion.

3. Faire de même à gauche selon une distribution $P_{\mathcal{L}}(L_i(m_i)|P,h,H)$, $P_{\mathcal{L}}(STOP|P,h,H)$ indique le début de l'expansion.

où m_i est la tête lexicale de la catégorie prédite (R_i ou L_i)

Modèle Collins [1997]

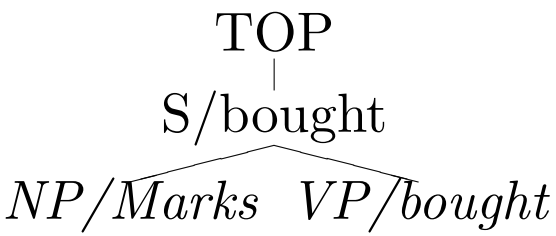
Exemple (1/6)



$\hookrightarrow$ on détermine le constituant qui portera la tête selon une distribution $P_{\mathcal{H}}$. Admettons ici que $p_{\mathcal{H}}(VP|\text{S, bought})$ gagne.

Modèle Collins [1997]

Exemple (2/6)

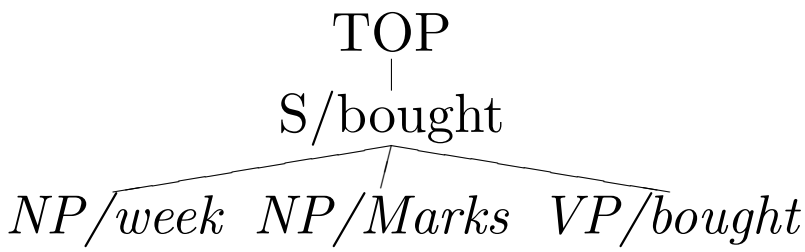


On génère à gauche de VP selon une distribution $p_{\mathcal{L}}(Cat(head)|S, VP, bought)$.

Admettons ici que $p_{\mathcal{L}}(NP(Marks)|S, VP, bought)$ l'emporte. . .

Modèle Collins [1997]

Exemple (3/6)

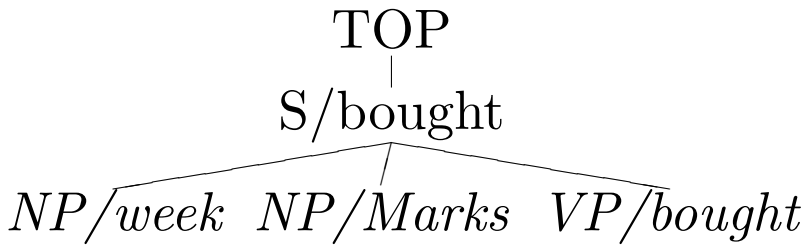


On génère à gauche de NP selon une distribution $p_{\mathcal{L}}(Cat(head)|S, VP, bought)$.

Admettons ici que $p_{\mathcal{L}}(NP(week)|S, VP, bought)$ l'emporte. . .

Modèle Collins [1997]

Exemple (4/6)



On génère à gauche de NP selon une distribution $p_{\mathcal{L}}(Cat(head)|S, VP, bought)$. Admettons ici que $p_{\mathcal{L}}(STOP|S, VP, bought)$ l'emporte. . .

On génère à droite de VP selon une distribution $p_{\mathcal{R}}(Cat(head)|S, VP, bought)$. Admettons ici que $p_{\mathcal{R}}(STOP|S, VP, bought)$ l'emporte. . .

Modèle Collins [1997]
Exemple (5/6)

La probabilité de la règle:

$$\overbrace{S(bought)}^{l_i} \rightarrow \overbrace{NP(week)NP(marks)VP(bought)}^{r_i}$$

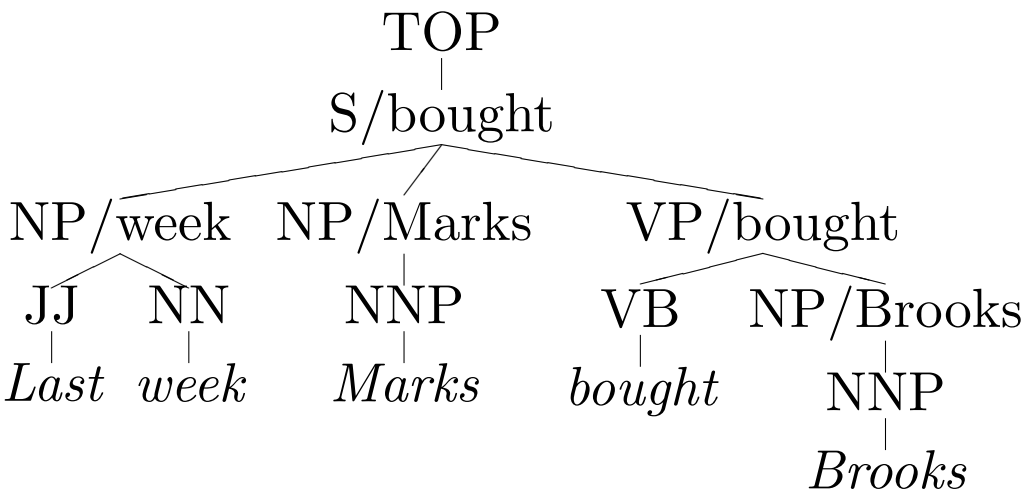
est donc estimée par:

$$\begin{aligned}
&p_{\mathcal{H}}(VP|\text{S, bought})\times \\
&p_{\mathcal{L}}(NP(marks)|S,VP,bought)\times \\
&p_{\mathcal{L}}(NP(week)|S,VP,bought)\times \\
&p_{\mathcal{L}}(STOP|S,VP,bought)\times \\
&p_{\mathcal{R}}(STOP|S,VP,bought)
\end{aligned}$$

$\Rightarrow$ hypothèse markovienne d'ordre 0 (en pratique une distance au mot de tête fait également partie du contexte conditionnant).

Modèle Collins [1997]

Exemple (6/6)



On continue jusqu'à générer l'arbre au complet.

Modèle Collins [1997]

test	LR %	LP %	CBs	0 CBs	≤ 2 CBs
	≤ 40 mots (2245 phrase)				
Collins [1996]	85.8	86.3	1.14	59.9%	83.6%
Modèle I	87.4	88.1	0.96	65.7%	86.3%
Modèle II	88.1	88.6	0.91	66.5%	86.9%
Modèle III	88.1	88.6	0.91	66.4%	86.9%
test	LR %	LP %	CBs	0 CBs	≤ 2 CBs
	≤ 100 mots (2416 phrase)				
Collins [1996]	85.3	85.7	1.32	57.2%	80.8%
Modèle I	86.8	87.6	1.11	63.1%	84.1%
Modèle II	87.5	88.1	1.07	63.9%	84.6%
Modèle III	87.5	88.1	1.07	63.9%	84.6%

PCFG + 3g > 3g Charniak [2001]

Soit le jargon suivant où c est un constituant (un nœud):

$h(c)$ la tête du constituant	$(manger)$
$t(c)$ 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 (toute information utile connue hors de c)	

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

$\hookrightarrow$ 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:

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

$\hookrightarrow p(h(c)|t(c), l(c), m(c), i(c), u(c))$ 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

Influence du corpus d'entraînement[1]

train	test	Recall	Prec.
WSJ	WSJ	86.1	86.6
WSJ	Brown	80.3	81.0
Brown	Brown	83.6	84.6
WSJ + Brown	Brown	83.9	84.8
WSJ + Brown	WSJ	86.3	86.9

performances d'une implémentation du modèle 1 de Collins [1997]

- Brown = 21 818 phrases arborées du corpus Brown,
- WSJ = 39 832 phrases du PTB

Lire Roark and Bacchiani [2003] pour des expériences plus positives en adaptation.

[1] Chiffres pris de Gildea [2001]

Ressource

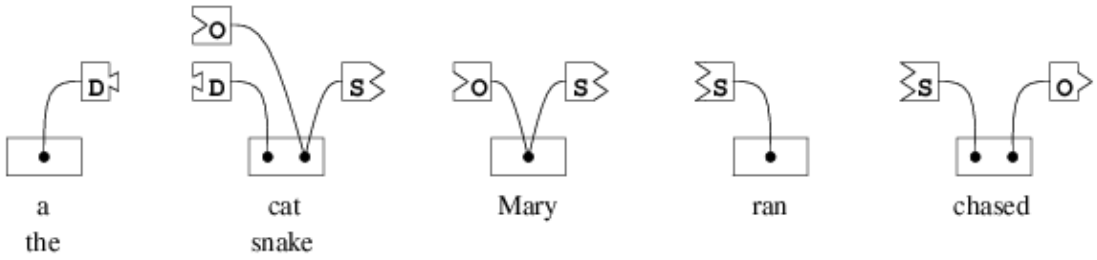
Beaucoup de travaux récents en analyse syntaxique probabiliste. Plusieurs analyseurs disponibles pour l'anglais, certains sont adaptés pour d'autres langues.

- Stanford Parser `nlp.stanford.edu/sofware/lex-parser.shtml`

- Parser de Collins:
  - `www.cis.upenn.edu/~dbikel/#stat-parser` (page de Daniel Bikel)
  - `people.csail.mit.edu/mcollins/code.html` (page de Michael Collins)

Les link grammars Sleator and Temperley [1991][2]

Link grammar = { un ensemble de mots (les terminaux)
des contraintes de liage pour chaque mot



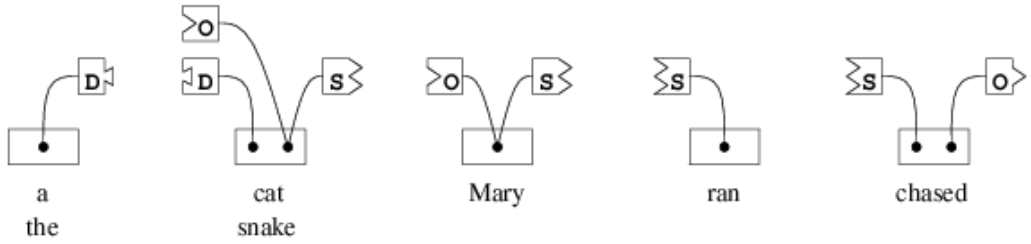
Une phrase fait partie du langage **ssi** il existe un moyen de tracer au dessus des mots des arcs (liens) tels que:

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

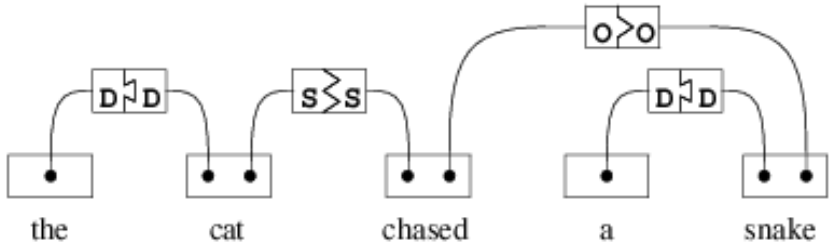
[2] Je reprends ici des exemples décrits dans Sleator and Temperley [1991]

Les link grammars

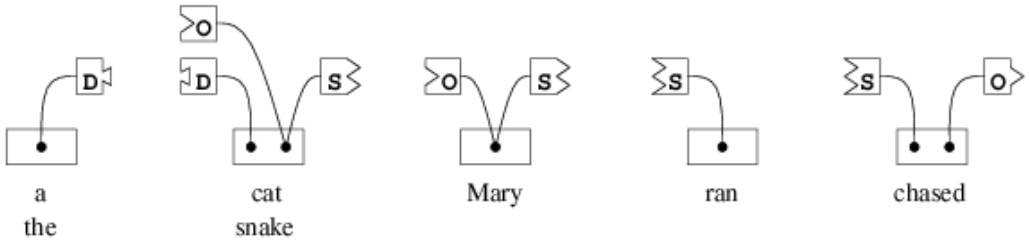
Sastisfaction des contraintes de liage de chaque mot



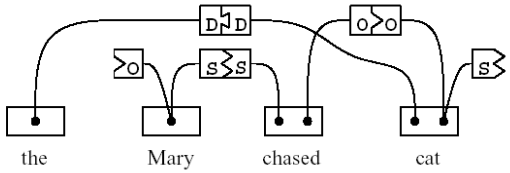
Les contraintes de liage de chaque mot sont exprimées par des connecteurs à gauche et/ou à droite. Un connecteur et un seul à gauche (resp. à droite) s'il en existe un, doit être statisfait (satisfait = emboîtement).



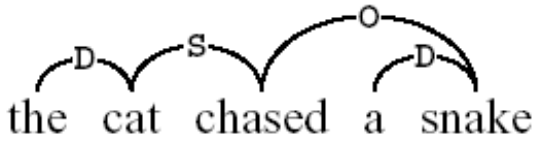
Les link grammars



Exemple de phrase qui ne fait pas parti du langage:

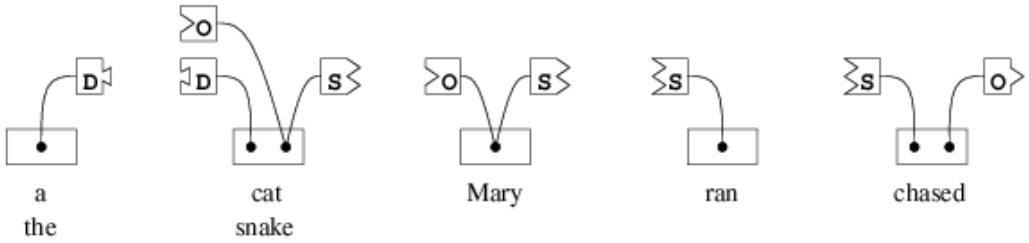


Un autre exemple avec une représentation simplifiée:



Les link grammars

Inutile d'être bon dessinateur pour faire des 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).

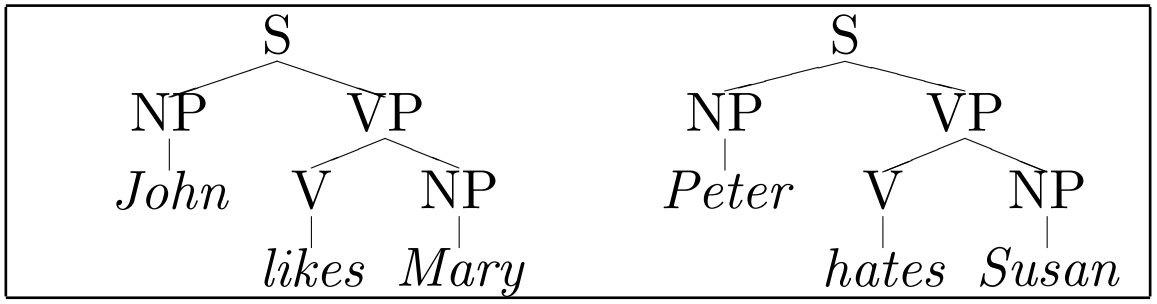
- Voir aussi **MiniPar**: `http://www.cs.ualberta.ca/~lindek/minipar.htm` (300 mots/sec.)

Modèles DOP Bod [1992]

Data Oriented Parsing

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

Ex: soit le treebank suivant:



$\hookrightarrow$ on peut alors construire une collection de sous arbres appelés arbres élémentaires à partir desquels nous allons construire – par composition – de nouvelles phrases (cad de nouveaux arbres)

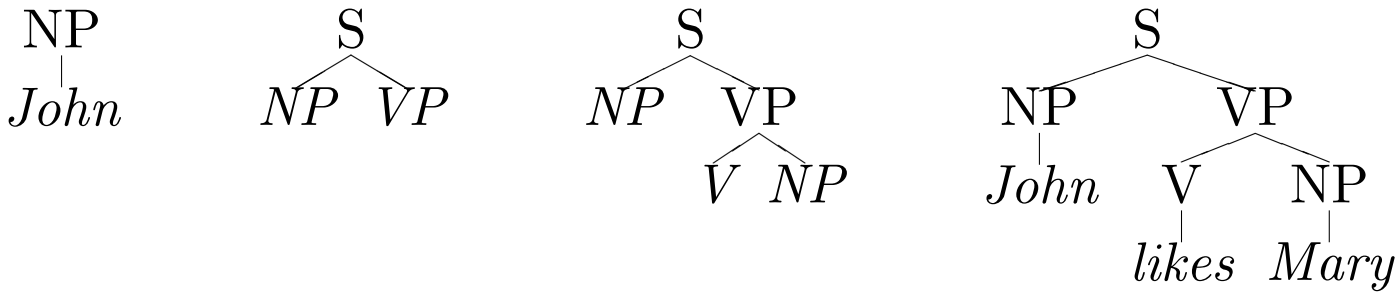
Modèles DOP Bod [1992]

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.

Ex:

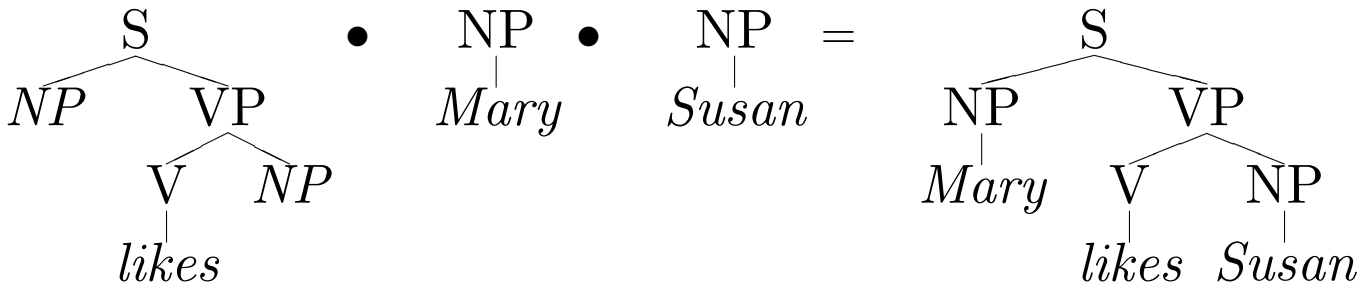


De notre PTB, on peut ainsi extraire un total de 34 fragments

Modèles DOP Bod [1992]

Mary likes Susan

Une composition possible ($\bullet$ est un opérateur compositionnel associatif à gauche):

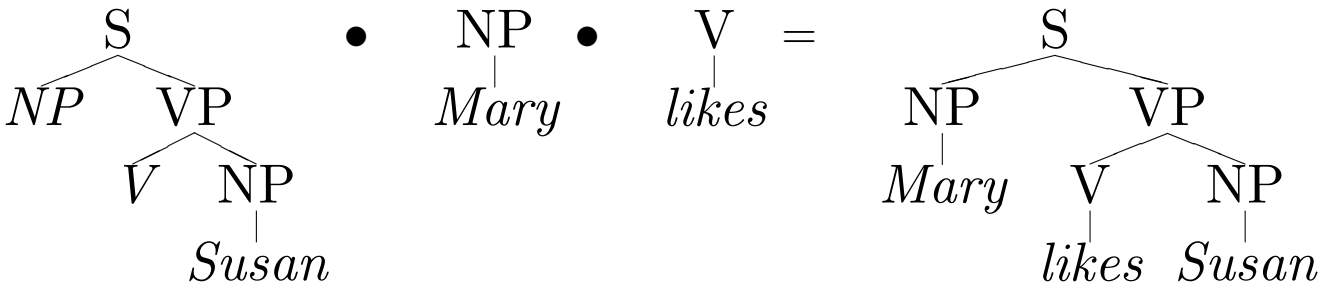


$\hookrightarrow$ L'opération de composition entre deux arbres ($t_1 \bullet t_2$) s'applique toujours sur le non-terminal (N) le plus à gauche de l'arbre t_1. Il faut de plus que la racine de t_2 soit égale à N

Modèles DOP Bod [1992]

Mary likes Susan

Une autre composition possible:



Modèles DOP Bod [1992]

DOP-I estime la probabilité d'un sous-arbre t par:

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

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

La probabilité d'une dérivation $D = t_1 \bullet \ldots \bullet t_n$ est donnée par:

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

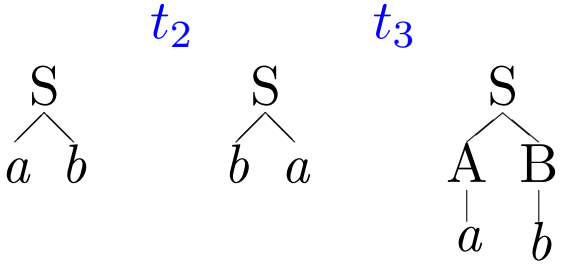
Et la probabilité d'un arbre est donnée par la somme des probabilités de chaque dérivation (d) menant à t:

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

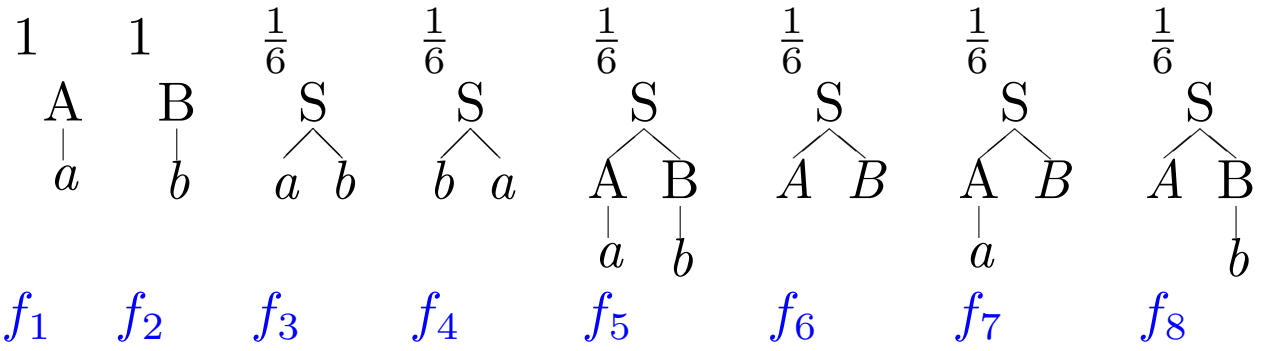
Modèles DOP 1

Exemple complet

Soit le TB:



La collection associée est constituée des 8 fragments suivants:



Modèles DOP 1

Exemple complet

Les 4 dérivations suivantes 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 $p_{dop}(t_3) = \sum_{i=1}^{4} p(d_i) = \frac{4}{6} = \frac{2}{3}$ et $p_{dop}(t_1) = \frac{1}{3}$

Note: $p_{dop}(t_3) > p_{dop}(t_1)$ donc DOP-1 préfère l'arbre d'analyse t_3 à t_1. De manière générale, les arbres les plus profonds dominent la masse de probabilité.

Modèles DOP Bod [1992]

Avantages

- La taille des sous-arbres n'est pas limitée (a priori): un sous-arbre peut donc potentiellement capturer des dépendances plus grandes qu'une PCFG (qui n'offre finalement que la concaténation de sous-arbres de profondeur 1).

- Ne semble pas posséder le biais des PCFGs en faveur des dérivations les plus courtes Bod [2000]

- Permet de mesurer "ce qu'il faudrait mettre" dans une grammaire Bod [2001]

Modèles DOP Bod [1992]

Cons

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

Références

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

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

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

Rens Bod. What is the minimal set of fragments that achieves maximal parse accuracy ? In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 66–73, Toulouse, France, July 2001.

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

Eugene Charniak. Immediate-head parsing for language models. In *Proceedings of the 39th*

Annual Meeting of the Association for Computational Linguistics, pages 116–124, Toulouse, 2001. Morgan Kaufmann Publishers. URL `citeseer.nj.nec.com/384171.html`.

Eugene Charniak. *Statistical Language Learning*. MIT Press, 1993.

Eugene Charniak. Tree-bank grammars. Technical report, Brown University, January 1996. CS-96-02.

Eugene Charniak. 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, 1997.

Ciprian Chelba and Frederick Jelinek. Exploiting syntactic structure for language modeling. In Christian Boitet and Pete Whitelock, 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, 1998. Morgan Kaufmann Publishers. URL `citeseer.nj.nec.com/chelba98exploiting.html`.

Ciprian Chelba, David Engle, Frederick Jelinek, Victor M. Jimenez, Sanjeev Khudanpur,

Lidia Mangu, Harry Printz, Eric Ristad, Ronald Rosenfeld, Andreas Stolcke, and Dekai Wu. Structure and performance of a dependency language model. In *Proc. Eurospeech '97*, pages 2775–2778, Rhodes, Greece, 1997. URL `citeseer.nj.nec.com/article/chelba97structure.html`.

Michael Collins. Three generative, lexicalized models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 16–23, Madrid, Spain, July 1997.

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

Michael Collins. Parameter estimation for statistical parsing models: Theory and practice of distribution-free methods, 2003. URL `citeseer.nj.nec.com/456841.html`.

Michael John Collins. A new statistical parser based on bigram lexical dependencies. In Arivind Joshi and Martha Palmer, editors, *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, pages 184–191, San Francisco, 1996. Morgan Kaufmann Publishers. URL `citeseer.nj.nec.com/collins96new.html`.

Daniel Gildea. Corpus variation and parser performance. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2001.

Joshua Goodman. *Parsing inside-out*. PhD thesis, Harvard University, 1998.

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

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

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

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

Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. Prentice Hall,

2000.

Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, 2003.

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

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

Dekang Lin. A dependency-based method for evaluating broad-coverage parsers. In *IJCAI*, pages 1420–1427, 1995. URL `citeseer.nj.nec.com/lin95dependencybased.html`.

David Magerman. 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, June 1995.

Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language*

Processing. MIT Press, 1999.

Ngoc Tran Nguyen. Étude de transformations grammaticales pour l' entraînement de grammaires probabilistes hors-contexte. Master's thesis, Université de Montréal, jan 2003.

F. Pereira and Y. Schabes. 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, June 1992.

B. Roark and M. Bacchiani. Supervised and unsupervised pcfg adaptation to novel domains. In *Human Language Technology Conference and Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, 2003.

Brian Roark. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276, 2001. URL `citeseer.nj.nec.com/540884.html`.

Yves Schabes, Michal Roth, and Randy Osborne. 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, June 1993.

Daniel D.K. Sleator and Davy Temperley. Parsing english with link grammar. Technical report, Carnegie Mellon University, Pittsburgh, oct 1991.

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