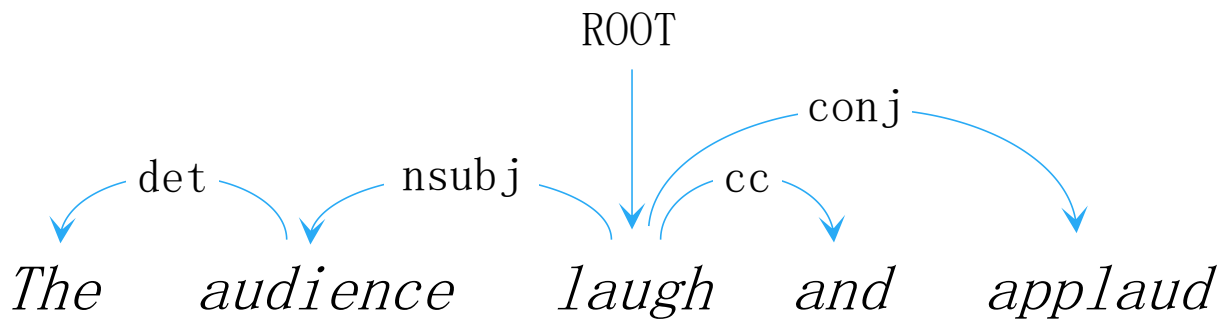
第11章：依存句法分析

课件制作：王新宇、周思锦、屠可伟
讲解人：王新宇、周思锦

依存句法分析（Dependency Parsing）

依存句法树

- 依存句法树是一棵有向树
  - 节点：句子中的词
    - 根节点（ROOT）：一个特殊的根节点
  - 边：两个词之间的依存关系
    - 依存关系可能有标签



依存关系

依存关系全称	英文
nsubj	nominal subject
casubj	clausal subject
dobj	direct object
iobj	indirect object
pobj	object of preposition
tmod	temporal modifier
appos	appositional modifier
det	determiner
prep	prepositional modifier

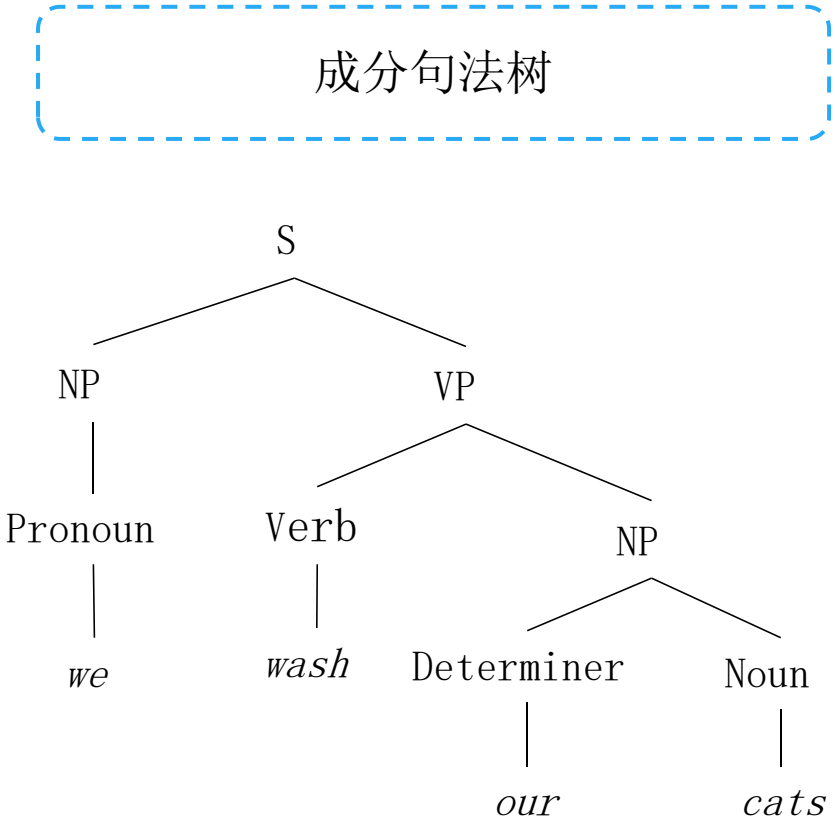
依存句法

优点

- 能更好地处理语序更自由的语言
  - 例如：捷克语（Czech），土耳其语（Turkish）
- 分析速度比成分句法分析更快
- 依存结构经常能直接反应下游应用中所需的句法关系
  - 基于成分句法分析的方法往往需要间接地从成分树中提取相同的信息

依存（Dependency）和成分（Constituency）

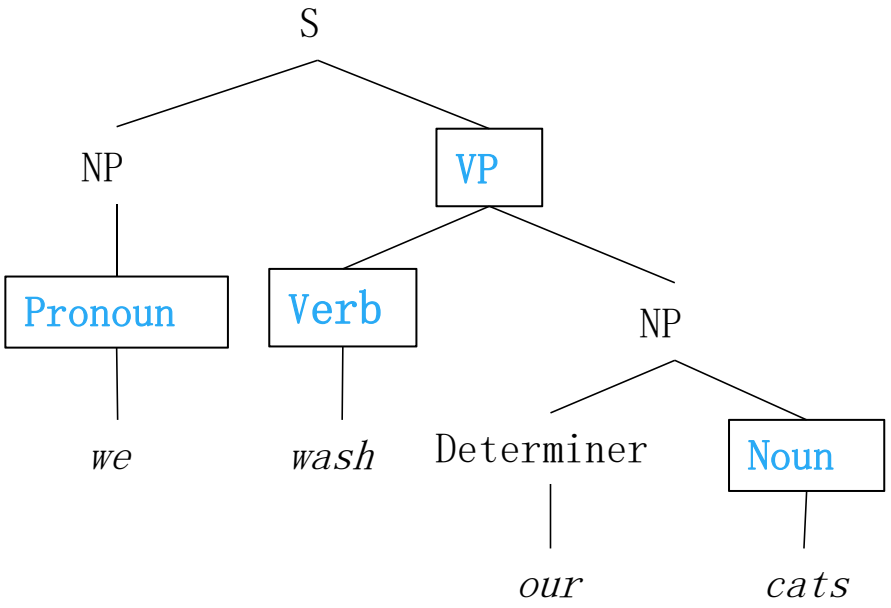
通常，我们可以从成分句法树（constituent parse tree）中提取出依存句法树（dependency parse tree）



依存（Dependency）和成分（Constituency）

通常，我们可以从成分句法树（constituent parse tree）中提取出依存句法树（dependency parse tree）

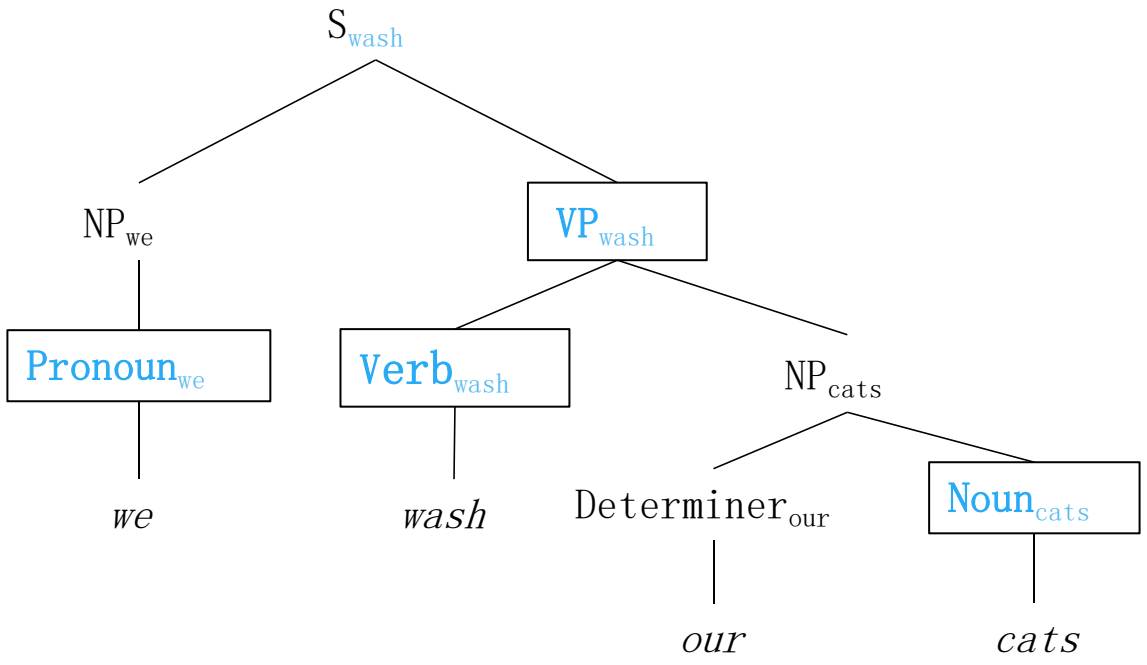
带有中心词的成分句法树



依存（Dependency）和成分（Constituency）

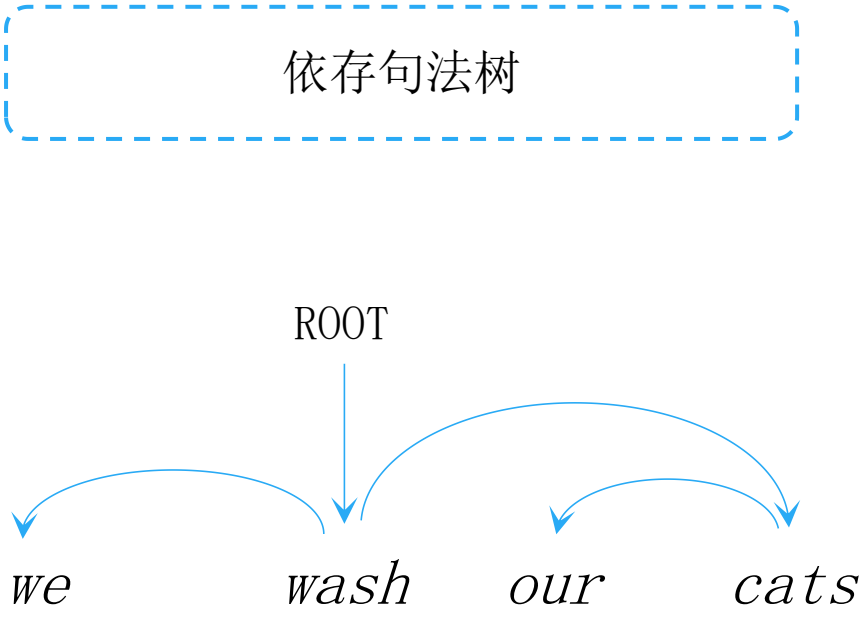
通常，我们可以从成分句法树（constituent parse tree）中提取出依存句法树（dependency parse tree）

带有中心词并确定中心词的成分句法树



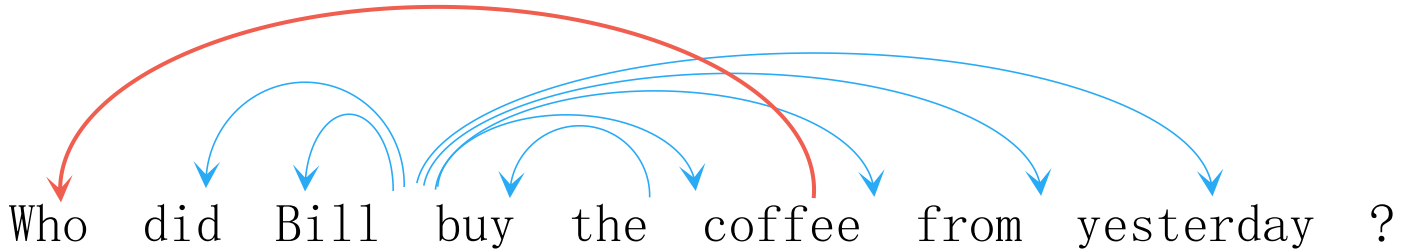
依存（Dependency）和成分（Constituency）

通常，我们可以从成分句法树（constituent parse tree）中提取出依存句法树（dependency parse tree）



投射性（Projectivity）

- 定义：假设句子中的单词呈线性排列，所有的依存边在单词之上，如果依存句法分析中不存在交叉依存边，则称其满足投射性
- 根据成分句法树提取出的依存句法树一定满足投射性
- 依存理论通常允许非投射性结构
  - 如果不用这些非投射性结构，将无法正确得到某些特定结构的语义



句法分析

- 句法分析（Parsing）是指对输入字符串赋予（最佳的）句法树（Parse Trees）的任务

算法

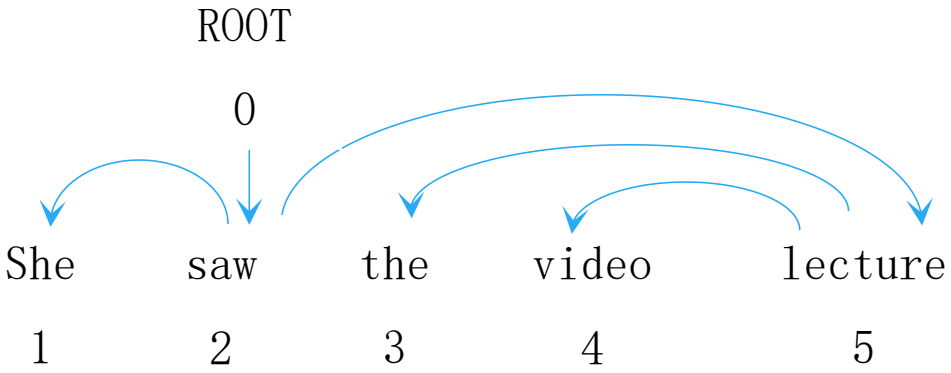
- 基于图的依存句法分析：MST，Eisner等
  - 可以找到全局最优解
- 基于转移的依存句法分析：arc-standard，arc-eager，arc-hybrid
  - 找到局部最优解，但速度更快

依存句法分析评价指标

- 无标签依存得分（unlabeled attachment score, UAS）
  - 测试集中找到其正确依存边（包括没有标注中心词的根节点）所占总词数的百分比

- 有标签依存得分（labeled attachment score, LAS）
  - 测试集中找到其正确依存边的词，并且依存关系类型也标注正确的词（包括没有标注中心词的根节点）占总词数的百分比

依存句法分析评价指标



$$Acc = \frac{\text{正确依存数}}{\text{总词数}}$$

UAS = 4/5 = 80%
LAS = 2/5 = 40%

标准分析

1	2	She	nsubj
2	0	saw	root
3	5	the	det
4	5	video	nn
5	2	lecture	obj

待分析分析

1	2	She	nsubj
2	0	saw	root
3	4	the	det
4	5	video	nsubj
5	2	lecture	ccomp

有监督方法

树库

- 通用依存树库（Universal Dependencies treebank）
- 截至2019年，UD中有超过100种树库，其中包括70多种不同语言

目标函数

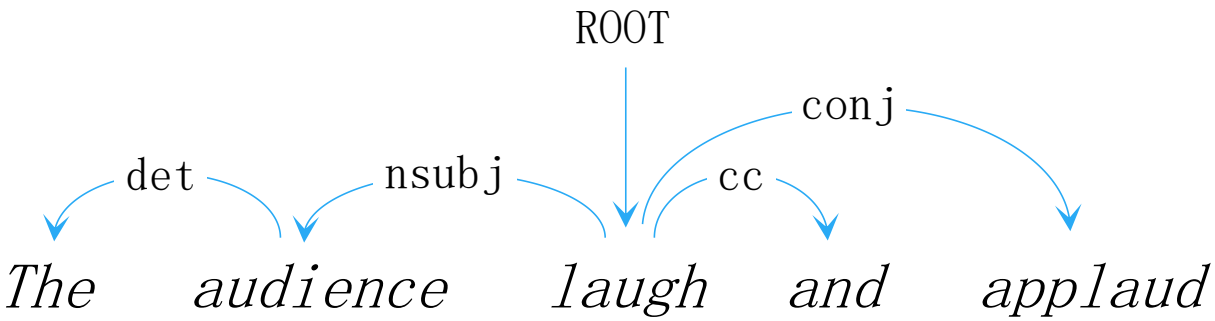
- 条件概率函数：P(parse | sentence)
- 基于边际距离的目标函数（margin-based）

优化方法

- 基于梯度的算法

依存句法分析

- 依存句法分析是一棵有向树
- 句法分析（Parsing）：输入字符串，输出最佳的句法树（Parse Trees）
- 例子：

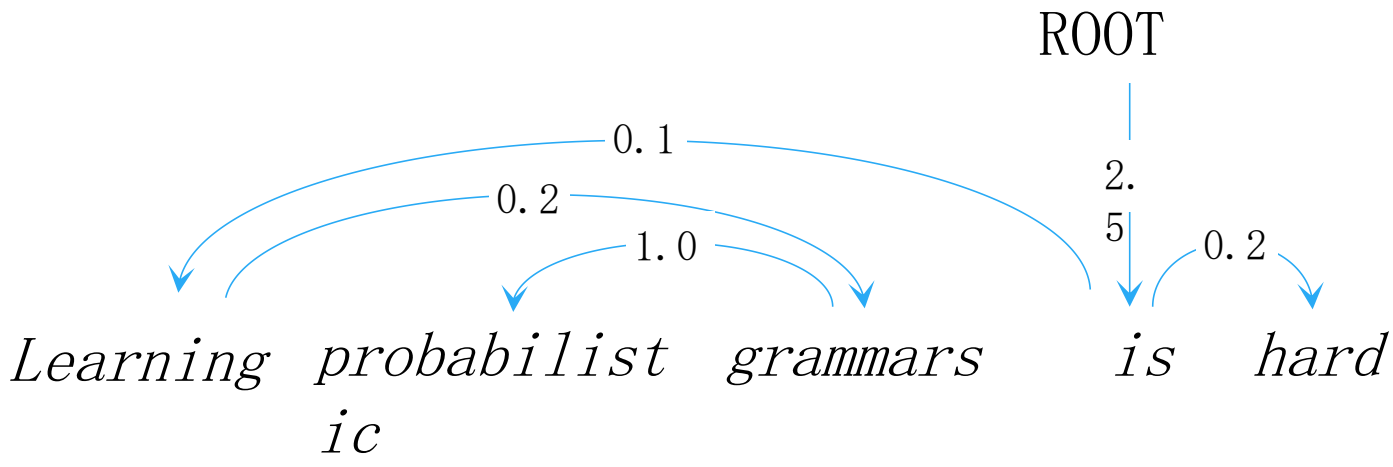


讲者口误，应是依存句法分析

一阶图依存句法分析

句法树分数

- 每条依存边都有一个分数。所有依存边的分数加起来就是句法树的分数
- 依存边的分数由它连接的两个单词的特征计算得到
  - 可能的特征：POS标签，相邻的单词，由LSTM计算的包含语境信息的词嵌入（embeddings）



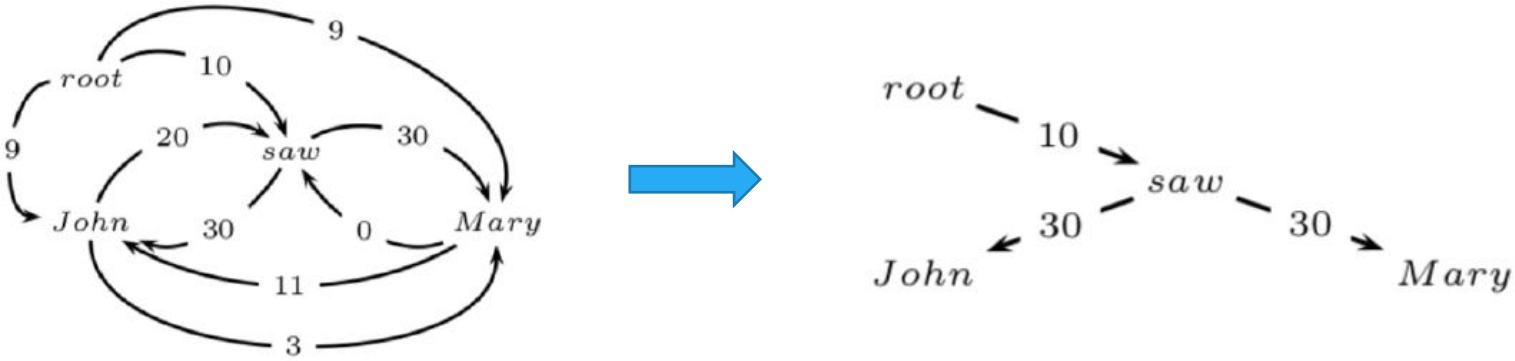
一阶图依存句法分析

句法树分数

- 每条依存边都有一个分数。所有依存边的分数加起来就是句法树的分数
- 依存边的分数由它连接的两个单词的特征计算得到

最大生成树

- 更准确的说法是生成树状图（spanning arborescence）



一阶图依存句法分析

句法树分数

- 每条依存边都有一个分数。所有依存边的分数加起来就是句法树的分数
- 依存边的分数由它连接的两个单词的特征计算得到

最大生成树

- 更准确的说法是生成树状图（spanning arborescence）

依存标签

- 对每一条依存边，只考虑它分数最高的标签

两类分析算法

非投射依存句法分析

- 允许存在交叉依存边（arc-crossing）
- Chu-Liu-Edmonds’ 算法
- 时间复杂度：最佳实现方式为$O(n^2 + n\log n)$

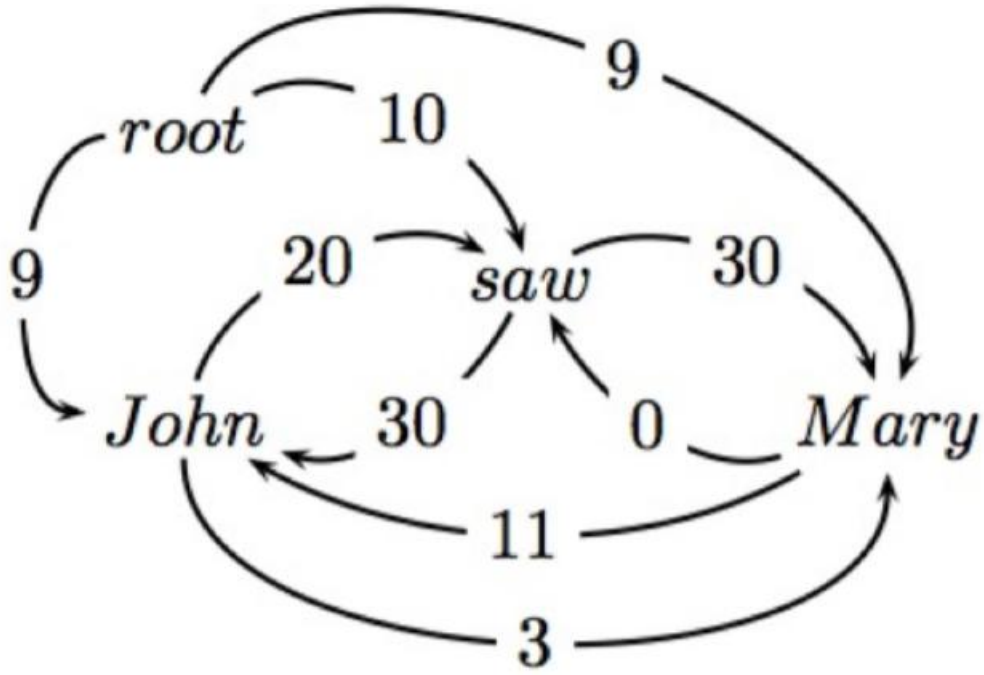
投射依存句法分析

- 不允许存在交叉依存边（arc-crossing）
- Eisner’s算法
- 时间复杂度：$O(n^3)$

讲者口误，应是依存句法分析

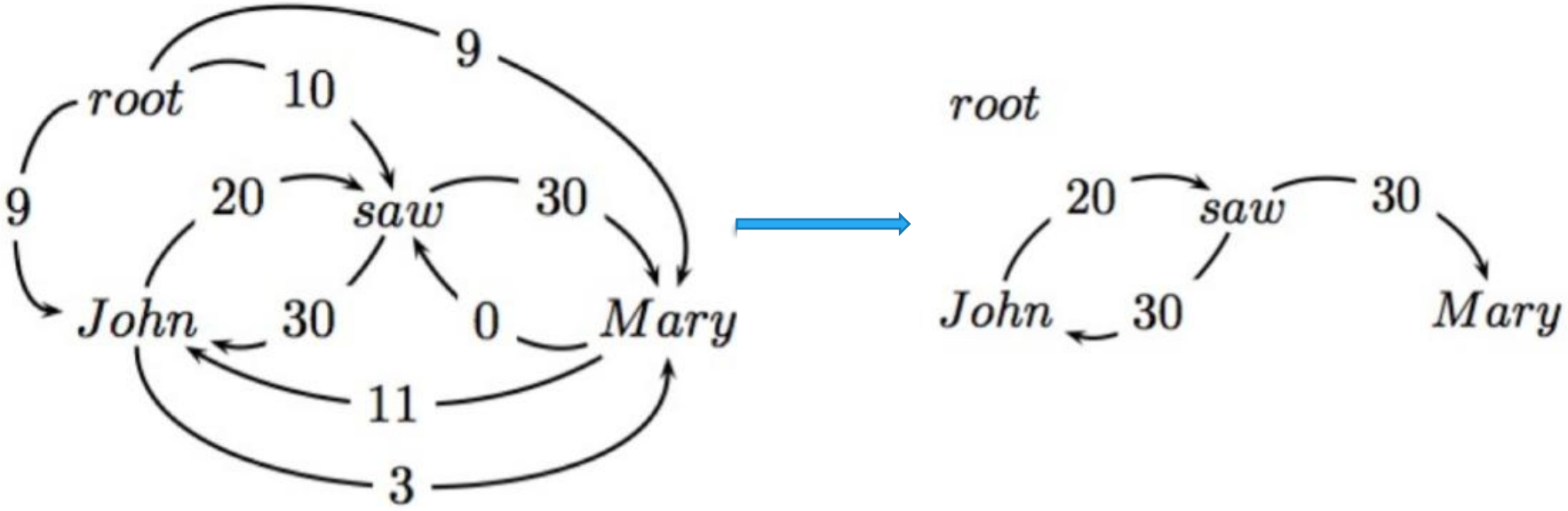
Chu-Liu-Edmonds’ 算法

- 待分析字符串：x = root John saw Mary



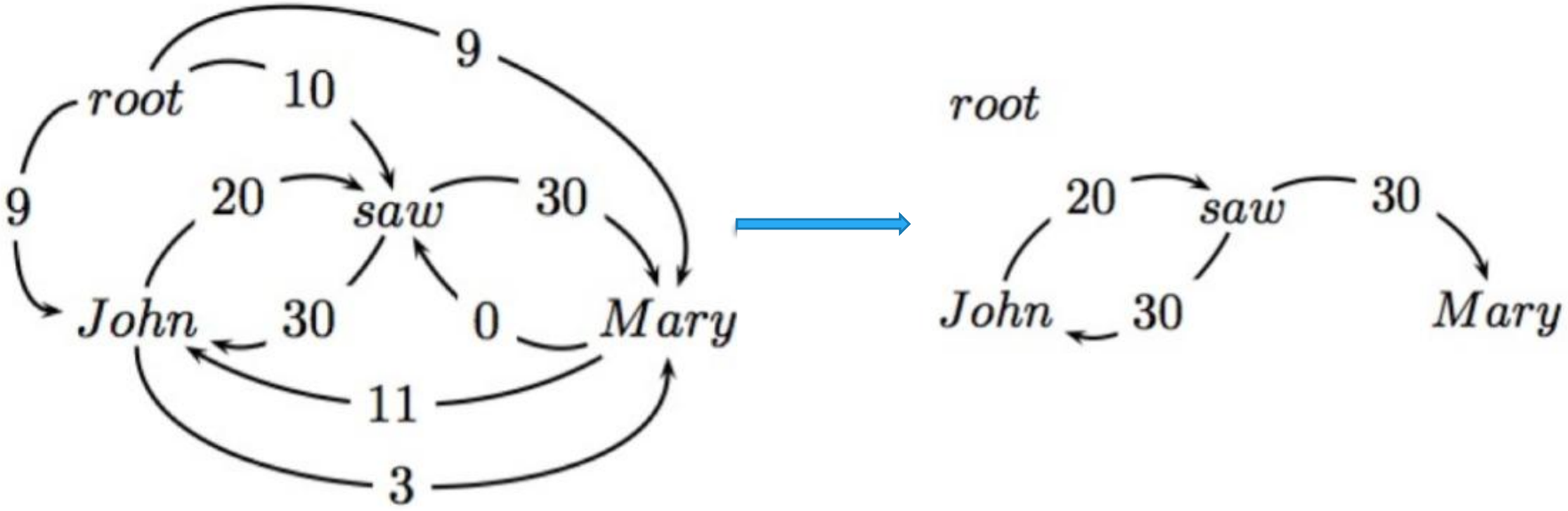
Chu-Liu-Edmonds’ 算法

- 为每个节点找到分数最高的入边



Chu-Liu-Edmonds’ 算法

- 为每个节点找到分数最高的入边

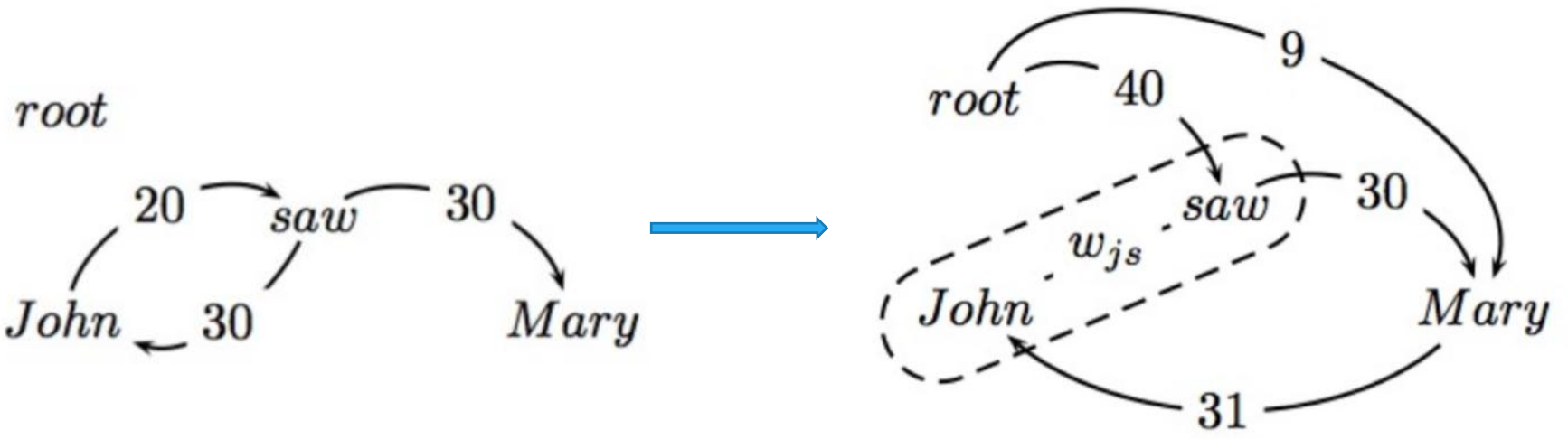


- 如果生成图是一棵树，我们就找到了最大生成树

为什么？

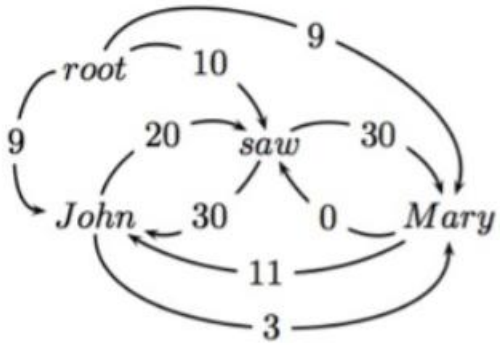
Chu-Liu-Edmonds’ 算法

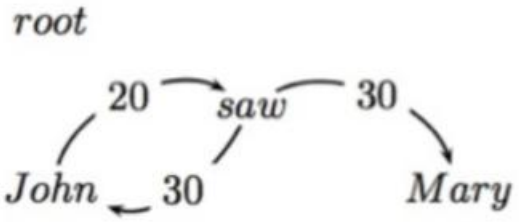
- 如果生成的图不是一棵树，寻找自环，并对环进行收缩
- 重新计算入环和出环的所有边的分数

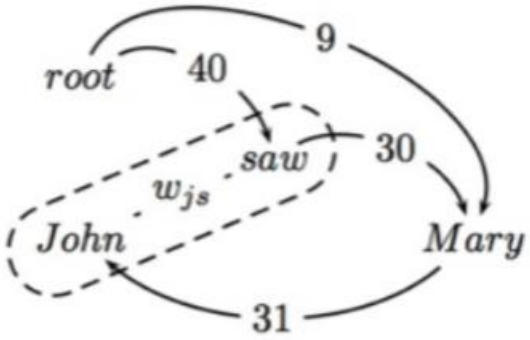


Chu-Liu-Edmonds’ 算法

- 出环边的分数：等于环内所有节点中出边分数最高的值
  - 例如：John->Mary = 3，saw->Mary = 30

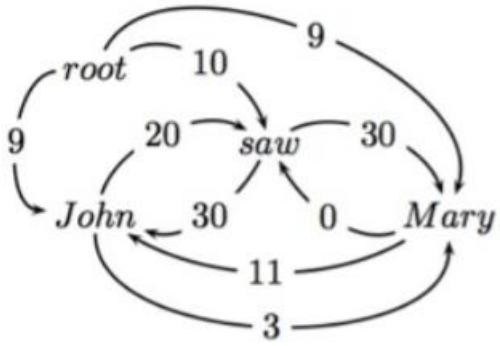


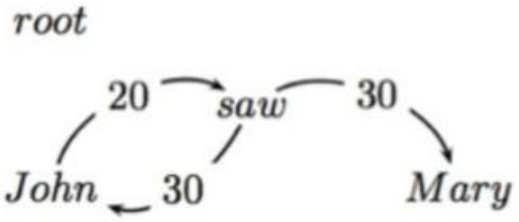


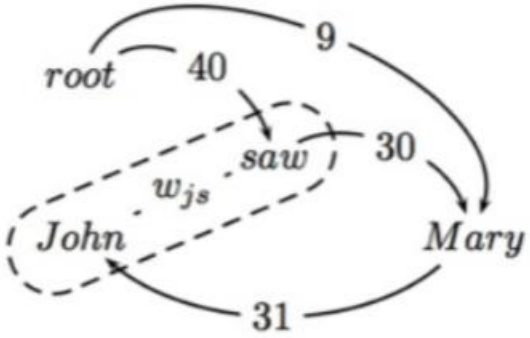


Chu-Liu-Edmonds’ 算法

- 入环边的分数：等于包括传入边的源点和环中的所有节点所构成的子图的最佳生成树的权重
  - 例如：root->saw->John = 40, root->John->saw = 29

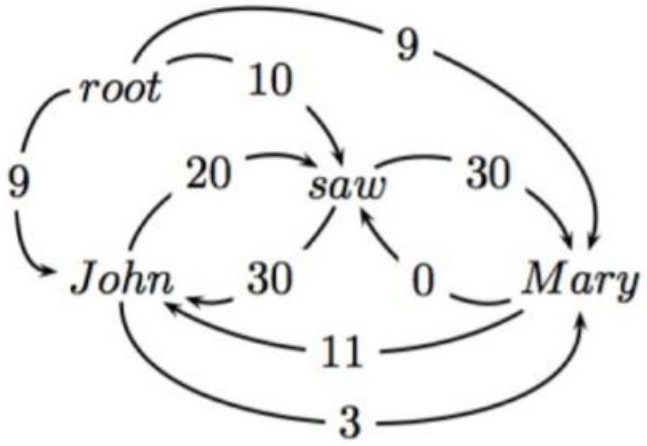


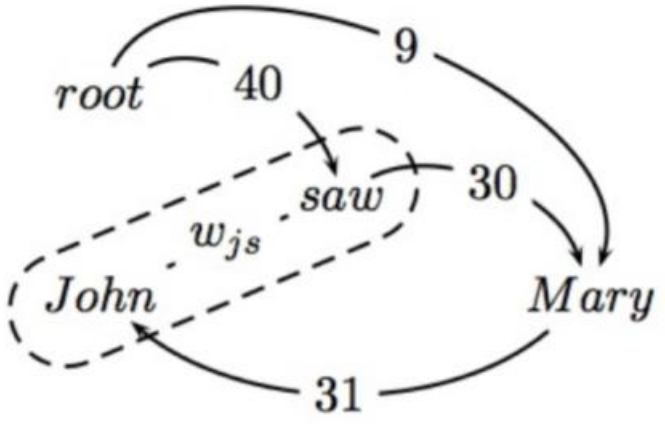




Chu-Liu-Edmonds’ 算法

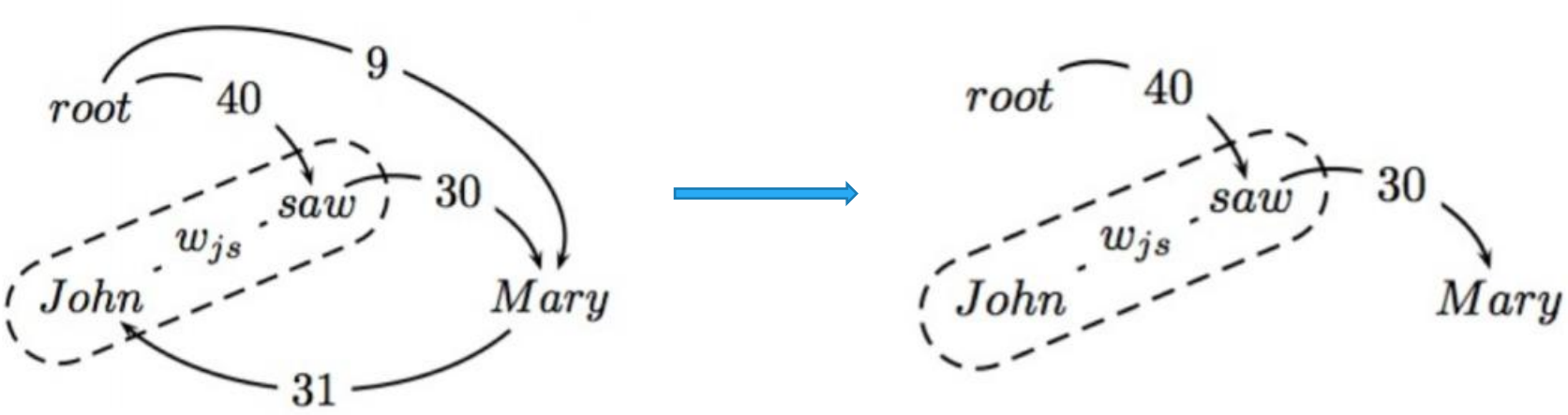
- 定理：收缩图的MST权值等于原图的MST权值
- 因此，在新的图上递归调用算法





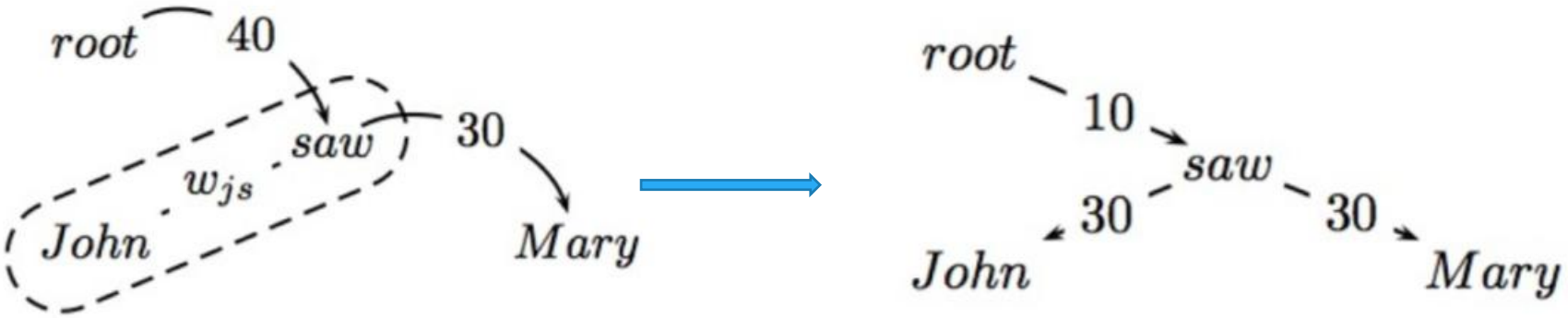
Chu-Liu-Edmonds’ 算法

- 下图是收缩图的MST



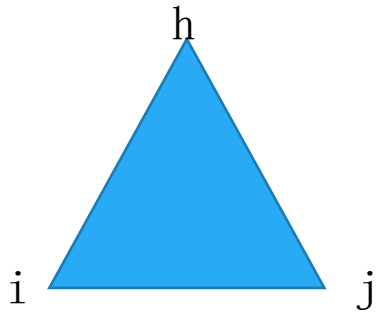
Chu-Liu-Edmonds’ 算法

- 倒推出原图的MST



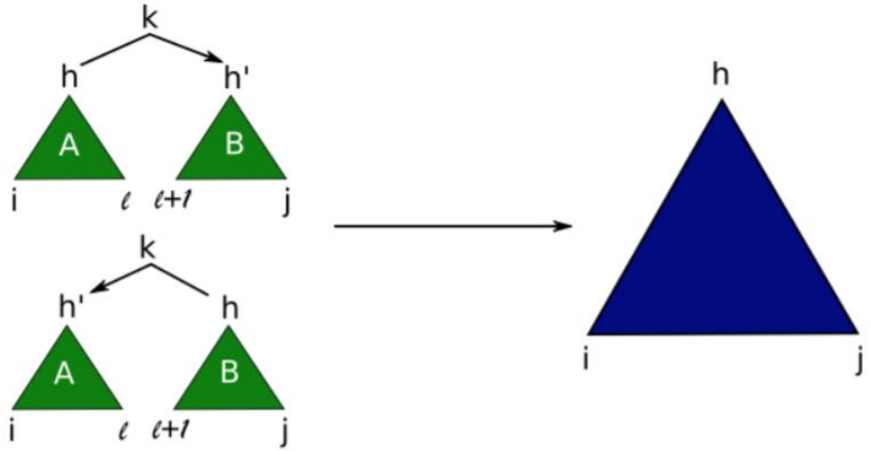
投射性分析

- 可以使用类似成分句法分析的分析算法
- 每个签名（Signature）代表以词h为根涵盖从i到j所有单词的最佳生成树的权值
  - 在CYK算法中，我们有签名[i， j， C]，C是非终结符
  - 这里我们有签名[i， j， h]，h是短语的中心词
- 基础情况（base case）
  - h=i=j：权值=1
- 目标
  - 以root为根，涵盖从0到n所有单词的签名：[0,n,root]



投射依存分析

- 使用CYK，自底向上算法
  - 首先处理长度为1的字符串，然后是长度为2的……

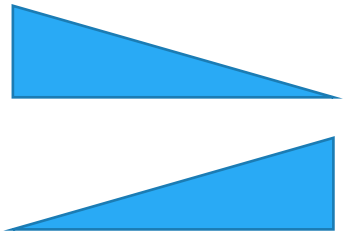


- 新签名的权值[i, j, h]：$\max_{l,h',k} w(A) + w(B) + w^{k}_{h,h'}$
- 计算完成后，使用后向指针重建树（像CYK算法一样）
- 时间复杂度：$O(|L|n^5)$

Eisner’s 算法 （具有投射性的树）

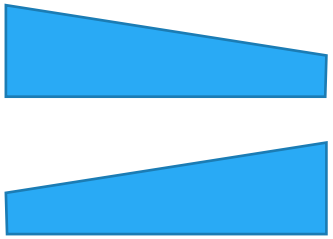
三角形

- 一棵子树，高边是根节点，其余单词均为其子孙节点，且它们的孩子均已在当前子树中



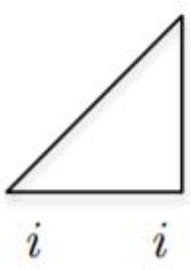
梯形

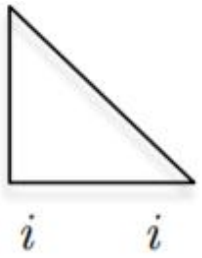
- 一棵子树，高边是根节点，矮边是其孩子节点，且矮边节点在它这一边仍可能有新的孩子节点



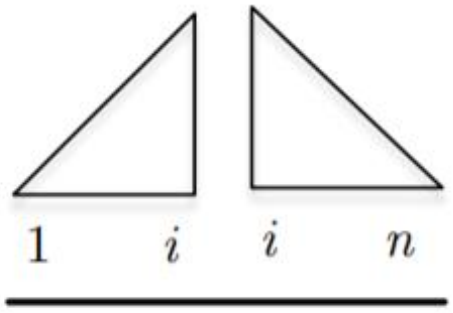
Eisner’s 算法

初始化:





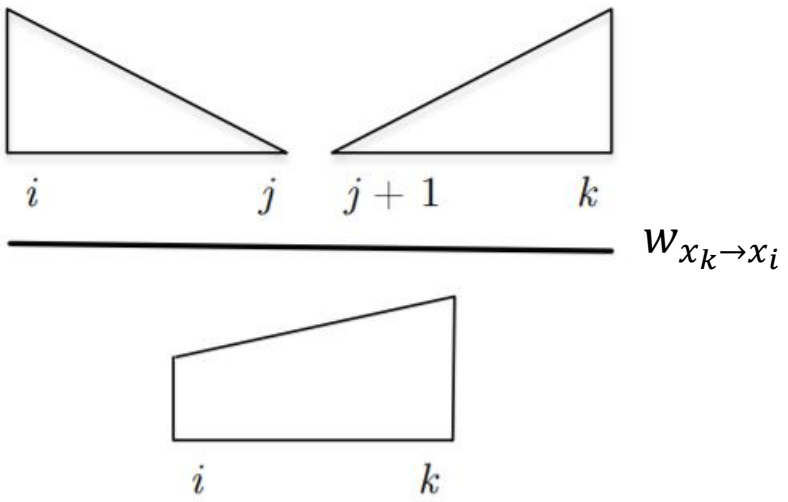
目标:



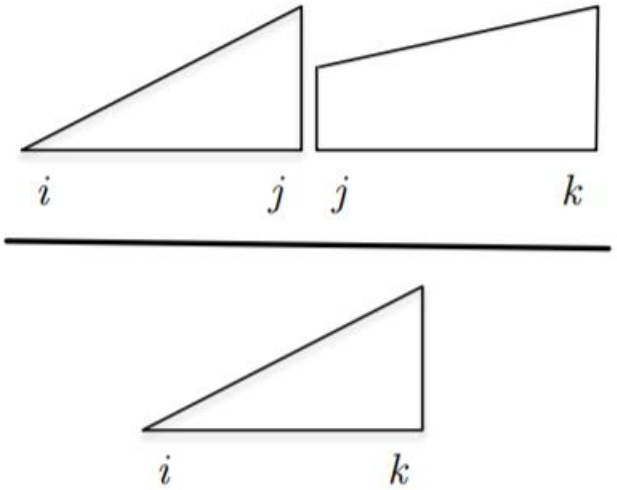
$w_{root \to x_i}$

Eisner’s 算法

附加左依存：

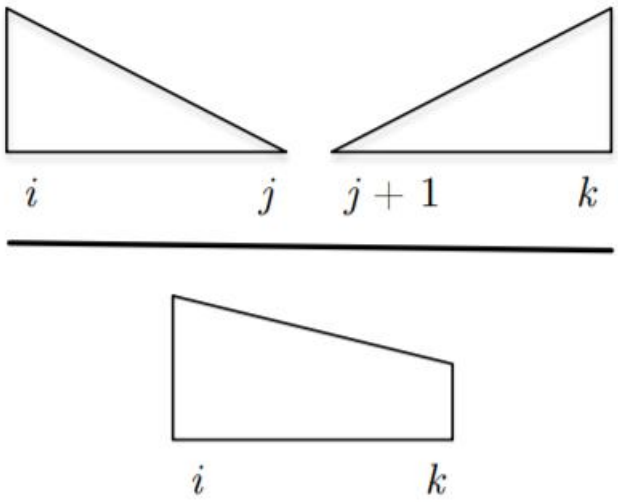


得到左孩子：



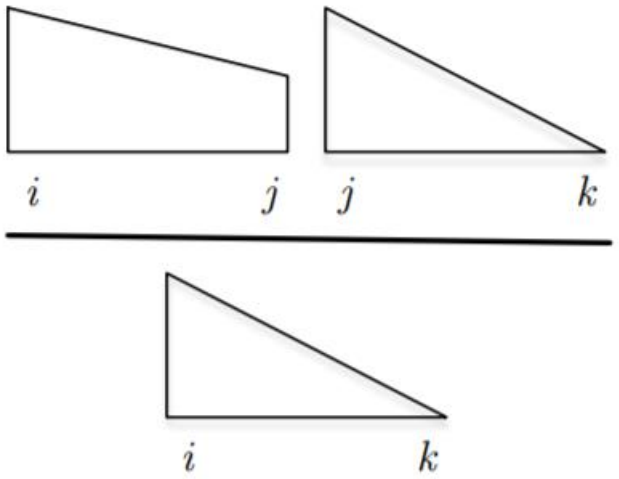
Eisner’s 算法

附加右依存：



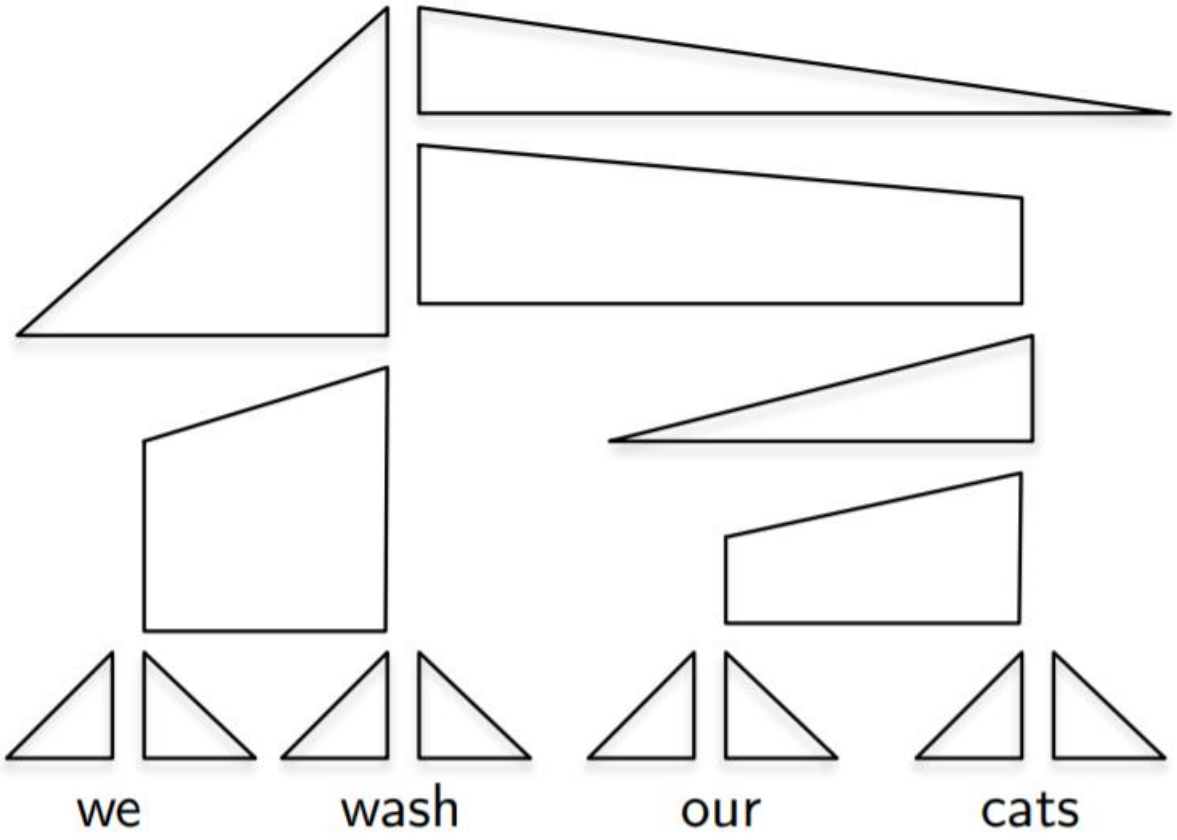
$w_{x_i \to x_k}$

得到右孩子：



Eisner’s 算法

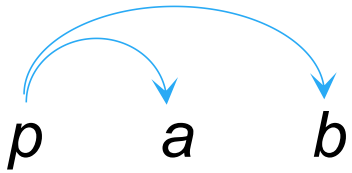
最终结果：



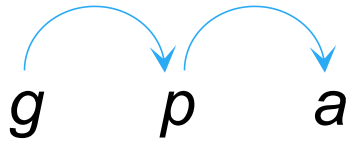
二阶图依存句法分析

句法树分数

- 每一对相连的边都有一个分数
- 树的分数是边对分数的总和



siblings

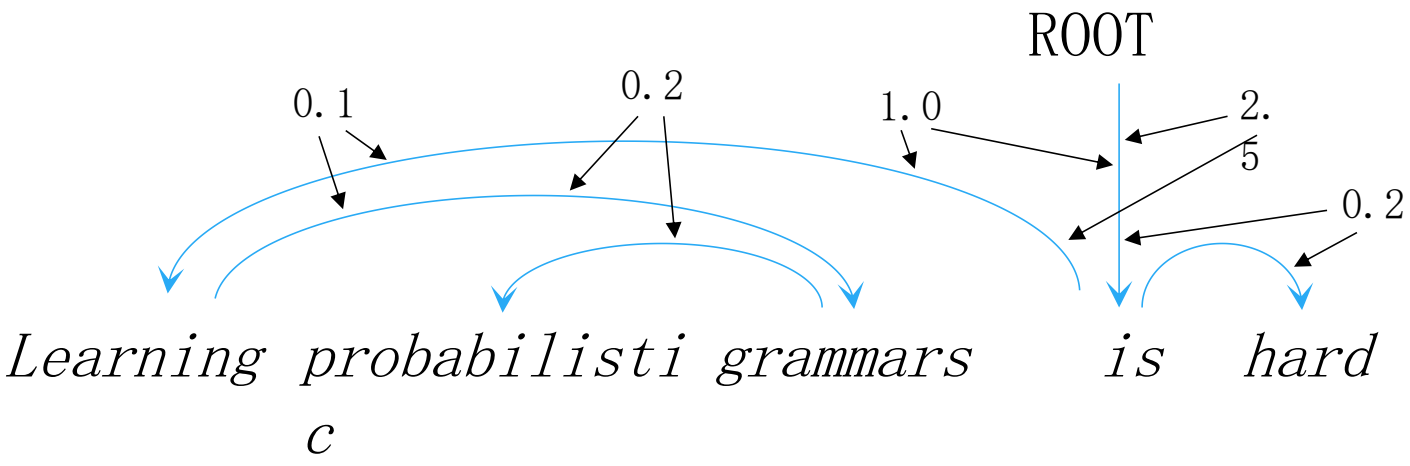


grandparent

二阶图依存句法分析

句法树分数

- 每一对相连的边都有一个分数
- 树的分数是边对分数的总和



二阶图依存句法分析

句法树分数

- 每一对相连的边都有一个分数
- 树的分数是边对分数的总和

分析

- 时间复杂度：
  - 投射性依存句法分析：$O(m^4)$
  - 非投射性依存句法分析：NP-hard
- 存在近似算法

基于转移的分析方法

- 依存树表示为一系列的转移（transitions）
- 转移（transitions）：在栈S（Stack）和 缓存区 B（Buffer）上执行的简单动作
- 在分析过程中，应用分类器决定下一个要执行的转移动作。贪心法；不回溯。

基于转移的分析方法：转移

分析状态

- 初始状态：缓存区包含待分析的字符串x，栈里只包含依存句法根ROOT
- 最终状态：缓存区为空，栈里包含整棵句法树

arc-standard 转移机制

- SHIFT：缓存区B队首的单词进入栈S
- RIGHT-ARC：u = pop(S)； v = pop(S)； push(S,v->u)
- LEFT-ARC： u = pop(S)； v = pop(S)； push(S,v<-u)

- 对于带标签分析，在RIGHT-ARC和LEFT-ARC动作中添加标签

基于转移的分析方法：示例

Buffer B:

we
vigorously
wash
our
cats
who
stink

Stack S:

ROOT

- 转移动作（actions）：

基于转移的分析方法：示例

Stack S:

we
ROOT

Buffer B:

vigorously
wash
our
cats
who
stink

- 转移动作（actions）： SHIFT

基于转移的分析方法：示例

Stack S:

vigorously
we
ROOT

Buffer B:

wash
our
cats
who
stink

- 转移动作（actions）：SHIFT， SHIFT

基于转移的分析方法：示例

Stack S:

wash
vigorously
we
ROOT

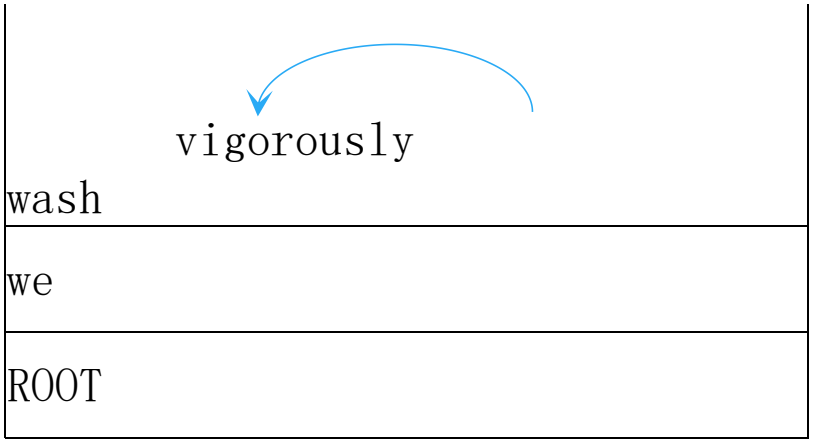
Buffer B:

our
cats
who
stink

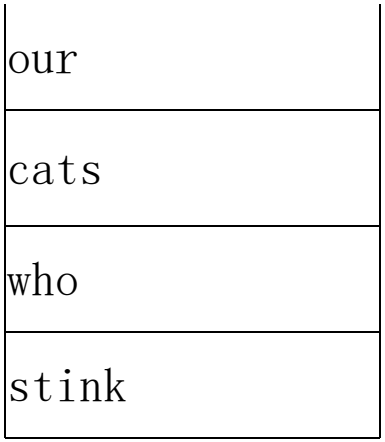
- 转移动作（actions）：SHIFT， SHIFT， SHIFT

基于转移的分析方法：示例

Stack S:

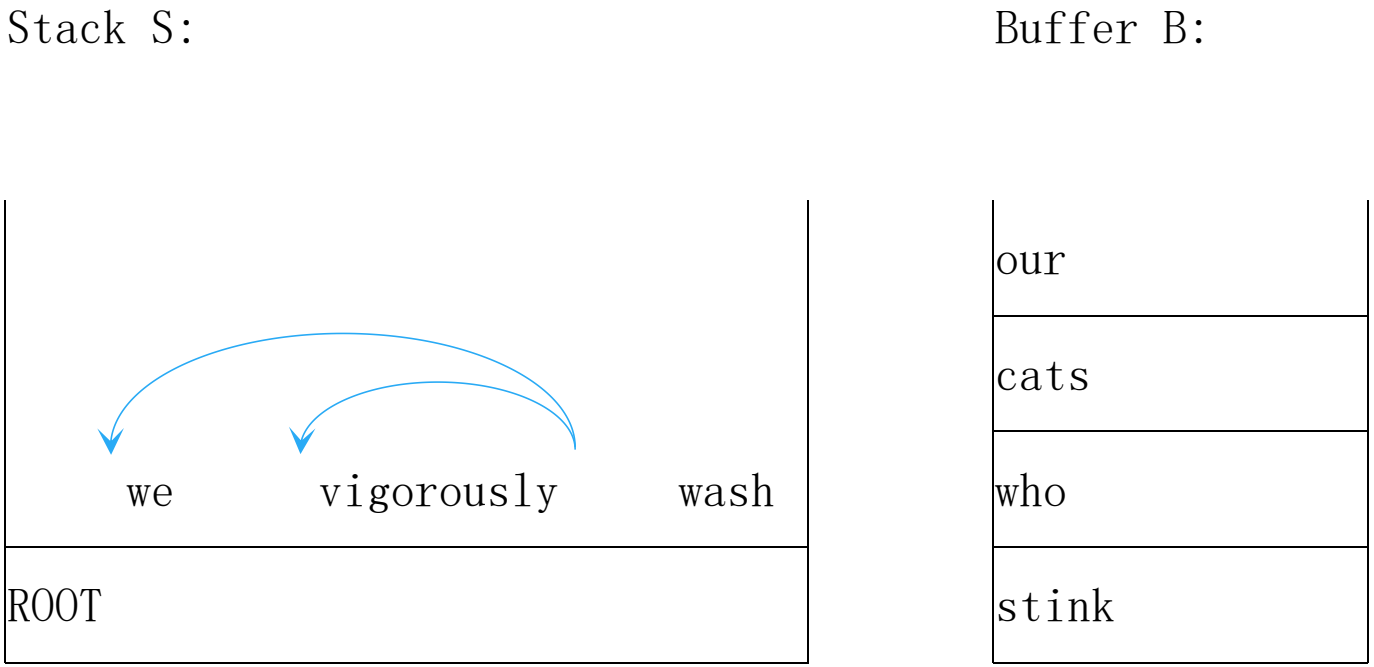


Buffer B:



- 转移动作（actions）：SHIFT， SHIFT， SHIFT，LEFT-ARC

基于转移的分析方法：示例

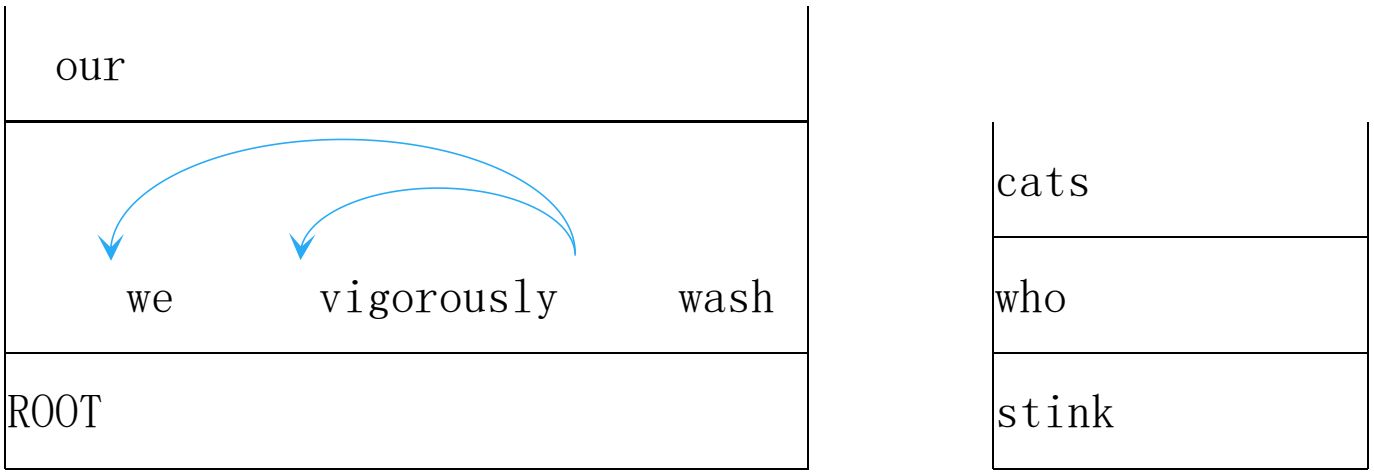


- 转移动作（actions）：SHIFT， SHIFT， SHIFT，LEFT-ARC，LEFT-ARC

基于转移的分析方法：示例

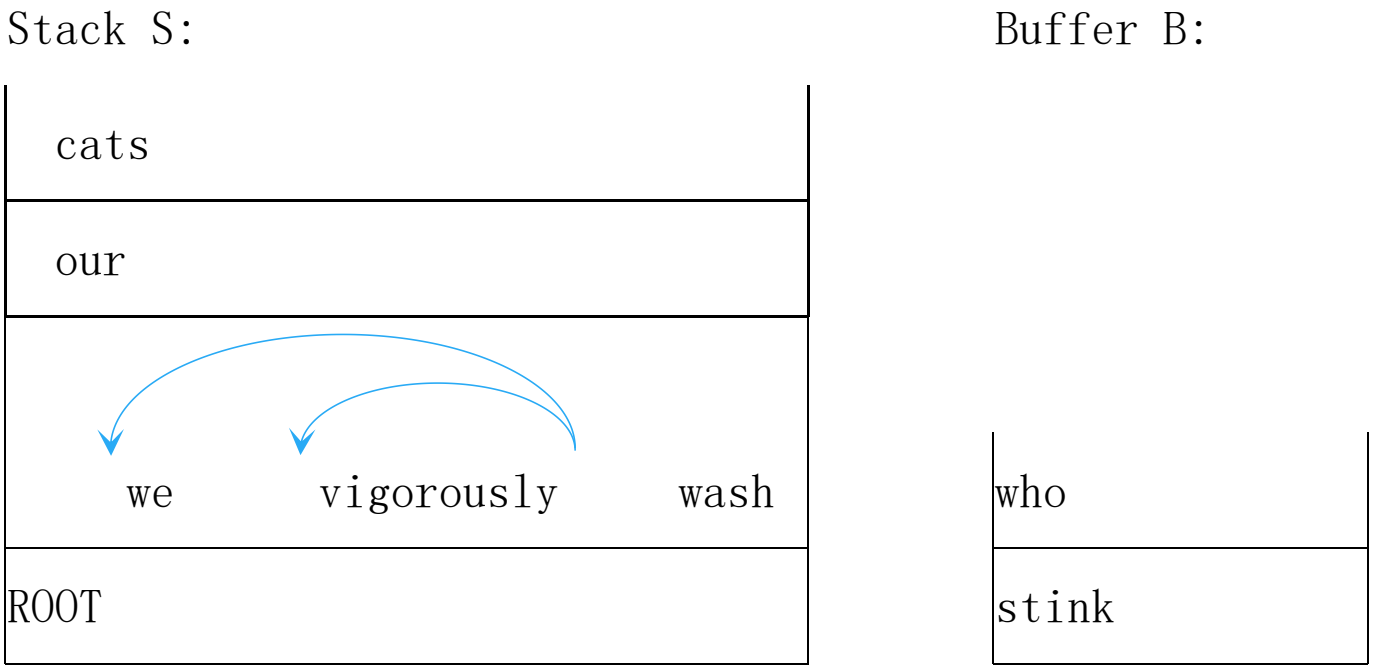
Stack S:

Buffer B:



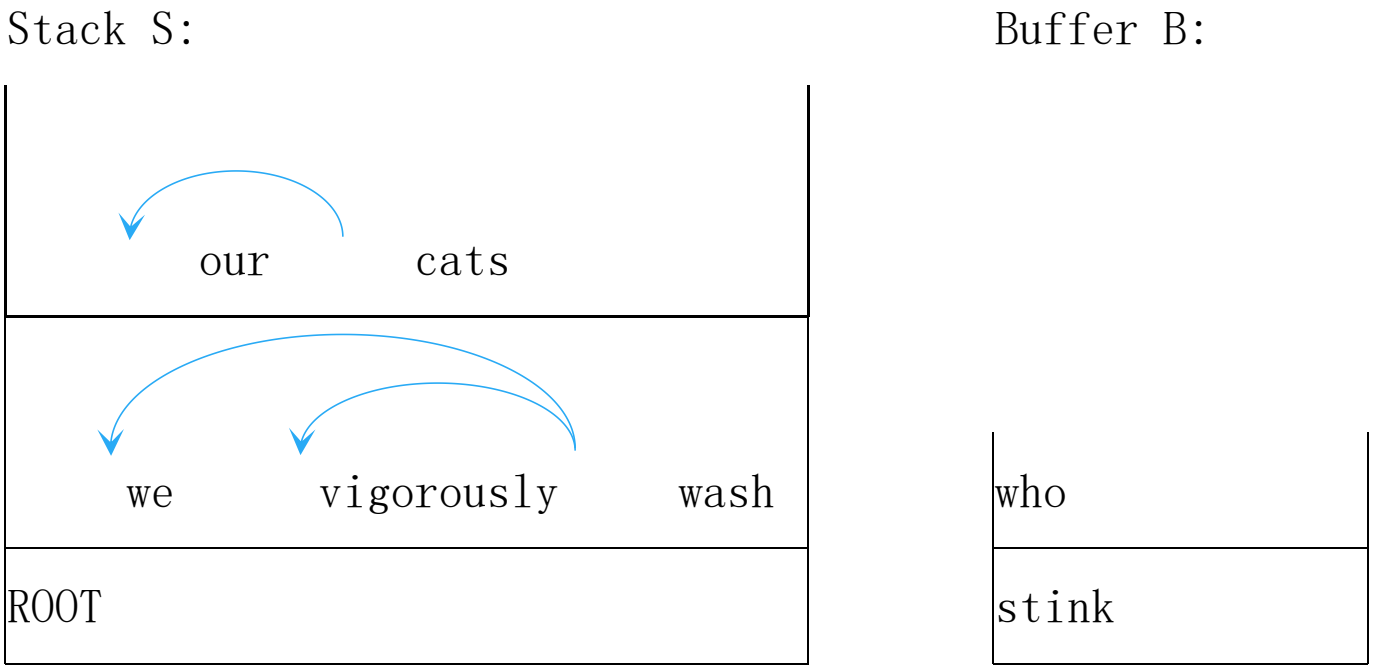
- 转移动作（actions）：SHIFT， SHIFT， SHIFT，LEFT-ARC，LEFT-ARC， SHIFT

基于转移的分析方法：示例



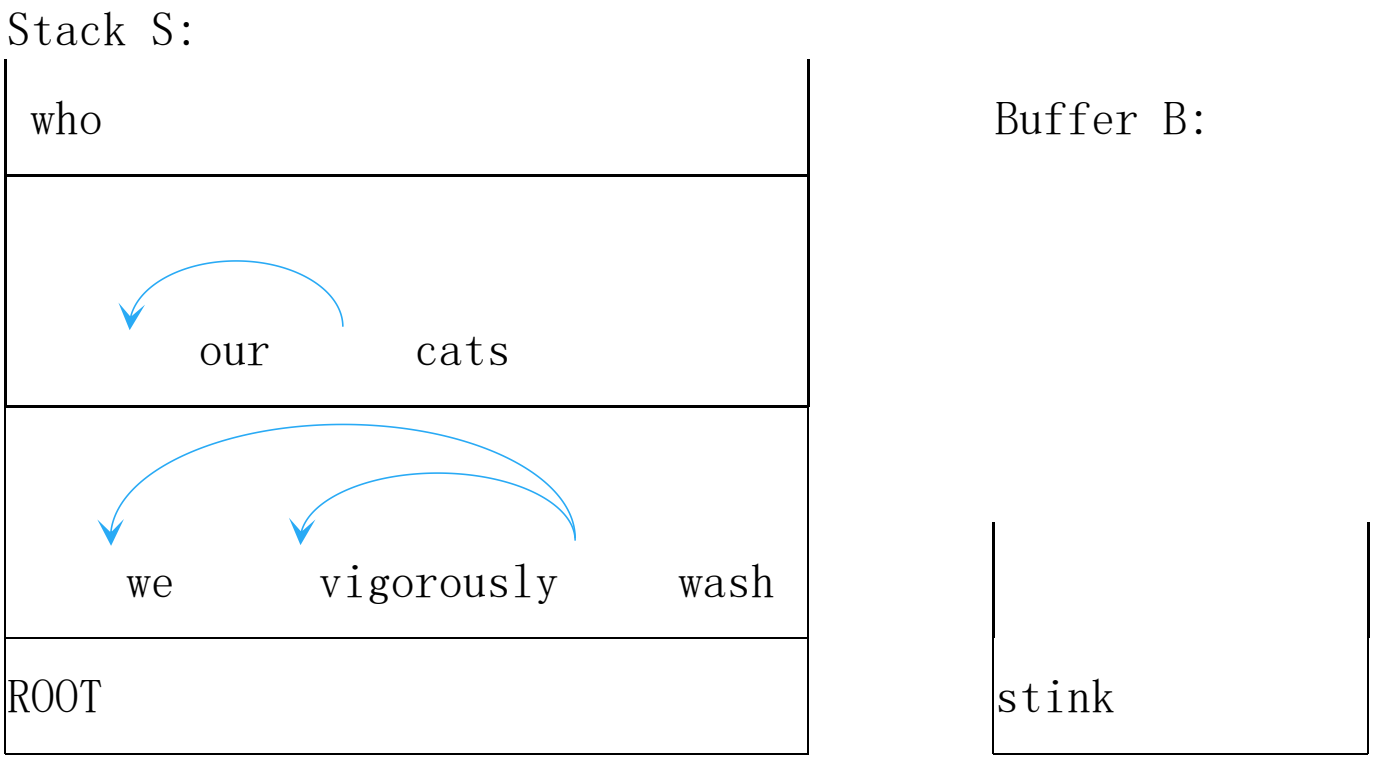
- 转移动作（actions）：SHIFT， SHIFT， SHIFT，LEFT-ARC，LEFT-ARC，SHIFT， SHIFT

基于转移的分析方法：示例



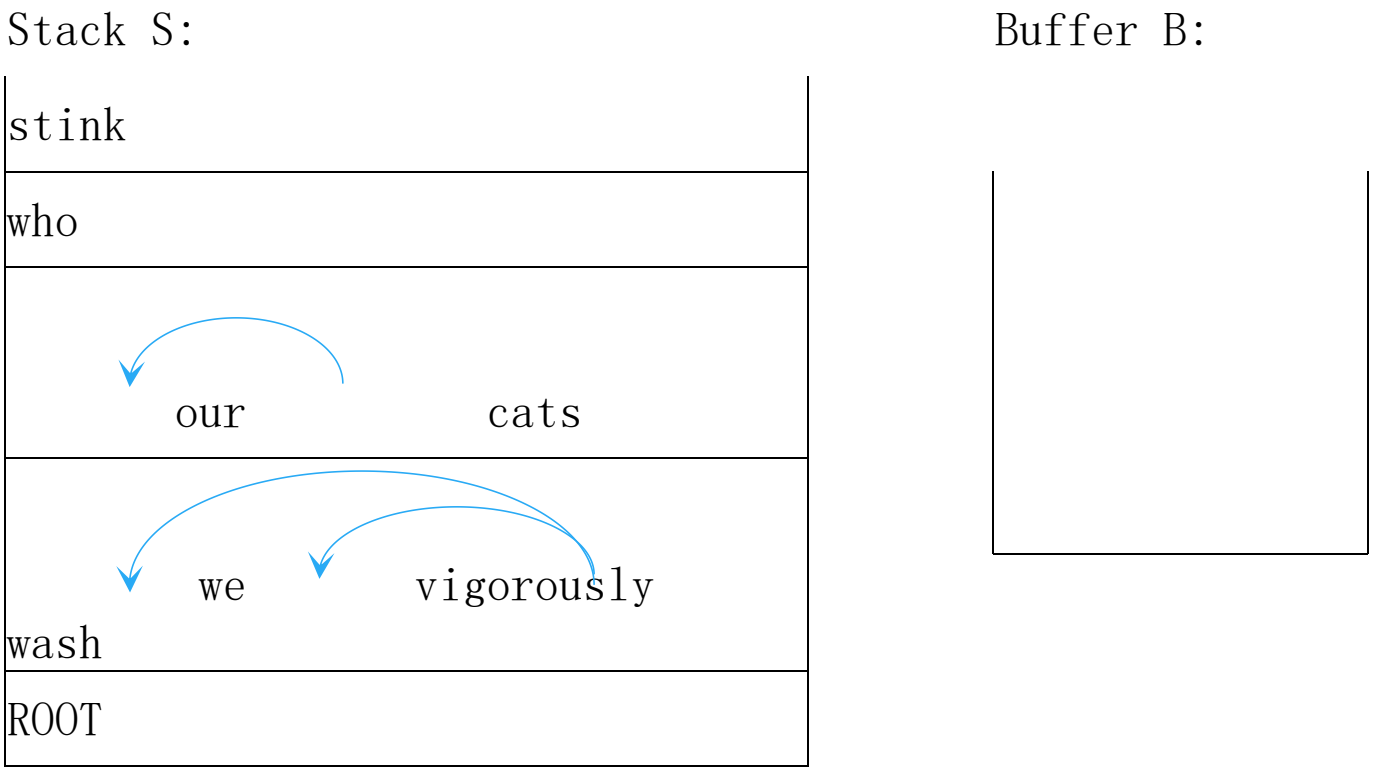
- 转移动作（actions）：SHIFT， SHIFT， SHIFT，LEFT-ARC，LEFT-ARC，SHIFT， SHIFT，LEFT-ARC

基于转移的分析方法：示例



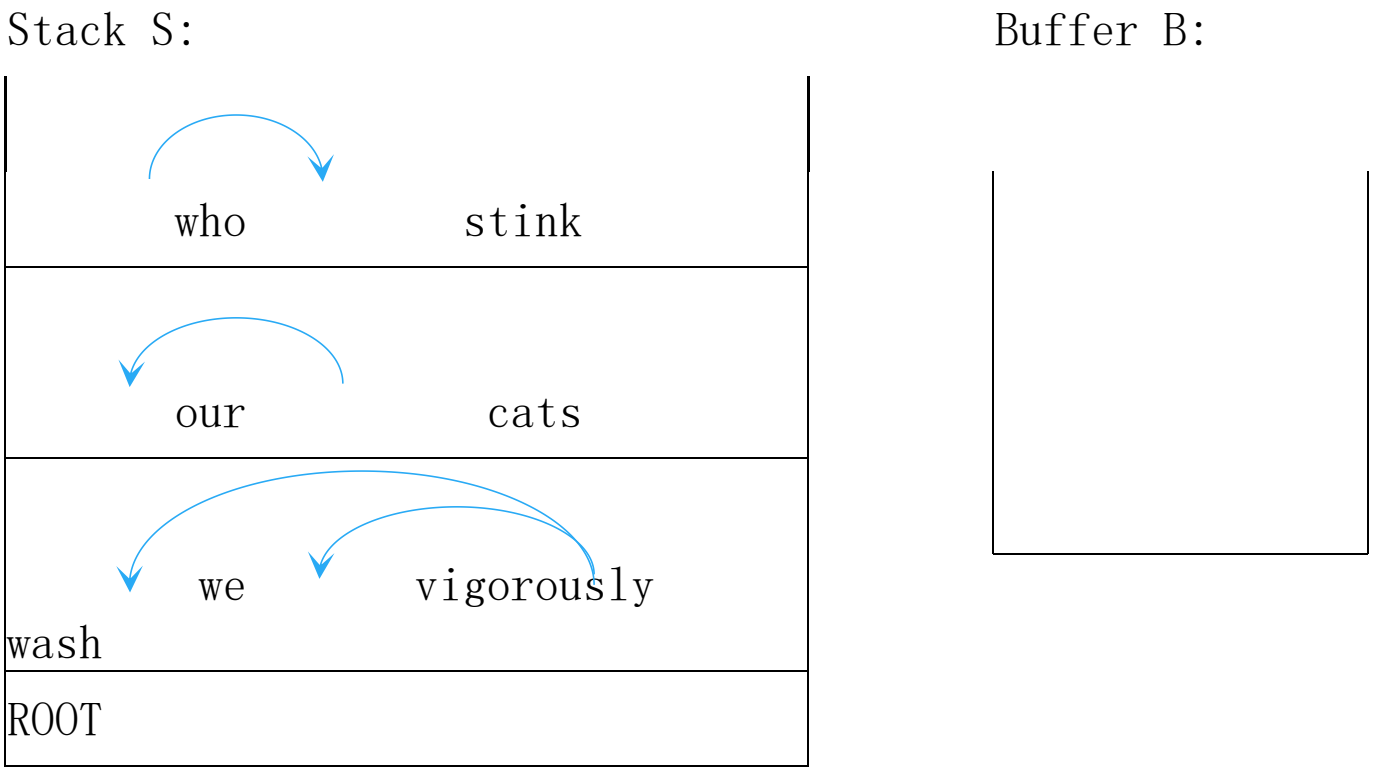
- 转移动作（actions）：SHIFT， SHIFT， SHIFT，LEFT-ARC，LEFT-ARC，SHIFT， SHIFT，LEFT-ARC， SHIFT

基于转移的分析方法：示例



- 转移动作（actions）：SHIFT， SHIFT， SHIFT，LEFT-ARC，LEFT-ARC，SHIFT， SHIFT，LEFT-ARC， SHIFT， SHIFT

基于转移的分析方法：示例

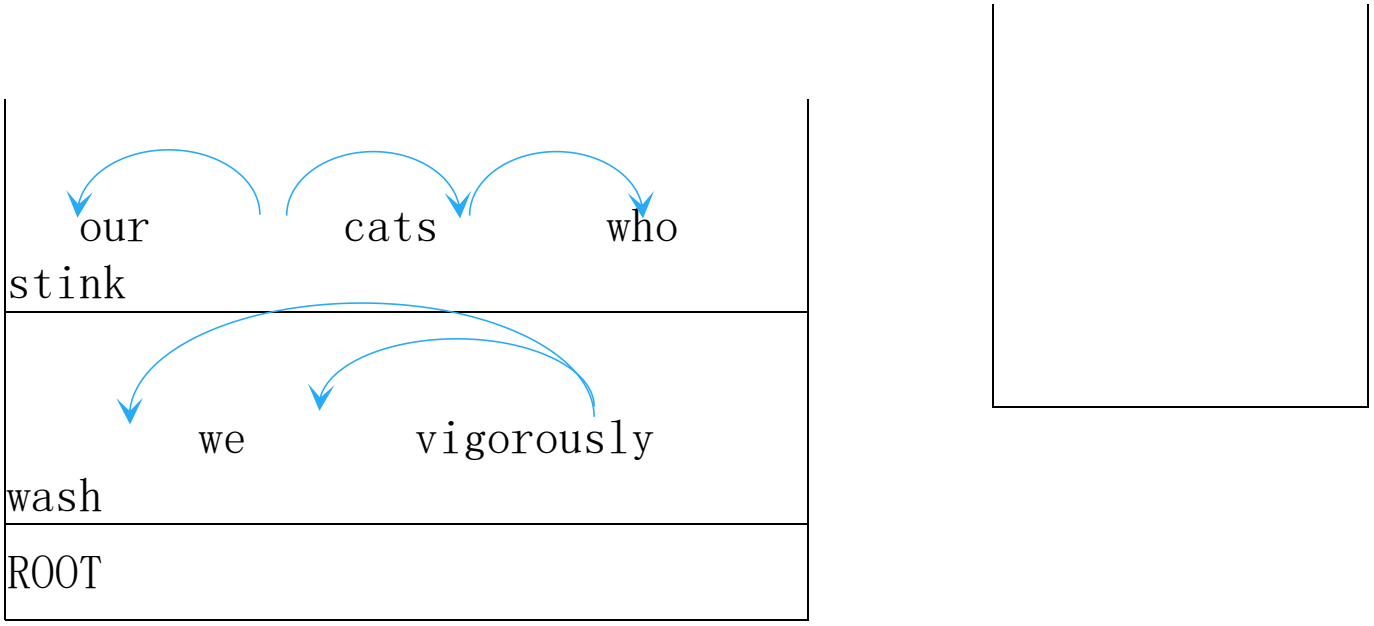


- 转移动作（actions）：SHIFT， SHIFT， SHIFT，LEFT-ARC，LEFT-ARC，SHIFT， SHIFT，LEFT-ARC， SHIFT， SHIFT， RIGHT-ARC

基于转移的分析方法：示例

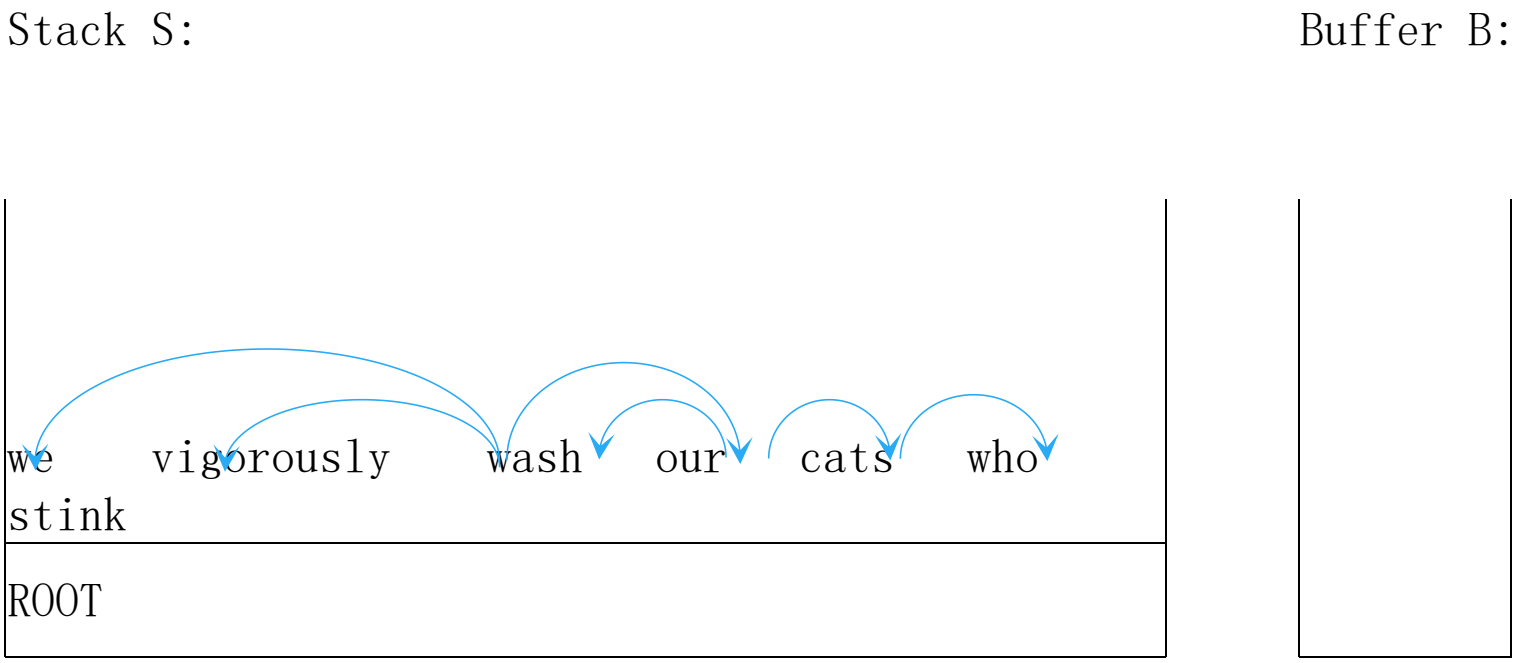
Stack S:

Buffer B:



- 转移动作（actions）：SHIFT， SHIFT， SHIFT，LEFT-ARC，LEFT-ARC，SHIFT， SHIFT，LEFT-ARC， SHIFT， SHIFT，RIGHT-ARC， RIGHT-ARC

基于转移的分析方法：示例

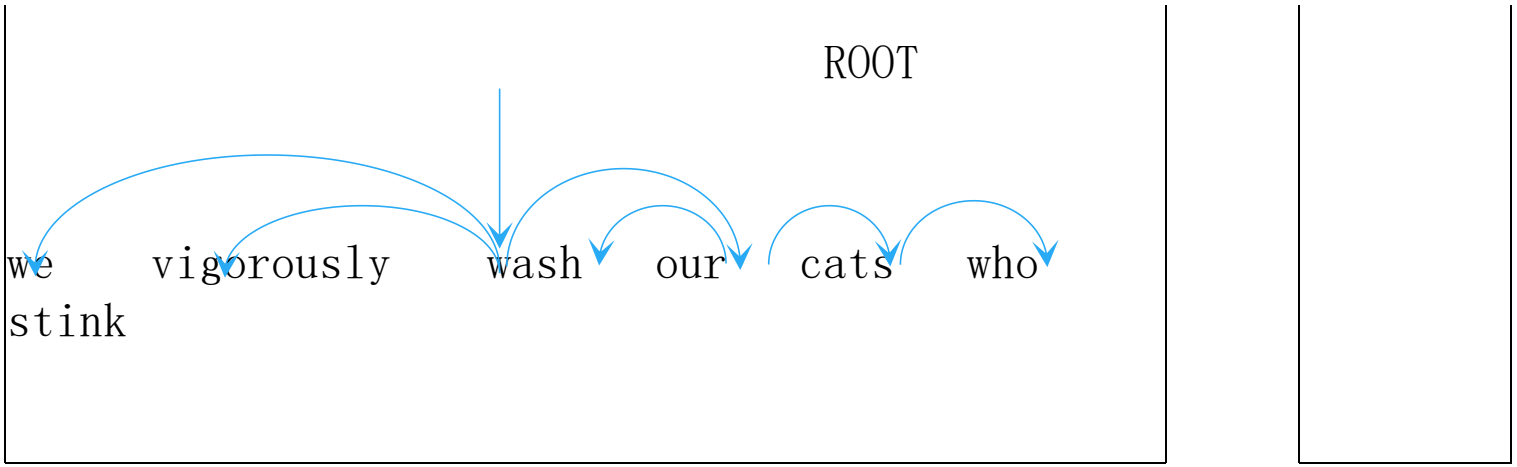


- 转移动作（actions）：SHIFT， SHIFT， SHIFT，LEFT-ARC，LEFT-ARC，SHIFT， SHIFT，LEFT-ARC， SHIFT， SHIFT，RIGHT-ARC， RIGHT-ARC， RIGHT-ARC

基于转移的分析方法：示例

Stack S:

Buffer B:



- 转移动作（actions）：SHIFT， SHIFT， SHIFT，LEFT-ARC，LEFT-ARC，SHIFT， SHIFT，LEFT-ARC， SHIFT， SHIFT，RIGHT-ARC， RIGHT-ARC， RIGHT-ARC， RIGHT-ARC

基于转移的分析方法：分类器

- 在每次迭代中，从转移动作集合{SHIFT，LEFT-ARC，RIGHT-ARC}中选择一个执行
  - 实际上，还有带有标签的变体LEFT-ARC和变体RIGHT-ARC
- 特征（feature）通过S、B和历史动作H得到——可以由LSTM模型自动地从S、B、H中提取特征
- 训练数据：将正确的树转换为$2n$个<状态（state），正确转移（correct transition）>的“标准（oracle）”转移序列
- 每个单词都会SHIFT一次，再作为孩子节点参与一次规约

线性时间

基于转移的分析方法

- 除了“arc-standard”方法外，还有其它的方法如：“arc-eager”，“arc-hybrid”
- 同样也能应用于短语结构句法的分析
- 做决策的算法不需要是贪婪的，考虑多个可能的转移
  - 例如：束搜索方法
- 潜在缺点：分类器通常在以前的分类决策都是正确的假设下进行训练
  - 一个常用的解决方案是动态标准（dynamic oracle）