

## Constituency Parsing

↓ 语义  
 | syntactic parsing 句法解析  
 | constituency parsing 语义解析

\* Constituency parsing overview

◆ Idea

单词 → 句子成分 → 更大句子成分 → 句子

⇒ 用树结构描述成分组合关系 ⇒ Constituency Parse Tree

Problem: 成分组合有歧义性 ⇒ 打分 (Scoring)

◆ Parsing → 找出得分最高的树  $t$

{ Method 1: Generative:  $\hat{t} = \arg\max_t S(t|x)$

(可理解为先加后加  $x$ )

→ (先加  $x$  后加  $t$ )

Method 2: Discriminative:  $\hat{t} = \arg\max_t S(t|x)$

$S$  算法计算: { Method 1: 从没树各点独立 → 加权的局部优解

Method 2: 独立假设, 全局搜索局部最优解

Span-based, Grammar-based → Transition-based

◆ Learning

监督 → Treebank 无监督 → 伪 Treebank 通过 BPE

评估 parsing tree 方法: 各边元组化 → 计算 precision, recall, F1

\* Span-based Constituency parsing

◆ Idea (Scoring)

仅二叉树 (binary parse tree),

$Score = \sum_l s(i,j,l) \rightarrow$  单节点  
 ↓  
 节点层级

根节点  $s(1, n, S) (l \neq S \Rightarrow s(l) = -\infty)$

parsing goal: find  $\arg\max \sum_l s(i,j,l)$

打分函数  $s$ : 对每跨段  $i, j$ , 每 Label  $l \rightarrow (i, j, l)$  元组都评分

\* Discriminative Parsing:

利用词向量 & Biaffine 表达式:

$$s(i, j, l) = \text{Biaffine}_l(\vec{r}_i, \vec{r}_j) = [\vec{r}_i; 1]^T W_l [\vec{r}_j; 1]$$

(还可用其他方法)

◆ Parsing: DP ⇒ CYK 解法 (Bottom up DP Algorithm)

$$\text{Base: } S_{best}(i, i) = s(i, i) \quad (s(i, j) = \max_l s(i, j, l))$$

$$S_{best}(i, j) = s(i, j) + \max_k (S_{best}(i, k) + S_{best}(k+1, j)) \rightarrow \text{动态规划}$$

↑ 旧部分数      ↓ 新子树分数

矩阵上执行: 由外圈逐圈扩展填数, 扩展域对角线

逐块对角线

以下每步迭代来源 → 还原向弦树

◆ 监督学习

$$P(t|x) = \frac{\exp S(t)}{Z(x)} \left( \text{softmax}(S(t)) \right) = \frac{\prod \exp s(i, j, l)}{\sum \exp s(i, j, l)}$$

$\Sigma(t)$  计算 ⇒ Inside-Algorithm (内面算法)

$$\alpha(i, j) = \sum_{t \in T_{i, j}} \exp s(t) \rightarrow \text{枚举所有左 i, 右 j 的树} \rightarrow \sum \exp$$

$$(S'(i, j) = \sum_l \exp s(i, j, l)) \rightarrow \text{与 CYK 一致处理树} \rightarrow \text{按边求和} \rightarrow \text{全局总分} = \sum \exp$$

$$\text{Base: } \alpha(i, i) = \sum_l \exp s(i, i, l) = S'(i, i)$$

$$\alpha(i, j) = \sum_A \sum_k \sum_{t_k} \exp s(i, j, A) \cdot \exp s(t_k) \cdot \exp s(t_{k+1})$$

逐边相乘 → 断点 → 子树 → 全部相加  $\sum \exp$

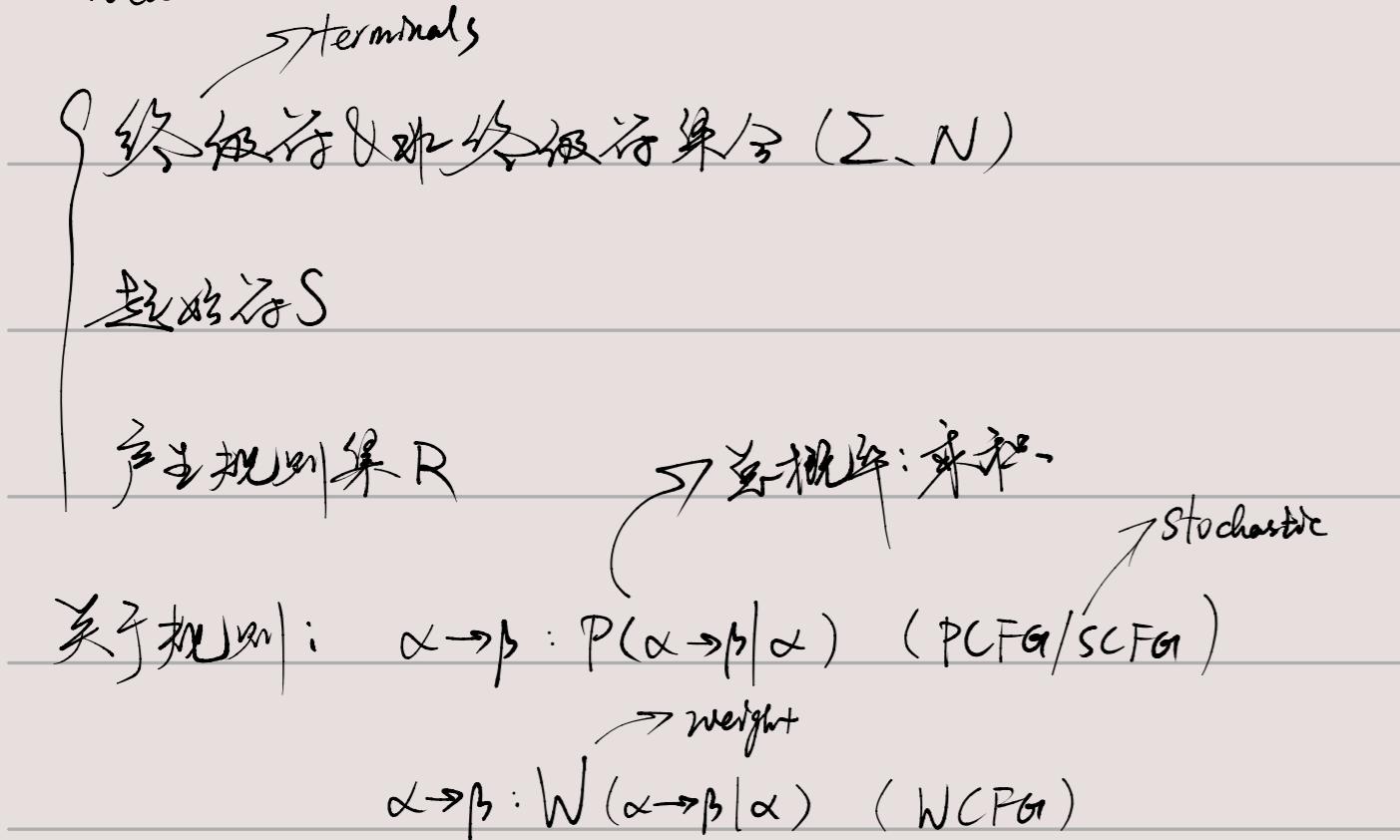
$$= S'(i, j) \times \sum_k (\alpha(i, k) \cdot \alpha(k+1, j))$$

优化: SGD

Alternative: marginal-based loss

## \* Context-Free Grammars

### Idea



### Parsing

CNF (Normal Form)  $\rightarrow 2\text{ non-terminal}$

Chomsky: Only 2 types of rules:  $\begin{cases} A \rightarrow BC \\ A \rightarrow w \end{cases}$  terminal

$\text{ex: } A \rightarrow B \ B \rightarrow C \Rightarrow A \rightarrow C$

$S \rightarrow ABC \Rightarrow S \rightarrow AX \ X \rightarrow BC$

### CYK (Bottom-up DP) $\rightarrow$ CYK for PCFG

区间表示: 打开区间

矩阵模拟: 内圈至外圈、折返总步数、存储器练习

滑坡: 去掉转移概率并加速

$$P_{A,i,j} = \max_{B,C,k} (P(A \rightarrow BC) \cdot P_{B,i,k} \cdot P_{C,k,j}) \rightarrow \text{转移方程}$$

全局概率      子树概率

$$\text{Base case: } P_{A,i+1,i} = P(A \rightarrow w_i) \rightarrow \text{单词概率}$$

$\rightarrow$  还可在 log space:

$$\begin{cases} S_{best}(i, i, A) = \log P_{A,i+1,i} = \log P(A \rightarrow w_i) \\ S_{best}(i, j, A) = \log P_{A,i+1,j} = \max_{B,C,k} (- - -) (P \text{ 已用 } S_{best} \text{ 表示}) \end{cases} \rightarrow dp[ ][ ]$$

$\Delta$  Span base CYK 方法  $\rightarrow$  子树极断点

### Learning

### 监督学习

### Generative Methods

$$\max P(x, t^*) \rightarrow \text{PCFG}$$

$\Rightarrow$  该问题: ~~该问题~~  $\rightarrow$  限制在统计概率 (MLE)

$\Rightarrow$  MLE 效果不佳  $\rightarrow$  Nonterminals in the bank are not sufficiently informative (无法通过规则学习到树结构)

### Discriminative Methods $\rightarrow$ for WCFG

$$\max P(t^* | x) \Rightarrow p(t|x) = \frac{\prod W(r|x)}{Z(x)} \rightarrow \text{归一化方法}$$

SGD 优化

$$\text{For Inside Algorithm: } \beta_j(p, q) = P(w_{pq} | N_{pq}^j, G) \rightarrow \text{局部树权值之和}$$

$$\therefore P(x=w_{1:m}) = \sum_{t \in T} P(t) = \beta_S(1, m) \rightarrow \text{所有树权值之和}$$

$$\text{Base: } \beta_j(k, k) = P(N^j \rightarrow w_k | G) \rightarrow \text{左右分离为根节点权值之和}$$

$$\text{Recursive: } \beta_j(p, q) = \sum_{r=1}^{q-1} P(N^j \rightarrow N^r N^q) \beta_r(p, r) \beta_q(q+1, q) \rightarrow \text{子树递归公式}$$

$\rightarrow$  权值分数求和

$\Delta$  best(i, j) parsing:  $\rightarrow$  CYK 内部算法:  $\max \downarrow \text{sum} \rightarrow$  权值得分布

Familiar:

HMM  $\left\{ \begin{array}{l} \text{Viterbi} \\ \text{Forward} \end{array} \right.$  max  $\rightarrow$  特殊 CYK  
sum  $\rightarrow$  特殊 inside

特殊 PCFG

可能需要整数: 双向问题  
对应二进法后缀及意义

$\Delta$  元监督:

结构搜索: 找到适当的 terminal, rules  $\rightarrow$  Bad

参数学习: 通过 terminal & rules, 寻找概率.

$\rightarrow$  限制训练语言的复杂度

o Parameter learning

目标: maximize  $P(\text{sentence})$

→ 通过句子边缘  
概率 (parse tree marginalized)

Compare with DP:

DP: 通过状态转移. 全局优化.  $O(n^3)$

→  $(i-j)$ , 局部化

算法: EM

E: 通过句子计算解分布分布 → expected

counts. & inside-outside algorithm

M: 更新参数. 有向式语义: expected-count 量化化

Method 2: 对  $P(\text{sentence})$  直接梯度下降

\* Transition-based

▲ B.S.

Transitions: SHIFT, REDUCE-X, UNARY-Y

→ 从二元语义加到一元语义加

▲ 配置:

Core: 通过分类器决定操作

config

策略: 模拟句法树生成 oracle-seg( $B_i, S_i, \text{transition}$ )

⇒ 利用 NN 存储序列信息 ⇒ 配置 classifier

Potential flaw: 只用 2-词频标注训练 → 遗训错误 not seen



during training

Dynamic Oracle: 配置中引入 Random errors

⇒ Find transitions → 调整结构直到 golden configuration

⇒ 使用该错误测量及最快替换方式训练

▲ Parsing

Action selection: Greedy

Beam search

长为 L 句子: L 次 SHIFT, L 次 UNARY, L-1 次

Reduce  $\Rightarrow O(3L-1) = O(L)$