

Dependency Parsing

1. Overview

Idea

Adv: 处理自由语序更优

以语义相关性为主线

更好捕捉下层的语义关系。

更快

drawback: 容易产生共识、不够清晰

projective parse: No crossing arcs

dep. to constituency: 叶节点按次序，一对 dependency arc 相

连出一个父节点 (为 dep head)

Constituency to dep: 从树叶结点找到匹配中心词 (词根)

⇒ 求化简图通过移

2. Parsing

Returning the best tree for story.

Algorithms: Graph based: MST. Eisner → 独立假设

及全局最优化

Transition-based: 独立假设及局部假设

Evaluating parsing: UAS (仅边相容)、LAS (带括号相容)

→ 增强型

3. Graph Based Parsing

Idea & Scoring

arc 打分 ⇒ 累加为 tree 总得分

由语义特征打分

Parsing

1. 完全独立选择所有边 X

2. Head Selection: 每词选择一个最大入边 X

3. MST (?)

Idea: 两词 → 1个 arc → 1个结点 → 相当于 1个 label

→ 类似 CGP → CYK 解析?

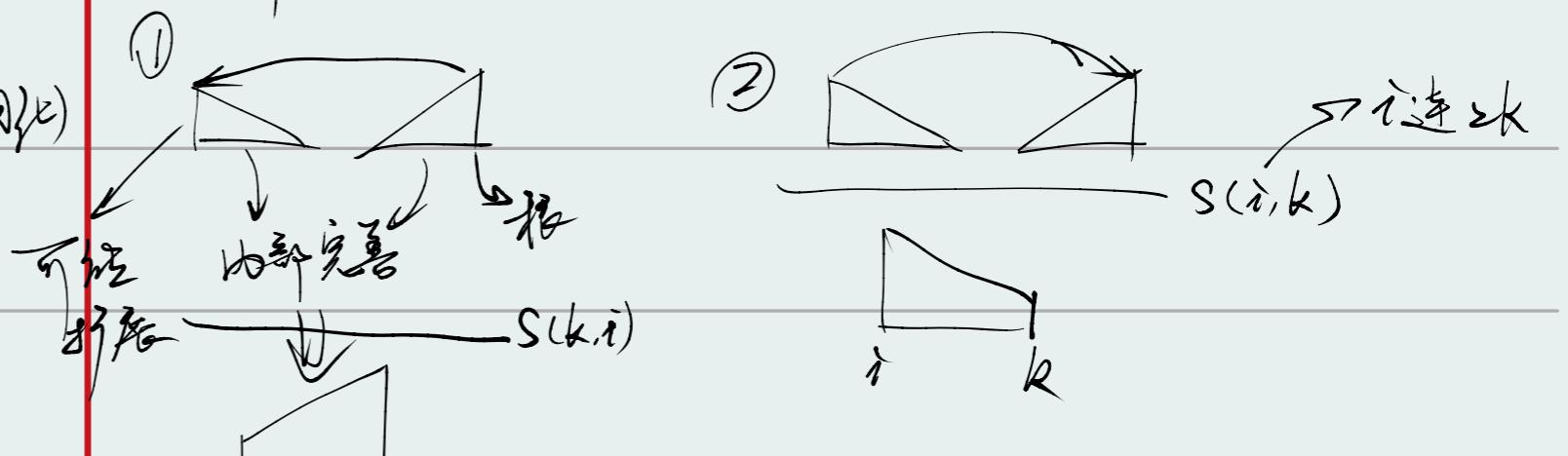
$O(n^3 |G|)$ rules = $O(n^2)$ $\rightarrow O(n^5)$

Eisner

4种情况. (后 * 表示是否构成了完整二叉树)

扩展子树 → 内部完成了 dependency parsing

合并类型:



Root

初始:

状态: $s(i, j, L/R, C/I)$

向左/向右: 完整路径 / 不完整路径

转移: ① ② ③ ④, i,j 中间嵌套断点 k max

$\Rightarrow O(n^3)$

前推: Projective

Non-projective Parsing: MST $\rightarrow O(n^3) \geq O(n^2 + n \log n)$

Second-order: no arcs between siblings/grandparents

non-projective

Parse: $O(n^4)$ or NP-hard

Learning

监督学习

树部分

$$\text{目标: } P(t|x) = \frac{\exp(\sum s(r))}{Z(x)} = \frac{\prod \exp(s(r))}{Z(x)}$$

$Z(x)$: for projective: Eisner 并查集, max 改为 sum

→ 先 inside

non-projective: Kirchhoff

无序标注: $P(t|x) = \prod P(b_i|x)$

优化: 梯度法

无监督

Generative: 通过 PCFG

优化: EM or SGD for $P(\text{sentence})$

Discriminative: CRF Autoencoder

优化: SGD

Transition-Based

option: SHIFT, RIGHT-ARC, LEFT-ARC

第二个指向第一个

从第一个指向第二个

