句法分析2

形式语法

- 形式语法是规定语言中允许出现的结构的形式化说明
- 形式语法可以追溯到1950s（Chomsky's PhD thesis ）
- 几个主要的形式语法：
  - context-free grammar (CFG)
  - lexical functional grammar (LFG)
  - head-driven phrase-structure grammar (HPSG)
  - tree adjoining grammars (TAG)
  - combinatory categorical grammar (CCG)

上下文无关语法

- Context-free grammars (CFGs)
- Hopcroft and Ullman, 1979
- 假设一个语言L是由语法G生成，则G可以表示为一个四元组：G = (T, N, S, R)
  - T：终结符（ terminal symbols）集合，通常包括句法树的叶子节点，如 *like*, *lecture*
  - N：非终结符（ nonterminal symbols）集合，句法树的中间节点，如 *NP*, *S*
  - S：开始符号，特殊的非终结符($S \in N$)，表示句子
  - R：重写规则（或产生式），具有形式 $X \rightarrow \gamma$，例如，*NP→DET JJ NN*
    - $X \in N$ 并且$\gamma \in (N \cup T)^*$

CFGs的特性

- 上下文无关特性：句法规则$X \rightarrow \gamma$的应用不依赖于X出现在什么上下文环境中

- 若s∈ T* 是由CFGs定义的语言，则至少有一种重写规则可以生成s

- 由CFGs生成的语言可能有不止一个短语结构 (结构歧义)

一个简单的CFGs的例子

- T = {sleeps, saw, man, woman, telescope, the, with, in}
- N = {S, NP, VP, PP, DT, Vi, Vt, NN, IN}
- S = S
- R =

$S \to NP\ VP$	$Vi \to$ sleeps
$VP \to Vi$	$Vt \to$ saw
$VP \to Vt\ NP$	$NN \to$ man
$VP \to VP\ PP$	$NN \to$ woman
$NP \to DT\ NN$	$NN \to$ telescope
$NP \to NP\ PP$	$DT \to$ the
$PP \to IN\ NP$	$IN \to$ with
……	$IN \to$ in
	……

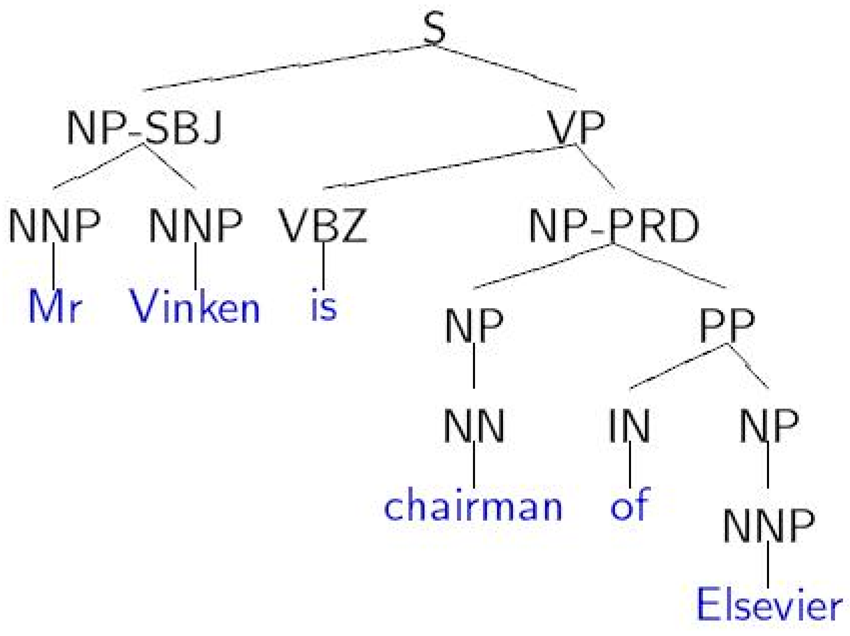
Chomsky 范式

- Chomsky Normal Form
- 一个受Chomsky范式约束的CFG句法 G = (T, N, S, R)，具有以下形式：
  - T：终结符集合
  - N：非终结符集合
  - S：开始符号，特殊的非终结符(S ∈ N)，表示句子
  - R：句法规则集合，具有以下两种形式：
    - $N^i \rightarrow N^j N^k$ for $N^i \in$ N, and N^j , $N^k \in$ N
    - $N^i \rightarrow w^j$ for $N^i \in$ N, and $w^j \in$ T

应用句法规则生成句子

Input	Rule	Output
S	S → NP VP	NP VP
NP VP	NP → PRO	PRO VP
PRO VP	PRO → *I*	*I* VP
I VP	VP → VP NP	*I* VP NP
I VP NP	VP → VB	*I* VB
I VB NP	VB → *like*	*I like* NP
I like NP	NP → DET JJ NN	*I like* DET JJ NN
I like DET JJ NN	DET → *the*	*I like the* JJ NN
I like the JJ NN	JJ → *interesting*	*I like the interesting* NN
I like the interesting NN	NN → *lecture*	*I like the interesting lecture*

应用句法规则构建句法树



S→NP-SBJ VP
NP-SBJ → NNP NNP
NNP → Mr
NNP → Vinken
VP → VBZ NP-PRD
VBZ → is
NP-PRD → NP PP
NP → NN
NN → chairman
PP → IN NP
IN → of
NP → NNP
NNP → Elsevier

应用句法规则构建句法树

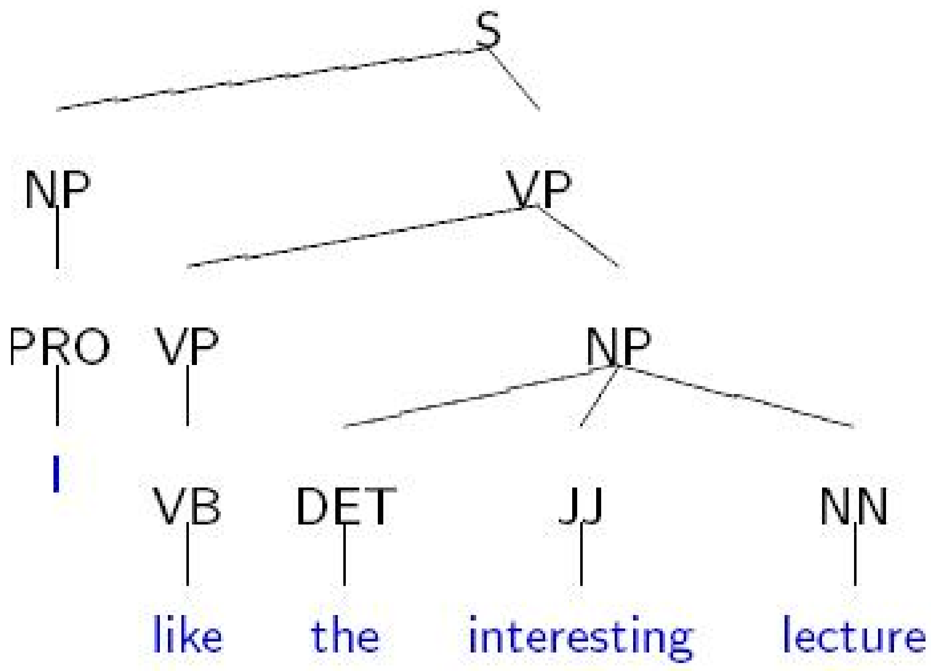
- 可以表述为一个搜索过程
  - 搜索空间：语法规则
  - 搜索过程：检查各种语法规则所有可能的组合方式
  - 搜索目的：最终找到一种组合，其中的语法规则能够生成一个用来表示句子结构的句法树
  - 搜索方向：自顶向下 vs 自底向上

Cocke-Kasami-Younger (CKY) Parsing

- 已有一组上下文无关语法：
  - S→NP VP, NP → PRO, PRO → I, VP → VP NP, VP →VB, VB → like, NP →DET JJ NN, DET → the, JJ → interesting, NN → lecture
- 输入：句子
  - I like the interesting lecture
- CKY句法分析：
  - 自底向上的句法分析算法
  - 采用一个线图（chart）存储中间结果

Example

- 最后得到完整句法树：



CYK算法

- Coke-Younger-Kasami (CYK) 算法
  - 对 Chomsky 文法进行范式化：

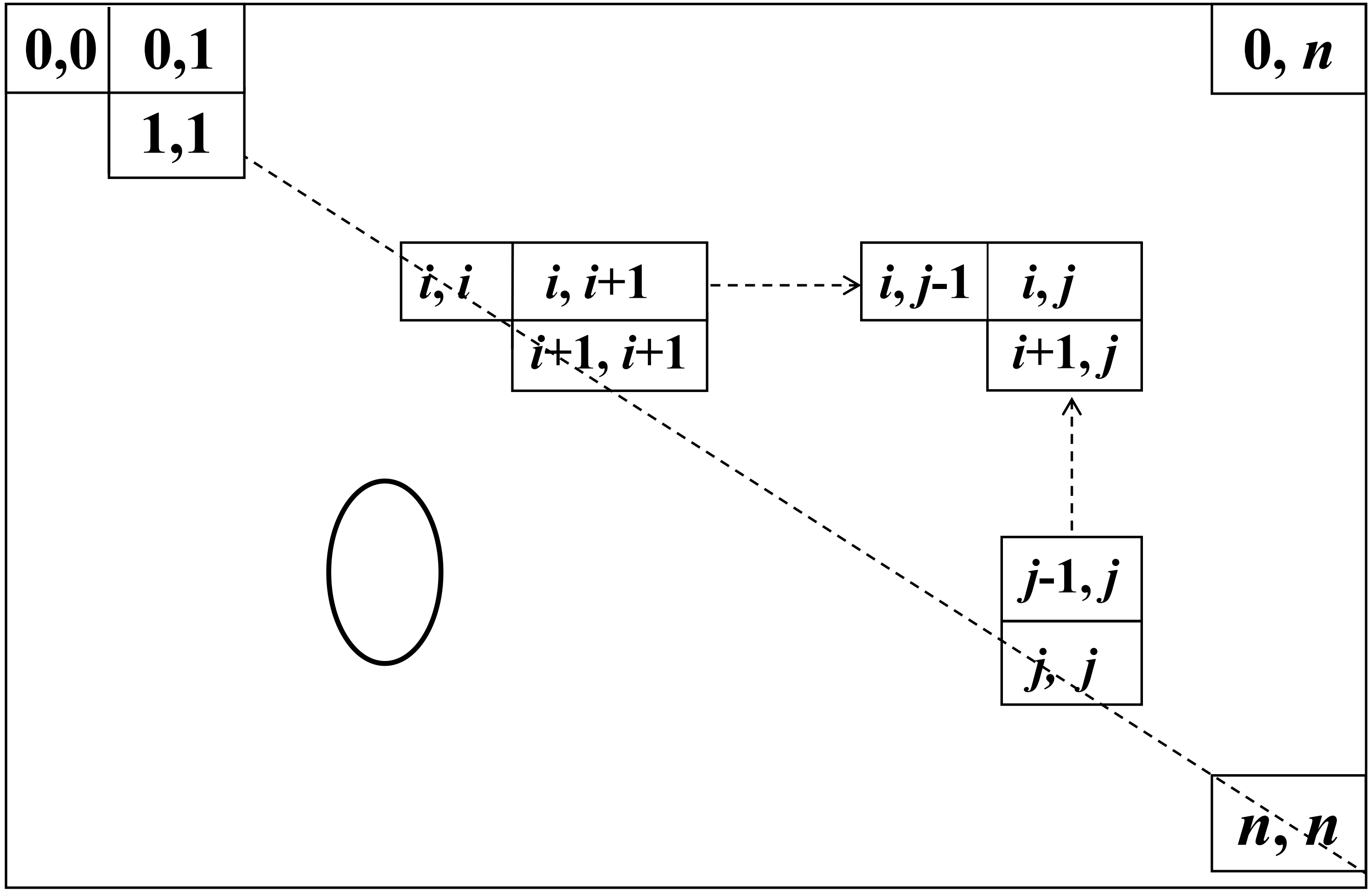
    $A \rightarrow w$ 或 $A \rightarrow BC$

    $A, B, C \in V_N$, $w \in V_T$, $G=(V_N, V_T, P, S)$
  - 自下而上的分析方法
  - 构造 $(n+1)\times(n+1)$ 识别矩阵，n为输入句子长度。假设输入句子 $x=w_1w_2\ldots w_n$，w_i 为构成句子的单词，$n=|x|$。

CYK算法

- 识别矩阵的构成
  - 方阵对角线以下全部为0
  - 主对角线以上的元素由文法G的非终结符构成
  - 主对角线上的元素由输入句子的终结符号(单词)构成

CYK算法



CYK算法

◆ 识别矩阵构造步骤

(1) 首先构造主对角线，令 $t_{0,0}$=0，然后，从 $t_{1,1}$ 到$t_{n,n}$ 在主对角线的位置上依次放入输入句子x 的单词w_i。

(2) 构造主对角线以上紧靠主对角线的元素$t_{i,\ i+1}$, 其中，$i = 0, 1, 2, \ldots, n\text{-}1$。对于输入句子 $x = w_1\, w_2 \ldots\, w_n$，从 w_1 开始分析。

CYK算法

如果在文法G 的产生式集中有一条规则：

$$A \rightarrow w_1$$

则 $t_{0,1}=A$。

依此类推，如果有 $A \rightarrow w_{i+1}$，则 $t_{i,\,i+1}=A$。

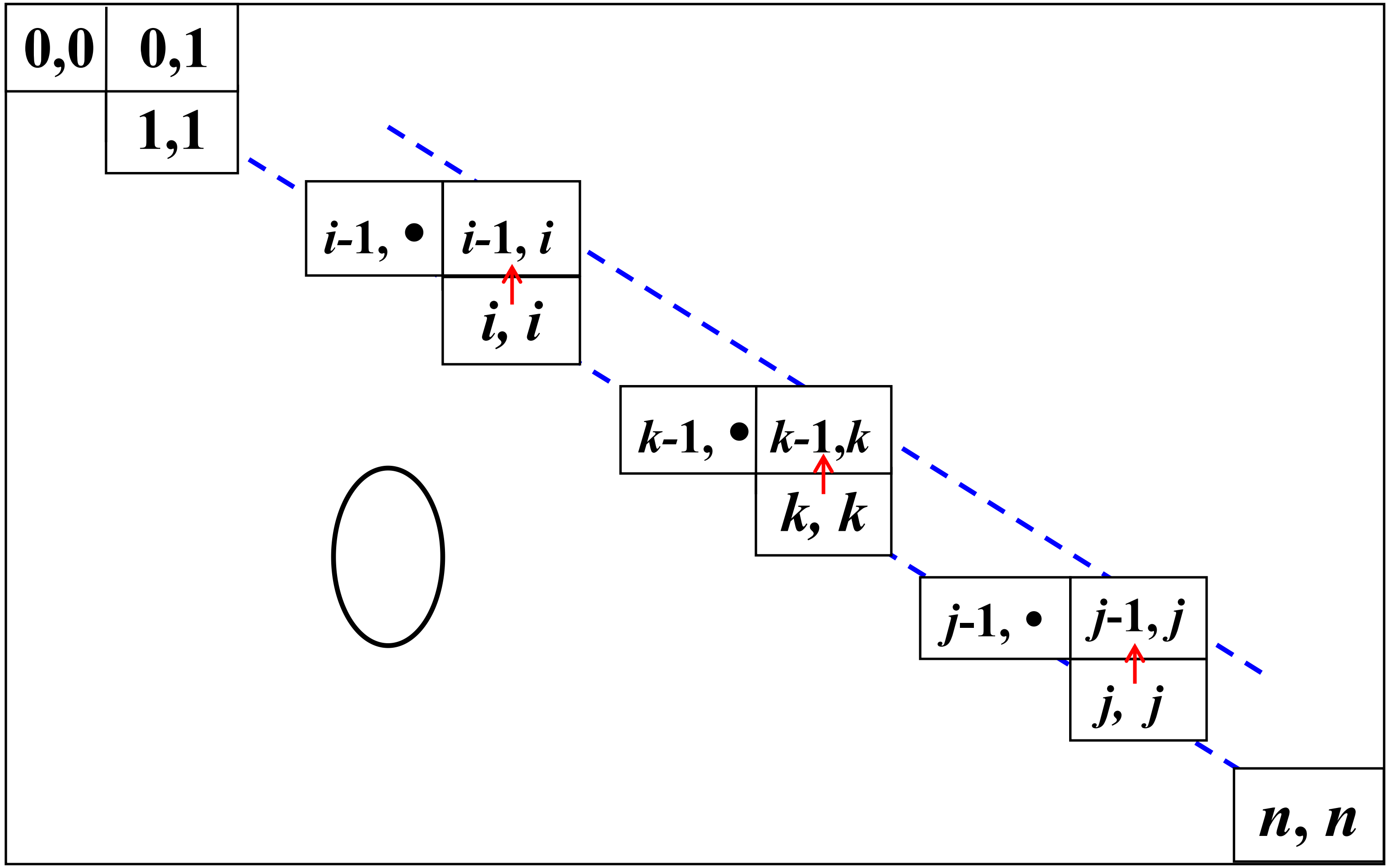
即，对于主对角线上的每一个终结符 w_i，所有可能推导出它的非终结符写在它的右边主对角线上方的位置上。

CYK算法

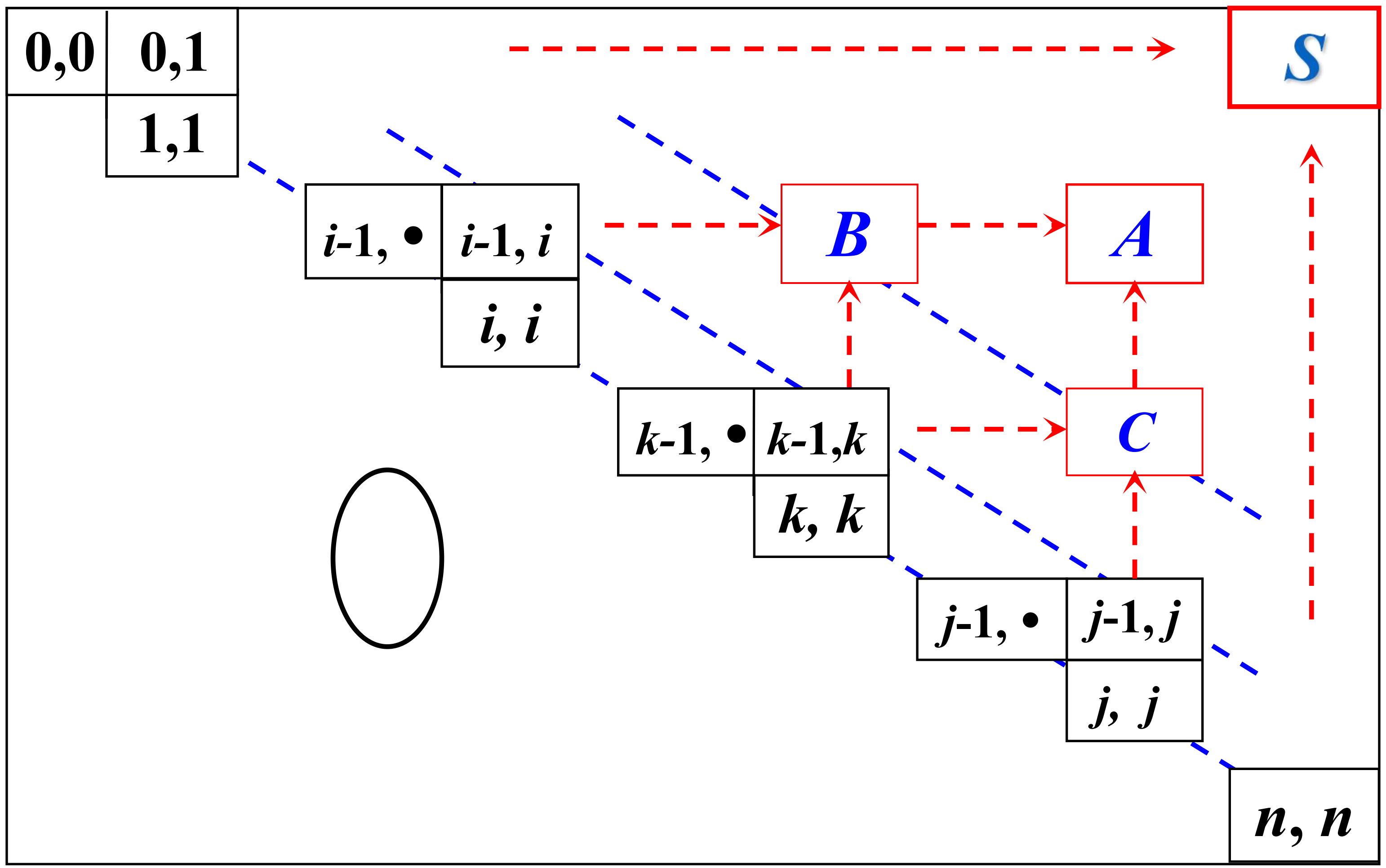
(3) 按平行于主对角线的方向，一层一层地向上填写矩阵的各个元素 $t_{i,j}$，其中，$i = 0,1, \ldots, n\text{-}d$, $j = d+i$，$d=2, 3, \ldots, n$。如果存在一个正整数 k, $i+1 \le k \le j\text{-}1$，在文法G的规则集中有产生式 $A \rightarrow BC$, 并且，$B \in t_{i,k}$, $C \in t_{k,j}$， 那么，将A写到矩阵 $t_{i,j}$ 位置上。

判断句子 x 由文法 G 所产生的充要条件是：$t_{0,n}=S$。

CYK算法



CYK算法



CYK算法

◆ 例子

给定文法 G(S)：

(1) $S \rightarrow P\ \ VP$

(2) $VP \rightarrow V\ \ V$

(3) $VP \rightarrow VP\ \ N$

(4) $P \rightarrow$ 他

(5) $V \rightarrow$ 喜欢

(6) $V \rightarrow$ 读

(7) $N \rightarrow$ 书

请用 CYK 算法分析句子：他喜欢读书

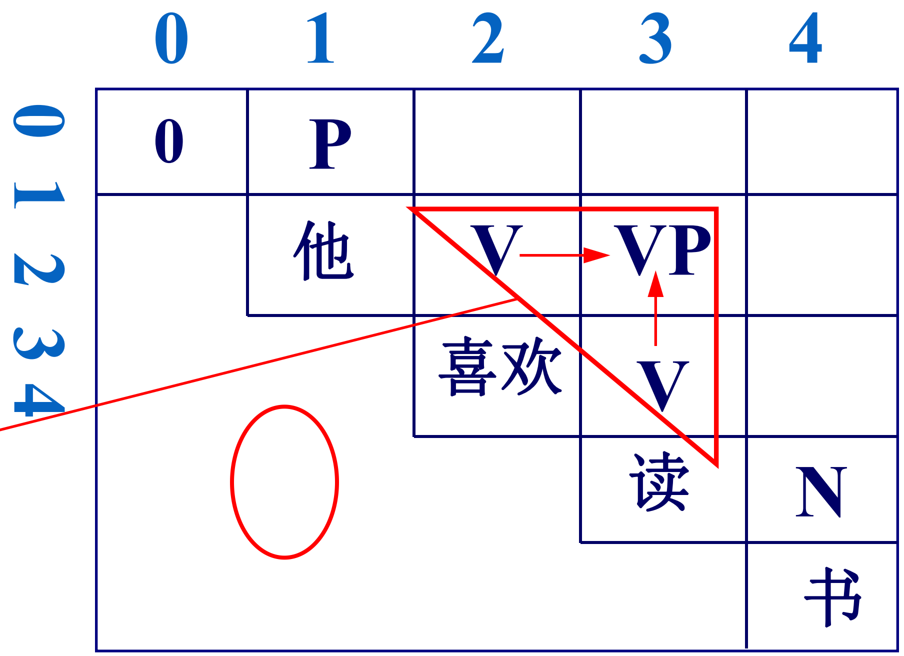
CYK算法

(1) 汉语分词和词性标注以后：

他/P 喜欢/V 读/V 书/N $\qquad n=4$

(2) 构造识别矩阵：

(3) 执行分析过程。



(1) S → P VP

(2) VP → V V

(3) VP → VP N

CYK算法

(1) 汉语分词和词性标注以后：

他/P 喜欢/V 读/V 书/N　　　n=4

(2) 构造识别矩阵：

(3) 执行分析过程。

	0	1	2	3	4
0	0	P	P		
1		他	V	VP	
2			喜欢	V	N
3				读	N
4					书

(1) S $\rightarrow$ P VP

(2) VP $\rightarrow$ V V

(3) VP $\rightarrow$ VP N

CYK算法

(1) 汉语分词和词性标注以后：

他/P 喜欢/V 读/V 书/N　　　　n=4

(2) 构造识别矩阵：

(3) 执行分析过程。

	0	1	2	3	4
0	0	P	P	S	?
1		他	V	VP	
2			喜欢	V	N
3				读	N
4					书

(1) S → P VP

(2) VP → V V

(3) VP → VP N

CYK算法

(1) 汉语分词和词性标注以后：

他/P 喜欢/V 读/V 书/N　　　n=4

(2) 构造识别矩阵：

(3) 执行分析过程。

	0	1	2	3	4
0	0	P	P		
1		他	V	VP	VP
2			喜欢	V	N
3				读	N
4					书

(1) $S \rightarrow P\ \ VP$

(2) $VP \rightarrow V\ \ V$

(3) $VP \rightarrow VP\ \ N$

CYK算法

(1) 汉语分词和词性标注以后：

他/P 喜欢/V 读/V 书/N　　　n=4

(2) 构造识别矩阵：

(3) 执行分析过程。

	0	1	2	3	4
0	0	P	P	P	
1		他	V	VP	VP
2			喜欢	V	N
3				读	N
4					书

$$(1)\ S \rightarrow P\ \ VP$$

$$(2)\ VP \rightarrow V\ \ V$$

$$(3)\ VP \rightarrow VP\ \ N$$

CYK算法

(1) 汉语分词和词性标注以后：

他/P 喜欢/V 读/V 书/N　　　n=4

(2) 构造识别矩阵：

(3) 执行分析过程。

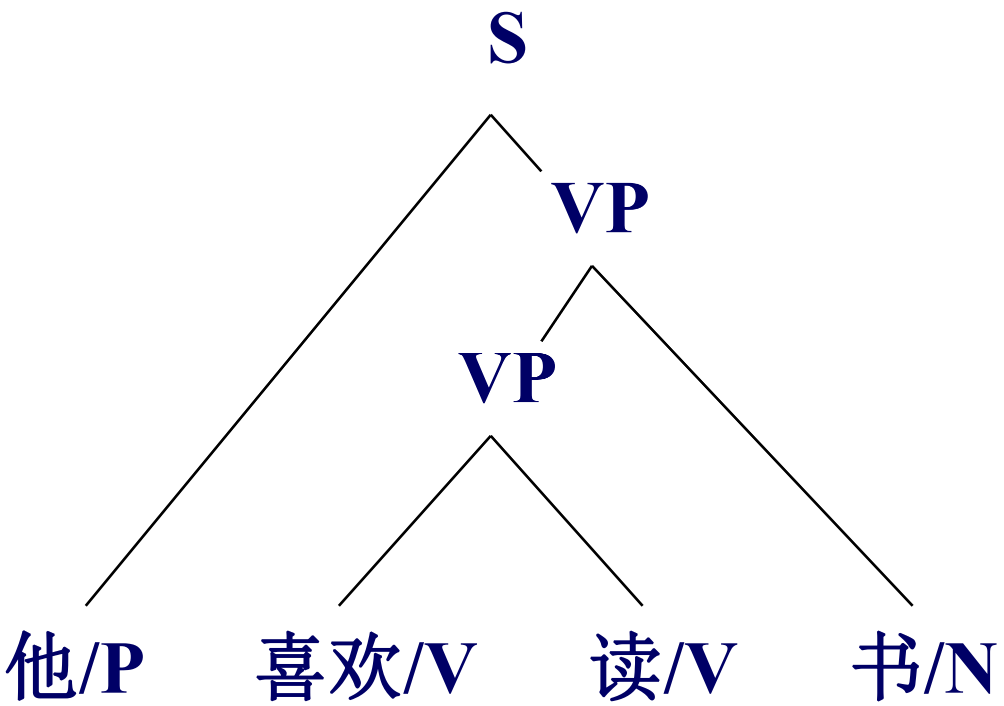
	0	1	2	3	4
0	0	P	P	P	S
1		他	V	VP	VP
2			喜欢	V	N
3				读	N
4					书

(1) S → P VP

(2) VP → V V

(3) VP → VP N

CYK算法



CYK算法

- CYK 算法的评价
  - 优点
    - 简单易行，执行效率高
  - 弱点
    - 必须对文法进行范式化处理
    - 无法区分歧义

CKY算法

- for all words w_i: // terminal rules
    - for all rules A→w_i: add new chart entry A at span [i, i]
- for length = 1 to sentence length n // non-terminal rules
    - for start = 1 to n − (length − 1)
      end = start + length − 1
    - for middle = start to end − 1: // binary rules
      for all non-terminals X in [start, middle]:
      for all non-terminals Y in [middle + 1, end]:
      for all rules A→X Y :
      add new chart entry A at position [start, end]
    - for all non-terminals X in [start, end]: // unary rules
      for all rules A → X:
      add new chart entry A at position [start, end]

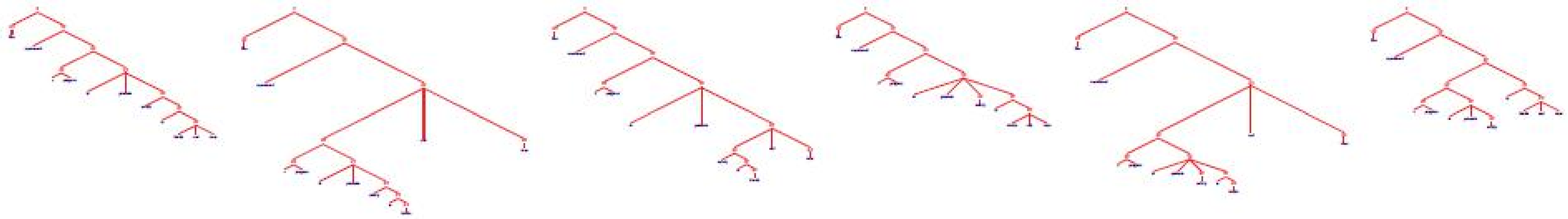
Why is parsing hard?

- **输入：**

  She announced a program to promote safety in trucks and vans

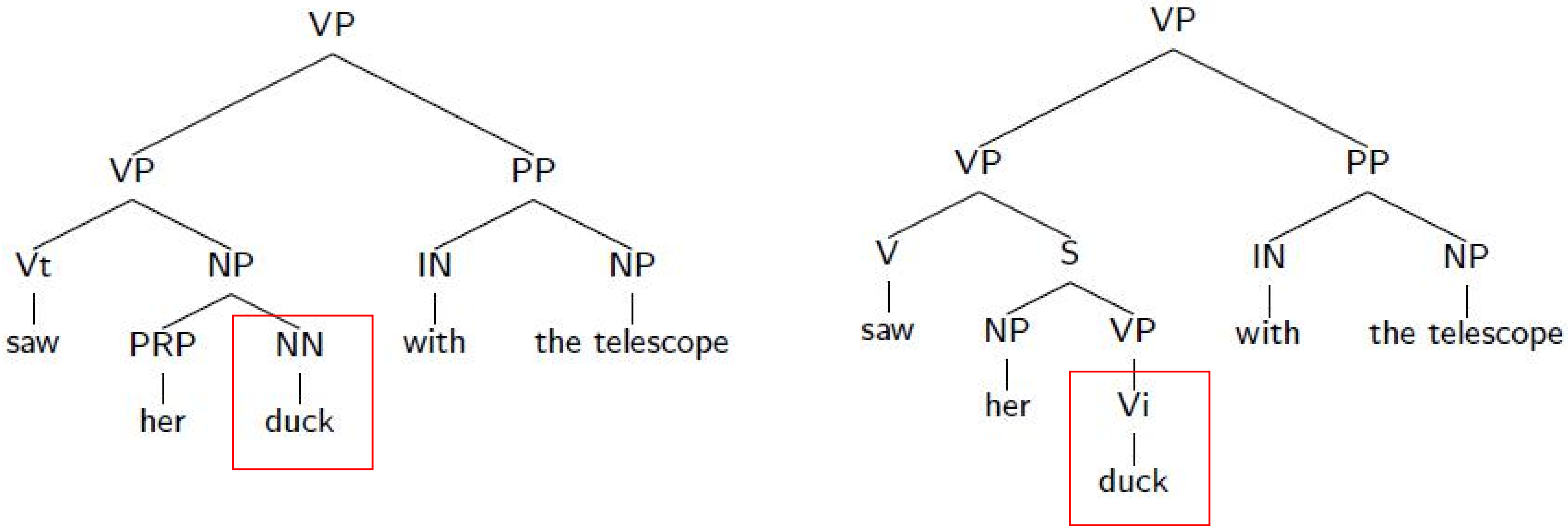
- **可能的输出：**

  - 还



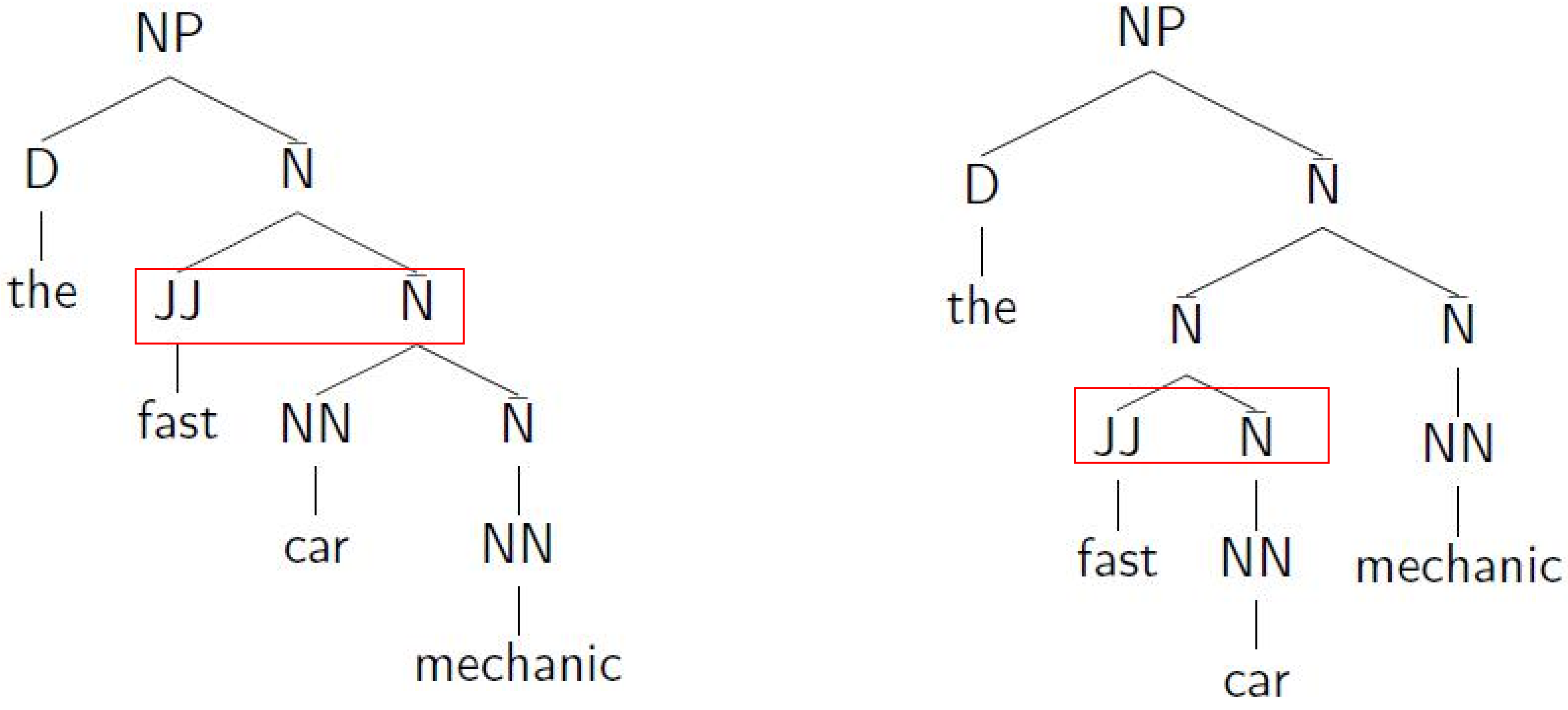
几种常见的歧义

- 词性歧义：
    - NN → duck
    - Vi → duck



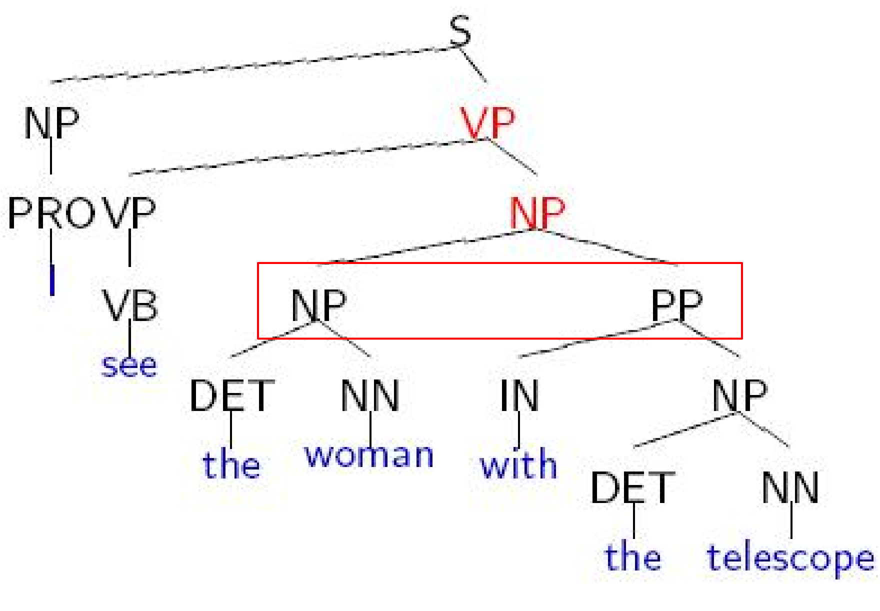
几种常见的歧义

- 名词修饰语歧义：



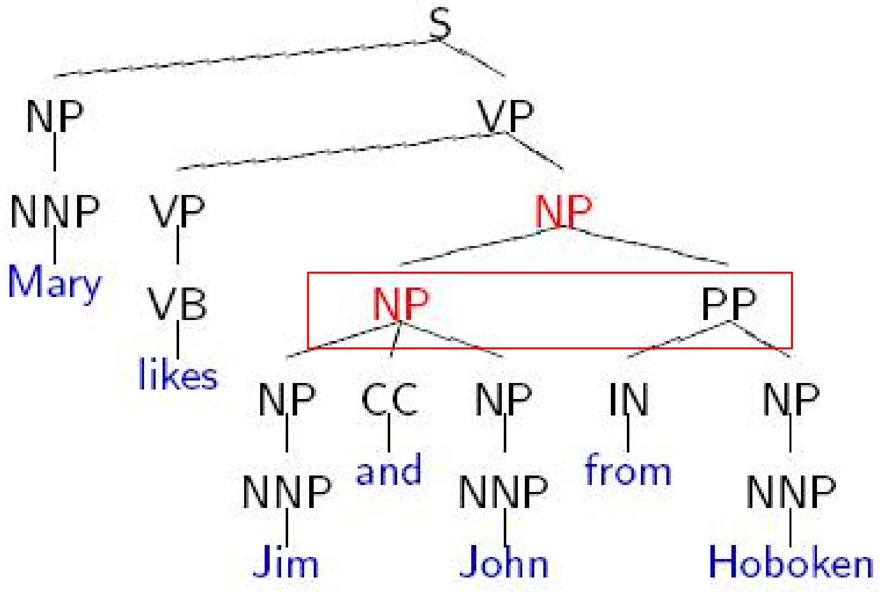
几种常见的歧义

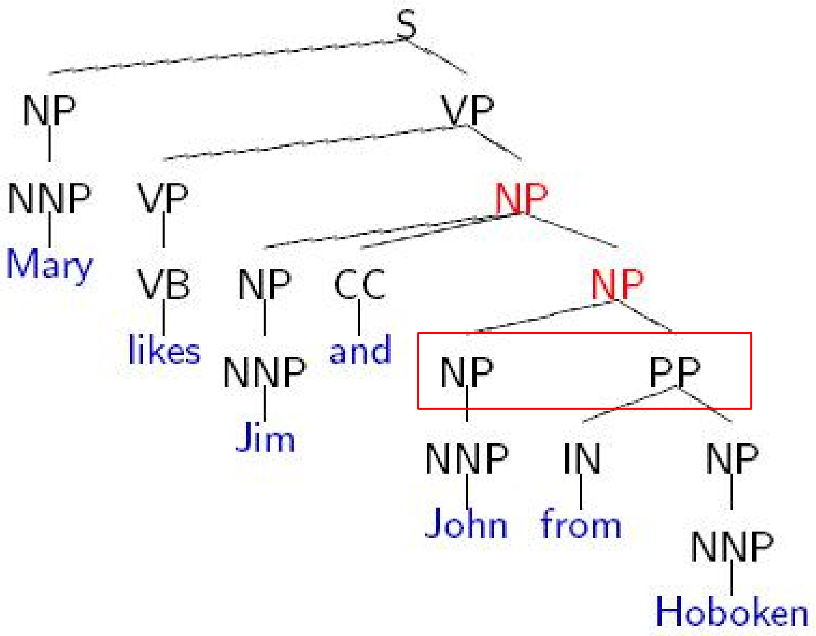
- 介词短语修饰语歧义：Who has the telescope?



几种常见的歧义

- 边界歧义：Is Jim also from Hoboken?





概率上下文无关文法

- Probabilistic context-free grammars (PCFGs)或– Stochastic context-free grammars (SCFGs)
- G = (T, N, S, R, P)
  - T：终结符（ terminal symbols）集合
  - N：非终结符（ nonterminal symbols）集合
  - S：开始符号，表示句子
  - R：重写规则（或产生式），具有形式 $X \rightarrow \gamma$，$X \in N$ 并且$\gamma \in (N \cup T)^*$
  - P：概率函数，为每个重写规则赋予一个概率值
    - P: R → [0,1]

$$\forall X \in N, \sum_{X \rightarrow \gamma \in R} P(X \rightarrow \gamma) = 1$$

$$P(\gamma) = 1$$

一个简单的PCFG例子

S → NP VP 1.0

VP → Vi 0.4

VP → Vt NP 0.4

VP → VP PP 0.2

NP → DT NN 0.3

NP → NP PP 0.7

PP → P NP 1.0

Vi → sleeps 1.0
Vt → saw 1.0
NN → man 0.7
NN → woman 0.2
NN → telescope 0.1
DT → the 1.0
IN → with 0.5
IN → in 0.5

- 设句法树t使用的规则有：$\alpha_1 \rightarrow \beta_1, \ldots, \alpha_n \rightarrow \beta_n$，规则$\alpha_i \rightarrow \beta_i$的概率为$q(\alpha_i \rightarrow \beta_i)$
- 则句法树t的概率为：

$$p(t) = \prod_{i=1}^{n} q(\alpha_i \rightarrow \beta_i)$$

PCFG的特性

- 为CFG规则下的每一棵句法导出树赋予一个概率
- 对于句子s和其可能的句法导出树集合Γ(s)，PCFG为Γ(s)中的每棵树t赋予一个概率 p(t)，即得到候选树按照概率的排序
- 句子s最可能的句法树为:

$$\arg\max_{t \in \Gamma(s)} p(t)$$

PCFG的特性

- 为CFG规则下的每一棵句法导出树赋予一个概率

两个问题

- 如何得到PCFG?
  - 句法规则学习
- 如何从多个候选树中找出一个概率最大的树?
  - 基于PCFG的句法分析

从treebank中学习语法

- 给定句法树样本 (树库，treebank)
- 从训练语料中统计观测到的重写规则，将其作为CFGs语法
- 并从中估计每个重写规则 $\alpha \rightarrow \beta$ 的概率：

$$q_{ML}(\alpha \rightarrow \beta) = \frac{Count(\alpha \rightarrow \beta)}{Count(\alpha)}$$

- 假设训练数据由其背后的PCFGs生成，则如果训练数据规模足够大，极大似然估计法得到的PCFG应该收敛于真实的PCFGs的概率分布

Penn treebank

- Penn treebank：标注了句法树的英文句子
  - 由the University of Pennsylvania构建
  - 标注了the Wall Street Journal的真实文本
  - 40,000个英文句子，约100万个词

```
( (S (NP-SBJ The move)
     (VP followed
         (NP (NP a round)
             (PP of
                 (NP (NP similar increases)
                     (PP by
                         (NP other lenders))
                     (PP against
                         (NP Arizona real estate loans)))))
         ,
         (S-ADV (NP-SBJ *)
                (VP reflecting
                    (NP (NP a continuing decline)
                        (PP-LOC in
                                (NP that market))))))
     .))
```

Penn treebank

- 包括多种语言：
  - German
  - French
  - Spanish
  - Arabic
  - **Chinese:** Chinese Penn Treebank (CTB)

- 树库提供了非常多有用的信息：
  - 可重用性
    - 可以基于此得到不同的词性标注器、句法分析器等
    - 语言学的重要资源
  - 大量的统计信息：频次、分布等
  - 提供了一种用于系统评价的标准数据集

Parsing with a PCFG

- 给定PCFG句法及句子 s，定义Γ(s)为s的候选句法树构成的集合

- 句法分析的目标：

$$\arg\max_{t \in \Gamma(s)} p(t)$$

PCFGs

- 一个概率上下文无关文法可以表示为一个五元组 G = (T, N, S, R, P)：
  - T：终结符（ terminal symbols）集合
  - N：非终结符（ nonterminal symbols）集合
  - S：开始符号，表示句子
  - R：重写规则（或产生式），具有形式 $X \to \gamma$，$X \in N$ 并且$\gamma \in (N \cup T)^*$
  - P：概率函数，为每个重写规则赋予一个概率值
    - P: R $\to$ [0,1]

$$\forall X \in N, \sum_{X \to \gamma \in R} P(X \to \gamma) = 1$$

$(\gamma) = 1$

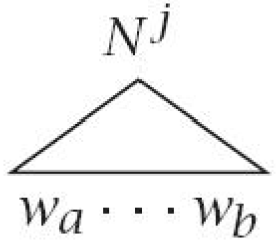
A simple PCFG

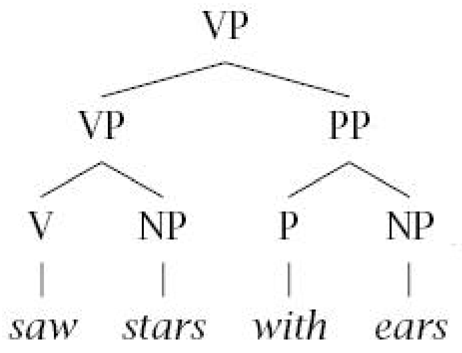
- S→*NP VP 1.0*
- PP → *P NP 1.0*
- VP → *V NP 0.7*
- VP → *VP PP 0.3*
- V → *saw 1.0*
- P → *with 1.0*

- *NP → NP PP 0.4*
- *NP → astronomers 0.1*
- *NP → saw 0.04*
- *NP → ears 0.18*
- *NP → stars 0.18*
- *NP → telescopes 0.1*

PCFG notation

- G：PCFG语法
- L：G生成的或G能接受的语言
- t：句法树
- $\{N^1,\ldots, N^n\}$：非终结符集合（N^1开始符号）
- *$\{w^1,\ldots, w^V\}$：终结符集合*
- $\{w_1,\ldots,w_m\}$：要处理的句子
- N^j_{pq}：管辖位置p到q的词串的非终结符N^j
- $\alpha(p, q, N^j)$：外向概率
- $\beta(p, q, N^j)$：内向概率





PCFG的假设

- 1. 位置不变性：

$$\forall k, P\left(N^j_{k(k+c)} \to \zeta\right)$$

- 2. 上下文无关：

$$P\left(N^j_{kl} \to \zeta \mid words\, outside\, w_k \ldots w_l\right) = P\left(N^j_{kl} \to \zeta\right)$$

- 3. 祖先节点无关：

$$P\left(N^j_{kl} \to \zeta \mid ancestor nodes\, of\, N^j_{kl}\right) = P\left(N^j_{kl} \to \zeta\right)$$

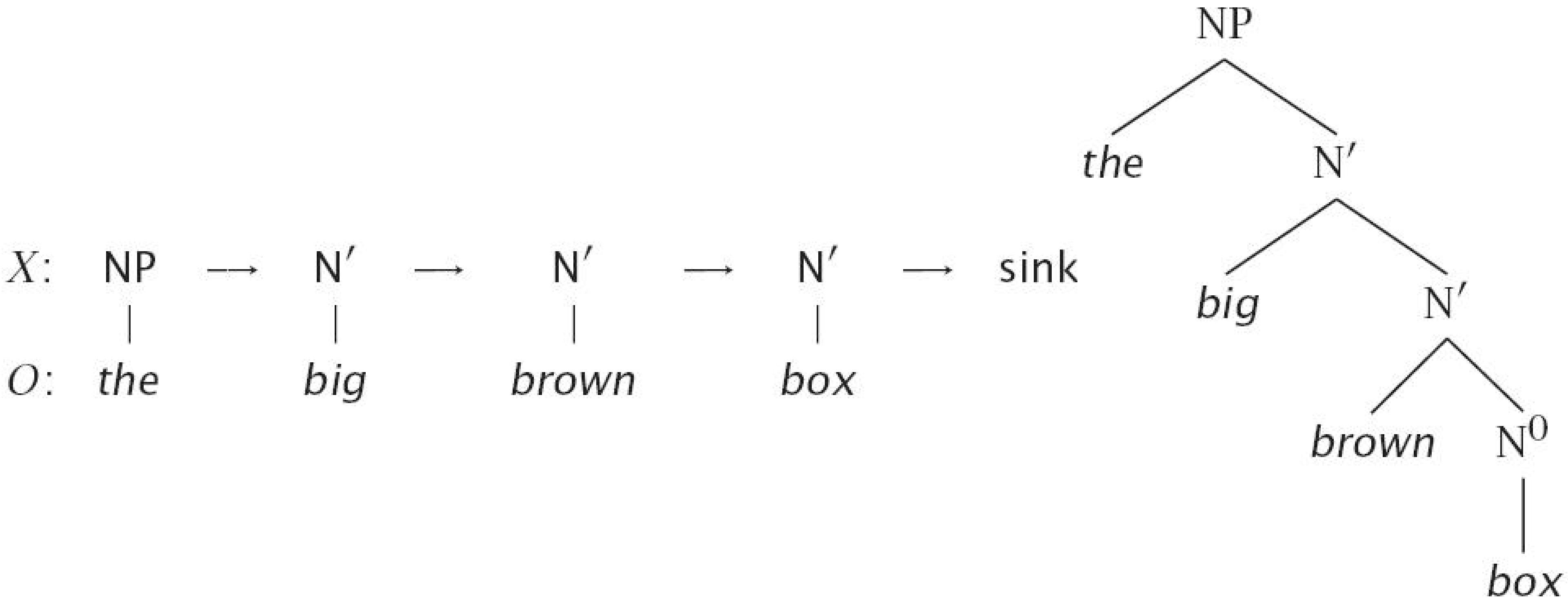
PCFG 参数

- CNF PCFG的句法规则：
  - $N^i \rightarrow N^j N^k$
  - $N^i \rightarrow w^j$
- CNF PCFG的参数：
  - $P(N^i \rightarrow N^j N^k)$: A n^3 matrix of parameters
  - $P(N^i \rightarrow w^j)$: An nV matrix of parameters
- 满足： $j=1, \ldots, n$,

$$\sum_{r,s} P\left(N^j \to N^r N^s\right) + \sum_k P\left(N^j \to w^k\right) = 1$$

HMMs与PCFGs的比较

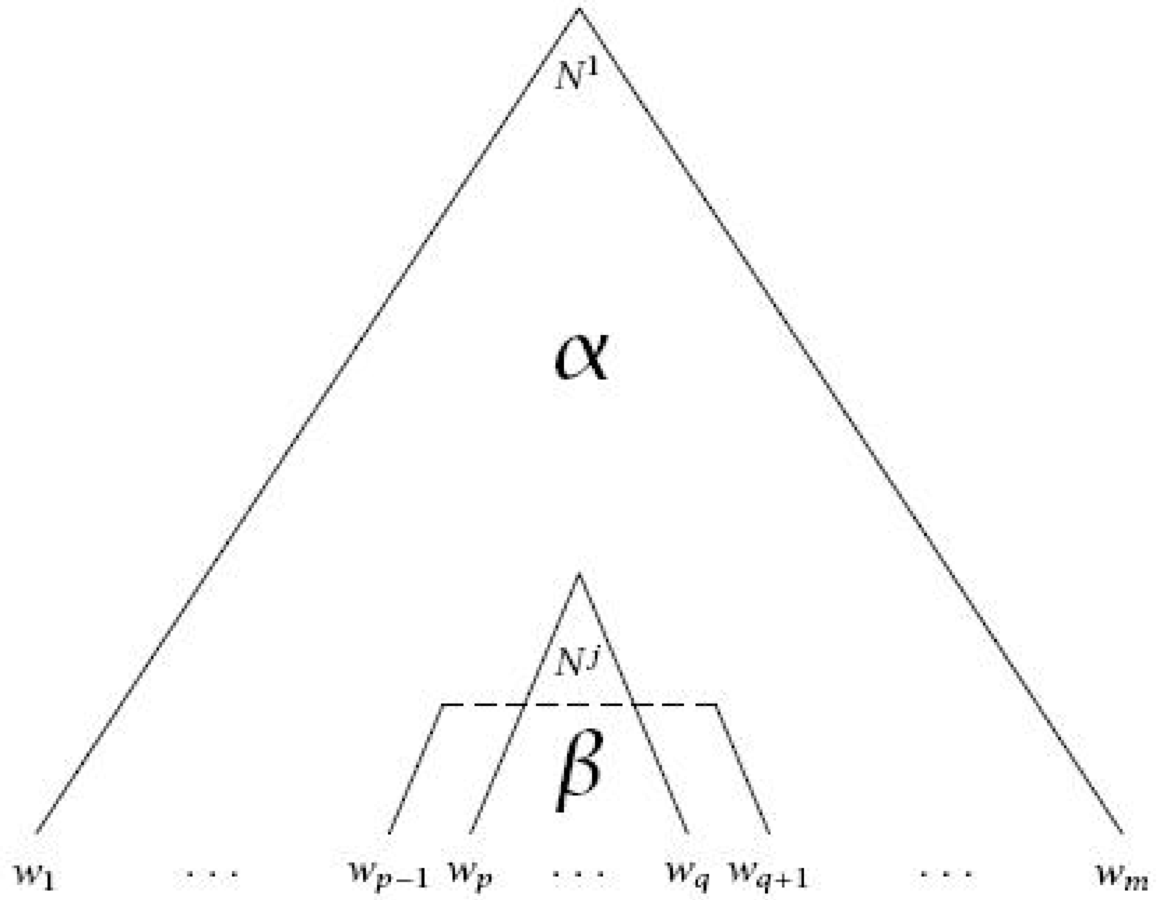
- *HMM:* **Probabilistic Regular Grammar**
    - $N^i \rightarrow w^j N^k$
    - $N^i \rightarrow w^j$
    - Start state, N^1



向内和向外概率

- HMM中定义了前向、后向概率：
  - Forwards = α_i (t)=$P(w_{1(t-1)}, X_t=i)$
  - Backwards = β_i(t)=P(w_{tT}|X_t=i)
- 同理，定义PCFG中的向外、向内概率：
  - Outside = $\alpha(p, q, N^j)=P(w_{1(p-1)}, N^j_{pq}, w_{(q+1)m}|G)$
  - Inside = $\beta(p, q , N^j)=P(w_{pq}|N^j_{pq}, G)$

向内和向外



$$\alpha_j(p, q)=P(w_{1(p-1)}, N^j_{pq}, w_{(q+1)m} \mid G)$$
$$\beta_j(p, q)=P(w_{pq} \mid N^j_{pq}, G)$$

PCFGs的三个问题

- 正如HMM的三个问题一样，PCFGs的三个基本问题如下：
  - 计算句子的概率：$P(w_{1m}|G)$
  - 为句子找到最优句法树：$\text{argmax}_t\ P(t|w_{1m};G)$
  - 参数学习：求解使得$P(w_{1m}|G)$ 最大的句法G

Problem2: Parsing

- 采用类似于Viterbi算法一样的思路，为句子找到最优句法树
- HMM：定义变量$\delta_j(t)$记录在t时刻到达状态j的 最优路径对应的概率（所有可能路径的概率的最大值）
- PCFG：定义变量π(i, j, X)记录由非终结符X推导出子串$w_i,\ldots w_j$的最大概率 树X_{ij}对应的概率（所有可能的导出结构的概率的最大值）

Problem2: Parsing

- 定义动态规划表：

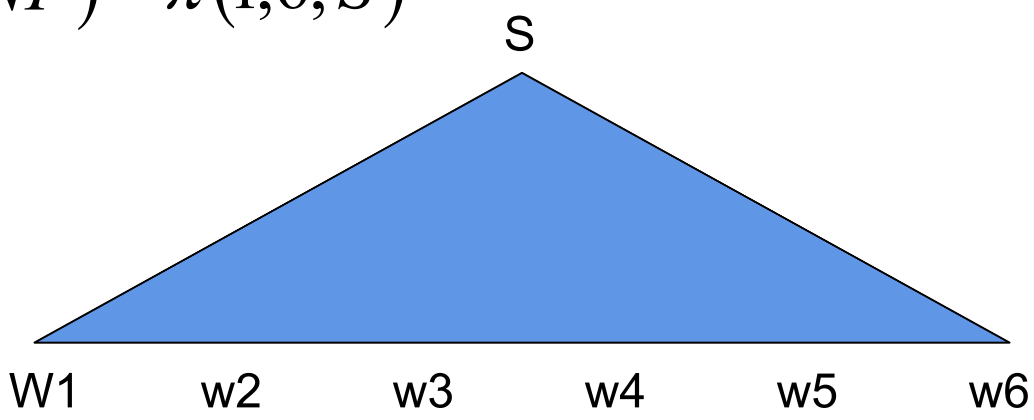
  π(i, j, X) =由非终结符X推导出子串$w_i,\ldots w_j$的最大概率

  = 子树X_{pq}最大的向内概率

- 目标是计算：

$$\max_{t\in\Gamma(s)} p(t) = \pi(1, n, S)$$

$$\pi(2,5,NP) \quad \pi(1,6,S)$$



A Dynamic Programming Algorithm

- Base case definition: for all i=1, …, n, for X∊N

$$\pi(i,i,X) = q(X \rightarrow \omega_i)$$

  - Note define $P(X \rightarrow w_i) = 0$ if $P(X \rightarrow w_i)$ is not in the grammar

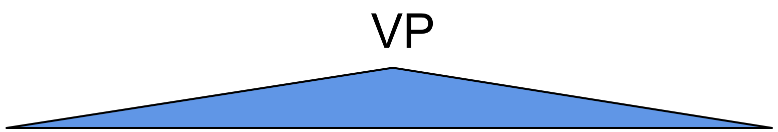
- Recursive definition: for all i=1…n-1, j=(i+1)…n, for X∊N

$$\pi(i,j,X) = \max_{\substack{X \rightarrow YZ \in R, \\ s \in \{i \ldots (j-1)\}}} \left(q(X \rightarrow YZ) \times \pi(i,s,Y) \times \pi(s+1,j,Z)\right)$$

An Example

$$\pi(i,j,X) = \max_{\substack{X \rightarrow YZ \in R,\\ s \in \{i \ldots (j-1)\}}} \left(q(X \rightarrow YZ) \times \pi(i,s,Y) \times \pi(s+1,j,Z) \right)$$

To compute: π(3,8,VP)



The	dog	saw	the	man	with	the	telescope
1	2	3	4	5	6	7	8

Suppose: q(VP→ V NP)=0.7, q(VP→ VP PP)=0.3

q(VP→ V NP) X π(3,3,V) X π(4,8,NP)
q(VP→ V NP) X π(3,4,V) X π(5,8,NP)
…
q(VP→ V NP) X π(3,7,V) X π(8,8,NP)

q(VP→ VP PP) X π(3,3,VP) X π(4,8,PP)
…
q(VP→ VP PP) X π(3,7,VP) X π(8,8,PP)

Exercise

- Consider the example sentence: the dog saw the man with the telescope
- Assume that we have π values such that
  - $\pi(3,3,V)\times\pi(4,8,NP)=0.01$, $\pi(3,5,VP)\times\pi(6,8,PP)=0.1$, $\pi(3,6,VP)\times\pi(7,8,NP)=0.1$, $\pi(3,7,VP)\times\pi(8,8,N)=0.01$
- For all other values of $s\in\{3\ldots7\}$ and $X\in N, Y\in N$, assume that $\pi(3,s,Y)\times\pi(s+1,8,X)=0$
- Also assume that the PCFG has the following parameters
  - $q(VP\rightarrow V\ NP)=0.2$
  - $q(VP\rightarrow VP\ PP)=0.5$
  - $q(VP\rightarrow VP\ NP)=0.2$
  - $q(VP\rightarrow VP\ N)=0.1$
- What is the value for $\pi(3,8,VP)$?

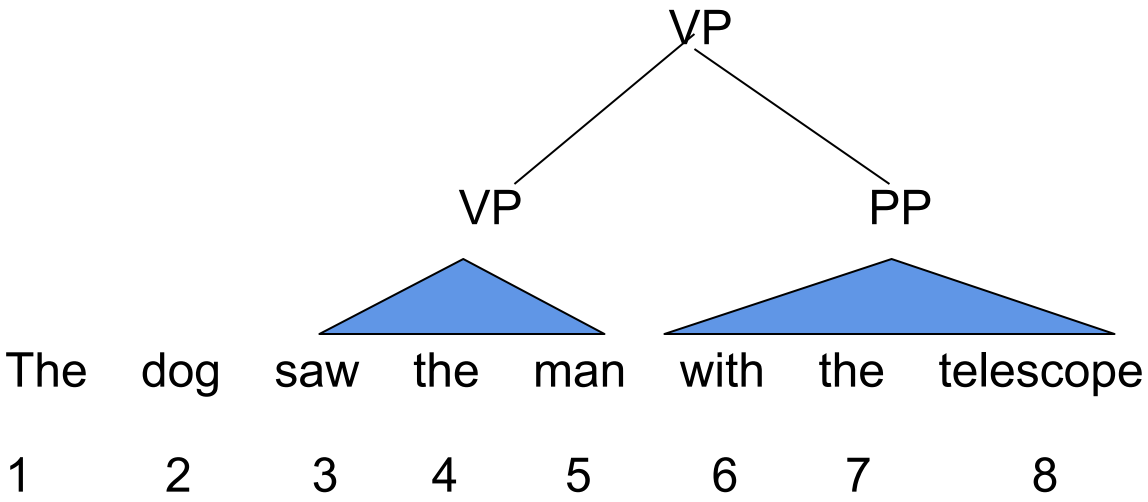
Exercise

- $\pi(3,8,VP) = 0.05$

$= q(VP \rightarrow VP\ PP)$

$\times\ \pi(3,5,VP)$

$\times \pi(6,8,PP)$



The Full Dynamic Programming Algorithm

- Input: a sentence s=x1….xn, a PCFG G=(N, ∑, S, R, q)
- Initializatior

$$\pi(i,i,X) = \begin{cases} q(X \rightarrow x_i) & \text{if } X \rightarrow x_i \in R \\ 0 & \text{otherwise} \end{cases}$$

- Algorithm:
  - For $l = 1 \ldots (n-1)$
    - For $i = 1 \ldots (n-l)$
      - Set $j = i + l$
      - For all $X \in N$, calculate

$$\pi(i,j,X) = \max_{\substack{X \rightarrow YZ \in R, \\ s \in \{i \ldots (j-1)\}}} \left(q(X \rightarrow YZ) \times \pi(i,s,Y) \times \pi(s+1,j,Z) \right)$$

and

$$bp(i,j,X) = \arg \max_{\substack{X \rightarrow YZ \in R, \\ s \in \{i \ldots (j-1)\}}} \left(q(X \rightarrow YZ) \times \pi(i,s,Y) \times \pi(s+1,j,Z) \right)$$

Problems with the Inside-Outside algorithm

- Slow
  - Each iteration is *O*(m^3n^3), *where m=sum_{i=1, w}, and n is the* number of nonterminals in the grammar.
- Local maxima are much more of a problem
  - Charniak reports that on each trial a different local maximum was found

Weakness of PCFG

- *Independence assumption* too strong
- Non-terminal rule applications do not use *lexical information*
- Not sufficiently sensitive to *structural differences* beyond parent/child node relationships

Some features of PCFGs

- Reasons to use a PCFG, and some idea of their limitations:
    - Partial solution for grammar ambiguity: a PCFG gives some idea of the plausibility of a sentence
    - But not a very good idea, as not lexicalized
    - Better for grammar induction (Gold 1967)
    - Robustness (Admit everything with low probability)

Some features of PCFGs

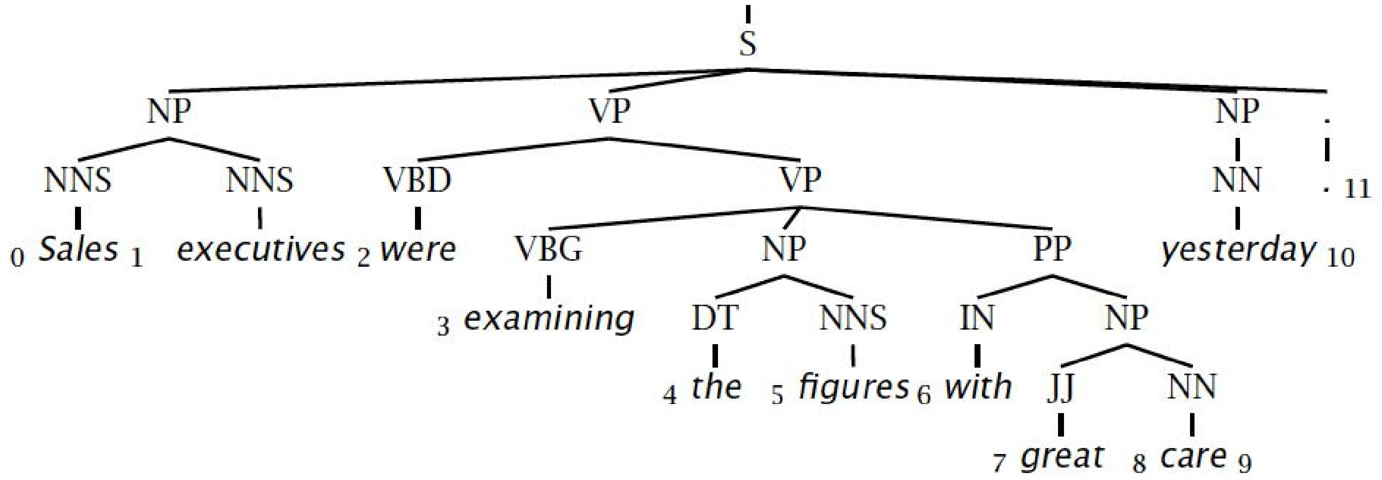
- Gives a probabilistic language model for English.
- In practice, a PCFG is a worse language model for English than a trigram model.
- Can hope to combine the strengths of a PCFG and a trigram model.
- PCFG encodes certain biases, e.g., that smaller trees are normally more probable.

Summary

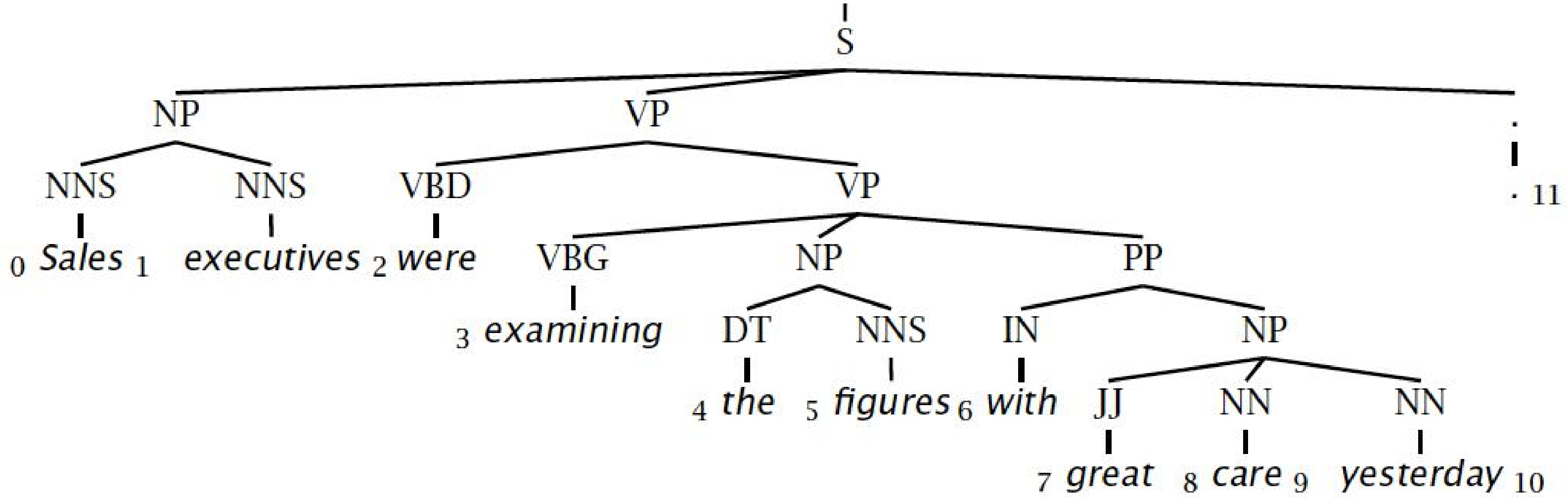
- PCFGs augments CFGs by including a probability for each rule in the grammar
- The probability for a parse tree is the product of probabilities for the rules in the tree
- To build a PCFG-parsed parser:
  - 1. Learn a PCFG from a treebank
  - 2. Given a test data sentence, use the CKY algorithm to compute the highest probability tree for the sentence under the PCFG

句法分析的评价

Gold standard brackets: **S-(0:11)**, **NP-(0:2)**, VP-(2:9), VP-(3:9), **NP-(4:6)**, PP-(6-9), NP-(7,9), NP-(9:10)



Candidate brackets: **S-(0:11)**, **NP-(0:2)**, VP-(2:10), VP-(3:10), **NP-(4:6)**, PP-(6-10), NP-(7,10)



句法分析的评价

标准结果：

S-(0:11), **NP-(0:2)**, VP-(2:9), VP-(3:9), **NP-(4:6)**, PP-(6-9), NP-(7,9), NP-(9:10)

系统输出结果：

S-(0:11), **NP-(0:2)**, VP-(2:10), VP-(3:10), **NP-(4:6)**, PP-(6-10), NP-(7,10)

Labeled Precision	3/7 = 42.9%
Labeled Recall	3/8 = 37.5%
F1	40.0%
Tagging Accuracy	11/11 = 100.0%

More Topics for Parser

- PCFG vs language model: lexicalized PCFG, head-driven PCFG
- Agenda-based/history-based PCFG
- Dependency parser
- Unified approach for POS tagging and parsing