



北京航空航天大学
BEIHANG UNIVERSITY

自然语言处理

人工智能研究院

主讲教师 沙磊



统计句法分析

基于统计的句法分析

- 句法分析，理论上可由两个阶段完成生成句子的所有句法树

句法排歧，找出正确的句法树

- 如何评价所有的句法树
- 引入概率，建立基于统计的句法分析
- 统计句法分析在一段时间内引起了较多的关注，并取得了较好的研究成果。

统计句法分析的基本思路

- 对给定的句子 S , 该句子的统计句法分析结果为:

$$\hat{T} = \arg \max_T P(T|S)$$

- 根据贝叶斯公式, 有:

$$\hat{T} = \arg \max_T P(T, S)$$

- 如何计算句法树的概率?
 - 概率上下文无关文法 (PCFG)
 - 改进的 PCFG

CKY算法

- 属于基于CFG的句法分析算法(与GLR、Earley类似)
- 由Cocke、Kasami及Younger, 上世纪60年代提出, 故命名为: Cocke-Kasami-Younger算法, 简称CKY算法。
- 属于自底而上的分析算法。
- 可以处理一般的文法, 但更适于处理乔姆斯基范式。
- 什么是乔姆斯基范式? 文法中只能有以下两种形式的重写规则:
 - $A \rightarrow BC$
 - $A \rightarrow w$

CKY算法

- 可以证明，对任何一个CFG，都存在与之等价的乔姆斯基范式。
 - $S \rightarrow aAB|BA \quad A \rightarrow BBB|a$
 - $B \rightarrow AS|b$
- 假定文法符合乔姆斯基范式不会损失CKY算法的一般性。 CKY算法的主要数据结构是一个二维表 $T[n,n]$ ，其中每个表元素定义如下：

$$t_{i,j} = \{A \mid A \xRightarrow{+} w_i w_{i+1} \dots w_j\}$$

- 即可以推导出词串 $w_i w_{i+1} \dots w_j$ 的非终结符号所组成的集合。

CKY 算法

- CKY 算法自底而上填写表格，首先

$$t_{i,i} = \{A \mid A \rightarrow w_i \in P\}$$

- 若有 $A \rightarrow BC \in P$, $B \in t_{i,k}$ 及 $C \in t_{k+1,j}$, 则有 $A \in t_{i,j}$ 。为什么?
- 分析过程举例，给定文法

$$S \rightarrow AA|AS|b$$

$$A \rightarrow SA|AS|a$$

试分析句子 abaab。

	1	2	3	4	5
1	A	S,A	S,A	S,A	S,A
2		S	A	S	S,A
3			A	S	S,A
4				A	S,A
5					S

CKY 算法

```
FUNCTION CKYRecognizer() begin  
    boolean chart[1..n][1..|N|][1..n] ← FALSE;  
    for  $k \leftarrow 1$  to  $n$  do  
        for each rule  $A \rightarrow w_k \in P$  do  
             $chart[k, A, k] \leftarrow \text{TRUE}$ ;  
    for  $l \leftarrow 2$  to  $n$  do  
        for  $p \leftarrow 1$  to  $n-l+1$  do  
            for  $t \leftarrow 1$  to  $l-1$  do  
                 $q \leftarrow p+l-1$ ;  $d \leftarrow p+t-1$ ;  
                for each rule  $A \rightarrow BC \in P$  do  
                     $chart[p, A, q] \leftarrow chart[p, A, q] \vee$   
                         $(chart[p, B, d] \wedge chart[d+1, C, q]);$   
    return chart[1,  $S$ ,  $n$ ];  
end.
```


什么是概率上下文无关文法？

- PCFG是CFG的一种扩展。一个PCFG
- G 是一个四元组 $G = (N, \Sigma, S, P)$

其中：

N 有限个非终结符号组成的集合

Σ 有限个终结符号组成的集合

S 文法的开始符号

P 是一组带有概率信息的重写规则组成的集合，每条规则形式如下：

$$A \rightarrow \alpha \quad [P(A \rightarrow \alpha)]$$

$\alpha \in (V_N \cup V_T)^*$, $P(A \rightarrow \alpha)$ 是重写规则的概率。且： $\sum_j P(A \rightarrow \alpha_j) = 1$

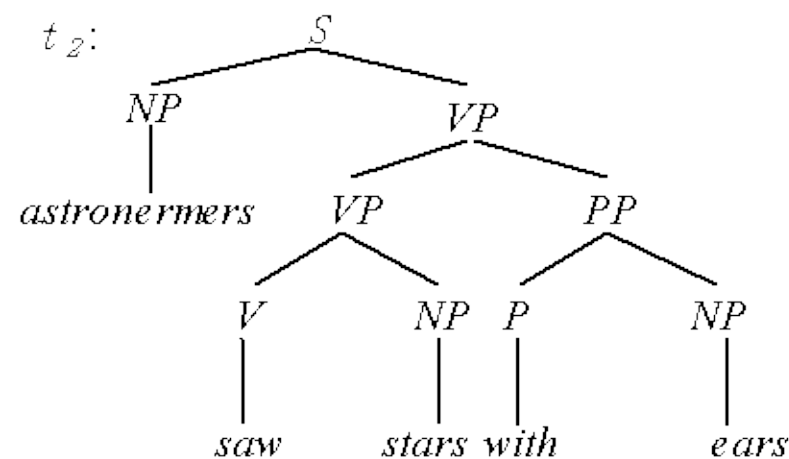
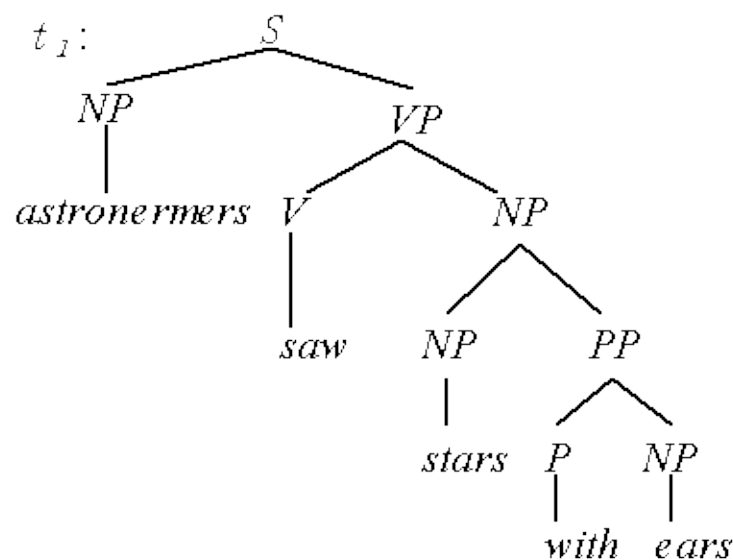
概率上下文无关文法举例

$S \rightarrow NP VP$	1.0	$NP \rightarrow NP PP$	0.4
$PP \rightarrow P NP$	1.0	$NP \rightarrow \textit{astronomers}$	0.1
$VP \rightarrow V NP$	0.7	$NP \rightarrow \textit{ears}$	0.18
$VP \rightarrow VP PP$	0.3	$NP \rightarrow \textit{saw}$	0.04
$P \rightarrow \textit{with}$	1.0	$NP \rightarrow \textit{stars}$	0.18
$V \rightarrow \textit{saw}$	1.0	$NP \rightarrow \textit{telescopes}$	0.1

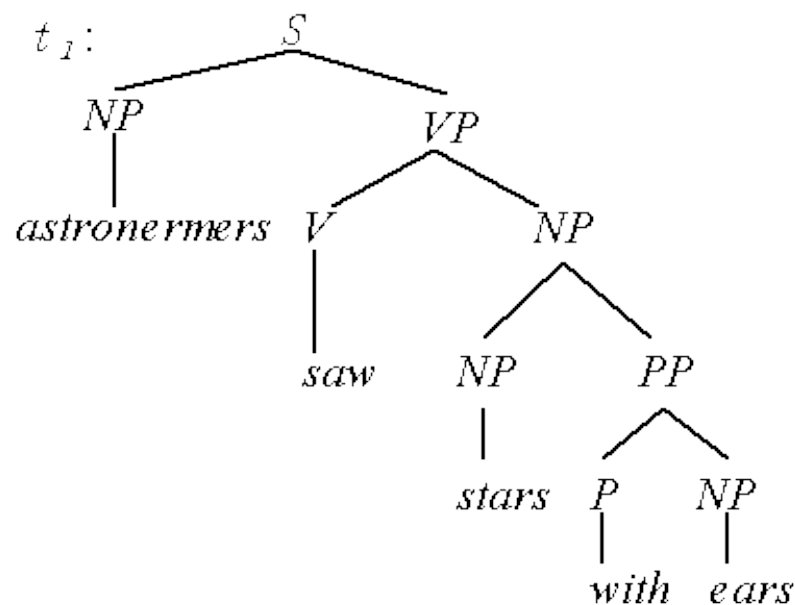
利用PCFG计算分析树的概率

$$P(t, S) = \prod_{i=1..n} P(r_i)$$

❖ 句子“astronomers saw stars with ears”的分析树



利用PCFG计算分析树的概率

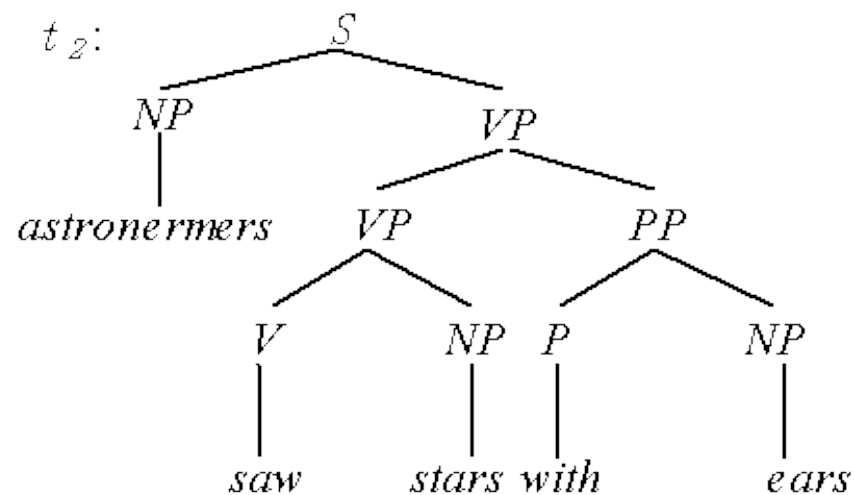


$$P(t_1) = 1.0 \times 0.1 \times 0.7 \times 1.0 \times 0.4 \times 0.18 \times 1.0 \times 1.0 \times 0.18$$

$$= 0.0009072$$

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

利用PCFG计算分析树的概率



$$P(t_2) = 1.0 \times 0.1 \times 0.3 \times 0.7 \times 1.0 \times 0.18 \times 1.0 \times 1.0 \times 0.18$$

$$= 0.0006804$$

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

PCFG 用于句法分析

- 基于PCFG可以计算分析树的概率值。
- 若一个句子有多个分析树，可以依据概率值对所有的分析树进行排序。
- PCFG可以用来进行句法排歧。面对多个分析结果，选择概率最大者为最终分析结果。

PCFG 用作语言模型

- 基于概率上下文无关文法，一个句子 w_{1m} 的概率为：

$$P(S) = \sum_t P(S, t)$$

- 句子 “astronomers saw stars with ears” 的概率

- $P(S) = P(t1) + P(t2) = 0.0009072 + 0.0006804 = 0.0015876$

- PCFG 提供了一种统计语言模型，同 n -gram 模型相比，基于 PCFG 的语言模型考虑了句子的结构信息，而 n -gram 模型则认为句子是线性结构。

PCFG的基本问题

- ① 给定一部概率上下文无关文法 G ，如何计算句子 S 的概率？即计算 $P(S|G)$ 的问题。（语言模型）
- ② 给定一部概率上下文无关文法 G 以及句子 S ，最为可能的分析树是什么，即计算 $\arg \max_t P(t|S, G)$ 的问题。（句法分析）
- ③ 如何为文法规则选择概率，使得训练句子的概率最大？即计算 $\arg \max_G P(S, G)$ 的问题。（模型训练）
- 可以通过计算每个分析树的概率，然后以求和或求最大值的方式解决上述第①、②个问题，缺点是效率不高。

向内变量和向外变量

- 为了有效解答PCFG的三个问题，定义向外变量(outside variable)和向内变量(inside variable).
 - 向外变量

$$\alpha_A(p, q) = P(S \xRightarrow{*} w_1 w_2 \dots w_{p-1} A w_{q+1} \dots w_n) = P(w_1 w_2 \dots w_{p-1} A w_{q+1} \dots w_n | S)$$

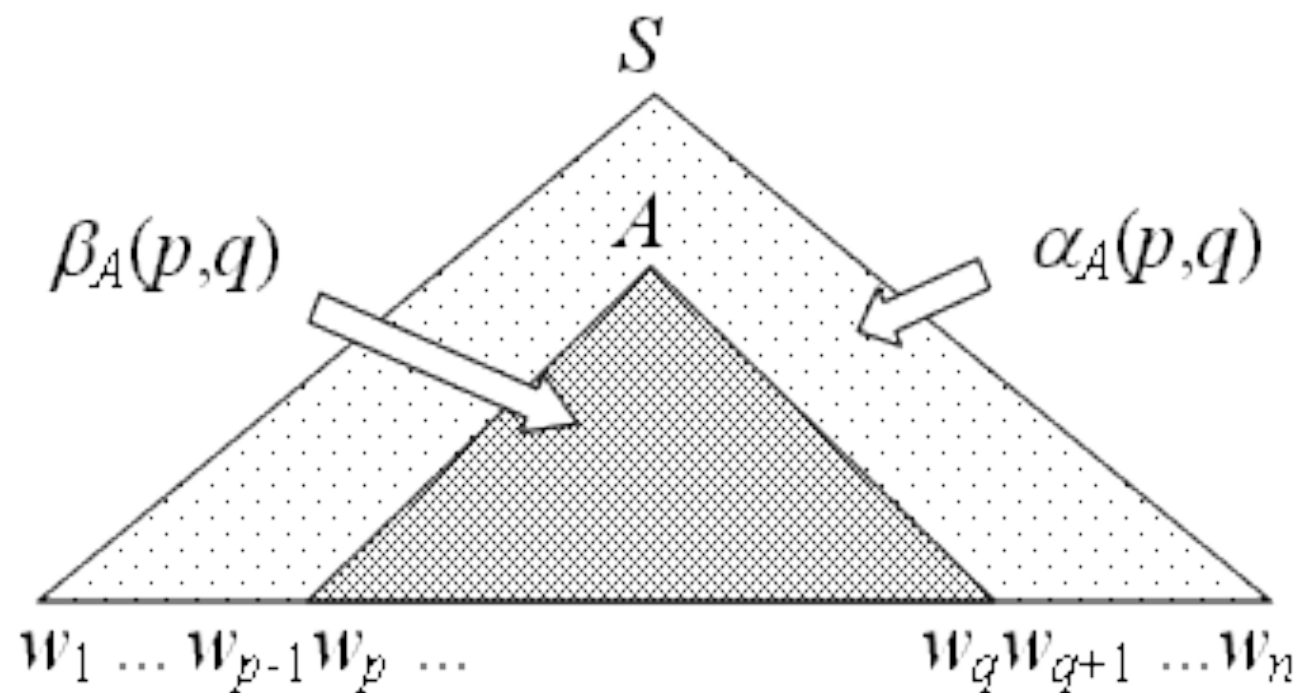
文法开始符号 S 推导出句型 $w_1 w_2 \dots w_{p-1} A w_{q+1} \dots w_n$ 的概率。

- 向内变量

$$\beta_A(p, q) = P(A \xRightarrow{*} w_p w_{p+1} \dots w_q) = P(w_p w_{p+1} \dots w_q | A)$$

非终结符号 A 推导出句子中子串 $w_p w_{p+1} \dots w_q$ 的概率，或者说以 A 为根、叶子为 $w_p w_{p+1} \dots w_q$ 的所有子树的概率。

向内变量和向外变量



计算句子 w_{1m} 的概率

- 向内变量和句子概率的关系

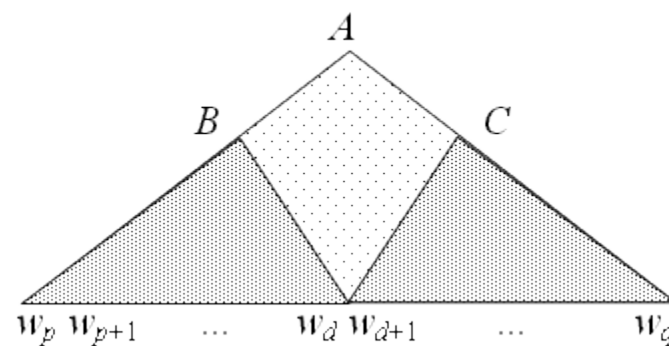
$$\beta_S(1, n) = P(S \xRightarrow{*} w_1 w_2 \dots w_n)$$

- 另有

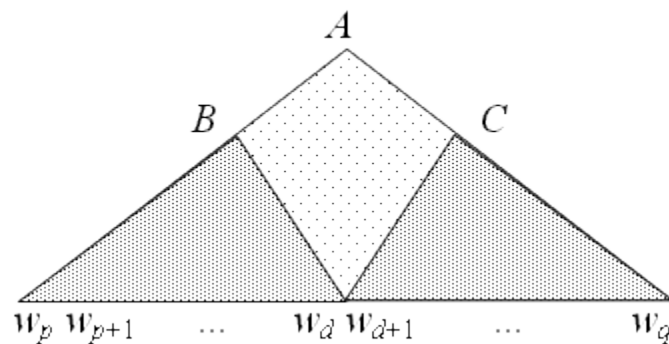
$$\beta_A(k, k) = P(A \xRightarrow{*} w_k)$$

向内算法(Inside Algorithm)

- 如何计算 $\beta_A(p, q)$, 其中 $p < q$
- 因为限制文法为 *Chomsky* 范式,
因此第一条使用的重写规则必为
 - $A \rightarrow BC$
- 子串 w_{pq} 一定在某个位置 d 被分成两个部分, 使得 B 支配子串 w_{pd} , 而 C 支配子串 $w_{(d+1)q}$,



向内算法



$$\begin{aligned}
 P(A \Rightarrow BC \xRightarrow{*} w_p w_{p+1} \dots w_d w_{d+1} \dots w_q) &= P(A \Rightarrow BC) P(B \xRightarrow{*} w_p w_{p+1} \dots w_d) P(C \xRightarrow{*} w_{d+1} w_{d+2} \dots w_q) \\
 &= P(A \rightarrow BC) \beta_B(p, d) \beta_C(d+1, q)
 \end{aligned}$$

而计算 $\beta_A(p, q)$ 应考虑所有可能的以 A 为左部的重写规则，而选定重写规则后，也应考虑将 $w_p w_{p+1} \dots w_q$ 分割成两个子串的不同情况，即要考虑为 d 做所有可能的选择。故有：

$$\beta_A(p, q) = \sum_{B, C} \sum_d P(A \Rightarrow BC \xRightarrow{*} w_p w_{p+1} \dots w_d w_{d+1} \dots w_q)$$

$$\beta_A(p, q) = \sum_{B, C} \sum_{d=p}^{q-1} P(A \rightarrow BC) \beta_B(p, d) \beta_C(d+1, q)$$

向内算法

① 初始化

$$\beta_A(k, k) = P(A \rightarrow w_k)$$

② 归纳计算 $\beta_A(p, q)$, 其中 $p < q$

$$\beta_A(p, q) = \sum_{B, C} \sum_{d=p}^{q-1} P(A \rightarrow BC) \beta_B(p, d) \beta_C(d+1, q)$$

③ 归纳终止

$$P(w_1 w_2 \dots w_n) = \beta_S(1, n)$$

向内算法自底而上的递归计算句子概率。

向内算法计算实例

	1	2	3	4	5
1	$\beta_{NP} = 0.1$		$\beta_S = 0.0126$		$\beta_S = 0.0015876$
2		$\beta_{NP} = 0.04$ $\beta_V = 1.0$	$\beta_{VP} = 0.126$		$\beta_{VP} = 0.015876$
3			$\beta_{NP} = 0.18$		$\beta_{NP} = 0.01296$
4				$\beta_P = 1.0$	$\beta_{PP} = 0.18$
5					$\beta_{NP} = 0.18$
	<i>Astronomers</i>	<i>saw</i>	<i>stars</i>	<i>with</i>	<i>ears</i>

$S \rightarrow NP VP$	1.0	$NP \rightarrow NP PP$	0.4
$PP \rightarrow P NP$	1.0	$NP \rightarrow \text{astronomers}$	0.1
$VP \rightarrow V NP$	0.7	$NP \rightarrow \text{ears}$	0.18
$VP \rightarrow VP PP$	0.3	$NP \rightarrow \text{saw}$	0.04
$P \rightarrow \text{with}$	1.0	$NP \rightarrow \text{stars}$	0.18
$V \rightarrow \text{saw}$	1.0	$NP \rightarrow \text{telescopes}$	0.1

单元格(p, q)内为
向内概率 $\beta_i(p, q)$

基于CKY算法的向内概率计算

```
FUNCTION InsideProbability()
begin
  float inside[1..n][1..|N][1..n]  $\leftarrow$  0;
  for  $k \leftarrow 1$  to  $n$  do
    for each rule  $A \rightarrow w_k \in P$  do
       $inside(k, A, k) \leftarrow P(A \rightarrow w_k)$ ;
  for  $l \leftarrow 2$  to  $n$  do
    for  $p \leftarrow 1$  to  $n-l+1$  do
      for  $t \leftarrow 1$  to  $l-1$  do
         $q \leftarrow p+l-1$ ;  $d \leftarrow p+t-1$ ;
        for each rule  $A \rightarrow BC \in P$  do
           $inside(p, A, q) \leftarrow inside(p, A, q) +$ 
             $inside(p, B, d) \times inside(d+1, C, q) \times P(A \rightarrow BC)$ 
      return  $inside(1, S, n)$ ;
end.
```

向外算法(outside algorithm)

- 如何计算 $\alpha_A(p,q)$?

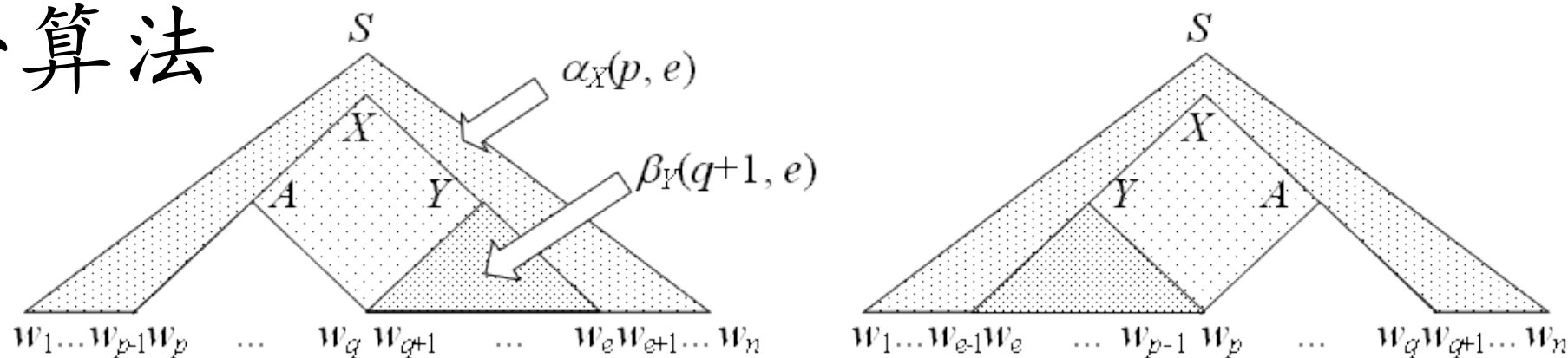
向外变量 $\alpha_A(p, q)$ 指的是 S 推导出句型 $w_1w_2\ldots w_{p-1}Aw_{q+1}\ldots w_n$ 的概率。因为限制文法是乔姆斯基范式，因此必然存在重写规则 $X \rightarrow AY$ 或重写规则 $X \rightarrow YA$ ，使得：

$$S \xRightarrow{*} w_1w_2\ldots w_{p-1}Xw_{q+1}\ldots w_n \Rightarrow w_1w_2\ldots w_{p-1}AYw_{q+1}\ldots w_n \text{ 且 } Y \xRightarrow{*} w_{q+1}w_{q+2}\ldots w_e$$

或者，

$$S \xRightarrow{*} w_1w_2\ldots w_{e-1}Xw_{q+1}\ldots w_n \Rightarrow w_1w_2\ldots w_{e-1}YAw_{q+1}\ldots w_n \text{ 且 } Y \xRightarrow{*} w_e w_{e+1}\ldots w_{p-1}$$

向外算法



若结点 A 作为父结点 X 的左子女，即：

$$P(S \xRightarrow{*} w_1 \dots w_{p-1} X w_{e+1} \dots w_n \Rightarrow w_1 \dots w_{p-1} A Y w_{e+1} \dots w_n, Y \xRightarrow{*} w_{q+1} \dots w_e) \\ = \alpha_X(p, e) P(X \rightarrow AY) \beta_Y(q+1, e)$$

同理，若结点 A 作为父结点 X 的右子女，则有：

$$P(S \xRightarrow{*} w_1 \dots w_{e-1} X w_{q+1} \dots w_n \Rightarrow w_1 \dots w_{e-1} Y A w_{q+1} \dots w_n, Y \xRightarrow{*} w_e \dots w_{p-1}) \\ = \alpha_X(e, q) P(X \rightarrow YA) \beta_Y(e, p-1)$$

所以，如何求 $\alpha_A(p, q)$? (5min)

向外算法

计算 $\alpha_A(p, q)$ 时, 对于这两种情况, 都需要考虑各种可能的重写规则及选择所有可能的 e 值, 故:

$$\begin{aligned}\alpha_A(p, q) = & \sum_{X, Y} \sum_{e=q+1}^n \alpha_X(p, e) P(X \rightarrow AY) \beta_Y(q+1, e) \\ & + \sum_{X, Y} \sum_{e=1}^{p-1} \alpha_X(e, q) P(X \rightarrow YA) \beta_Y(e, p-1)\end{aligned}$$

因为句法树的根结点总是文法的开始符号, 而不会是其它非终结符号, 所以有 $P(S \Rightarrow S) \models 1$ 及 $P(S \Rightarrow X) = 0$, 即:

$$\alpha_S(1, n) = 1, \quad \alpha_X(1, n) = 0 \quad (X \neq S)$$

此外, 根据向内变量及向外变量的定义, 有:

$$\begin{aligned}P(S \Rightarrow w_1 w_2 \dots w_{p-1} A w_{q+1} \dots w_n \Rightarrow w_1 w_2 \dots w_{p-1} w_p \dots w_q w_{q+1} \dots w_n) \\ = P(S \Rightarrow w_1 w_2 \dots w_{p-1} A w_{q+1} \dots w_n) P(A \Rightarrow w_p w_{p+1} \dots w_q) \\ = \alpha_A(p, q) \beta_A(p, q)\end{aligned}$$

向外算法

向外算法自顶向下
递归计算句子的概率

(1) 初始化, 令

$$\alpha_A(1, n) = \begin{cases} 1, & \text{if } A = S \\ 0, & \text{if } A \neq S \end{cases}$$

(2) 归纳计算, 对所有的 p, q , 且 $p < q$, 令

$$\begin{aligned} \alpha_A(p, q) = & \sum_{X, Y} \sum_{e=q+1}^n \alpha_X(p, e) P(X \rightarrow AY) \beta_Y(q+1, e) \\ & + \sum_{X, Y} \sum_{e=1}^{p-1} \alpha_X(e, q) P(X \rightarrow YA) \beta_Y(e, p-1) \end{aligned}$$

(3) 归纳终止, 对任意非终结符号

$$P(w_1 w_2 \dots w_n) = \sum_A \alpha_A(p, q) \beta_A(p, q)$$

基于CKY算法的向外概率计算

FUNCTION *OutsideProbability*()

begin

 float *outside*[1..*n*][1..*N*][1..*n*]:=0;

outside[1, *S*, *n*+1] \leftarrow 1;

 for *l* \leftarrow *n* down to 2 do

 for *s* \leftarrow 1 to *n*-*l*+1 do

 for *t* \leftarrow 1 to *l*-1 do

$p_1 \leftarrow s; e_1 \leftarrow s+l-1; q_1 \leftarrow s+t-1;$

$p_2 \leftarrow s+t; e_2 \leftarrow s; q_2 \leftarrow s+l-1;$

 for each rule $A \rightarrow BC \in P$ do

$outside(p_1, B, q_1) = outside(p_1, B, q_1) +$

$outside(p_1, A, e_1) \times inside(q_1+1, C, e_1) \times P(A \rightarrow BC)$

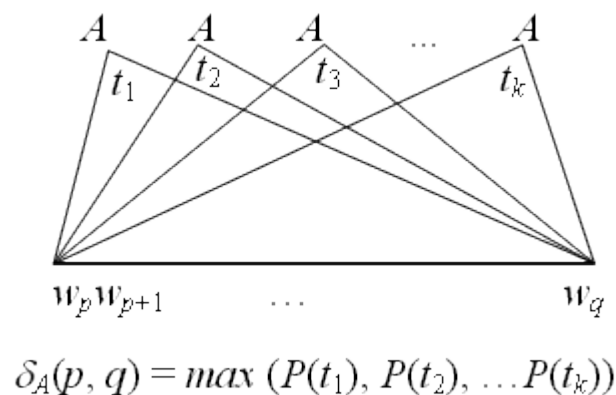
$outside(p_2, C, q_2) = outside(p_2, C, q_2) +$

$outside(e_2, A, q_2) \times inside(e_2, B, p_2-1) \times P(A \rightarrow BC)$

end.

寻找最佳的分析树

- PCFG的第二个基本问题是在给定文法 G 和句子 $w_1 w_2 \dots w_m$ 的前提下，如何有效找出最为可能的分析树，这可以通过韦特比算法求得。
- 韦特比变量 $\delta_A(p, q)$
- 以 A 为根并且叶结点是 $w_p w_{p+1} \dots w_q$ 的所有子树中概率最大的子树的概率。



韦特比算法(Viterbi Algorithm)

- (1) 初始化, 对所有的 k ($1 \leq k \leq n$), 令

$$\delta_A(k, k) = P(A \rightarrow w_k)$$

- (2) 归纳计算, 对所有的 p, q , 且 $p < q$, 令

$$\delta_A(p, q) = \max_{\substack{p \leq d < q \\ B, C \in N}} P(A \rightarrow BC) \delta_B(p, d) \delta_C(d+1, q)$$

$$\psi_A(p, q) = \arg \max_{(B, C, d)} P(A \rightarrow BC) \delta_B(p, d) \delta_C(d+1, q)$$

- (3) 归纳终止

$$P(\hat{t}) = \delta_S(1, n)$$

- (4) 根据数组 $\psi_A(p, q)$ 中记录的信息构造概率最大的分析树 \hat{t} 。

a) \hat{t} 的根结点为 S 。

b) 若 A 是 \hat{t} 的内部结点(非叶子结点), 并且若 $\psi_A(p, q) = (B, C, d)$, 则 A 的左子女是 B , 右子女是 C 。且 B 支配子串 $w_p w_{p+1} \dots w_d$, C 支配子串

$w_{d+1} w_{d+2} \dots w_q$ 。

模型训练

- 如何为语法规则选择概率，使得训练句子的概率最大？也就是如何得到语法规则的概率的问题。
- 有指导训练
- 无指导训练，向内向外算法

模型训练

- 有指导的训练
- 树库(Treebank), 是标记了句法树结构的语料库。

$$\hat{P}(A \rightarrow BC) = \frac{C(A \rightarrow BC)}{\sum_{\gamma} C(A \rightarrow \gamma)} \quad \hat{P}(A \rightarrow w) = \frac{C(A \rightarrow w)}{\sum_{\gamma} C(A \rightarrow \gamma)}$$

- 树库的构建的工作量巨大, 耗时耗力, 但在没有可靠的无指导训练技术的前提下, 树库的构造必须进行
- 美国宾夕法尼亚大学一直致力于树库的构建工作, 其构建的树库被称作 *Penn Treebank*。其中英文树库规模较大、汉语树库的规模较小
- *Penn treebank* 尽管规模很小, 但为统计句法分析研究 提供了一个很好的基础。

向内向外算法(inside-outside算法)

- 无指导训练算法是IO算法
- 同Baum-Welch 算法类似，IO算法也是一个反复迭代、逐步求精的算法。
- 通常要首先给定一组不准确的参数，以反复迭代计算的方式调整模型参数，最终使参数稳定 在一个可以接受的精度。
- IO算法不能保证求得最优模型，一般能得到一个局部最优模型。

向内向外算法的基本原理

- 没有树结构，无法准确统计：
- $C(A \rightarrow BC)$ 、 $C(A \rightarrow w)$
- 通过CKY算法，可得到给定句子的所有树结构。基于所有树结构，计算规则的期望频次。

$$C'(A) = \sum P(t|S) C(A)$$

$$C'(A \rightarrow BC) = \sum P(t|S) C(A \rightarrow BC)$$

$$C'(A \rightarrow w) = \sum P(t|S) C(A \rightarrow w)$$

- 基于上述期望频次，进行参数估计

$$P'(A \rightarrow BC) = C'(A \rightarrow BC) \div C'(A)$$

$$P'(A \rightarrow w) = C'(A \rightarrow w) \div C'(A)$$

向内向外算法的基本原理

- 如何计算 $P(t|S)$?
- 前提是参数已知。"蛋生鸡、鸡生蛋"的问题。
- 循环迭代、逐步求精
- 首先给定一组初始参数;
 - 循环, 直到得到一组合理的参数 基于当前参数,
 - 计算 $P(t|S)$
 - 计算 $C'(A)$ 、 $C'(A \rightarrow BC)$ 以及 $C'(A \rightarrow w)$
 - 计算新参数
- 计算实例
- 效率问题: 逐棵计算树的概率

首先，非终结符号 A 在句法树中支配子串 $w_p w_{p+1} \dots w_q$ 的期望次数，可由下面的公式给出：

$$\begin{aligned}
 & P(A \xRightarrow{*} w_p w_{p+1} \dots w_q | S \xRightarrow{*} w_1 w_2 \dots w_n) \\
 &= \frac{P(A \xRightarrow{*} w_p w_{p+1} \dots w_q, S \xRightarrow{*} w_1 w_2 \dots w_n)}{P(S \xRightarrow{*} w_1 w_2 \dots w_n)} \\
 &= \frac{P(S \xRightarrow{*} w_1 w_2 \dots w_{p-1} A w_{q+1} \dots w_n \xRightarrow{*} w_1 w_2 \dots w_{p-1} w_p \dots w_q w_{q+1} \dots w_n)}{P(S \xRightarrow{*} w_1 w_2 \dots w_n)} \\
 &= \frac{\alpha_A(p, q) \beta_A(p, q)}{P(S \xRightarrow{*} w_1 w_2 \dots w_n)} \quad (\#1\#)
 \end{aligned}$$

令， $\pi = P(S \xRightarrow{*} w_1 w_2 \dots w_n)$ ， 则：

$$P(A \xRightarrow{*} w_p w_{p+1} \dots w_q | S \xRightarrow{*} w_1 w_2 \dots w_n) = \frac{\alpha_A(p, q) \beta_A(p, q)}{\pi}$$

考虑到所有可能的 $p < q$ ， 则非终结符号 A 在句法树中出现的期望次数为：

$$P(A \text{ 在句法树中出现}) = \sum_{p=1}^n \sum_{q=p}^n \frac{\alpha_A(p, q) \beta_A(p, q)}{\pi}$$

(#2#)

同理，考虑重写规则 $A \rightarrow BC$ 在句法树中出现的期望次数：

$$\begin{aligned}
 & P(A \Rightarrow BC \xRightarrow{*} w_p w_{p+1} \dots w_q | S \xRightarrow{*} w_1 w_2 \dots w_n) \\
 &= \frac{P(A \Rightarrow BC \xRightarrow{*} w_p w_{p+1} \dots w_q, B \xRightarrow{*} w_p \dots w_d, C \xRightarrow{*} w_{d+1} \dots w_q, S \xRightarrow{*} w_1 w_2 \dots w_n)}{P(S \xRightarrow{*} w_1 w_2 \dots w_n)} \\
 &= \frac{\sum_{d=p}^{q-1} \alpha_A(p, q) P(A \rightarrow BC) \beta_B(p, d) \beta_C(d+1, q)}{\pi}
 \end{aligned}$$

考虑到所有可能的 $p < q$ ，重写规则 $A \rightarrow BC$ 在句法树中出现的期望次数为

$$P(A \rightarrow BC \text{ 在句法树中出现}) = \frac{\sum_{p=1}^{n-1} \sum_{q=p+1}^n \sum_{d=p}^{q-1} \alpha_A(p, q) P(A \rightarrow BC) \beta_B(p, d) \beta_C(d+1, q)}{\pi}$$

因此有：

$$\begin{aligned}
 \hat{P}(A \rightarrow BC) &= \frac{P(A \rightarrow BC \text{ 在句法树中出现})}{P(A \text{ 在句法树中出现})} \\
 &= \frac{\sum_{p=1}^{n-1} \sum_{q=p+1}^n \sum_{d=p}^{q-1} \alpha_A(p, q) P(A \rightarrow BC) \beta_B(p, d) \beta_C(d+1, q)}{\sum_{p=1}^m \sum_{q=p}^m \alpha_A(p, q) \beta_A(p, q)}
 \end{aligned}$$

(#4#)

对 $A \rightarrow w$, 也可做类似的推导, 若 $w_k = w$, 有:

$$\begin{aligned}
 & P(A \rightarrow w_k | S \xRightarrow{*} w_1 w_2 \dots w_n) \\
 &= \frac{P(A \rightarrow w_k, S \xRightarrow{*} w_1 w_2 \dots w_{k-1} A w_{k+1} \dots w_n)}{P(S \xRightarrow{*} w_1 w_2 \dots w_n)} \\
 &= \frac{\alpha_A(k, k) \beta_A(k, k)}{\pi}
 \end{aligned}
 \tag{\#3\#}$$

考虑到所有可能的位置 k , 根据 w_k 是否为 w , 则:

$$P(A \rightarrow w \text{ 在句法树中出现}) = \frac{\sum_{k=1}^n \alpha_A(k, k) \beta_A(k, k) \delta(w_k, w)}{\pi}$$

$$\hat{P}(A \rightarrow w) = \frac{P(A \rightarrow w \text{ 在句法树中出现})}{P(A \text{ 在句法树中出现})}$$

$$= \frac{\sum_{k=1}^n \alpha_A(k, k) \beta_A(k, k) \delta(w_k, w)}{\sum_{p=1}^m \sum_{q=p}^m \alpha_A(p, q) \beta_A(p, q)}$$

(#5#)

IO算法描述

- EM算法特例
 - 不保证收敛到全局最优点
 - 对初始参数较敏感
- (1) 任意设定一组概率文法参数 $\hat{P}_0(A \rightarrow BC)$ 及 $\hat{P}_0(A \rightarrow w)$
 - (2) 令 $i \leftarrow 1$
 - (3) 循环执行下面的步骤，直到文法参数收敛
 - 1) E-Step: 基于 $\hat{P}_{i-1}(A \rightarrow BC)$ 及 $\hat{P}_{i-1}(A \rightarrow w)$ 计算公式(#1#)、(#2#)、(#3#)的值
 - 2) M-Step: 将 E-Step 的计算结果代入公式(#4#)和(#5#)，得到一组更新的参数 $\hat{P}_i(A \rightarrow BC)$ 及 $\hat{P}_i(A \rightarrow w)$
 - 3) 令 $i \leftarrow i+1$
 - (4) 算法结束，输出概率文法参数

基于PCFG的句法分析

- PCFG把概率引入上下文无关文法，将统计方法和规则方法进行有效的融合，具有十分重要的意义，但是PCFG缺陷也是十分明显的。
- 作为一种统计句法分析方法，基于PCFG的句法分析效果有限。
 - PCFG没有考虑结构之间的依存关系。
 - PCFG没有考虑词汇对句法结构的影响。
- 需要针对PCFG的句法分析表现出来的缺陷进行改进。

结构依存关系

- 代词作为主语的可能性高于作为宾语的可能性

- 主题与述题

- 主题通常是已知信息，在句子中做主语

- 代词用来指代已经陈述过的已知信息

- 述题引入新信息

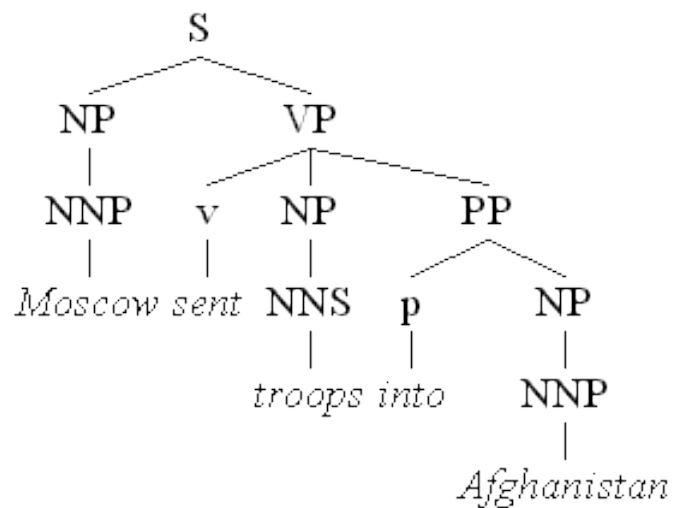
- 统计数据，来自Switchboard corpus:

- ◆ 陈述句中91%的主语是代词
- ◆ 陈述句中66%的直接宾语不是代词

$NP \rightarrow Pron$ $NP \rightarrow Det Noun$ 的概率应和其在句中所处的位置有关，处在VP前后概率应该不同。

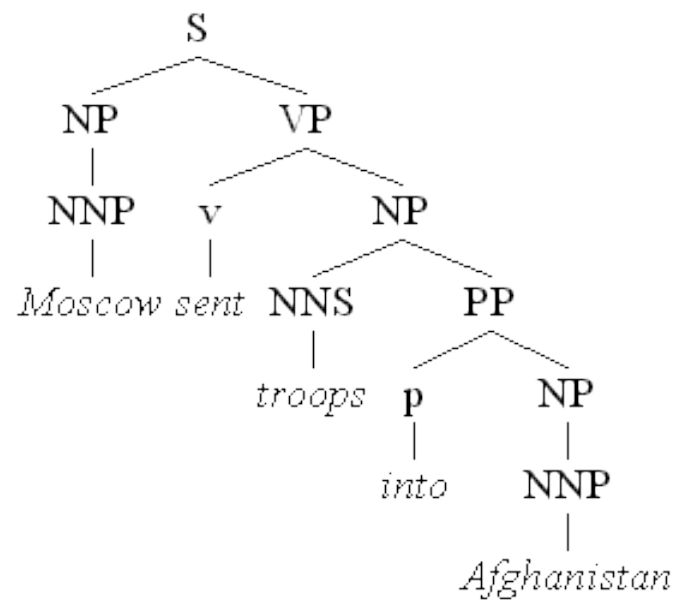
词汇依存关系

Moscow sent troops into Afghanistan



正确的句法树

PP 可以修饰 *VP* , *PP* 也可以修饰 *NP*, 但不同的动词, *PP* 修饰 *NP* 和 *VP* 的可能性并不相同。



错误的句法树

基于PCFG的统计语言模型

- 作为一种统计语言模型，其效果甚至还不如n-gram模型以及HMM模型(原因同上)
- the green banana the green time

句法分析的评价

◆ 标记准确率

$$\text{Labeled Precision} = \frac{\text{number of correct constituents in proposed parse}}{\text{number of constituents in proposed parse}}$$

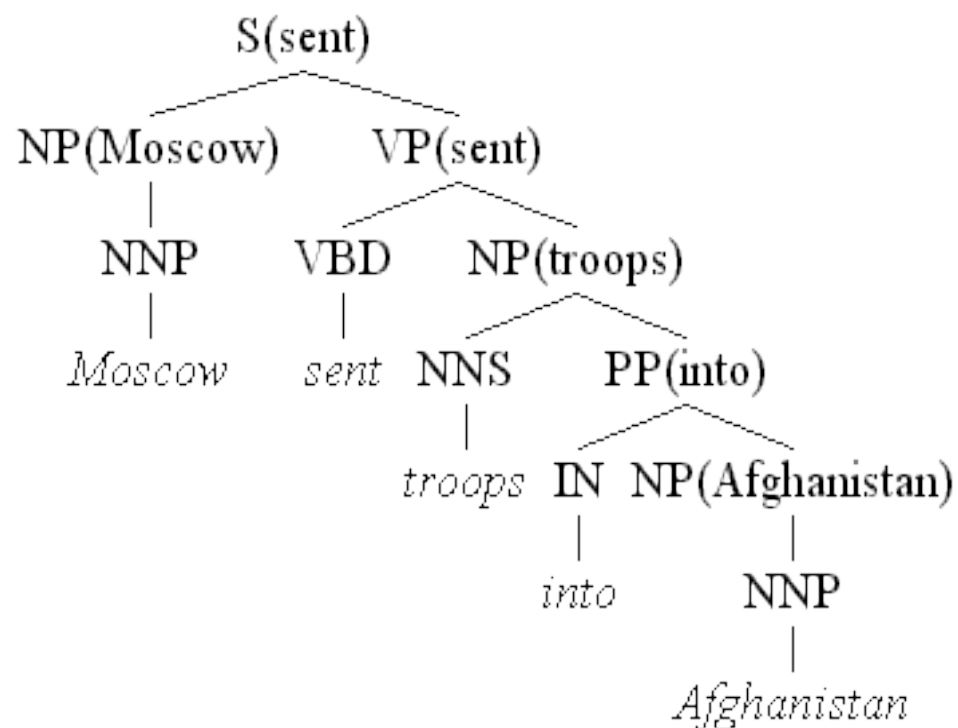
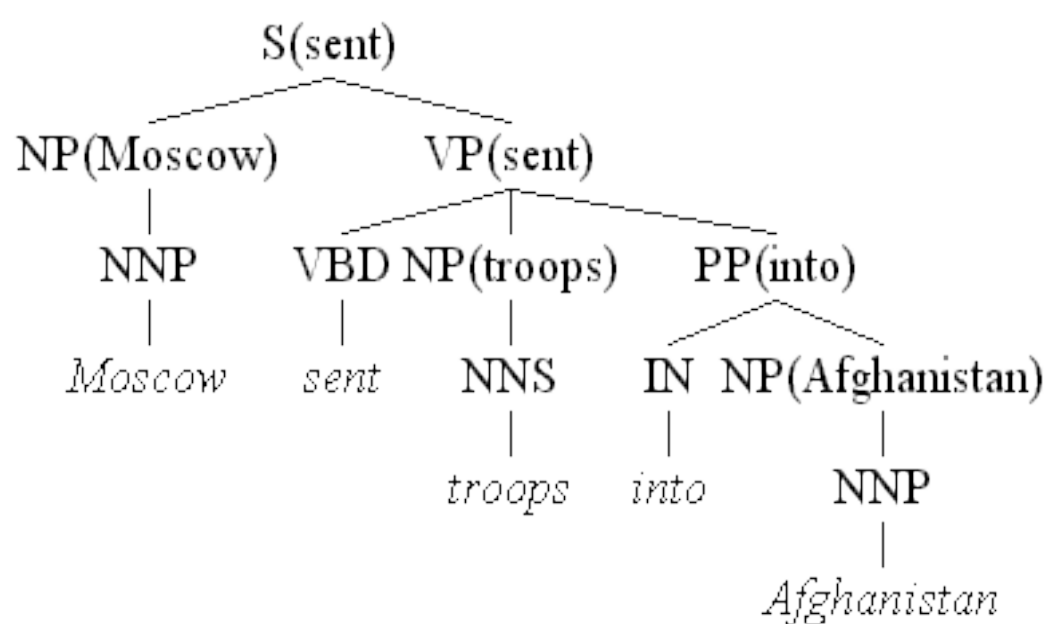
◆ 标记召回率

$$\text{Labeled Recall} = \frac{\text{number of correct constituents in proposed parse}}{\text{number of constituents in treebank parse}}$$

◆ 括号交叉数

$$\text{Crossing Brackets} = \text{number of constituents which violate constituent boundaries with a constituent in the treebank parse}$$

词汇化的句法分析简介



词汇化PCFG

$S(\text{sent}) \rightarrow NP(\text{Moscow}) VP(\text{sent})$

$NP(\text{Moscow}) \rightarrow NNP(\text{Moscow})$

$VP(\text{sent}) \rightarrow VBD(\text{sent}) NP(\text{troops}) PP(\text{into})$

$NP(\text{troops}) \rightarrow NNS(\text{troops}) PP(\text{into})$

$PP(\text{into}) \rightarrow IN(\text{into}) NP(\text{Afghansitan})$

$NP(\text{Afghanistan}) \rightarrow NNP(\text{Afghanistan})$

$NNP(\text{Moscow}) \rightarrow \text{Moscow}$

$NNP(\text{Afghanistan}) \rightarrow \text{Afghanistan}$

$IN(\text{into}) \rightarrow \text{into}$

$VBD(\text{sent}) \rightarrow \text{sent}$

$NNS(\text{troops}) \rightarrow \text{troops}$

非终结符号的数量激烈膨胀。

参数估计时，数据稀疏问题十分严重！

词汇化PCFG

- 为避免数据稀疏问题，概率计算需要分解
- 在词汇化PCFG中

$$P(h) \rightarrow L_n(l_n) \dots L_1(l_1) H(h) R_1(r_1) \dots R_m(r_m)$$

其中， H 是短语的中心成分

h 是成分的中心词

L_i 和 R_i 分别是中心成分左右的修饰性成分

l_i 和 r_i 分别是左右修饰成分的中心词

- 在左右两端增加 $STOP$
- 令 $L_{n+1} = STOP$ $R_{n+1} = STOP$

词汇化PCFG

$$P(L_{n+1}(l_{n+1}) \dots L_1(l_1)H(h)R_1(r_1) \dots R_{m+1}(r_{m+1})|P(h)) = \\ P_h(H|P(h)) \times$$

$$\prod_{i=1 \dots n+1} P_l(L_i(l_i)|L_1(l_1) \dots L_{i-1}(l_{i-1}), P(h), H) \times$$

$$\prod_{j=1 \dots n+1} P_r(R_j(r_j)|L_1(l_1) \dots L_{n+1}(l_{n+1}), R_1(r_1) \dots R_{j-1}(r_{j-1}), P(h), H)$$

- 作如下的独立性假设

$$P_l(L_i(l_i)|L_1(l_1) \dots L_{i-1}(l_{i-1}), P(h), H) = P_l(L_i(l_i)|P(h), H)$$

$$P_r(R_j(r_j)|L_1(l_1) \dots L_{n+1}(l_{n+1}), R_1(r_1) \dots R_{j-1}(r_{j-1}), P(h), H) = P_r(R_j(r_j)|P(h), H)$$

词汇化PCFG

$$P(L_{n+1}(l_{n+1}) \dots L_1(l_1)H(h)R_1(r_1) \dots R_{m+1}(r_{m+1})|P(h)) = P(H|P(h)) \times \\ \prod_{i=1 \dots n+1} P_l(L_i(l_i)|P(h), H) \times \prod_{j=1 \dots m+1} P_r(R_j(r_j)|P(h), H)$$

- 句法树可以视为自顶向下按照三个步骤产生：

1. 以概率 $P_H(H|P, h)$ 生成短语的中心成分标签；

2. 以概率 $\prod_{i=1 \dots n+1} P_L(L_i(l_i)|P, h, H)$ 生成中心词左面的修饰性成分，其中

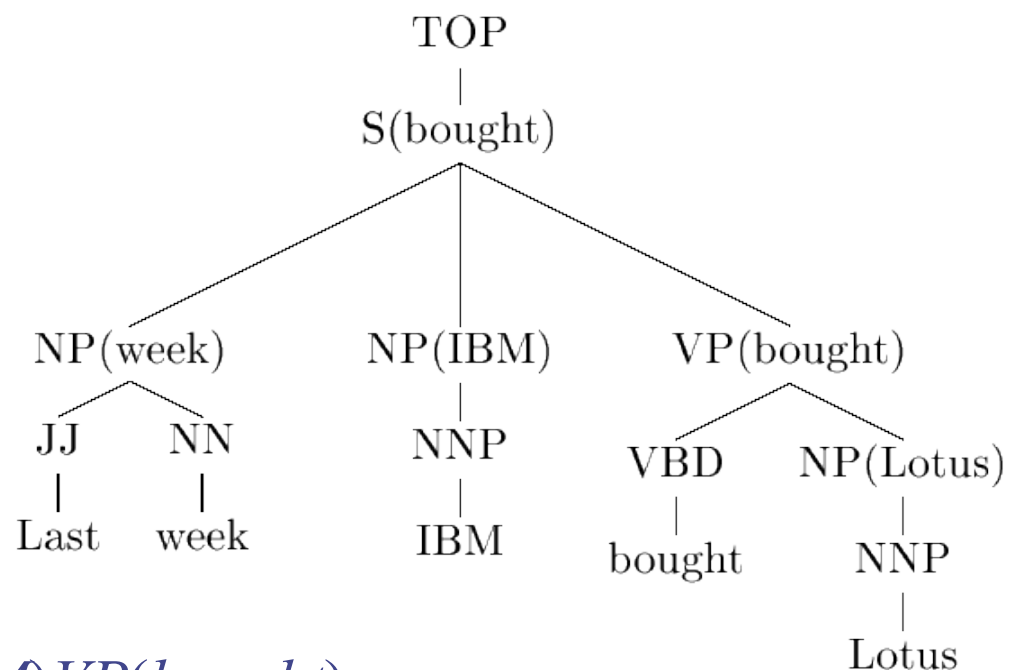
$L_{n+1}(l_{n+1}) = STOP$ 。 $STOP$ 可视作一个特殊的非终结符号，模型在生成 $STOP$ 标记时结束生成。

3. 以概率 $\prod_{i=1 \dots m+1} P_R(R_i(r_i)|P, h, H)$ 生成中心词右面的修饰性成分，其中

$R_{m+1}(r_{m+1}) = STOP$ 。

词汇化PCFG

- 例子



$S(bought) \rightarrow NP(week)NP(IBM)VP(bought)$

$P_h(VP | S, bought) \times P_l(NP(IBM) | S, VP, bought) \times$
 $P_l(NP(week) | S, VP, bought) \times P_l(STOP | S, VP, bought) \times$
 $P_r(STOP | S, VP, bought)$

Thank you!