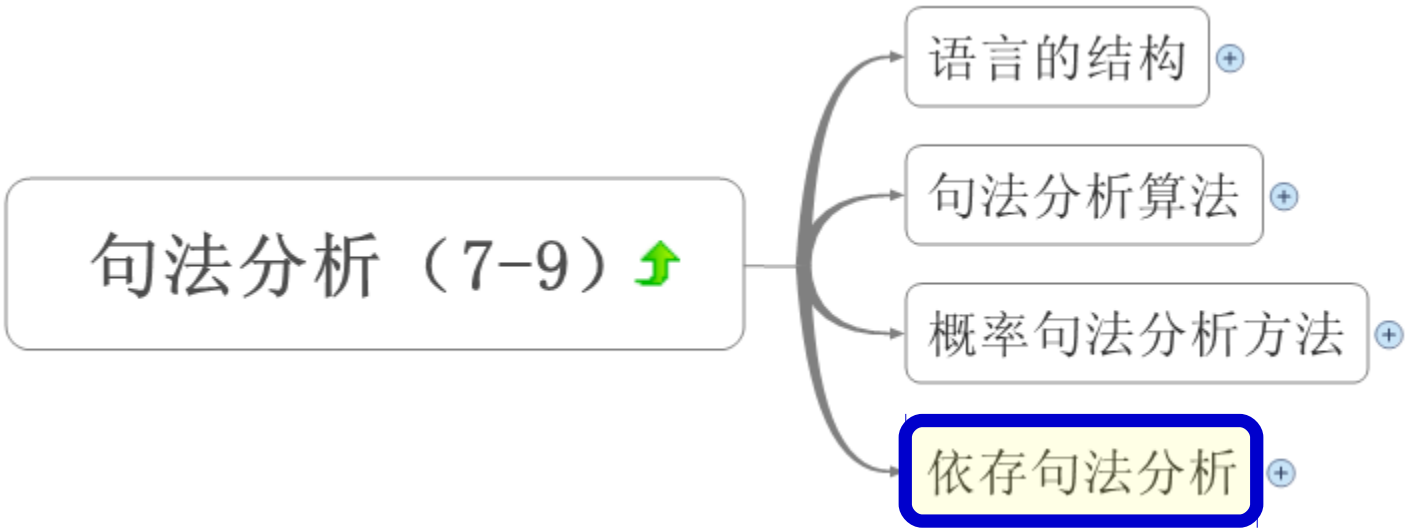
计算语言学

第 9 讲　句法分析（三）

刘群
中国科学院计算技术研究所
liuqun@ict.ac.cn

中国科学院研究生院 2012 年春季课程讲义

内容提要

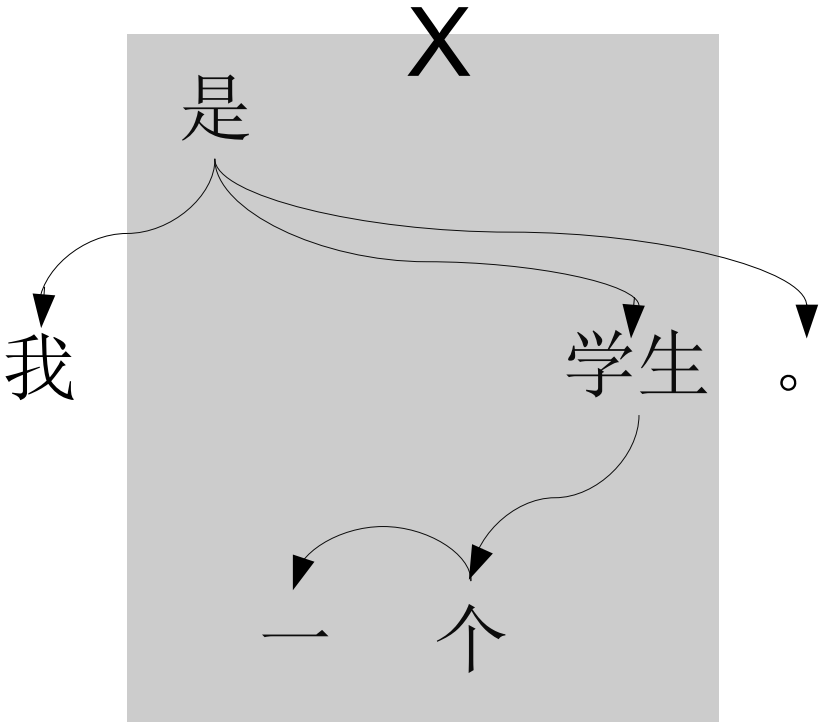


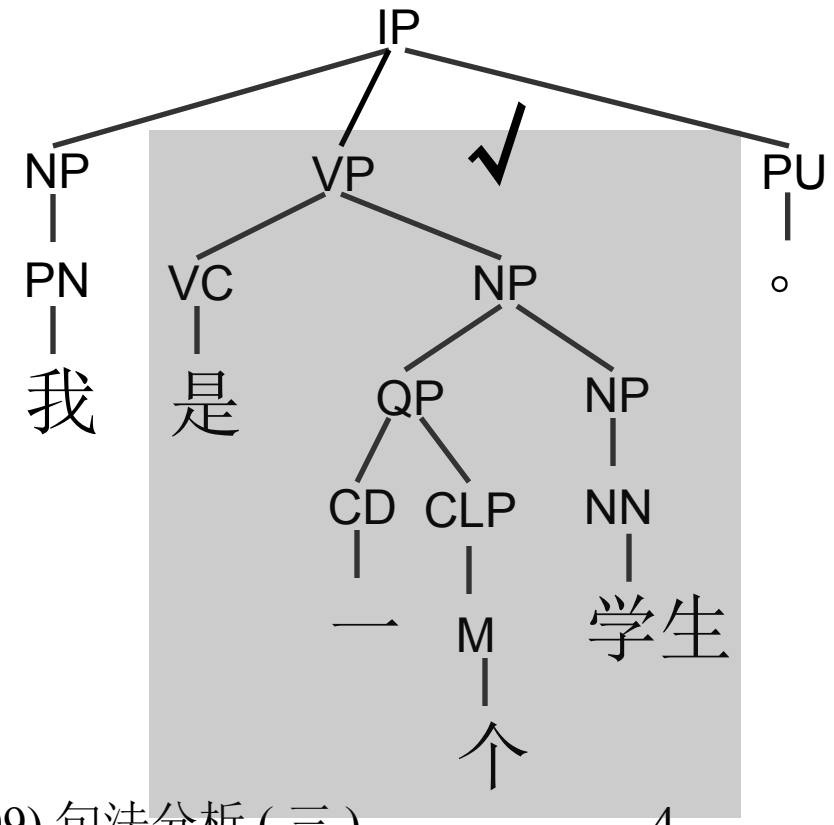
依存分析

- 依存结构和依存语法
- 短语结构树转依存树
- 专门的依存分析模型
  - 概率依存模型
  - 最大生成树模型
  - 状态转移模型

依存分析简介

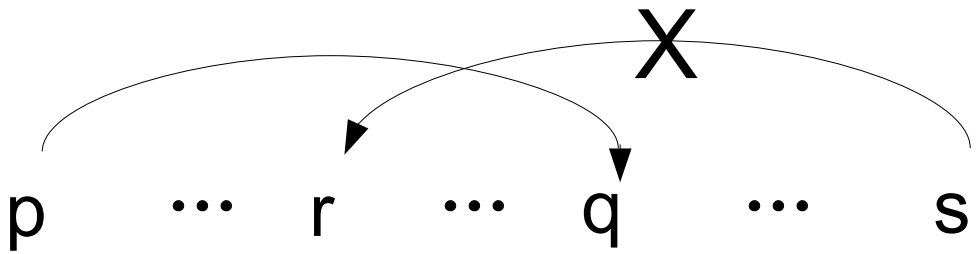
- 依存分析与短语结构分析类似，但有所不同：
  依存分析丢掉了跨度信息和跨度上的句法标识





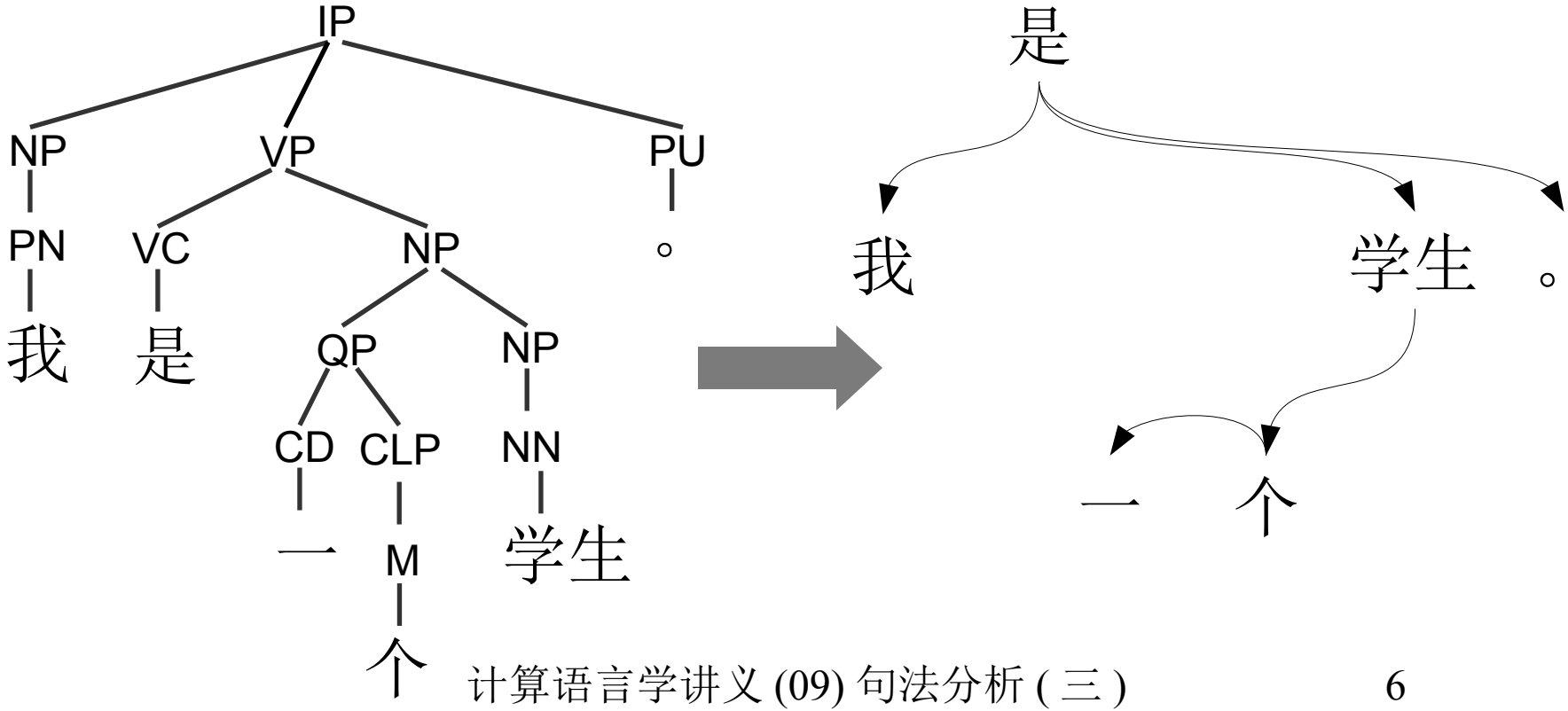
依存分析简介

- 大多数语言，包括汉语和英语，满足投射性。所谓投射性是指：如果词 p 依存于词 q，那么 p 和 q 之间的任意词 r 就不能依存到 p 和 q 所构成的跨度之外



短语结构树转依存树

- 任何短语结构树句法分析模型输出的句法树，通过 Yamada and Matsumoto (2003) 的中心词映射规则即可转化为依存结构树



短语结构树转依存树

- 中心词映射规则示例
  - 规则： IP right { IP VP }
  - 意义：对于句法树中标识为 IP 的节点，自右向左扫描该节点的所有孩子，第一个出现在列表 { IP VP } 中的孩子即为中心孩子节点。其他孩子节点的中心词将依存到中心孩子节点的中心词
- 对于给定的短语结构树，自底向上应用中心词映射规则，即可确定各词之间的依存关系

依存分析模型

- 生成式依存模型
  - 词汇依存概率模型（Collins 模型）
  - 依存生成概率模型（Eisner 模型）
- 判别式依存模型
  - 最大生成树模型
  - 状态转移模型

词汇依存概率模型

- Collins, 1996
- 训练：
  通过极大似然估计，在树库中统计出任意两个词之间存在特定依存关系的概率
- 对于给定的两个词，存在和不存在依存关系的概率之和为 1 。

词汇依存概率模型

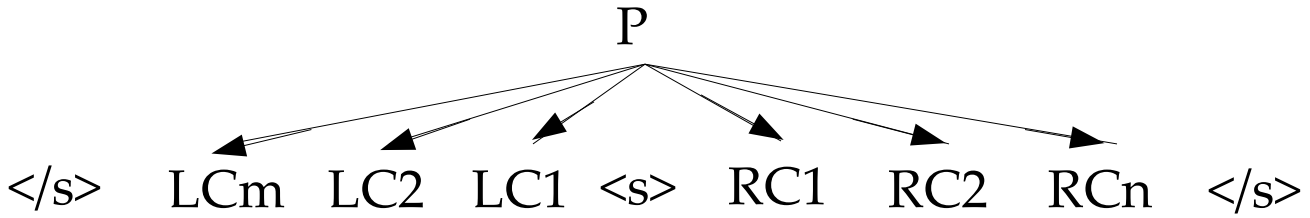
- 解码：
  - 寻找使得所有依存词对的依存概率的乘积最大的依存树，采用自底向上分析法
  - 可以采用 Viterbi 算法（ CYK ）
    - 不存在非终结符
    - 每次两个依存子树合并成一个更大的依存子树时，要考虑左子树根结点依存于右子树根结点和右子树根结点依存于左子树根结点两种可能性
    - 同一个 span 上根结点相同的依存树可以归并，只保留概率最大的依存树
    - 复杂度是句子长度的 3 次方
    - 可以确保找到的依存树满足投射性约束

依存分析模型

- 生成式依存模型
  - 词汇依存概率模型（Collins 模型）
  - 依存生成概率模型（Eisner 模型）
- 判别式依存模型
  - 最大生成树模型
  - 状态转移模型

概率依存模型
(Eisner, 1996)

给定一个带词性标记的输入语句，对于一棵可能的依存树，设该树中任一节点 P，它的左孩子由近及远分别为 LC1, LC2, …, LCm；右孩子分别为 RC1, RC2, …, RCn



定义 P 生成其所有孩子的概率为：

$$\mathrm{Gen}(P) = \prod_{i=1}^{m} \Pr(LC_i.word \mid LC_{i-1}.POS, P.word)$$

$$\times \prod_{i=1}^{n} \Pr(RC_i.word \mid RC_{i-1}.POS, P.word)$$

概率依存模型
(Eisner, 1996)

- 对于每棵候选依存树 T，整棵树的生成概率定义为树中所有节点生成概率的乘积

$$Gen(T) = \prod_{x \in T} Gen(x)$$

- 该模型的一个特点是考虑了兄弟依存节点对概率分布的影响
- 依存句法分析任务就是寻找生成概率最大的依存树
- 可以采用前述的 CYK 形式的 Viterbi 算法

依存分析模型

- 生成式依存模型
  - 词汇依存概率模型（Collins 模型）
  - 依存生成概率模型（Eisner 模型）
- 判别式依存模型
  - 最大生成树模型
  - 状态转移模型

最大生成树模型—基本思想

- McDonald et al., 2005
- McDonald and Pereira, 2006
- 给定一个包含 N 个词的句子，任意两个词之间都可能存在依存关系，共有 N*(N-1) 种可能的依存边（不能含有依存到自己的自环），只是依存强弱不同
- 将依存强弱表示为这个完全有向图中边的分数。于是，寻找最可能的依存树的任务就转化为寻找这个完全有向图的最大生成树

最大生成树模型—解码搜索

- 在有向图中确定最大生成树，在图论中有现成的算法，如 Chu-Liu-Edmonds 算法，时间复杂度为 O(VE)，还有一些改进的算法可以获得更好的时间复杂度
- 原始的最大生成树算法不能确保找到的最大生成树满足投射性约束，可以对原始算法加以改进，确保找到的最大生成树能够满足投射性约束

最大生成树模型—模型定义

- 每条边 p→c 的分数定义为

$$\text{score}(p \to c) = f(p \to c) \cdot w$$

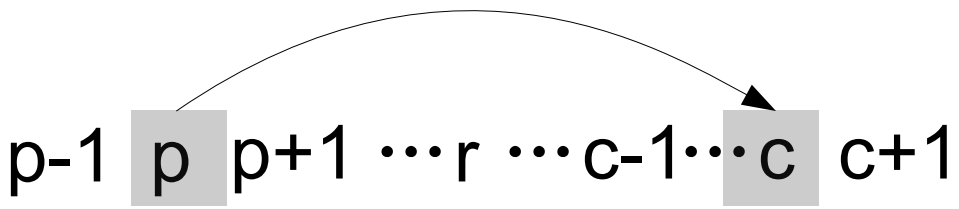
- f() 函数返回依存边 p→c 的特征向量； w 为权重向量，它由判别式训练得到

最大生成树模型—特征设计

- 特征设计针对边进行，而非节点
- 任意一条 p→c 的特征可以取那些呢？

最大生成树模型—特征设计

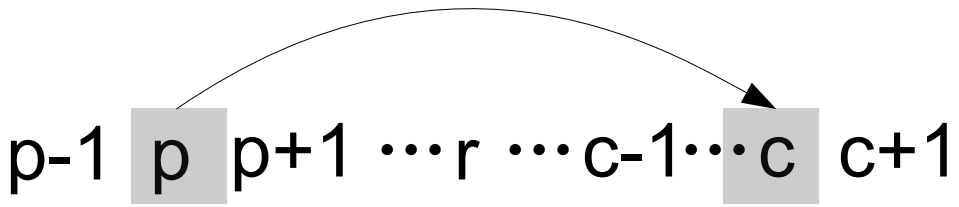
- 一元特征



Pword, Ppos
Pword
Ppos
Cword, Cpos
Cword
Cpos

最大生成树模型—特征设计

- 二元特征



Pword, Ppos, Cword, Cpos

Ppos, Cword, Cpos

Pword, Cword, Cpos

Pword, Ppos, Cpos

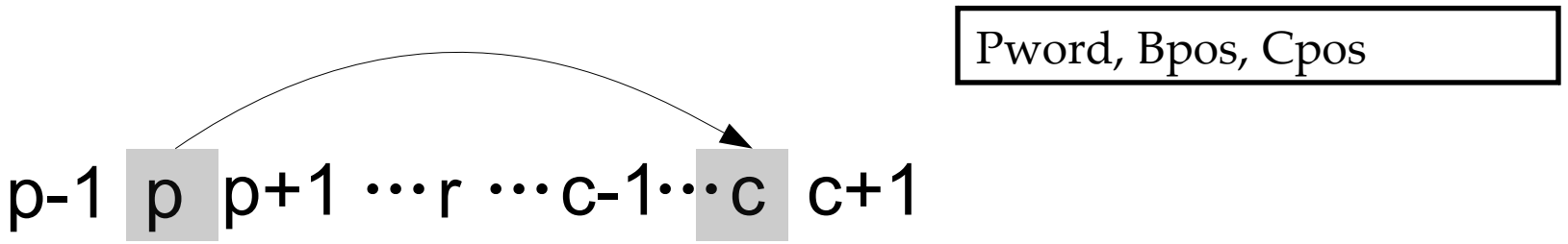
Pword, Ppos, Cword

Pword, Cword

Ppos, Cpos

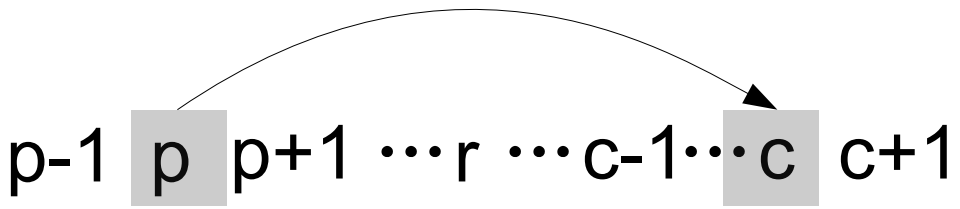
最大生成树模型—特征设计

- 词间词性标注特征， Bpos 为父亲 p 和儿子 c 之间的一个词的词性标注



最大生成树模型—特征设计

- 父亲儿子周围词性标注特征



Ppos,Ppos+1,Cpos-1, Cpos
Ppos-1, Ppos, Cpos-1, Cpos
Ppos, Ppos+1, Cpos, Cpos+1

最大生成树模型—特征设计

- 除以上特征本身，所有特征都加上父亲 - 儿子的顺序 ord 和距离 dis ，构成一组新的，更细化的特征
- Ord = (Index(p) > Index(c)) ? Left : Right
- Dis = abs(Index(p) – Index(c))

最大生成树模型—参数训练

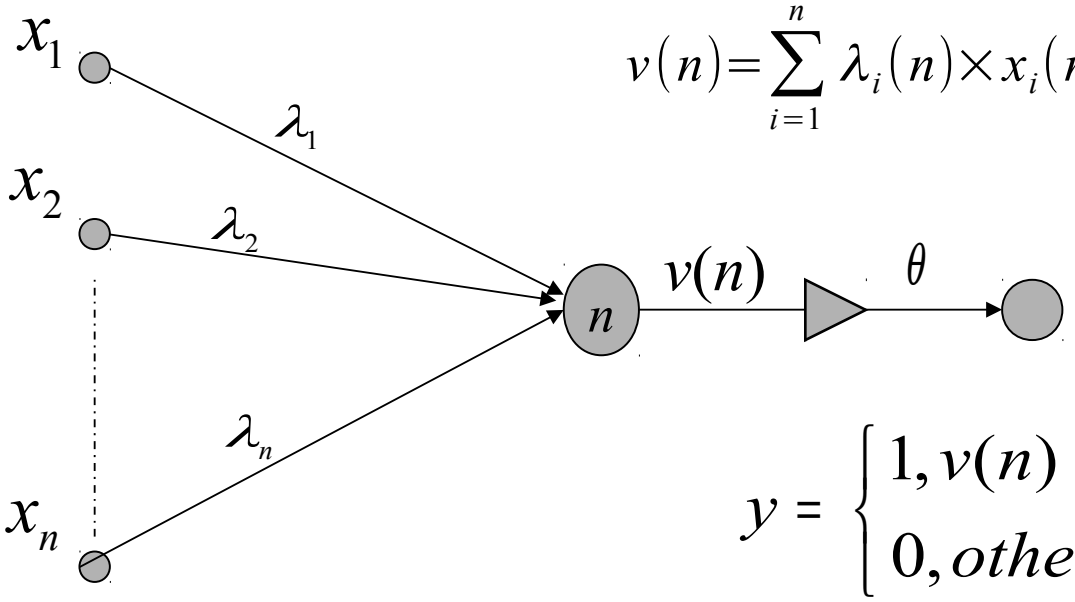
- 对于上述模型，我们可以采用感知机算法来进行模型参数的训练，其基本思想是：
  - 任意假设初始参数
  - 循环直到收敛
    - 对每个句子
      - 采用最大生成树算法进行依存分析，得到最优依存树
      - 对最优依存树上每一条依存边：
        - » 如果该条依存边出现在正确依存树上，但没有出现在分析结果中，那么调整该依存边的所有特征对应的权重，使得该依存边得分增加
        - » 如果该条边不出现在正确的依存树上，却出现在分析结果中，那么调整该依存边的所有特征对应的权重，使得该依存边得分减少

感知机简介

- 感知机是是一种双层神经网络模型，一层为输入层，另一层具有计算单元，可以通过监督学习建立模式判别的能力，在判别训练中广泛应用。
- 学习的目标是通过改变权值使神经网络由给定的输入得到给定的输出。
- 用于解决二值分类问题。

感知机简介

感知机模型示意图：



$$v(n)=\sum_{i=1}^{n}\lambda_i(n)\times x_i(n)=\Lambda^T(n)\times X(n)$$

$$y=\begin{cases}1, v(n)>\theta\\ 0, otherwise\end{cases}$$

在线性可分的情况下，上面的 $v(n)$ 就是分类的超平面方程。

自然语言处理中的感知机

- 传统的感知机，只用于解决二元分类问题
- 自然语言处理处理中，我们通常用扩展后的感知机算法来解决更复杂的问题，包括序列标注问题和句法分析问题，这些问题通常都不是二元分类问题，但可以简化为多元标注问题

自然语言处理中的感知机

- 对每一个可能的标注，与条件组合，得到一个特征向量，作为感知机的输入
- 标注的选择方法之一：
  - 对于每一个可能的标注，用一个感知机判断输出结果是 0 还是 1 ， 0 解释为错误标注， 1 解释为正确标注
  - 问题：如果有多个标注判断为正确，怎么办？
- 标注的选择方法之一：
  - 采用无阈值的感知机，对每一个输入的特征向量，只计算出特征加权和 $v(n)$ ，取 $v(n)$ 最大的标注作为最优标注
  - 目前自然语言处理中通常采用这种方法！

感知机应用

- 传统的感知机算法主要应用在两类的分类问题上
- 当前，在自然语言处理方面有如下应用：
  - 词性标注 (Michael Collins)
  - 增量式句法分析 (Michael Collins)
  - 语言模型的训练 (于浩，步丰林，高剑峰)
  - 机器翻译 (Percy Liang etc.)

感知机—问题定义

- 给定一组已经正确标注的训练样本：

$$W_1^R, W_2^R, \ldots W_N^R$$

- 在这些样本上去除标注信息，得到未标注样本：

$$W_1, W_2, \ldots W_N$$

- 对于这些未标注样本，采用某种方法进行标注，得到新的已标注样本：

$$W_1^t, W_2^t, \ldots W_N^t$$

感知机—模型定义

- 在任意一个已标注样本上，定义一组特征：

$$f_1(W_i^t), f_2(W_i^t), \ldots, f_M(W_i^t)$$

- 对于上述每一个特征 *f* 赋予一个权重 λ，就得到一个感知机模型：

$$\lambda_1, \lambda_2, \ldots \lambda_M$$

- 对于任意未标注样本，可以求解最优标注：

$$W_i^* = \underset{W_i^t}{argmax} \sum_{m=1}^{M} \lambda_m h_{f_m}(W_i^t)$$

h_{f_m} 为特征 f_m 出现的次数

感知机训练的目标函数

- 以分类错误平方最小化作为优化目标，使用梯度下降方法求解损失函数极值：

$$MSELoss(\lambda)=\frac{1}{2}\sum_{i=1}^{N}\left(Score(W_i^R,\lambda)-Score(W_i^*,\lambda)\right)^2$$

- *MSELoss* 可以用梯度下降的方式求极值，对 λ_m 求偏导：

$$\Delta\lambda_m=\frac{\partial MSELoss(\lambda)}{\partial\lambda_m}$$

$$=\sum_{i=1}^{N}\left(Score(W_i^R,\lambda)-Score(W_i^*,\lambda)\right)\times\left(h_{f_m}(W_i^R)-h_{f_m}(W_i^*)\right)$$

感知机的训练算法（之一）

for $iter = 1 \text{ to } T$

$\Delta\lambda_m = 0, m = 1, \ldots, D$

foreach $training - data : i$

$N = Score(W_i^R, \lambda) - Score(W_i^*, \lambda)$

foreach $c_m = h_{f_m}(W_i^R) - h_{f_m}(W_i^*)$

$\Delta\lambda_m = \Delta\lambda_m + \eta N c_m$

$\lambda_m = \lambda_m + \Delta\lambda_m$

参数 η 用于控制训练的步长，避免收敛过慢或者产生震荡。

感知机的训练算法（之二）

上述算法要等所有训练样本遍历一遍才能调整参数，为加快训练过程，可以使用梯度下降随机近似方法代替梯度下降，也就是对每一个样本处理后马上调整相关特征参数：

for $iter = 1 \text{ to } T$

  foreach $training-data : i$

    $N = Score(W_i^R, \lambda) - Score(W_i^*, \lambda)$

      foreach $c_m = h_{f_m}(W_i^R) - h_{f_m}(W_i^*)$

        $\Delta\lambda_m = \eta N c_m, \lambda_m = \lambda_m + \Delta\lambda_m$

感知机算法的扩展：平均感知机

- Averaged perceptron
  - 如果用 $\lambda_s^{t,i}$ 表示特征 f_s 在经过 t 次迭代之后，在输入第 i 个训练样本之后的值，则采用参数平均化的方式计算 f_s 的权值

$$\lambda_s = \frac{\sum_{t=1...T;i=1...N} \lambda_s^{t,i}}{NT}$$

平均感知机的使用是为了避免学习速度过快导致训练过程中出现的震荡现象。

最大生成树模型—感知机定义

- 由于句法分析是对每个句子进行分析，无法对单条边进行独立判断，因此需要在整个依存句法树上定义感知机：

$$score(tree) = \sum_{(p \to c) \in tree} \text{score}(p \to c)$$

$$= \sum_{(p \to c) \in tree} f(p \to c) \cdot w$$

$$= \left[\sum_{(p \to c) \in tree} f(p \to c) \right] \cdot w$$

最大生成树模型—感知机训练

训练集 T={(xt, yt), t=1..|T|}

For n = 1 .. N

- For t = 1 .. |T|
  - y′ = Decode(**w**, xt);
  - **w** = **w** + **F**(xt,yt) – **F**(xt,y′)
- End for

End for

w 为所有特征权重所构成的向量

对于每一个句子

任何你选用的解码方法得到一颗依存树

计算正确依存树的所有依存边的特征向量之和

计算分析所得依存树的所有依存边的特征向量之和

对于每个句子而言，只有那些正确依存树和解码所得依存树不一致的依存边，其相关的特征权重才会被调整

最大生成树模型—平均参数感知机训练

- 训练集 T={(xt, yt) , t=1..|T|}
- For n = 1 .. N
  - **v** = 0
  - For t = 1 .. |T|
    - y' = Decode(**w**, xt);
    - **w** = **w** + **F**(xt,yt) – **F**(xt,y')
    - **v** = **v** + **w**
  - End for
  - **w_avg** = **v** / (N*|T|)
- End for

依存分析模型

- 生成式依存模型
  - 词汇依存概率模型（Collins 模型）
  - 依存生成概率模型（Eisner 模型）
- 判别式依存模型
  - 最大生成树模型
  - 状态转移模型

状态转移模型

- 分析过程的任一时刻称为一个状态，依据该状态下的特征做出某种决策，从而转入新的状态。状态转移模型有两类：
  - **多伦扫描状态转移模型**
  - 移进归约状态转移模型

多轮扫描状态转移模型

- Yamada and Matsumoto, 2003
- 多轮扫描方式：对输入语句的词语序列进行多趟扫描，每趟都试图将存在依存关系的相邻词对 (或者子树对) 合并为更大的子树，直到形成一棵完整的依存树或者再也没有新的依存对产生

多轮扫描状态转移模型

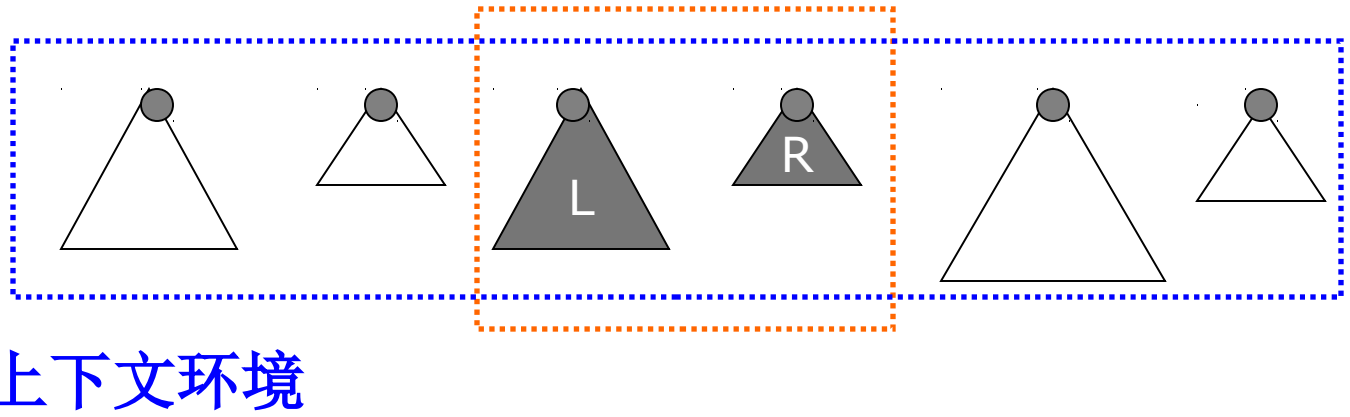
- Yamada and Matsumoto, 2003
- 基本思想
- 状态和转移
- 训练和解码

多轮扫描状态转移模型—基本思想

- 自左到右一趟趟扫描输入串，合并可以合并的相邻子树或词（左依存于右，或反之），直到整个输入串合并为一个依存树或者再也没有相邻子树能够合并为止

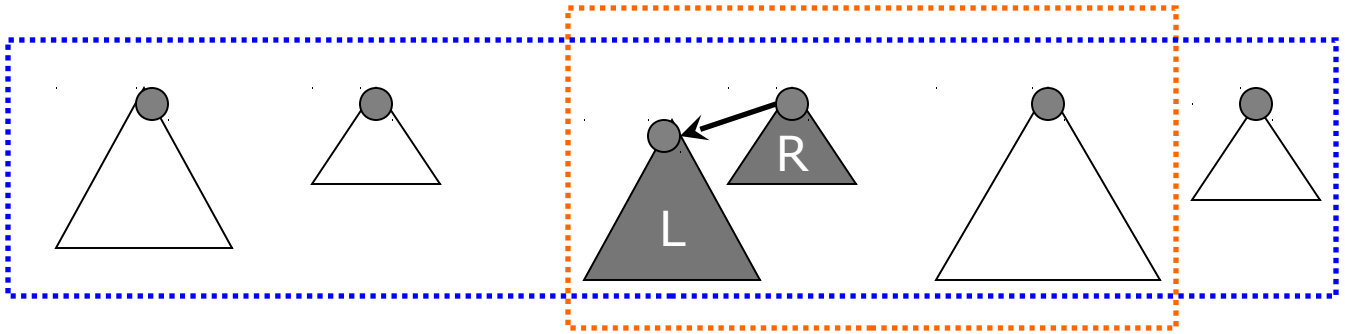
状态和转移

- 在逐趟扫描的过程中，任意时刻的状态是指：当前子树对及它们所处的特定上下文环境
- 任意状态下，三种可能的转移动作：left， right 和 shift



状态和转移

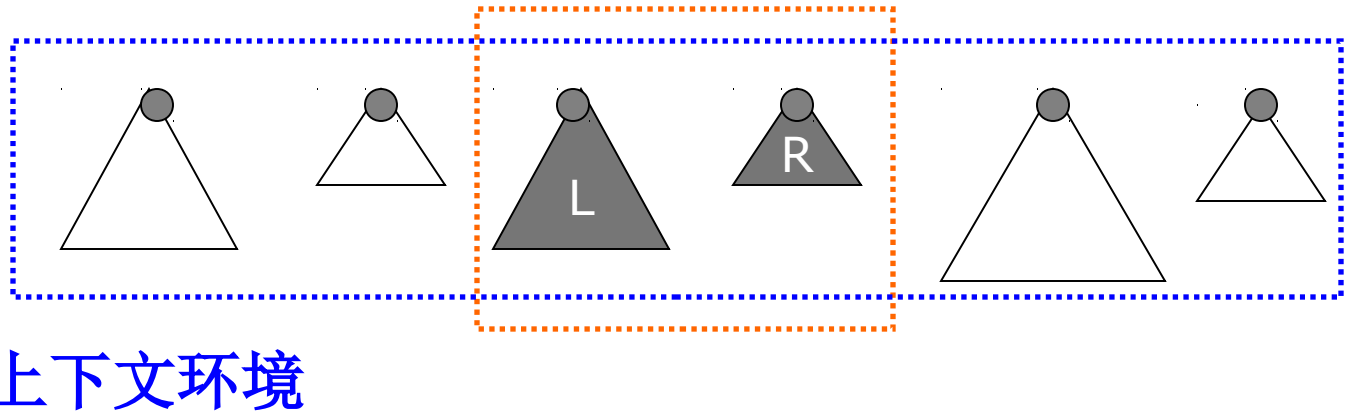
- 在逐趟扫描的过程中，任意时刻的状态是指：当前子树对及它们所处的特定上下文环境
- 任意状态下，三种可能的转移动作：left， right 和 shift



Left： L←R

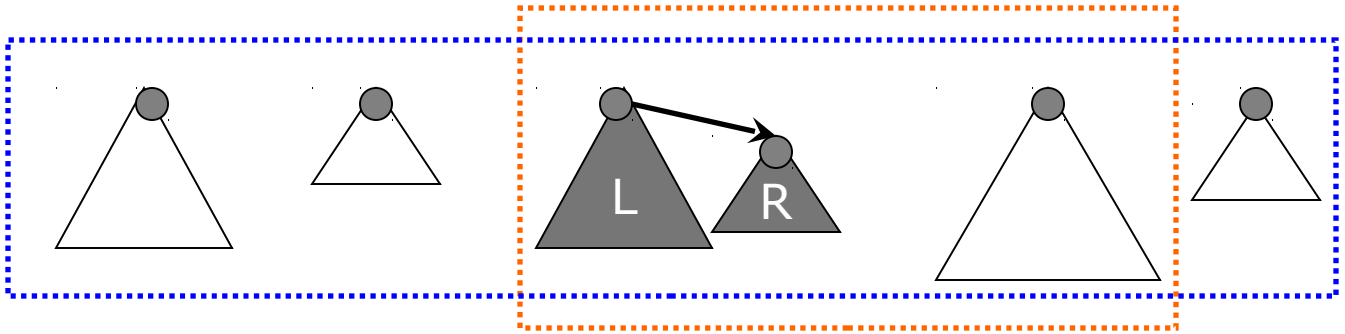
状态和转移

- 在逐趟扫描的过程中，任意时刻的状态是指：当前子树对及它们所处的特定上下文环境
- 任意状态下，三种可能的转移动作：left， right 和 shift



状态和转移

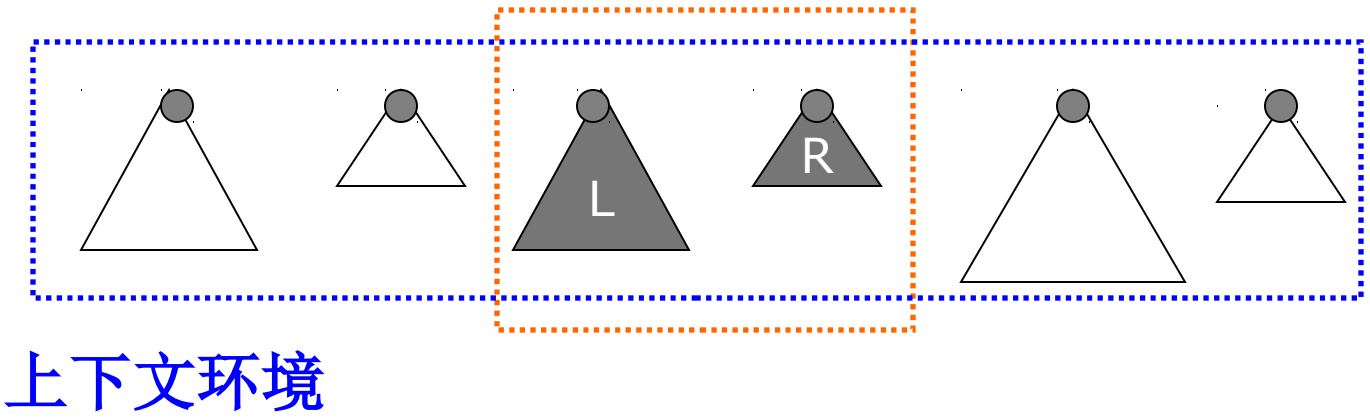
- 在逐趟扫描的过程中，任意时刻的状态是指：当前子树对及它们所处的特定上下文环境
- 任意状态下，三种可能的转移动作：left， right 和 shift



Right： L→R

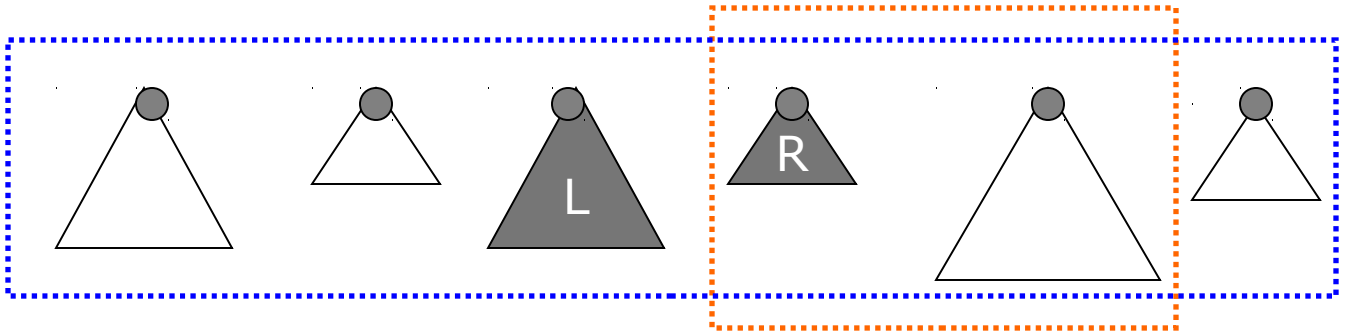
状态和转移

- 在逐趟扫描的过程中，任意时刻的状态是指：当前子树对及它们所处的特定上下文环境
- 任意状态下，三种可能的转移动作：left， right 和 shift



状态和转移

- 在逐趟扫描的过程中，任意时刻的状态是指：当前子树对及它们所处的特定上下文环境
- 任意状态下，三种可能的转移动作：left， right 和 shift



Shift

训练和解码

- 训练：
  在树库的训练集中抽取状态转移实例，训练判别式分类器，比如 SVM
- 解码：
  在逐趟扫描的过程中，分类器根据当前子树对及其上下文的特征，预测需要执行的状态转移操作并转移到新的状态。这一过程持续至结束条件满足
- 问题：可能最后不能得到完整的依存树

状态转移模型

- 分析过程的任一时刻称为一个状态，依据该状态下的特征做出某种决策，从而转入新的状态。状态转移模型有两类：
  - 多伦扫描状态转移模型
  - **移进归约状态转移模型**

移进规约状态转移模型

- Nivre et al. , 2006
- 基本思想
- 状态和转移
- 训练和解码
- nbest 全局训练

移进归约状态转移模型—基本思想

- 在分析过程中维护一个堆栈和一个队列，堆栈用以存储到目前为止所有的依存子树，队列存储尚未被分析到的词。堆栈顶端和队列的头部确定了当前分析器的状态，依据该状态决定进行移进、规约或者建立栈顶元素与队首元素的依存关系的操作，从而转入新的状态

状态和转移

- 在逐趟扫描的过程中，任意时刻的状态是指：当前堆栈和当前队列的格局
- 任意状态下，四种转移动作：
  left， right， shift 和 reduce

状态和转移

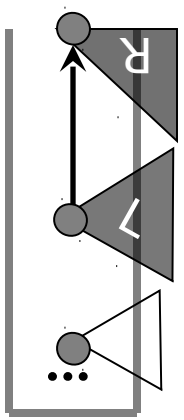
- 在逐趟扫描的过程中，任意时刻的状态是指：当前堆栈和当前队列的格局
- 任意状态下，四种转移动作：left， right， shift 和 reduce

状态和转移

- 在逐趟扫描的过程中，任意时刻的状态是指：当前堆栈和当前队列的格局
- 任意状态下，四种转移动作：
  left， right， shift 和 reduce

状态和转移

- 在逐趟扫描的过程中，任意时刻的状态是指：当前堆栈和当前队列的格局
- 任意状态下，四种转移动作：
  left， right， shift 和 reduce



栈顶

队首

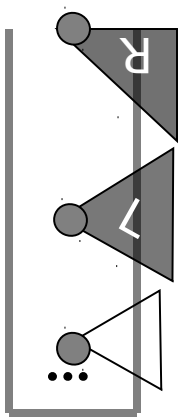
Right： L→R

状态和转移

- 在逐趟扫描的过程中，任意时刻的状态是指：当前堆栈和当前队列的格局
- 任意状态下，四种转移动作：
  left， right， shift 和 reduce

状态和转移

- 在逐趟扫描的过程中，任意时刻的状态是指：当前堆栈和当前队列的格局
- 任意状态下，四种转移动作：
  left， right， shift 和 reduce



栈顶

队首

Shift

状态和转移

- 在逐趟扫描的过程中，任意时刻的状态是指：当前堆栈和当前队列的格局
- 任意状态下，四种转移动作：left， right， shift 和 reduce

状态和转移

- 在逐趟扫描的过程中，任意时刻的状态是指：当前堆栈和当前队列的格局
- 任意状态下，四种转移动作：left， right， shift 和 reduce

状态和转移

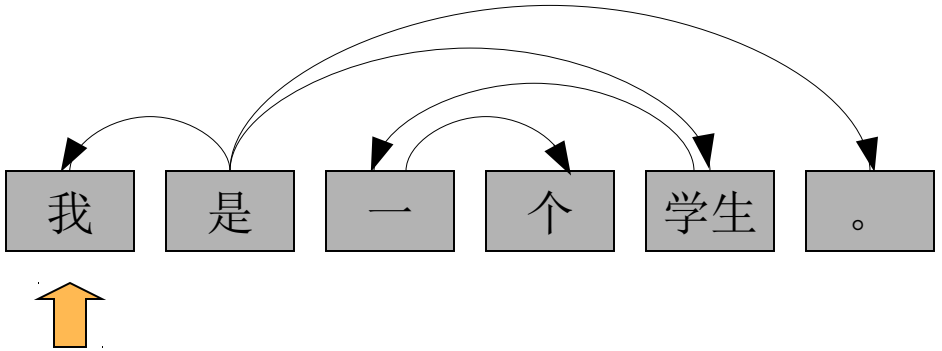
- 动作可以执行的条件
  - Left： 栈顶元素尚无 head
  - Right：无
  - Shift：输入队列不空
  - Reduce：栈顶元素有 head
- 想一想，为什么？

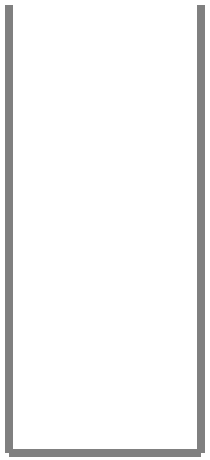
训练和解码

- 训练：
  - 状态转移实例抽取——模拟分析树库中每棵依存树
  - 判别式训练分类器，如 SVM
- 解码：
  在单趟扫描的过程中，分类器根据当前栈顶、当前队首以及栈顶和队首所处的上下文，预测需要执行的状态转移操作并转移到新的状态。这一过程持续至输入队列为空
- 问题：可能最后不能得到完整依存树
- 解决办法？

模拟分析

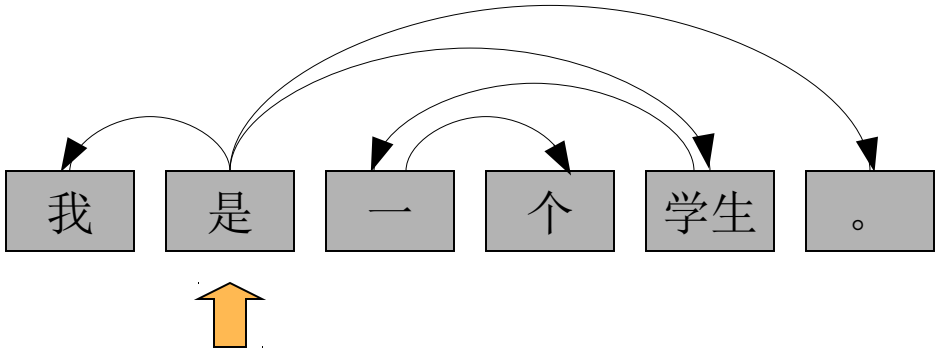
- ((我 /PN) 是 /VC ((一 /CD (个 /M)) 学生 /NN) (。 /PU))

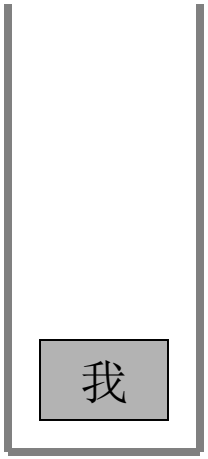




模拟分析

- ((我 /PN) 是 /VC ((一 /CD (个 /M)) 学生 /NN) (。 /PU))

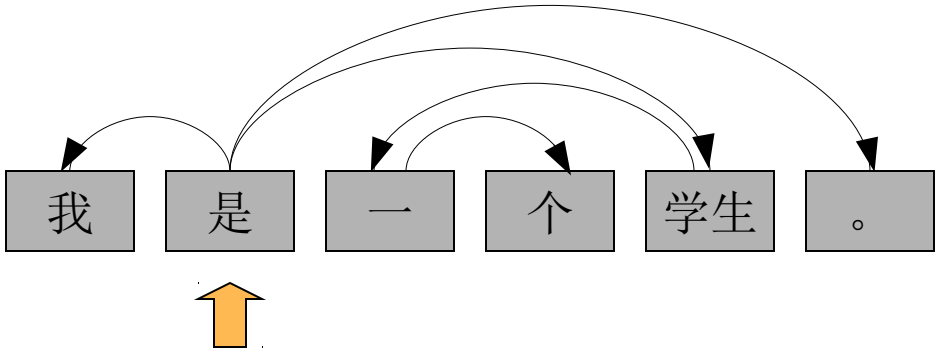


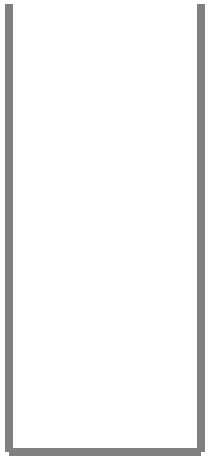


Shift(我)

模拟分析

- ((我 /PN) 是 /VC ((一 /CD (个 /M)) 学生 /NN) (。 /PU))

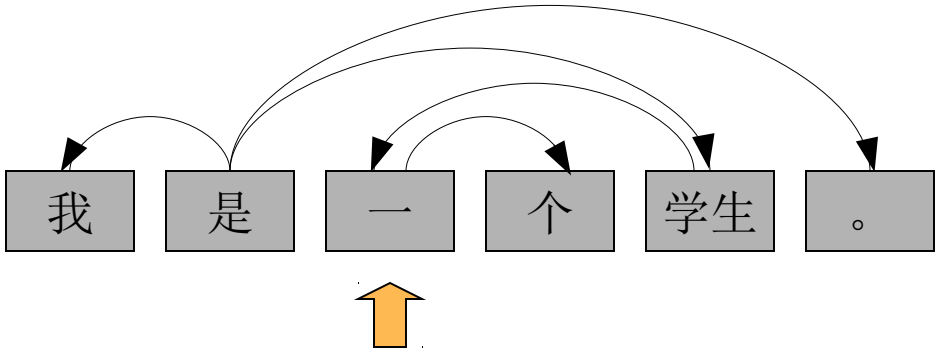


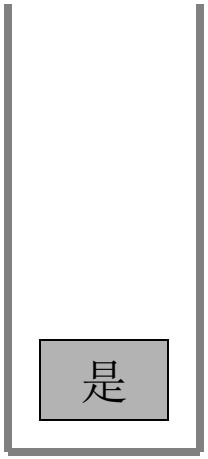


Left(我←是)

模拟分析

- ((我 /PN) 是 /VC ((一 /CD (个 /M)) 学生 /NN) (。 /PU))

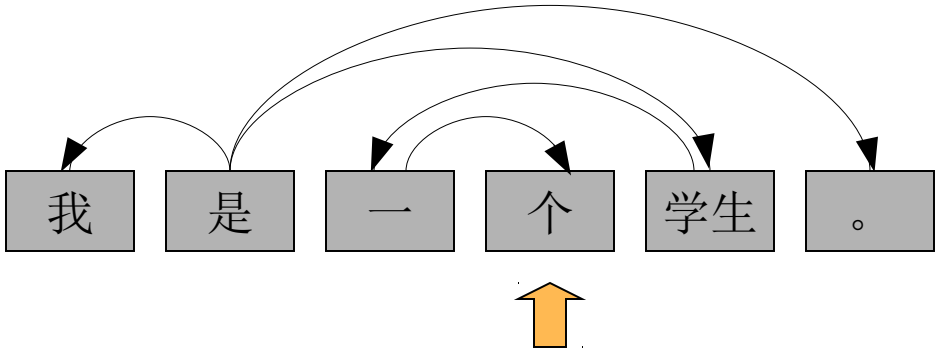


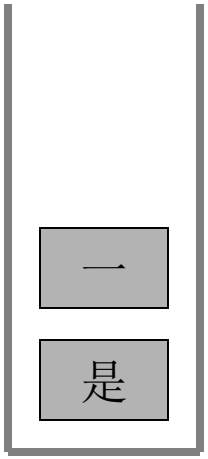


Shift(是)

模拟分析

- ((我 /PN) 是 /VC ((一 /CD (个 /M)) 学生 /NN) (。 /PU))

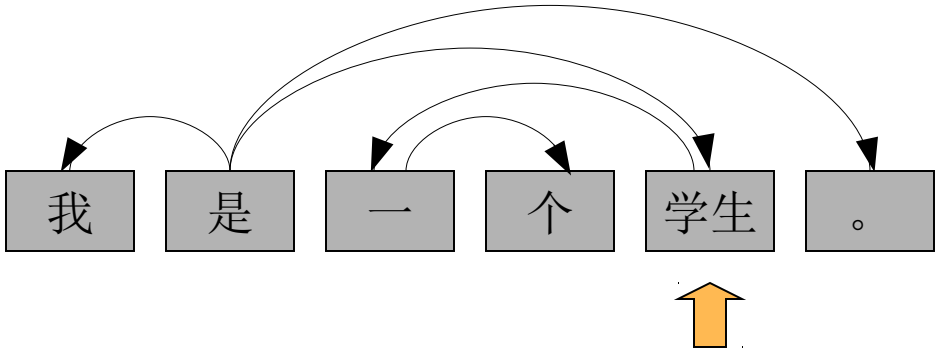


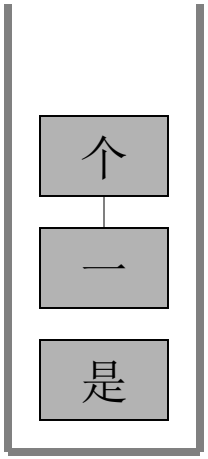


Shift(一)

模拟分析

- ((我 /PN) 是 /VC ((一 /CD (个 /M)) 学生 /NN) (。 /PU))

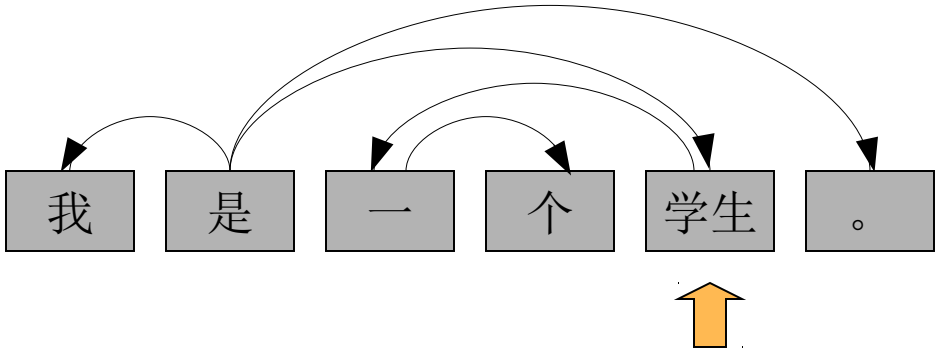


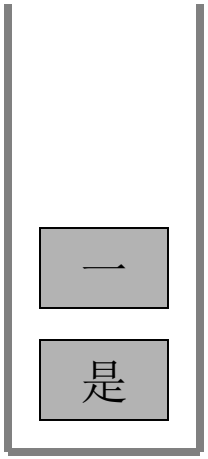


Right(一→个)

模拟分析

- ((我 /PN) 是 /VC ((一 /CD (个 /M)) 学生 /NN) (。 /PU))

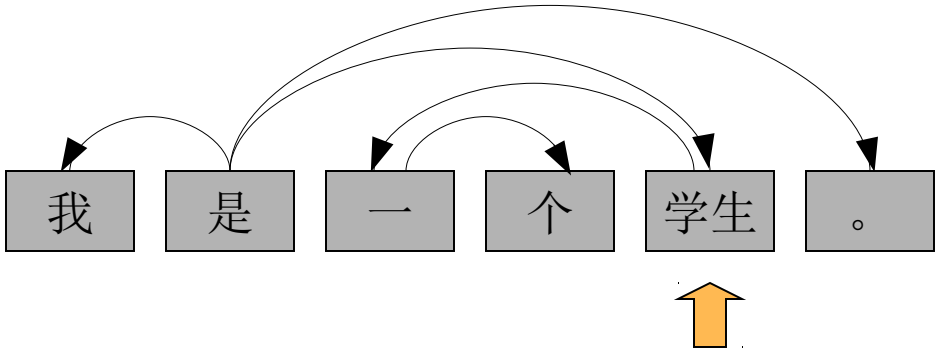


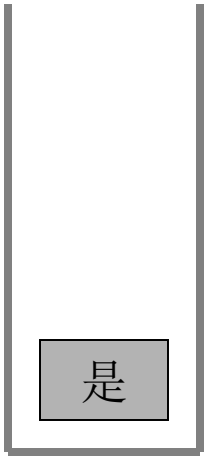


Reduce(个)

模拟分析

- ((我 /PN) 是 /VC ((一 /CD (个 /M)) 学生 /NN) (。 /PU))

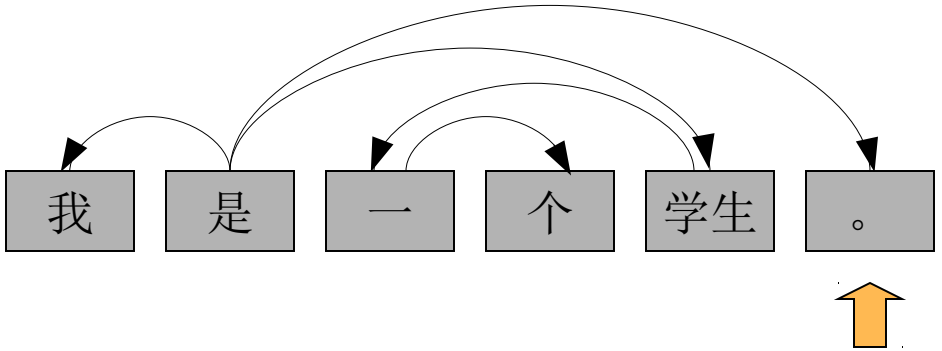


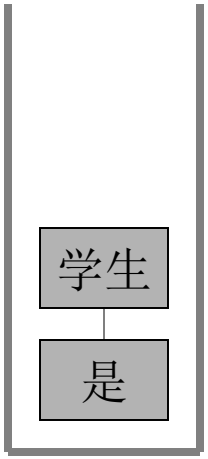


Left(一←学生)

模拟分析

- ((我 /PN) 是 /VC ((一 /CD (个 /M)) 学生 /NN) (。 /PU))

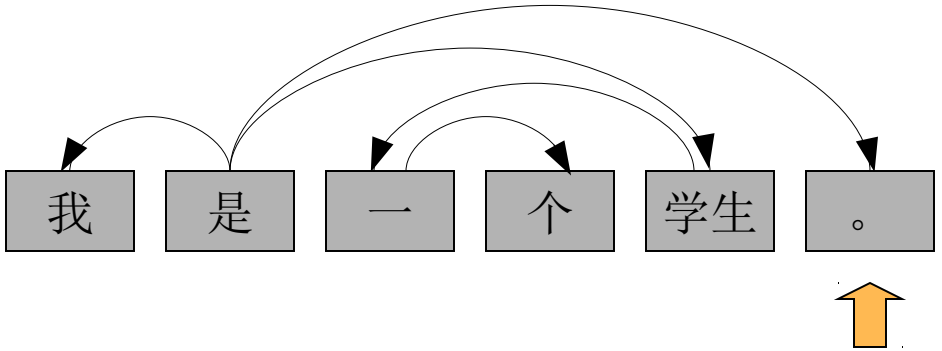


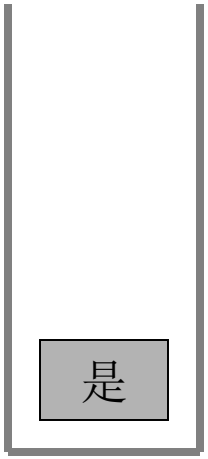


Right(是→学生)

模拟分析

- ((我 /PN) 是 /VC ((一 /CD (个 /M)) 学生 /NN) (。 /PU))

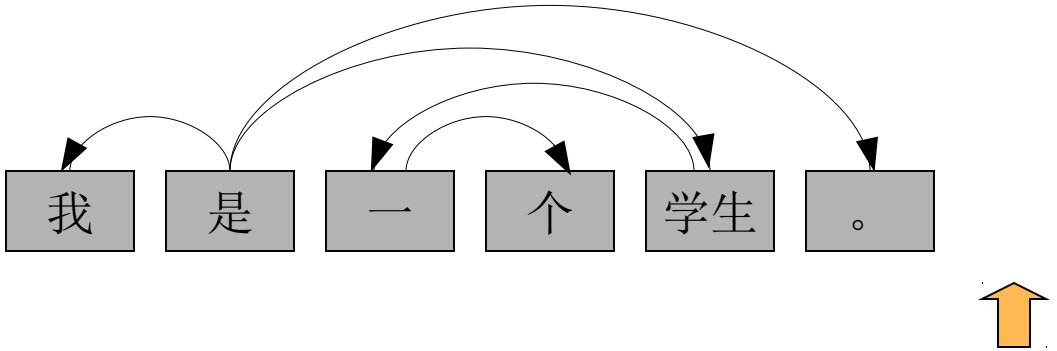


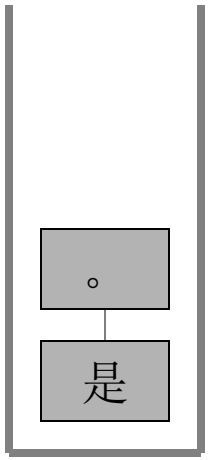


Reduce(学生)

模拟分析

- ((我 /PN) 是 /VC ((一 /CD (个 /M)) 学生 /NN) (。 /PU))

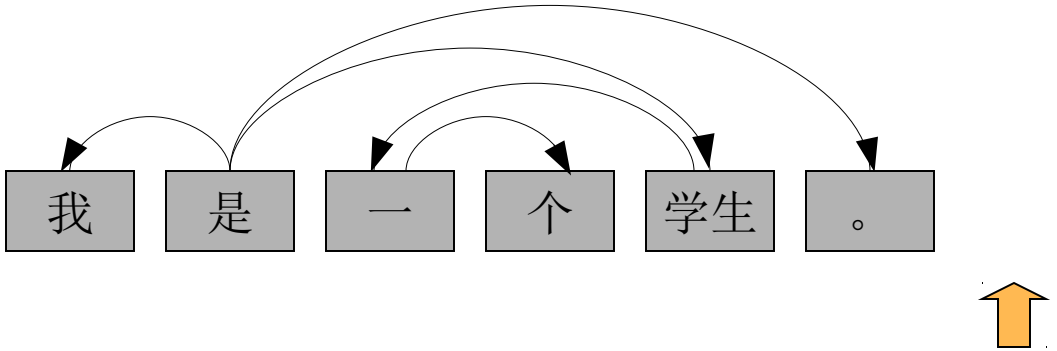


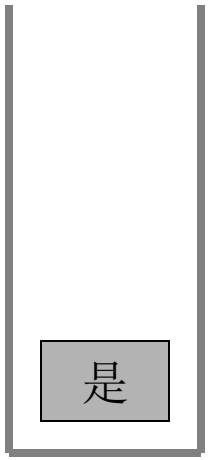


Right(是→。)

模拟分析

- ((我 /PN) 是 /VC ((一 /CD (个 /M)) 学生 /NN) (。 /PU))

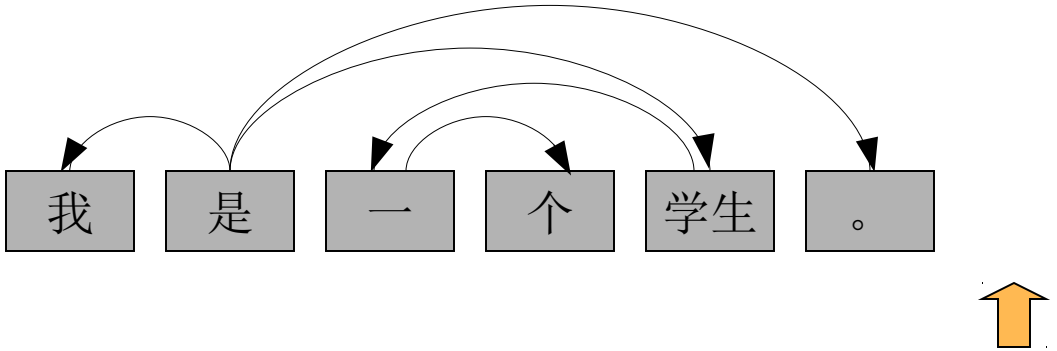


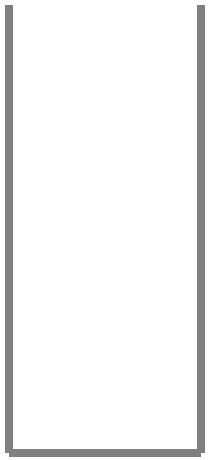


Reduce(。)

模拟分析

- ((我 /PN) 是 /VC ((一 /CD (个 /M)) 学生 /NN) (。 /PU))





Reduce(是)

nbest 全局训练

依存分析模型比较

- 生成式依存模型通过简单的极大似然估计即可完成训练，且模型较小。缺点是分析准确率较低
- 最大生成树模型和状态转移模型则需要在训练语料上进行多轮迭代以调节参数，训练耗时长且模型较大。优点是分析准确率高
- 目前流行的是最大生成树模型和状态转移模型中的移进规约模型。其中，最大生成树模型擅于确定远距离的依存关系，而移进归约模型则对近距离依存关系识别准确率更高