Natural Language Processing
#L6

# 句法分析

袁彩霞

yuancx@bupt.edu.cn

人工智能学院 智能科学与技术中心

# The path so far

- 最初，将语言视为由词构成的序列
  - n-gram（语言模型）
- 接着，引入词的句法属性
  - part-of-speech tagging（词性标注）
- 现在，考察词之间的句法关系
  - Syntactic parsing（句法分析）

# 语法和语义

- 大部分情况下，一个不合乎语法的句子也可以被理解
  - The boy quickly in the house the ball found
  - 看清楚路先兄弟
- 合乎语法的句子也可能无法理解
  - Are gyre and gimble in the wabe? (non-sense words)
  - 不会做饭的裁缝不是一个好司机
- 但句法规则可以传递如下信息：
  - 句子的语法
  - 词的顺序
  - 短语成分
  - 句子的层次结构
  - 句法关系，例如主语、宾语
  - ……

# 句法分析的应用

- 句法分析被广泛且成功地应用到NLP的各个方面：
  - Meaning Representation [Jeffrey Flanigan, et al., ACL 2014]
  - High precision question answering [Pasca and Harabagiu, SIGIR 2011]
  - Source sentence analysis for machine translation [Xu et al., 2009]
  - Syntactically based sentence compression [Lin and Wilbur, 2007]
  - Extracting opinions about products [Bloom et al., NAACL 2007]
  - Relation extraction systems [Fundel et al., *Bioinformatics* 2006]
  - Improved interaction in computer games [Gorniak and Roy, 2005]
  - Helping linguists find data [Resnik et al., BLS 2005]
  - Improving biological named entity finding [Finkel et al., JNLPBA 2004]

# Parsing on ACL 2019

- *30 papers:*
  - Learning Structured Natural Language Representations for Semantic Parsing
  - Neural AMR: Sequence-to-Sequence Models for Parsing and Generation
  - A* CCG Parsing with a Supertag and Dependency Factored Model
  - A Minimal Span-Based Neural Constituency Parser
  - ……

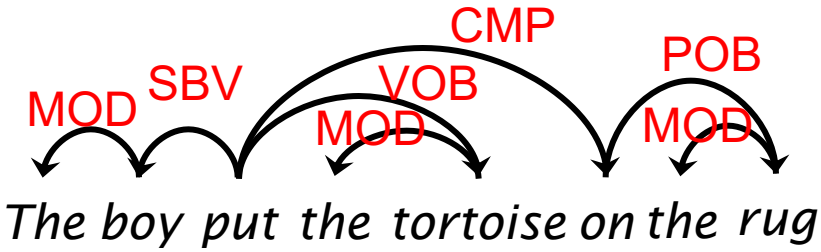
# 一个简单的句子

- I like the interesting lecture
- PRO VB DET JJ NN

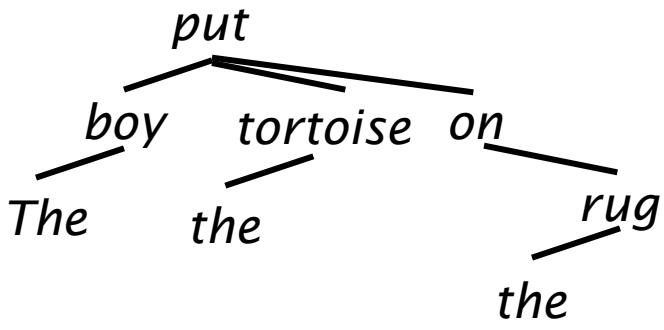
- 除了以上的词性标注，往往还关心：
  - 动词 *like*的主语是代词 *I*，说明 who is doing the liking
  - 动词 *like*的宾语是名词*lecture*，说明 what is being liked
  - 定冠词*the* 指出说明名词*lecture*
  - 形容词*interesting* 给出更多关于名词*lecture*的信息

# 两种不同的句法结构

- 依存结构（Dependency structure）：
  - 说明词和其它词之间的依赖关系（从属关系、支配关系等）

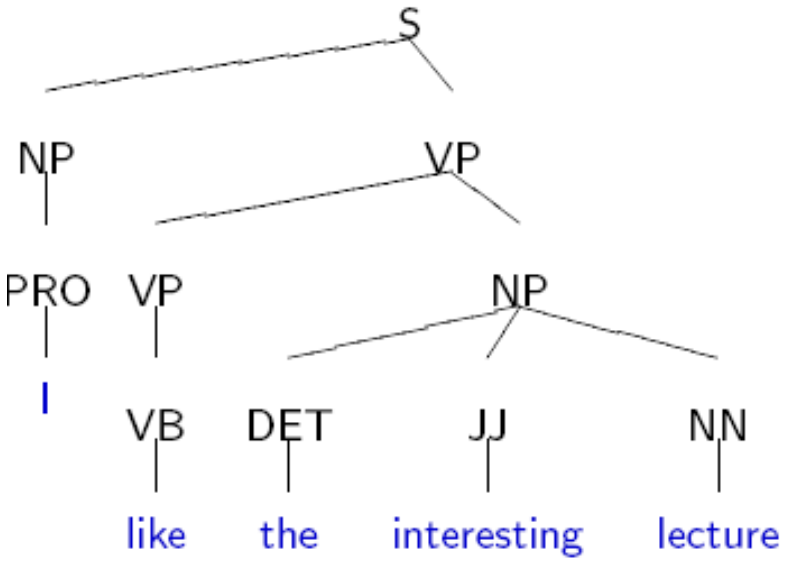



  - 可以表示为一个依存树（dependency tree）：



# 两种不同的句法结构

- 短语结构（Phrase structure）：
  - 将句子表示成嵌套的短语成分
- 父节点将子节点组合成较大的短语单元
  - 例如将DET JJ NN组合成NP



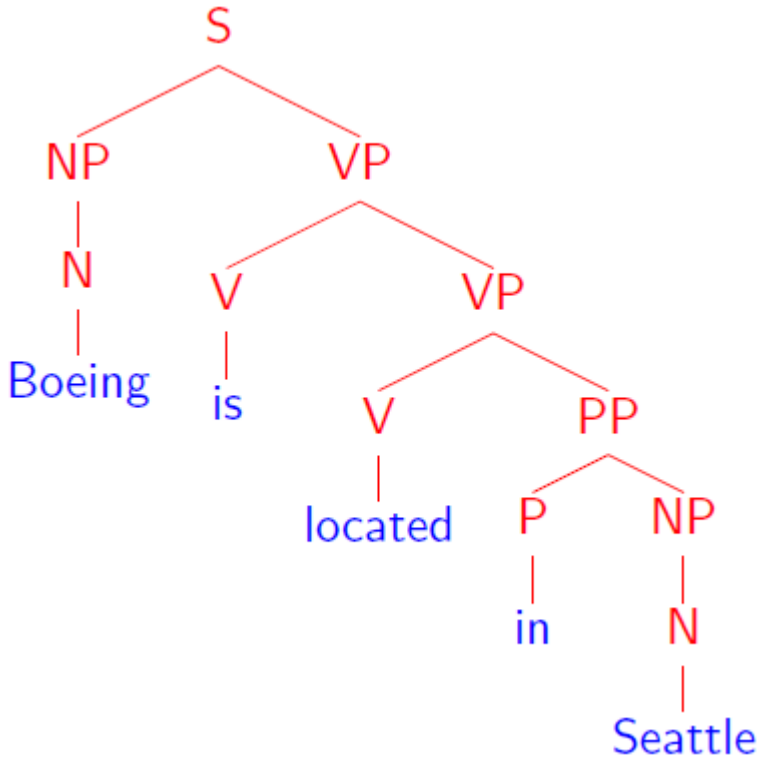
# 句法分析

- 这里的句法分析（Parsing）：专指短语结构分析

INPUT: 句子

Boeing is located in Seattle

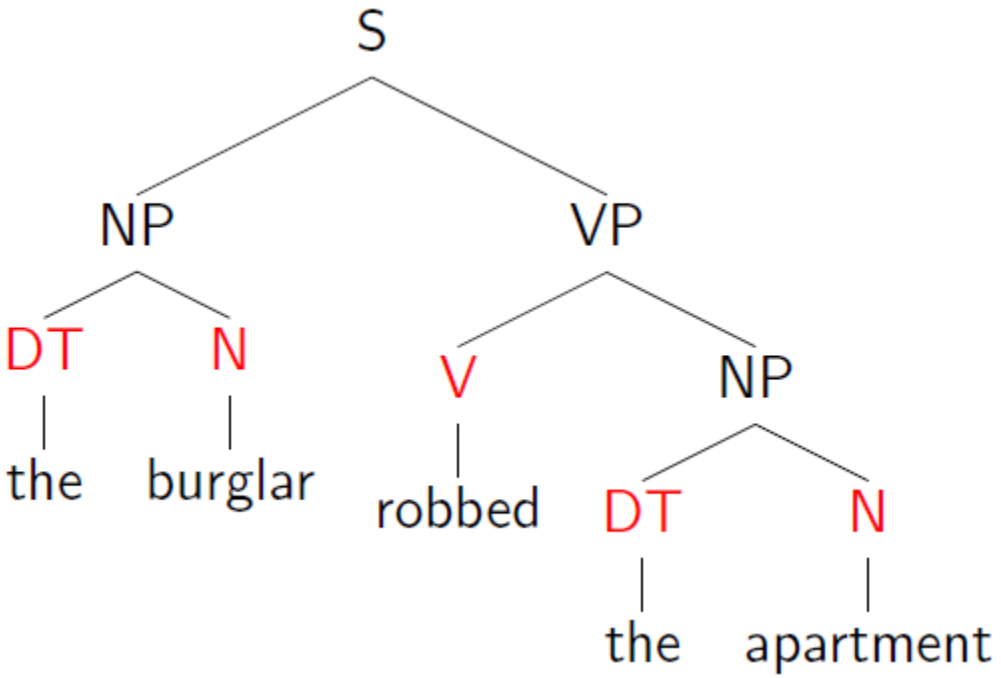
OUTPUT: 句法树



# 句法树中包含的信息

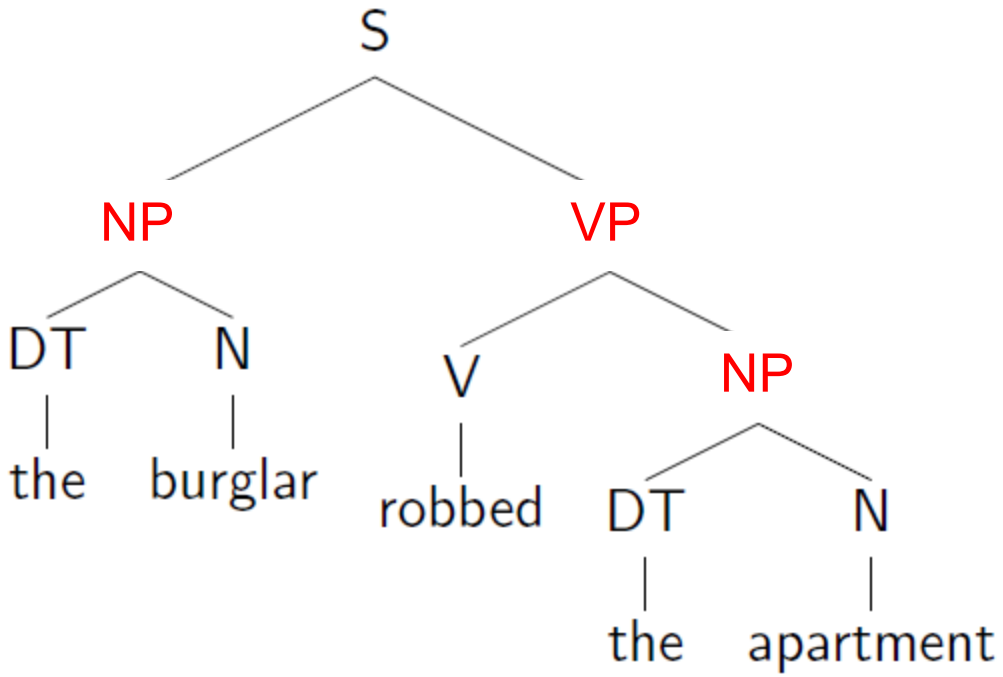
- (1) 词的词性类别

  (N = noun, V = verb, DT = determiner)



# 句法树中包含的信息

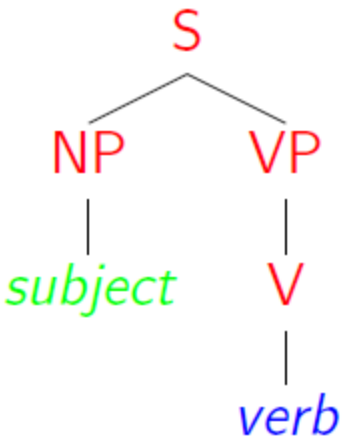
- (2) 短语

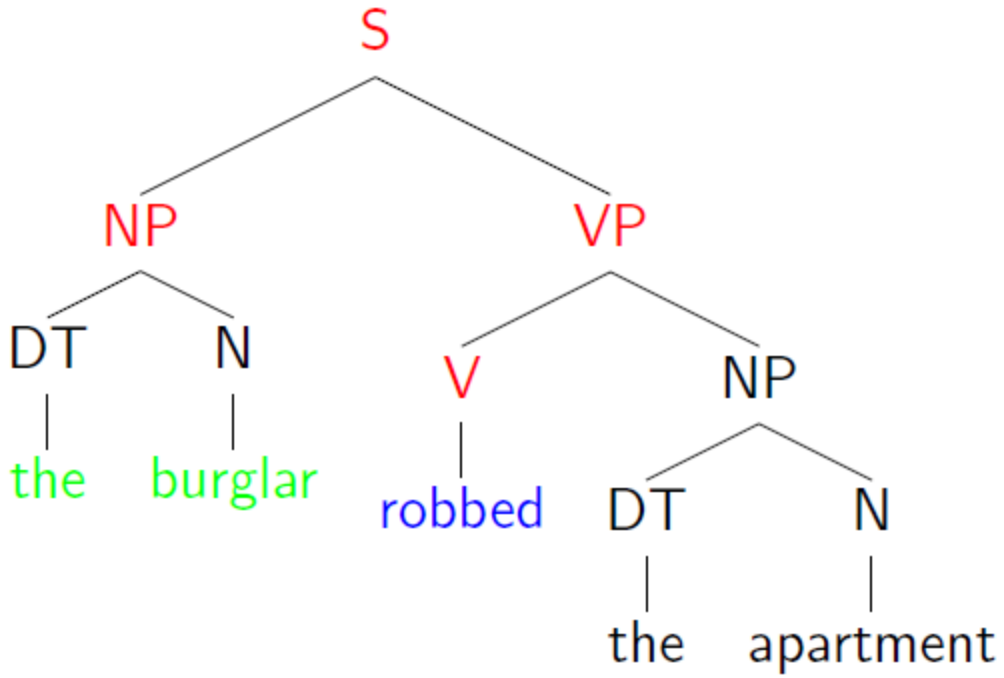



- 名词短语(NP): "the burglar", "the apartment"
- 动词短语 (VP): "robbed the apartment"
- 句子 (S): "the burglar robbed the apartment"

# 句法树中包含的信息

- 有用的关系：




- “the burglar”是“ robbed”的主语

# 句法分析

- 两个目的：
  - 判断输入句子是否合乎给定的语法
  - 识别句子各部分是如何依据语法规则组成合法句子，同时生成句法树
- 两个准备：
  - 语言的形式化描述（规定该语言中允许出现的结构）
  - 句法分析技术（根据语法来分析句子并确定其结构）

# 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 ∈ 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$s \in T^*$ 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

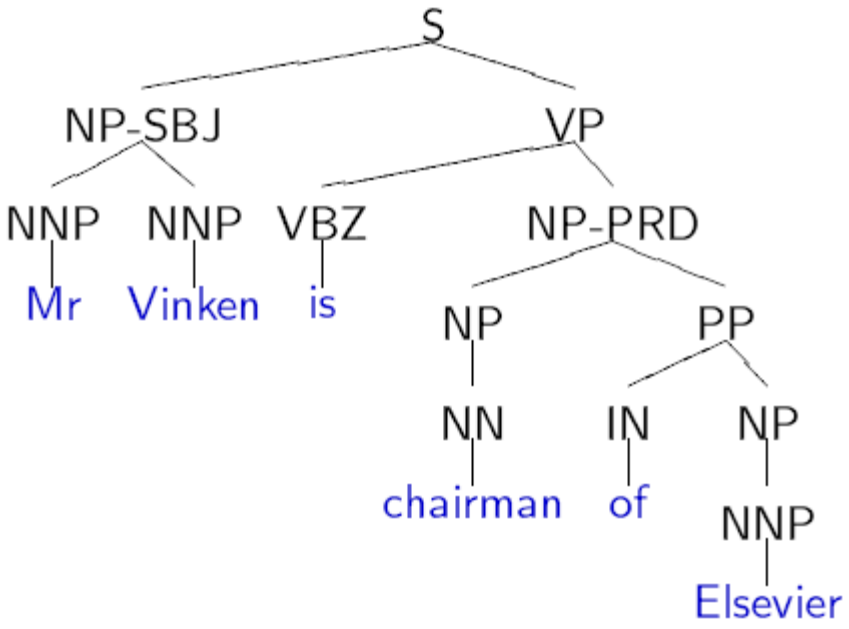
- T = {sleeps, saw, man, woman, telescope, the, with, in}
- N = {S, NP, VP, PP, DT, Vi, Vt, NN, IN}
- S = S
- R =

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

# 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

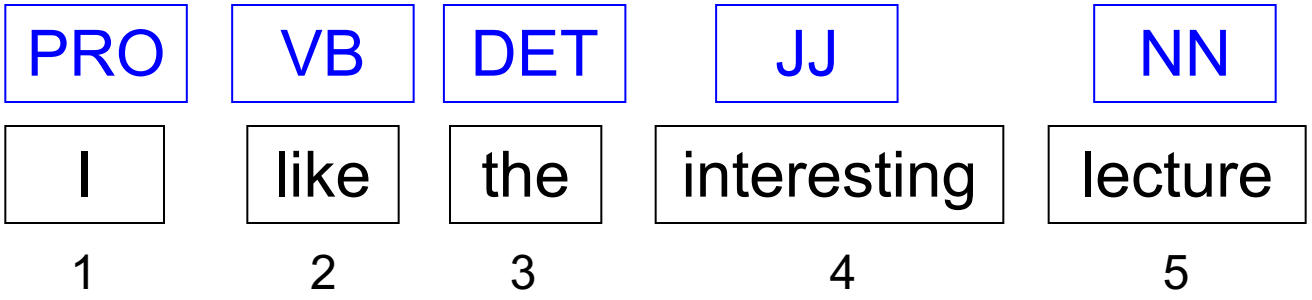
- 可以表述为一个搜索过程
  - 搜索空间：语法规则
  - 搜索过程：检查各种语法规则所有可能的组合方式
  - 搜索目的：最终找到一种组合，其中的语法规则能够生成一个用来表示句子结构的句法树
  - 搜索方向：自顶向下 vs 自底向上

# 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

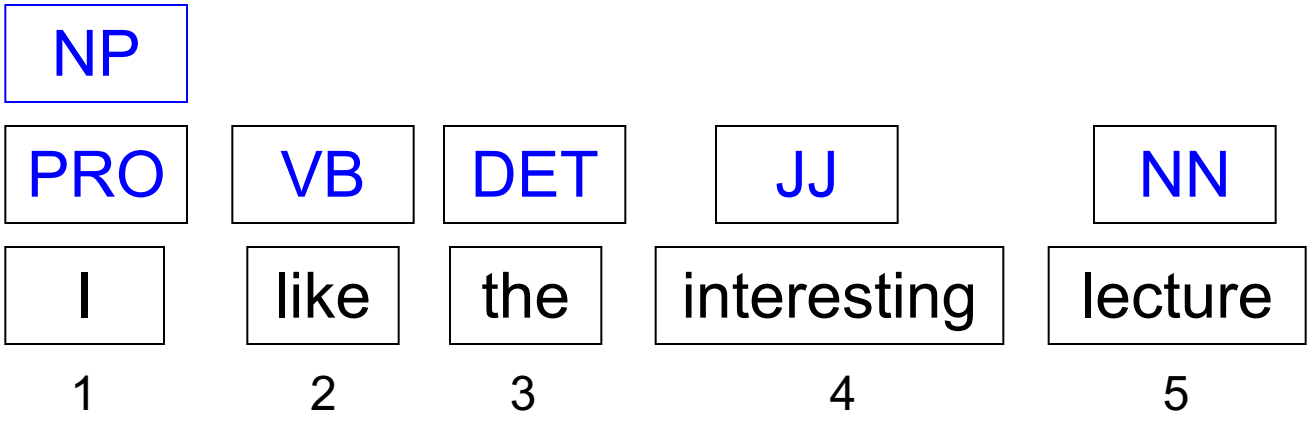
# Example

- 初始化：采用词初始化线图
- 首先应用终结符的导出规则：PRO→I, VB → like ……



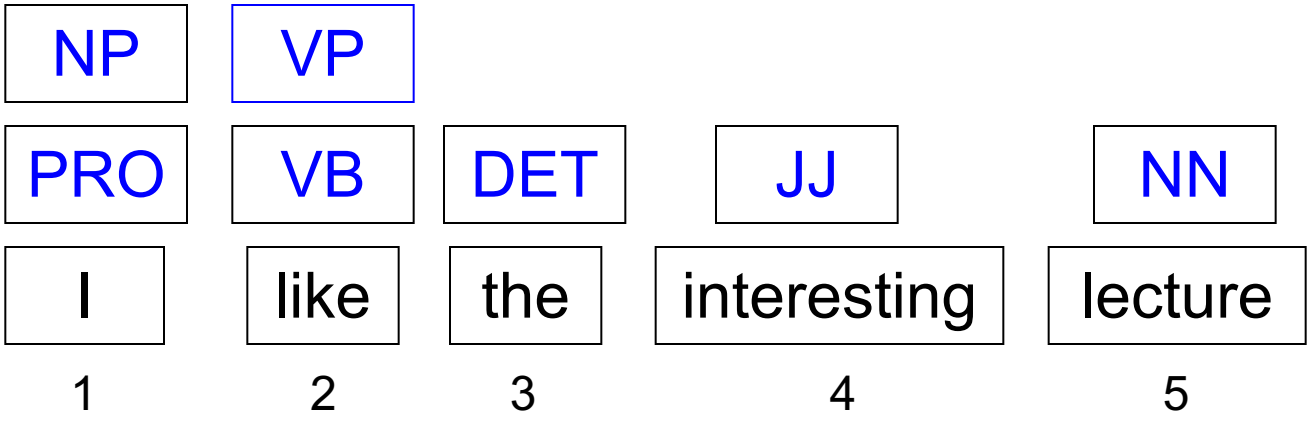
# Example

- 先为第一个词搜索可能的非终结符重写规则：?→PRO
  - NP→PRO
- 递归：继续为第一个词搜索可能的非终结符重写规则：？→NP
  - 没有可匹配的规则



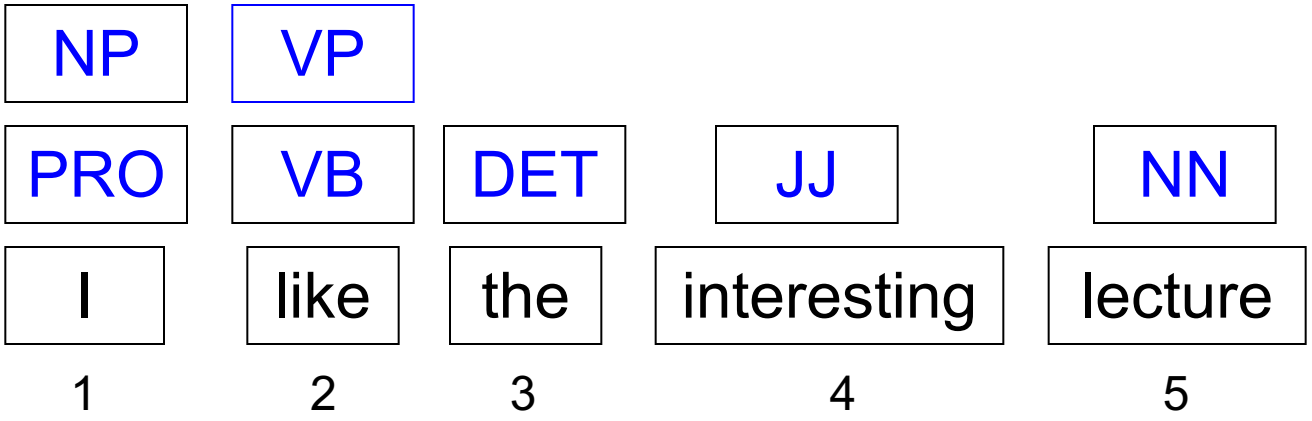
# Example

- 为第二个词搜索可能的非终结符重写规则：?→VB
  - VP→VB
- 递归：继续为第二个词搜索可能的非终结符重写规则：？→VP
  - 没有可匹配的规则



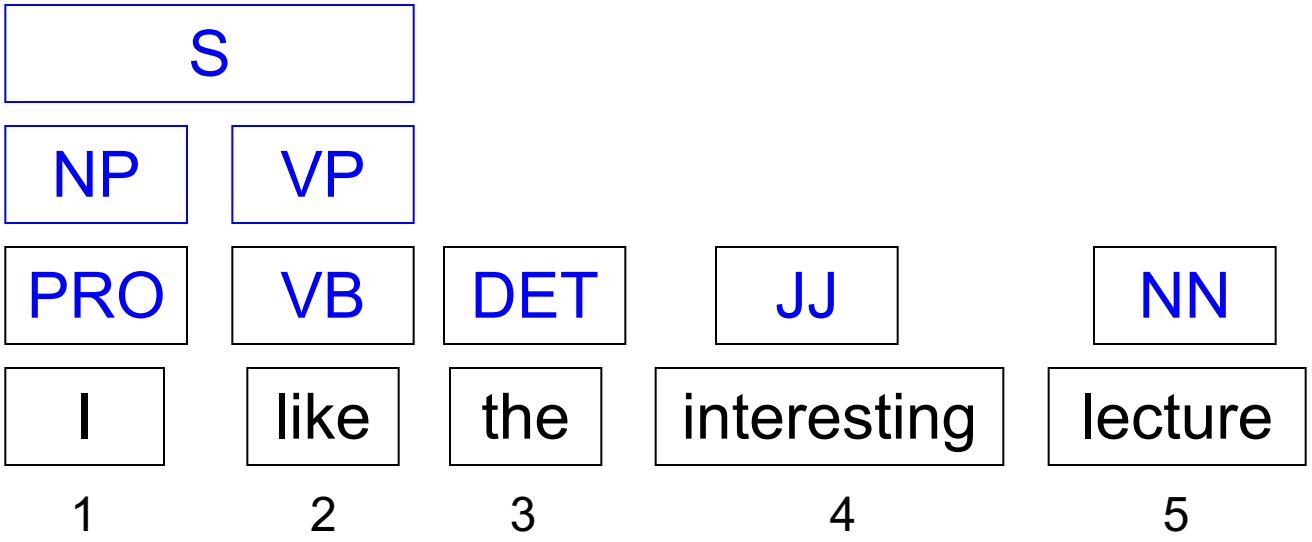
# Example

- 为第三个词搜索可能的非终结符重写规则：?→DET
  - 没有可匹配的规则



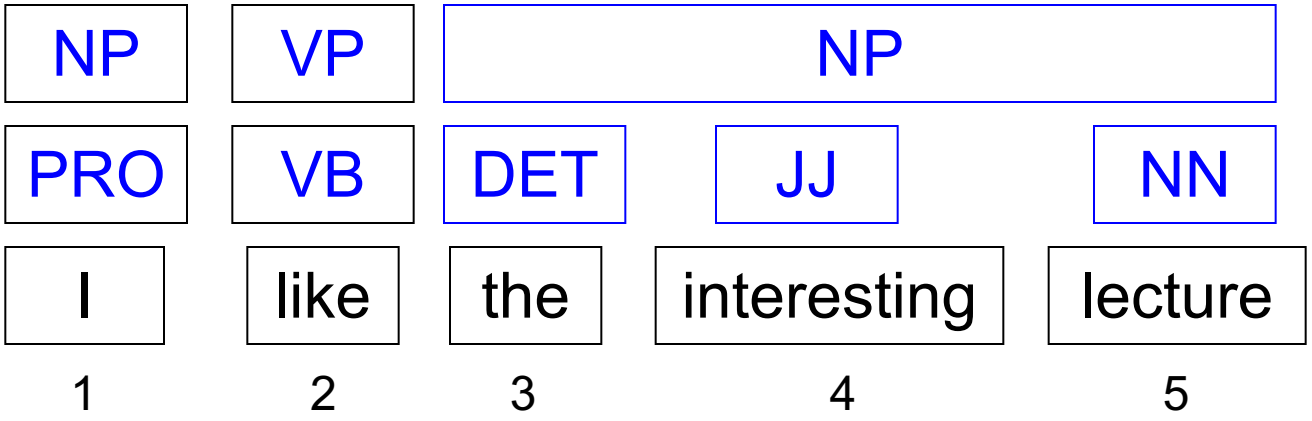
# Example

- 为前两个词搜索非终结符重写规则：?→NP VP
  - S→NP VP
- 然而得不到一颗完整的树



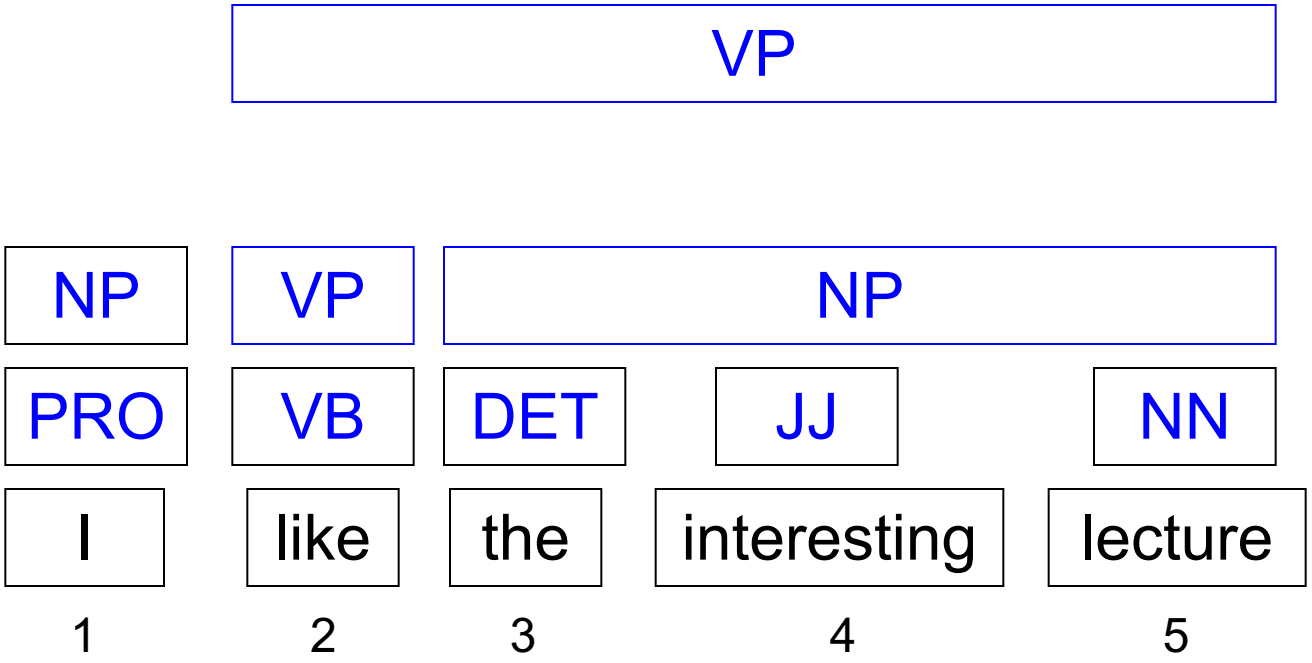
# Example

- 为后三个词搜索非终结符重写规则：
  ?→DET JJ NN
  - NP→DET JJ NN



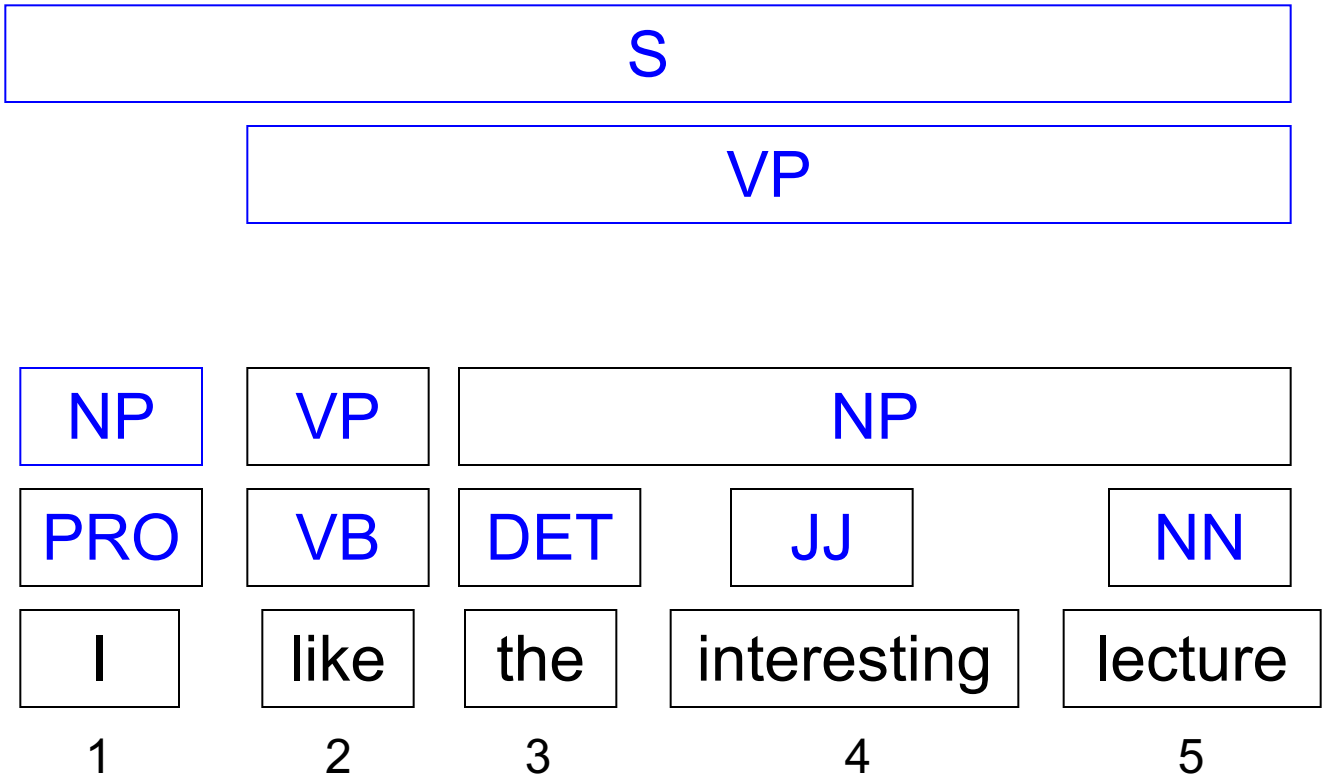
# Example

- 为后四个词搜索非终结符重写规则：?→VP
  NP：
  - VP →VP NP



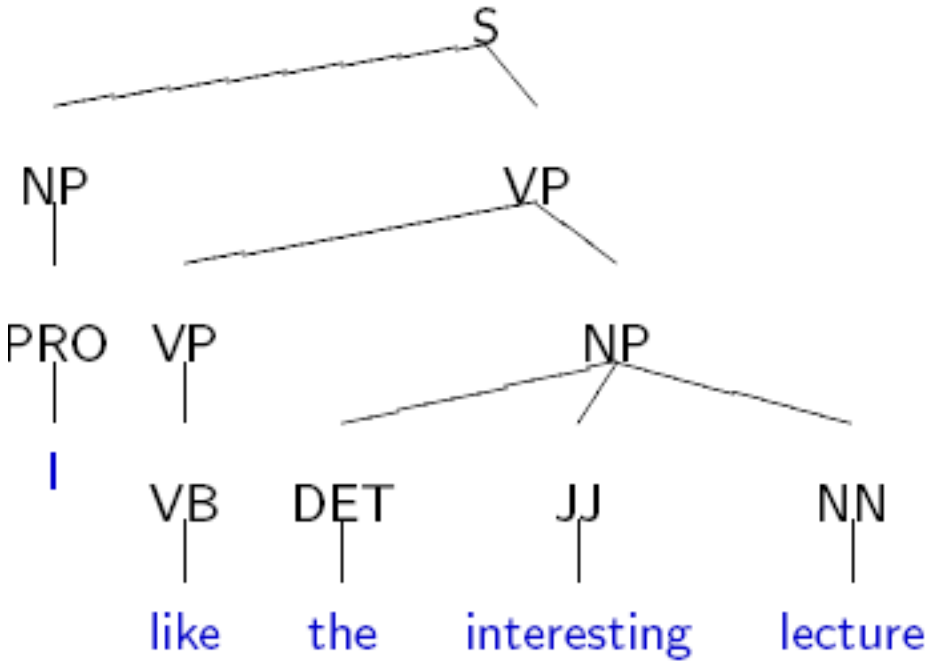
# Example

- 为五个词搜索非终结符重写规则： ?→NP VP
  - S→NP VP



# Example

- 最后得到完整句法树：



# CKY算法

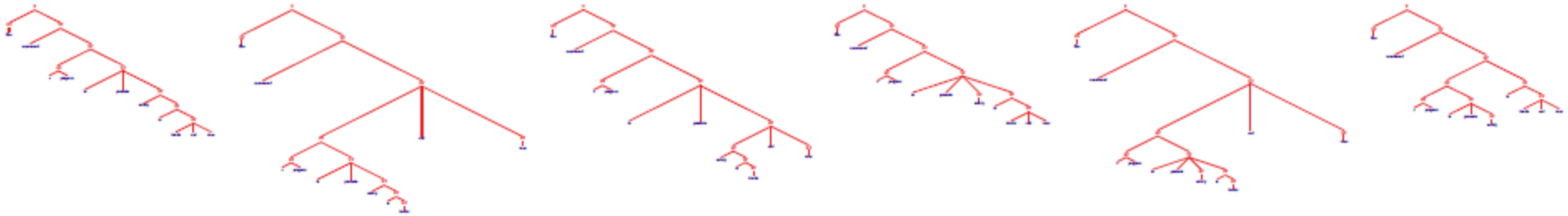
- for all words $w_i$: // terminal rules
  - for all rules A→$w_i$: add new chart entry A at span [i, i]
- for length = 1 to sentence length n // non-terminal rules
  - for start = 1 to n − (length − 1)
    end = start + length − 1
    - for middle = start to end − 1: // binary rules
      for all non-terminals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

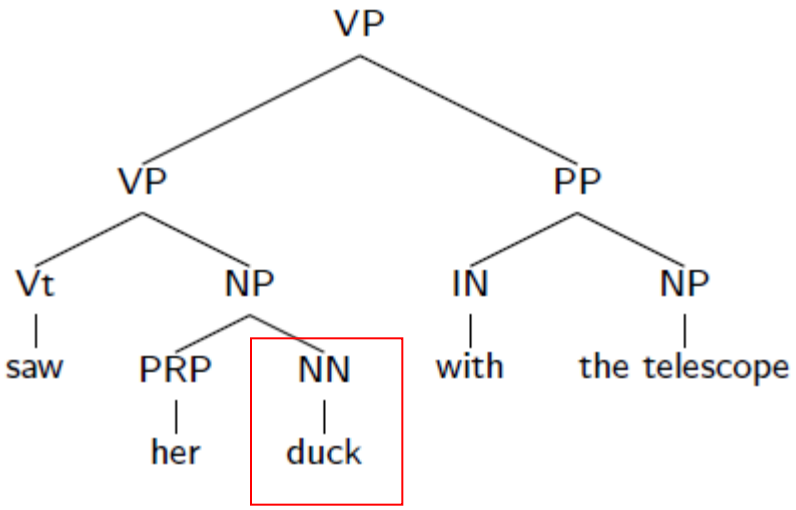
- **可能的输出：**

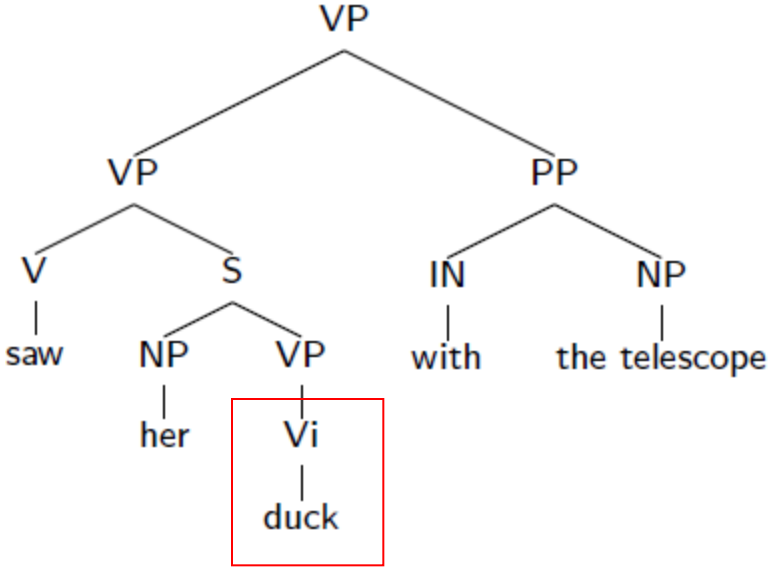


– 还有很多…

# 几种常见的歧义

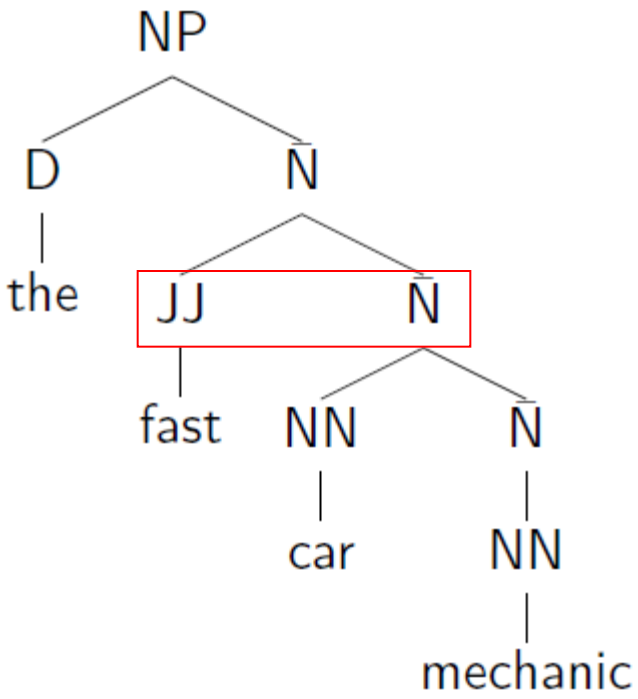
- 词性歧义：
  - NN → duck
  - Vi → duck

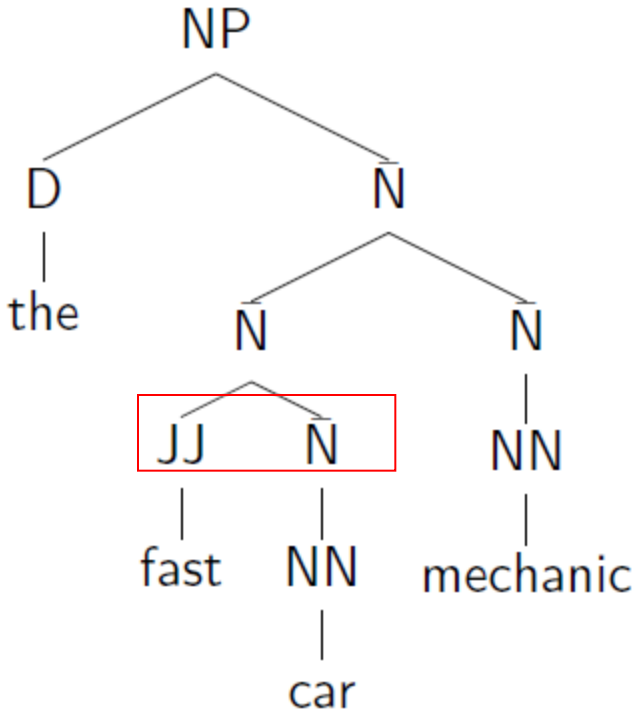


# 几种常见的歧义

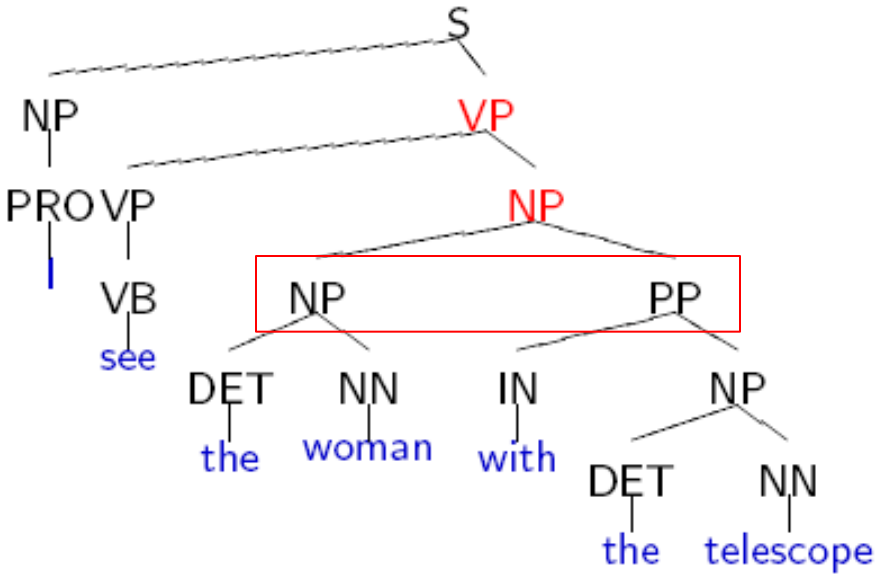
- 名词修饰语歧义：

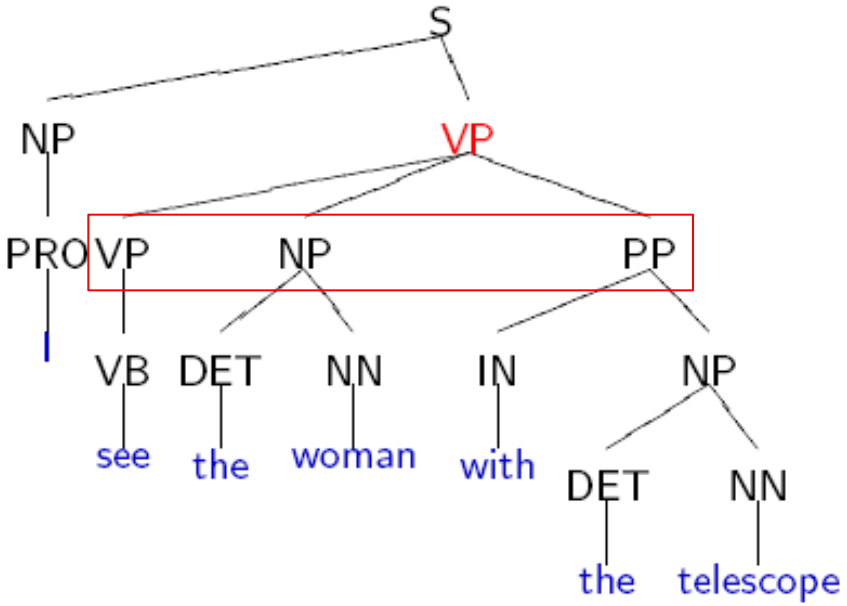


# 几种常见的歧义

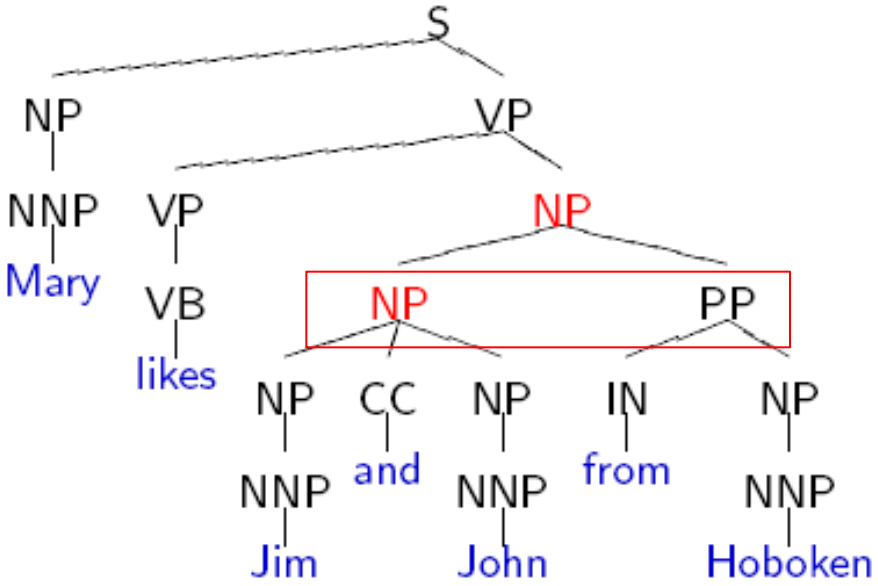
- 介词短语修饰语歧义：Who has the telescope?

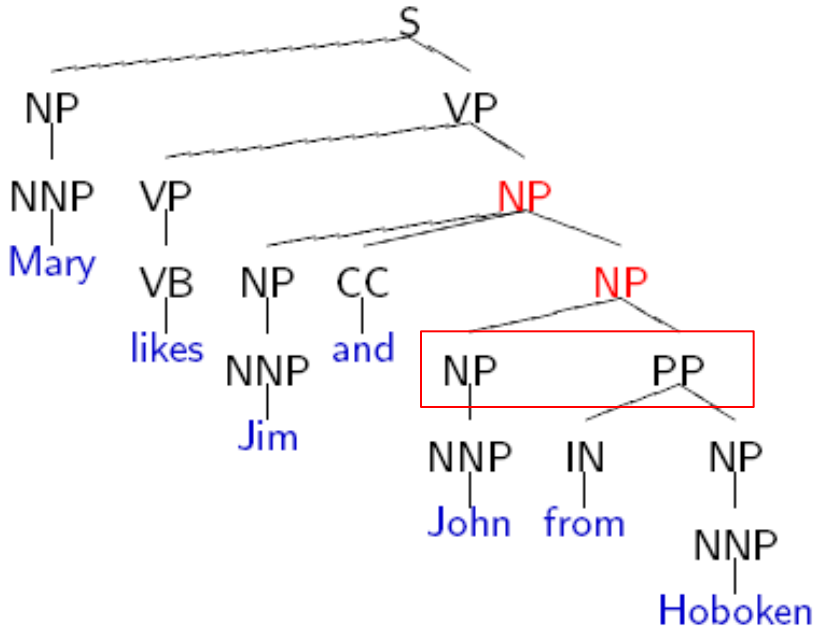


# 几种常见的歧义

- 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 $\rightarrow$ [0,1]

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

$$\sum_{\gamma \in T^*} P(\gamma) = 1$$

# Chomsky 范式

- Chomsky Normal Form
- 一个受Chomsky范式约束的PCFG句法 G = (T, N, S, R, P)，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
  - P：概率函数，为每个重写规则赋予一个概率值

# 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的特性

- 为CFG规则下的每一棵句法导出树赋予一个概率
- 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：

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

- 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：

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

# 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

- 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))))))
     .))
```

# 从treebank中学习语法

- 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

# 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 $\alpha \to \beta$ 的概率：

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

- 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

# Parsing with a PCFG

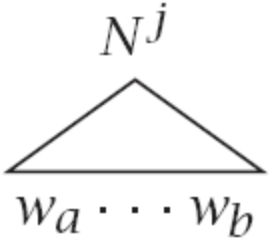
- 给定PCFG句法及句子s，定义Γ(s)为s的候选句法树构成的集合

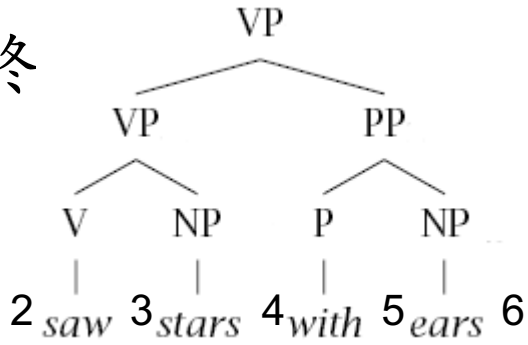
- 句法分析的目标：

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

# 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（特殊的，$N^1$为开始符号）
- $\{w^1, \ldots, w^V\}$：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
- $\alpha(p, q, N^j)$：外向概率
- $\beta(p, q, N^j)$：内向概率



# PCFG的假设

- 1. 位置不变性：

$$\forall k, P\left(N^j_{k(k+c)} \rightarrow \zeta\right) \quad \text{不变}$$

- 2. 上下文无关：

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

- 3. 祖先节点无关：

$$P\left(N^j_{kl} \rightarrow \zeta \mid ancestornodes\ of\ N^j_{kl}\right) = P\left(N^j_{kl} \rightarrow \zeta\right)$$

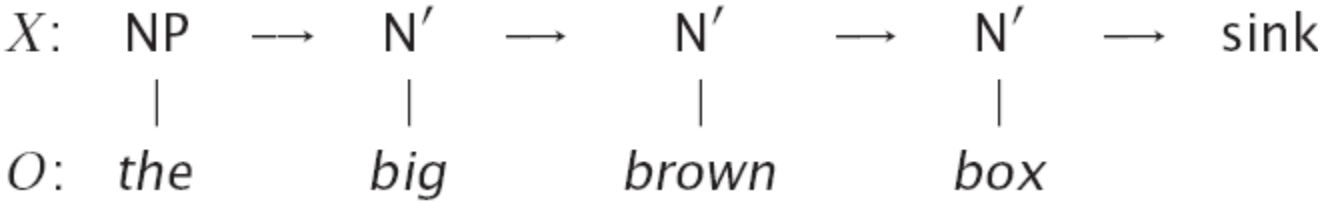
# PCFG 参数

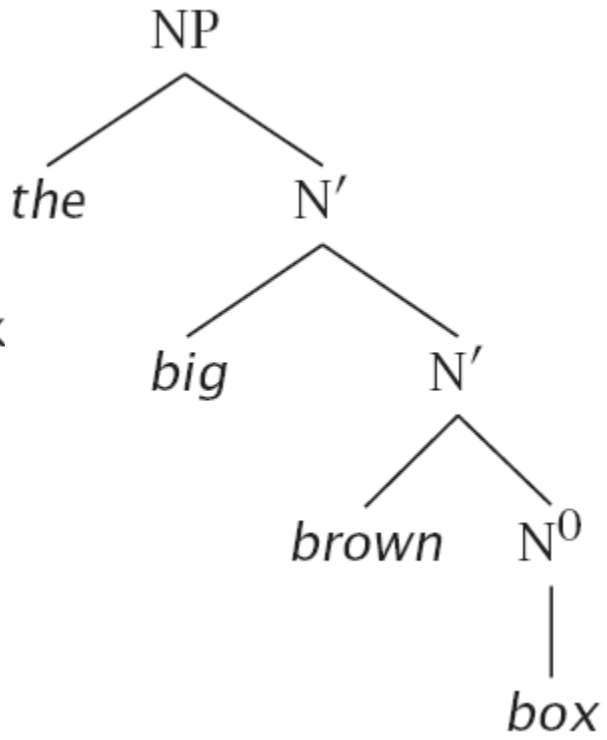
- CNF PCFG的句法规则：
  - $N^i \rightarrow N^j N^k$
  - $N^i \rightarrow w^j$
- CNF PCFG的参数：
  - *$P(N^i \rightarrow N^j N^k)$: A $n^3$ matrix of parameters*
  - *$P(N^i \rightarrow w^j)$: An nV matrix of parameters*
- 满足： *j=1, …, n,*

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

# HMMs与PCFGs的比较

- *HMM:* **Probabilistic Regular Grammar**
  - $N^i \rightarrow w^j N^k$
  - $N^i \rightarrow w^j$
  - 起始状态： $N^1$



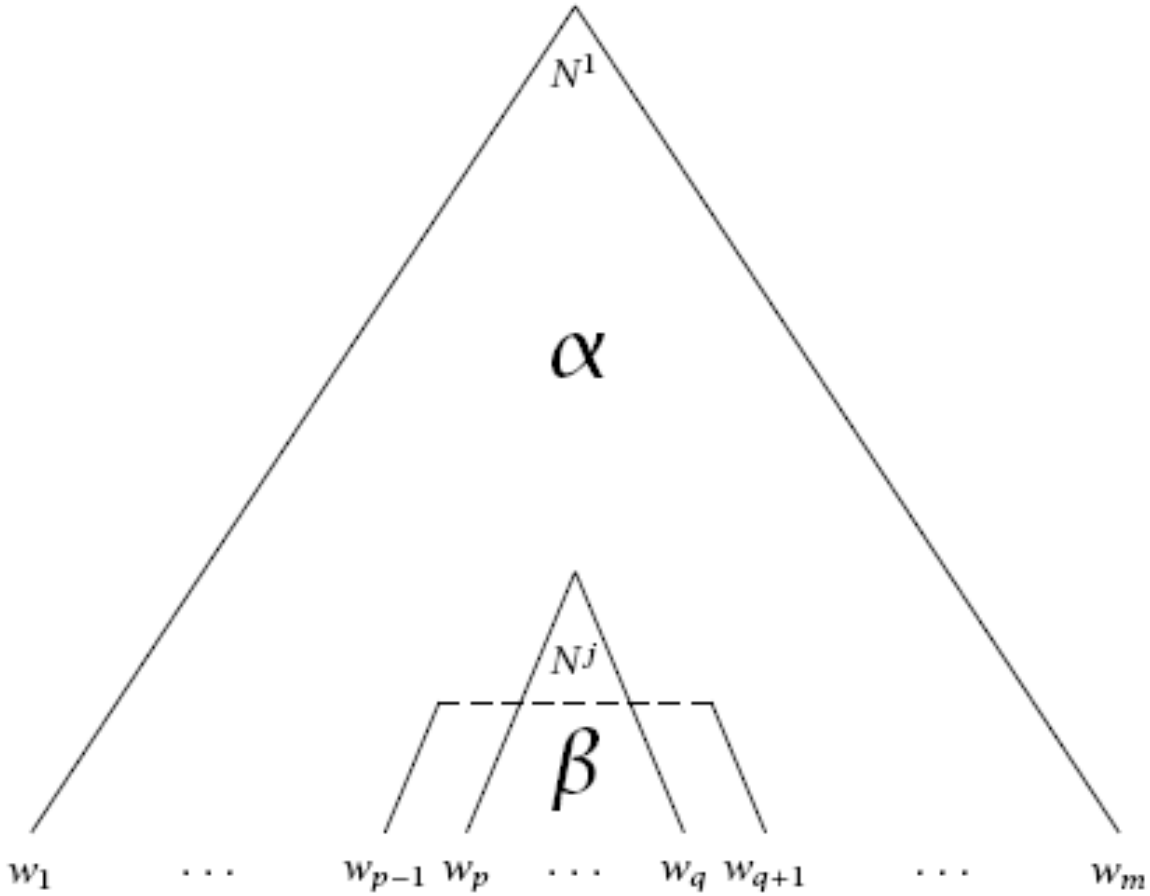



# 向内和向外概率

- HMM中定义了前向、后向概率：
  - Forwards = $\alpha_i$ (t)=$P(w_{1(t-1)}, X_t=i)$
  - Backwards = $\beta_i$(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 = $\beta(p, q , N^j)=P(w_{pq}|N^j_{pq}, G)$

# 向外和向内概率




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

# 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

# Problem 2: 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$\pi(i, j, X)$记录由非终结符X推导出子串$w_i,...w_j$的最大可能树$X_{ij}$对应的概率（所有可能的导出结构的概率的最大值）

# Problem 2: Parsing

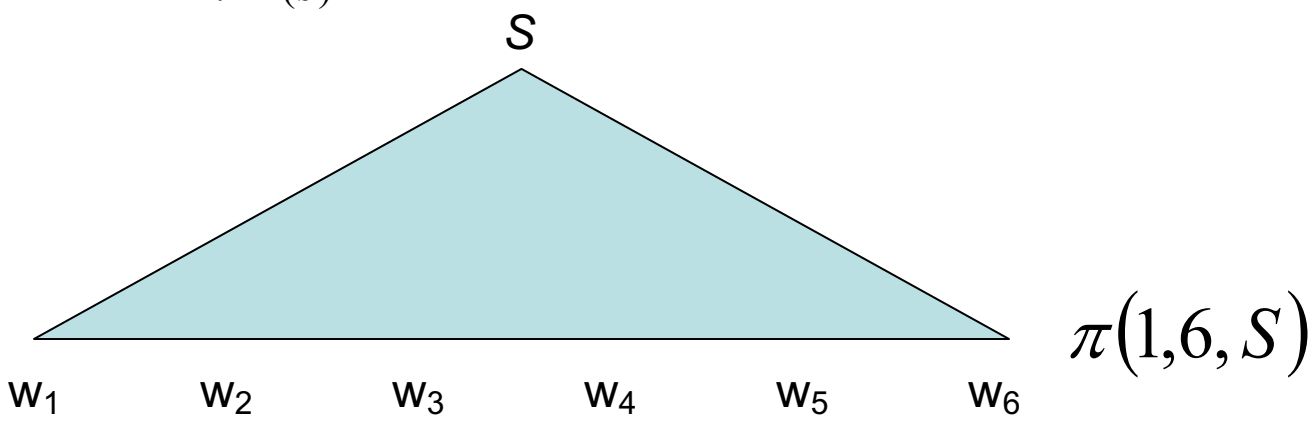
- 定义动态规划表：

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

= 子树$X_{pq}$最大的向内概率

- 目标是计算：

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




$\pi(1,6,S)$

# Inside-Outside algorithm

- 向内向外算法：一种动态规划方法
- 基本要素： for all i=1, …, n, for X∈N

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

  - 注意如果P(X→$w_i$)不在语法表中，则 P(X→$w_i$) = 0

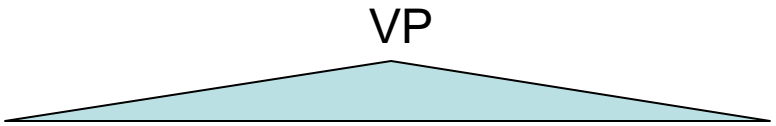
- 递归定义： for all i=1…n-1, j=(i+1)…n, for X∈N

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

# 一个例子

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

例如：计算π(3, 8, VP)




The dog saw the man with the telescope

1 2 3 4 5 6 7 8

假设已知：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

- 考虑例句： the dog saw the man with the telescope

- 假设已知PCFGs的参数为：
  - $q$(VP→V NP)=0.2
  - $q$(VP→VP PP)=0.5
  - $q$(VP→VP NP)=0.2
  - $q$(VP→VP N)=0.1

- 假设已知 $\pi$ 值的计算为：
  - $\pi(3,3,V)\times\pi(4,8,NP)=0.01$, $\pi(3,5,VP)\times\pi(6,8,PP)=0.1$, $\pi(3,6,VP)\times\pi(7,8,NP)=0.1$, $\pi(3,7,VP)\times\pi(8,8,N)=0.01$
  - 其它所有 $s\in\{3...7\}$ 且$X\in N$, $Y\in N$的情况，均假设 $\pi(3,s,Y)\times\pi(s+1,8,X)=0$

- 计算： $\pi(3,8,VP)$=?

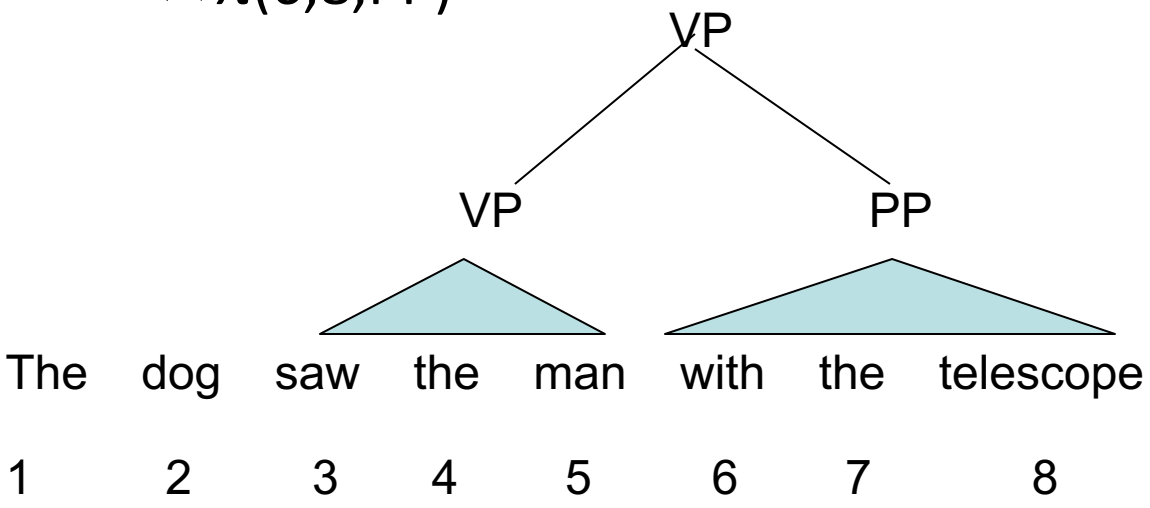
# Exercise

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

  $= q(\text{VP} \rightarrow \text{VP PP})$

  $\times\ \pi(3,5,\text{VP})$

  $\times \pi(6,8,\text{PP})$



# 完整的向内-向外算法

- Input: a sentence $s=x_1....x_n$, a PCFG $G=(N, \sum, S, R, q)$
- Initialization:

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

- Algorithm:

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

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

and

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

# Inside-Outside算法总结

- 慢…
  - 每次迭代的复杂度是$O(m^3n^3)$，其中$m=sum_{i=1, w}$，$n$是语法中非终结符的个数
- 局部最优
  - 每次不同的实验，往往会得到不同的局部最大值
    - 采用模拟退火？
    - Restrict rules by initializing some parameters to zero?
    - Or HMM initialization?
    - Reallocate nonterminals away from “greedy” terminals?

# PCFG的不足

- 过强的独立性假设（*Independence assumption*）
- 非终结符的语法规则，不考虑词汇信息
  - 会出现什么问题？？
- 不关心父节点和/子节点之外的节点的结构

# PCFGs的特性

- 使用PCFGs的几个原因及其限制：
  - 部分解决了语法歧义：PCFGs给出了一种判断句子合理性的方法
  - 一般优于早期的句法归纳方法 (Gold 1967)
  - 健壮性好（接受低概率句法）
  - 但由于缺少词汇化，PCFGs并不是一个最好的选择

# PCFGs的特性

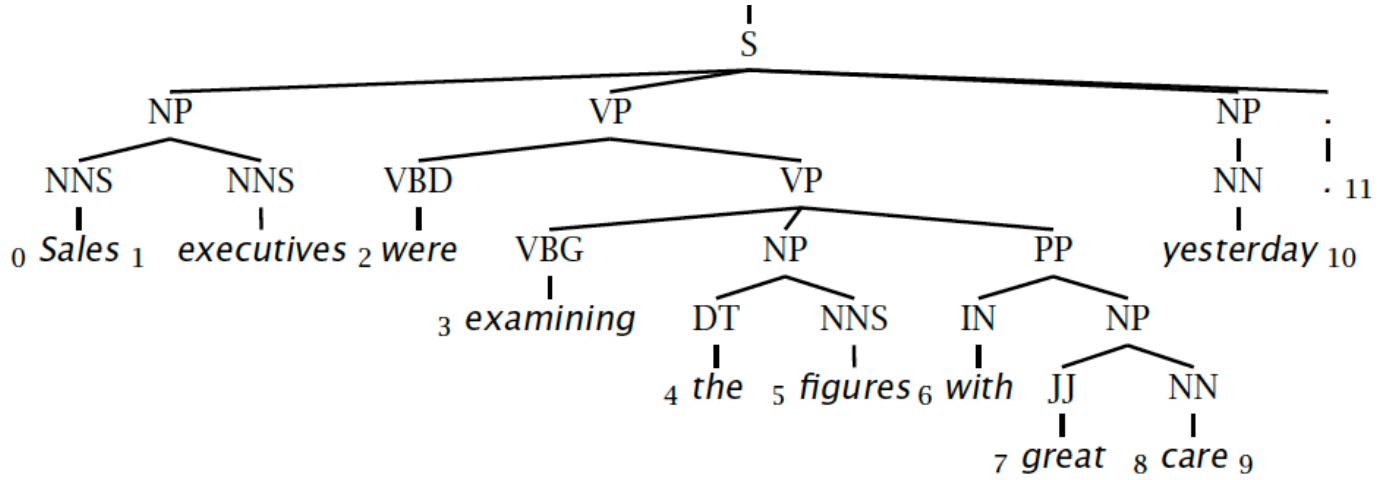
- Gives a probabilistic language model for English.
- In practice, a PCFG is a worse language model for English than a trigram model.
- Can hope to combine the strengths of a PCFG and a trigram model.
- PCFG encodes certain biases, e.g., that smaller trees are normally more probable.

# 总结

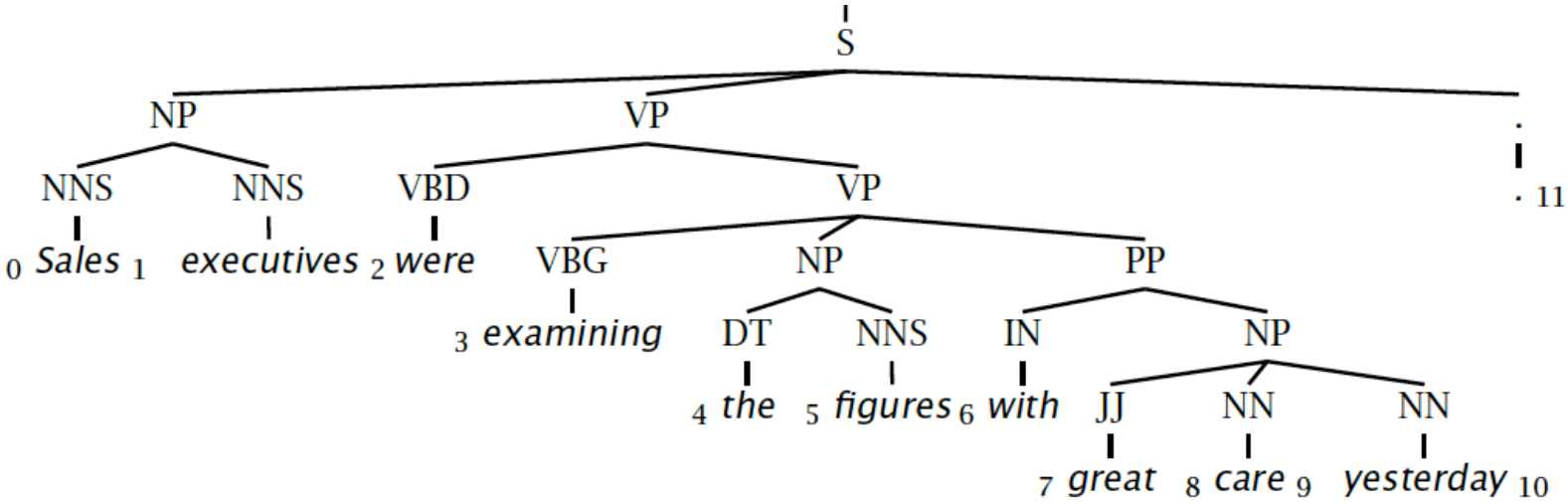
- PCFGs augments CFGs by including a probability for each rule in the grammar
- The probability for a parse tree is the product of probabilities for the rules in the tree
- To build a PCFG-parsed parser:
  - 1. Learn a PCFG from a treebank
  - 2. Given a test data sentence, use the CKY algorithm to compute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

# Several Free Parsers

- Michael Collins's Parser
  - http://people.csail.mit.edu/mcollins/code.html
  - English
- Dan Bikel's Parser
  - http://www.cis.upenn.edu/~dbikel/software.html#stat-parser
  - English / Chinese / Arabic
- Stanford Parser
  - http://www-nlp.stanford.edu/software/lex-parser.shtml
  - English / Chinese / German
- David Chiang's Parser
  - http://www.isi.edu/~chiang/
  - English / Chinese

- Next 4-5 lectures: sentence/text representation