第 8 章　句 法 分 析

2/2

中科院信息工程研究所第二研究室

胡玥

huyue@iie.ac.cn

内 容 提 要

第一部分：完全句法分析

第二部分：局部句法分析

第三部分：依存关系分析

局部句法分析概述

局部句法分析任务

输入： 警察/NN 正在/AD 详细/AD 调查/VV 事故/NN 原因/NN

输出：

部分独立成分

识别句子中某些结构相对简单的独立成分

- 规则法
- 概率统计
- 深度学习

技术方法

特点：不需要句法分析模型

11.2 局部句法分析

局部句法分析（partial parsing）：

也叫**浅层句法分析**(shallow parsing), 或语块分析(chunk parsing), 它是与完全句法分析相对的。传统的句法分析要得到句子的完整的句法树；而浅层句法分析则不要求得到完整的句法分析树,它只要求识别其中的某些结构相对简单的成分,如非递归的名词短语、动词短语等。这些识别出来的结构通常被称作语块(chunk),语块和短语这两个概念可以换用。由 S. Abney (1991) 首先提出。

11.2 局部句法分析

语块定义

S. Abney 对语块的解释：语块是介于词和句子之间的具有非递归特征的核心成分。S. Abney对英语组块定义三个层次：

- 词 (words)
- 非递归的名词短语 (NPs)、动词词组 (VGs)、副词短语 (DPs) 和 介词短语 (PPs)
- 子句 (clause)

局部句法分析主要任务是识别第二层名词短语、动词短语等

11.2 局部句法分析

浅层句法分析方法

- 基于规则的方法（略）
- 基于统计的方法
- 神经网络方法

} 可以将组块的分析转化成序列标注问题解决

局部句法分析性能评测：

三个基本的评测指标：

- 精度(precision)
- 召回率(recall)
- F-measure

内 容 提 要

第一部分：完全句法分析

第二部分：局部句法分析

第三部分：依存关系分析

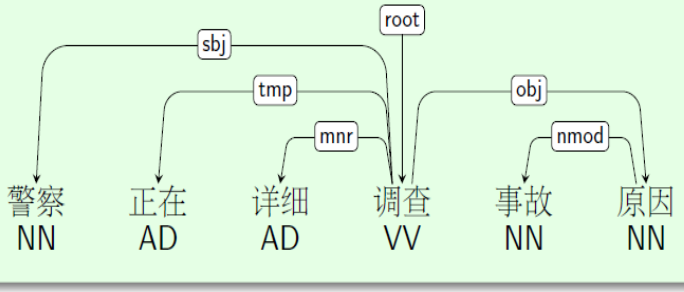
依存句法分析概述

依存句法分析任务

输入：警察/NN 正在/AD 详细/AD 调查/VV 事故/NN 原因/NN

依存文法

输出：依存结构树



分析出词与词之间的依存关系

规则法

概率统计

深度学习

技术方法

依存句法分析-内容提要

11.3.1 依存文法

现代依存语法理论（dependency grammar）

法国语言学家吕西安·泰尼埃（Lucien Tesnière, 1893-1954）。1953年在出版的专著《结构句法概要》中创立。对语言学的发展产生了深远的影响，特别是在计算语言学界备受推崇。

L. Tesnière 的理论认为：

在一个句子中，词语不是一盘散沙，而是有机联系的整体。因此，句子中词语和词语的“关联”是句子的“生命线”。依存文法通过这种句子中词与词的**相互依存关系**揭示句子的结构。

句法关系和词义体现在词之间的依存关系中

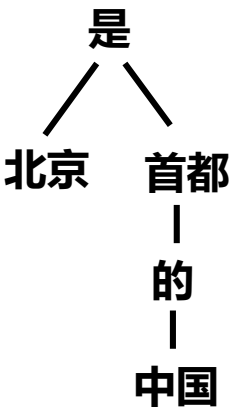
11.3.1 依存文法

依存语法：

1.词之间的依存关系

参加组成一个结构的成分(词)之间是不平等的，一些成分从属于另一些成分，每个成分只能从属于至多一个成分。从属与被从属的关系就决定了结构的性质。即，词与词之间支配与被支配的关系。这种关系是**不对等的**，是**有方向的**
各种依存关系依据“**词和词之间的语义关联**”有着不同的含义

如：　　北京　是 中国 的 首都



11.3.1 依存文法

哪个两个词之间有依存关系是根据句法规则和词义来定义的

句法和依存语法的观点：

（1）修饰语从属于被修饰语

（2）主语、宾语从属于谓语

（3）介词结构中的介词宾语从属于介词

（4）由连词构成的联合结构中分支成分从属于连词

（5）从句从属于从句的引导成分。

……

11.3.1 依存文法

在实际应用中可以按需求，对依存关系进行定义

如，哈工大语言技术平台（http://www.ltp-cloud.com/） 依存分析中，将依存关系细化为15 种：

关系类型	Tag	Description	Example
主谓关系	SBV	subject-verb	我送她一束花 (我 <-- 送)
动宾关系	VOB	直接宾语，verb-object	我送她一束花 (送 --> 花)
间宾关系	IOB	间接宾语，indirect-object	我送她一束花 (送 --> 她)
前置宾语	FOB	前置宾语，fronting-object	他什么书都读 (书 <-- 读)
兼语	DBL	double	他请我吃饭 (请 --> 我)
定中关系	ATT	attribute	红苹果 (红 <-- 苹果)

11.3.1 依存文法

状中结构	ADV	adverbial	非常美丽 (非常 <-- 美丽)
动补结构	CMP	complement	做完了作业 (做 --> 完)
并列关系	COO	coordinate	大山和大海 (大山 --> 大海)
介宾关系	POB	preposition-object	在贸易区内 (在 --> 内)
左附加关系	LAD	left adjunct	大山和大海 (和 <-- 大海)
右附加关系	RAD	right adjunct	孩子们 (孩子 --> 们)
独立结构	IS	independent structure	两个单句在结构上彼此独立
标点	WP	punctuation	。
核心关系	HED	head	指整个句子的核心

11.3.1 依存文法

2 .句子中心词

动词是句子的中心，它支配着别的成分，而不受其他成分支配。直接受动词支配的有**名词词组**和**副词词组**，名词词组形成“行动元”（不超过三个：主语、宾语1、宾语2）；副词词组形成“状态元”（可以无限个）；

动词支配行动元的个数称为 该动词的“**价**”

在汉语中：

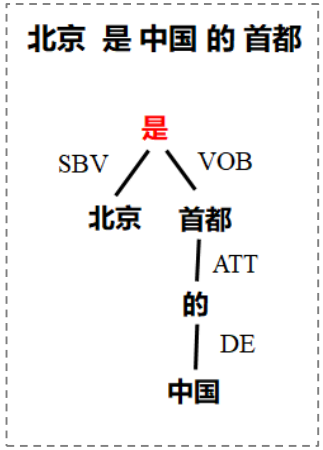
零价动词：地震、刮风；

一价动词：病、醉、休息、咳嗽、游泳、跑 等；

二价动词：爱、采、参观、讨论、是 等；

三价动词：给、送、告诉 等。

如：



11.3.1 依存文法

1970年计算语言学家J. Robinson在论文《依存结构和转换规则》中提出了依存语法的四条公理为依存语法奠定了基础

四条公理：

(1) 一个句子只有一个独立的成分；

(2) 句子的其他成分都从属于某一成分；

(3) 任何一成分都不能依存于两个或多个成分；

(4) 如果成分A直接从属于成分B，而成分C在句子中位于A和B之间，那么，成分C或者从属于A，或者从属于B，或者从属于A和B之间的某一成分。

11.3.1 依存文法

这四条公理相当于对依存图和依存树的形式约束为：

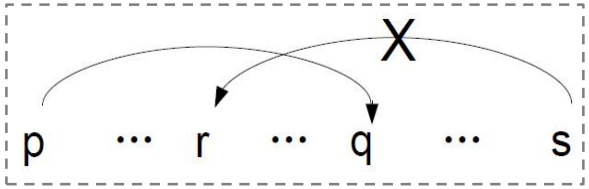
- 单一父结点(single headed)
- 连通(connective)
- 无环(acyclic)
- 可投射(projective)

由此保证句子的依存分析结果是一棵有“根(root)”的树结构

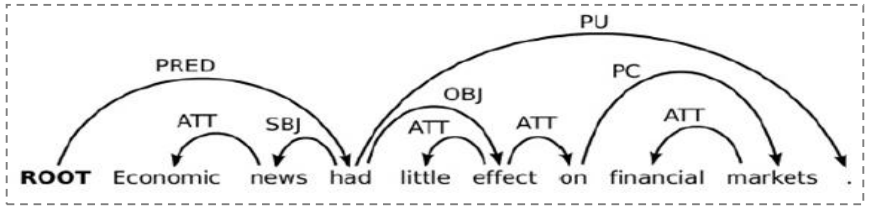
11.3.1 依存文法

投射性：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。

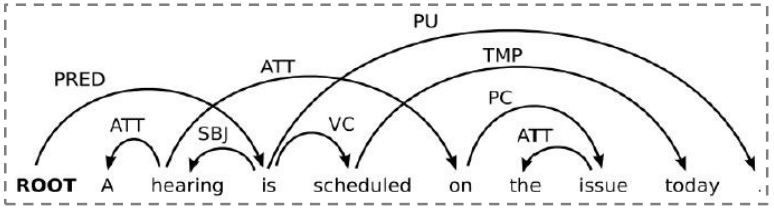
非投射：没有上述限制，依存边会有交叉。



如：



投射

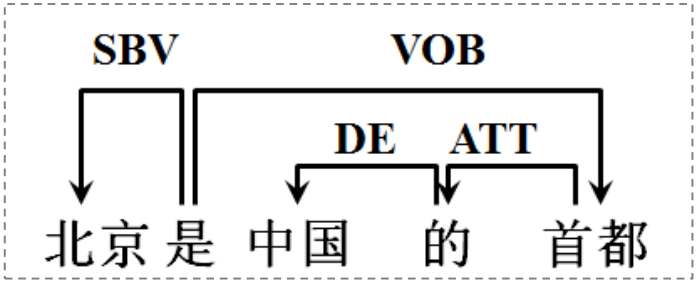


非投射

11.3.1 依存文法

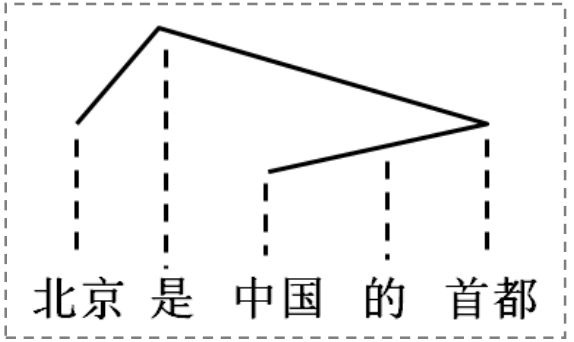
依存语法表示方式：

有向图

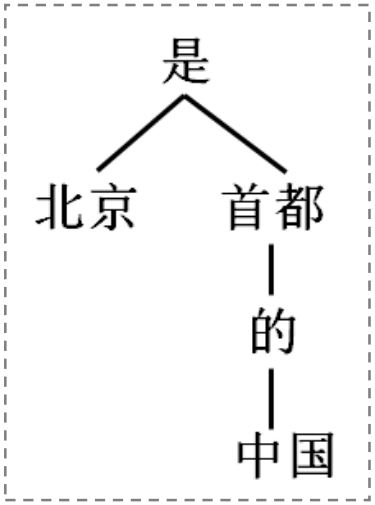


依存投射树

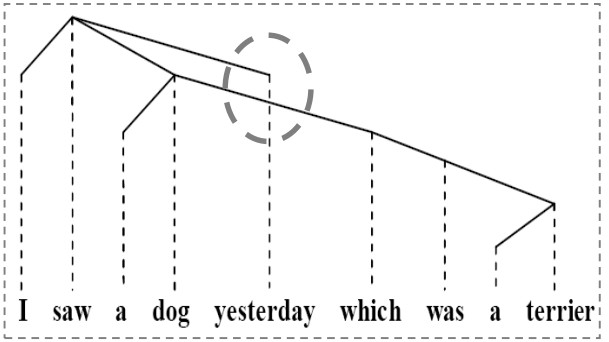
投射：



依存树



非投射：



11.3.1 依存文法

◆ **依存语法的优势**

简单，直接按照词语之间的依存关系工作，是天然词汇化的；

不过多强调句子中的固定词序，对自由语序的语言分析更有优势；

◆ **依存语法的局限性**

在汉语中，难以分析连词的联合结构、连谓结构、述补结构。

依存句法分析-内容提要

11.3.2.1 中文依存树库（语料）

中文依存树库

汉语依存结构树库是根据句子中词与词之间直接的句法关系构建的树库

第二届自然语言处理与中文计算会议（NLP&CC 2013）的技术评测样例中文树库：

- 哈尔滨工业大学依存语料库（8千句）
- 清华大学语义依存网络语料（2万句）

这些树库语料都是CoNLL格式的（以.conll 结尾）

下载地址：

http://tcci.ccf.org.cn/conference/2013/dldoc/evsam05.zip

11.3.2.1 中文依存树库（语料）

依存树库的 CoNLL 格式

CONLL标注格式包含10列，分别为：

ID	FORM	LEMMA	CPOSTAG	POSTAG	FEATS	HEAD	DEPREL	PHEAD	PDEPREL

本文只用到前 8 列，其含义分别为：

1 ID 当前词在句子中的序号，1 开始.

2 FORM 当前词语或标点

3 LEMMA 当前词语（或标点）的原型或词干，在中文中，此列与FORM相同

4 CPOSTAG 当前词语的词性（粗粒度）

5 POSTAG 当前词语的词性（细粒度）

6 FEATS 句法特征，在本次评测中，此列未被使用，全部以下划线代替。

7 HEAD 当前词语的中心词

8 DEPREL 当前词语与中心词的依存关系

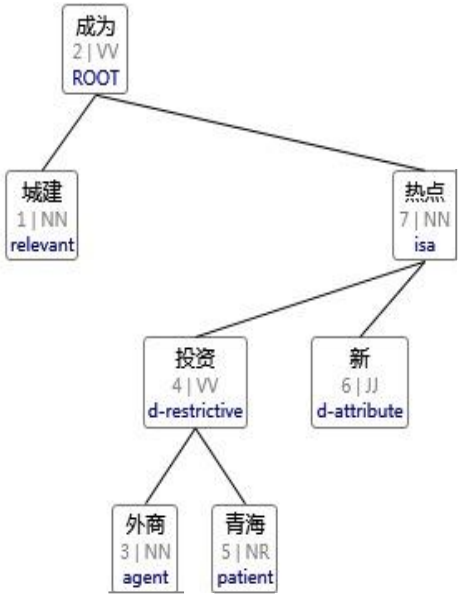
在CONLL格式中，每个词语占一行，无值列用下划线'_'代替，列的分隔符为制表符'\t'，行的分隔符为换行符'\n'；句子与句子之间用空行分隔。

11.3.2.1 中文依存树库（语料）

哈尔滨工业大学依存语料库（8千句）

序号	词语	原型（中文词语=原型）	粗粒度词性	细粒度词性	句法特征	中心词	依存关系		
1	城建	城建	NN	NN	_	2	relevant	_	_
2	成为	成为	VV	VV	_	0	ROOT	_	_
3	外商	外商	NN	NN	_	4	agent	_	_
4	投资	投资	VV	VV	_	7	d- restrictive	_	_
5	青海	青海	NR	NR	_	4	patient	_	_
6	新	新	JJ	JJ	_	7	d-attribute	_	_
7	热点	热点	NN	NN	_	2	isa	_	_

树形图：



有向图：

(1) 词性标注说明:

a 形容词	nl 处所名词
b 区别词	ns 地名
c 连词	nt 时间名词
d 副词	nz 其它专名
e 叹词	o 拟声词
g 语素字	p 介词
h 前接成分	q 量词
i 习用语	r 代词
j 简称	u 助词
k 后接成分	v 动词
m 数词	wp 标点
n 名词	ws 字符串
nd 方位名词	x 非语素字
nh 人名	
ni 团体、机构、组织的专名	

(2)依存关系标注说明: (24种依存关系)

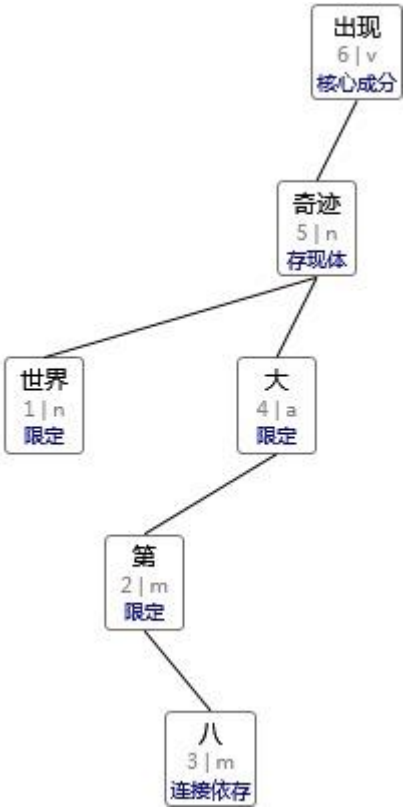
定中关系ATT（attribute）
数量关系QUN（quantity）
并列关系COO（coordinate）
同位关系APP（appositive）
前附加关系LAD（left adjunct）
后附加关系RAD（right adjunct）
动宾关系VOB（verb-object）
介宾关系POB（preposition-object）
主谓关系SBV（subject-verb）
比拟关系SIM（similarity）
核心HED（head）
连谓结构VV（verb-verb）
关联结构CNJ（conjunctive）
语态结构MT（mood-tense）
独立结构IS（independent structure）
状中结构ADV（adverbial）
动补结构CMP（complement）
“的”字结构DE
“地”字结构DI
“得”字结构DEI
“把”字结构BA
“被”字结构BEI
独立分句IC（independent clause）
依存分句DC（dependent clause）

11.3.2.1 中文依存树库（语料）

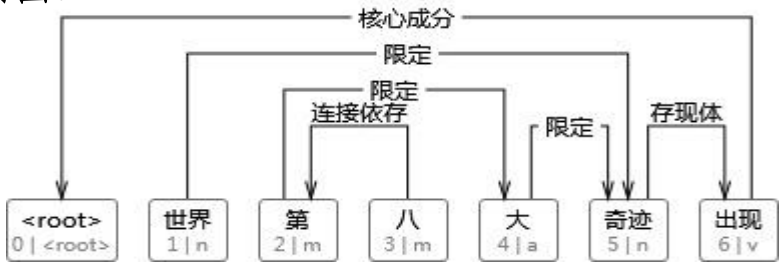
清华大学语义依存网络语料（2万句）

序号	词语	原型（中文词语=原型）	粗粒度词性	细粒度词性	句法特征	中心词	依存关系
1	世界	世界	n	n	_	5	限定
2	第	第	m	m	_	4	限定
3	八	八	m	m	_	2	连接依存
4	大	大	a	a	_	5	限定
5	奇迹	奇迹	n	n	_	6	存现体
6	出现	出现	v	v	_	0	核心成分

树形图：



有向图：



11.3.2.1 中文依存树库（语料）

依存关系标注说明: (69 种依存关系)

..是..的依存 441
的 字依存 11091
伴随 56
比较量 37
比较内容 8
并列 97
部分 110
材料 22
参照体 262
程度 3616
除了 2
处所 2036
触及部件 6
存现体 589
代价 28
递进 7
动量 245
范围 757
方式 3205
方位词依存 2284
方向 150
根据 102
工具 66
关联词依存 1855
关系主体 2184
核心成分 15354
后延时段 50
接续 182
结果 287
结果事件 738
介词依存 7788
进程时段 351
经验者 2028
来源 102
类指 719
连接依存 8531
领有者 1
描述 3368
描写体 1188
目标 1337
目的 425
内容 3746
频率 288
评论 4477
起始时间 186
趋向动词依存 535
让步 7
施事 **7430**
时间 2401
时距 133
时态依存 3283
时态语态依存 199
事件过程 12
手段 78
受事 6153
数量 5627
条件 114
通过处所 38
同位语 942
限定 36014
相伴体 775
语气依存 91
原处所 198
原因 284
原状态 96
整体 180
终处所 388
终止时间 76
终状态 84

11.3.2.1 中文依存树库（语料）

1	赵宁	赵宁	NR	NR	_	4	d-genetive	_	_
2	的	的	DEG	DEG	_	1	aux-depend	_	_
3	哀乐	哀乐	NN	NN	_	4	d-attribute	_	_
4	中年	中年	NN	NN	_	9	scope	_	_
5	，	，	PU	PU	_	9	PU	_	_
6	转折	转折	NN	NN	_	9	patient	_	_
7	不	不	AD	AD	_	9	negation	_	_
8	可	可	VV	VV	_	9	modal	_	_
9	谓	谓	VV	VV	_	0	ROOT	_	_
10	不	不	AD	AD	_	11	negation	_	_
11	大	大	VA	VA	_	9	d-attribute	_	_
12	。	。	PU	PU	_	9	PU	_	_

1	司法院	司法院	NN	NN	_	2	d-genetive	_	_
2	秘书长	秘书长	NN	NN	_	3	d-category	_	_
3	杨仁寿	杨仁寿	NR	NR	_	9	agent	_	_
4	在	在	P	P	_	5	prep-depend	_	_
5	记者会	记者会	NN	NN	_	9	location	_	_
6	上	上	LC	LC	_	5	aux-depend	_	_
7	阐述	阐述	VV	VV	_	9	succeeding	_	_
8	解释文	解释文	NN	NN	_	7	content	_	_
9	指出	指出	VV	VV	_	0	ROOT	_	_
10	，	，	PU	PU	_	9	PU	_	_
11	行政院	行政院	NN	NN	_	12	agent	_	_
12	停建	停建	VV	VV	_	14	relevant	_	_
13	核四	核四	NN	NN	_	12	patient	_	_
14	属于	属于	VV	VV	_	40	s-coordinate	_	_
15	国家	国家	NN	NN	_	17	d-genetive	_	_
16	重要	重要	JJ	JJ	_	17	d-attribute	_	_
17	政策	政策	NN	NN	_	19	d-host	_	_
18	之	之	DEG	DEG	_	17	aux-depend	_	_
19	变更	变更	NN	NN	_	14	isa	_	_
20	，	，	PU	PU	_	40	PU	_	_
21	行政院	行政院	NN	NN	_	40	agent	_	_
22	未	未	AD	AD	_	31	negation	_	_
23	依	依	P	P	_	25	prep-depend	_	_
24	宪法	宪法	NN	NN	_	25	d-restrictive	_	_
25	程序	程序	NN	NN	_	31	basis	_	_
26	适时	适时	AD	AD	_	31	manner	_	_
27	向	向	P	P	_	28	prep-depend	_	_
28	立法院	立法院	NN	NN	_	29	patient	_	_
29	报告	报告	VV	VV	_	31	s-coordinate	_	_
30	并	并	CC	CC	_	31	aux-depend	_	_
31	备询	备询	VV	VV	_	40	s-coordinate	_	_
32	，	，	PU	PU	_	40	PU	_	_
33	有	有	VE	VE	_	40	s-coordinate	_	_
34	程序	程序	NN	NN	_	36	d-restrictive	_	_
35	上	上	LC	LC	_	34	aux-depend	_	_
36	瑕疵	瑕疵	NN	NN	_	33	existent	_	_
37	，	，	PU	PU	_	40	PU	_	_
38	需	需	VV	VV	_	40	s-coordinate	_	_
39	尽速	尽速	AD	AD	_	40	manner	_	_
40	补行	补行	VV	VV	_	9	content	_	_
41	报告	报告	NN	NN	_	44	s-coordinate	_	_
42	及	及	CC	CC	_	44	aux-depend	_	_
43	备询	备询	NN	NN	_	44	d-restrictive	_	_
44	程序	程序	NN	NN	_	40	content	_	_
45	。	。	PU	PU	_	9	PU	_	_

1	此外	此外	AD	AD	_	5	aux-depend	_	_
2	，	，	PU	PU	_	5	PU	_	_
3	立法院	立法院	NN	NN	_	5	possessor	_	_
4	也	也	AD	AD	_	5	aux-depend	_	_
5	有	有	VE	VE	_	31	s-succession	_	_
6	听取	听取	VV	VV	_	10	d-content	_	_
7	行政院	行政院	NN	NN	_	8	d-restrictive	_	_
8	报告	报告	NN	NN	_	6	content	_	_
9	的	的	DEC	DEC	_	6	aux-depend	_	_
10	义务	义务	NN	NN	_	5	possession	_	_
11	，	，	PU	PU	_	31	PU	_	_
12	立法院	立法院	NN	NN	_	13	agent	_	_
13	听取	听取	VV	VV	_	31	s-succession	_	_
14	报告	报告	NN	NN	_	13	content	_	_
15	后	后	LC	LC	_	13	aux-depend	_	_
16	，	，	PU	PU	_	31	PU	_	_
17	若	若	CS	CS	_	18	aux-depend	_	_
18	采	采	VV	VV	_	31	s-condition	_	_
19	反对	反对	NN	NN	_	20	d-restrictive	_	_
20	决议	决议	NN	NN	_	18	possession	_	_
21	时	时	LC	LC	_	18	aux-depend	_	_
22	，	，	PU	PU	_	31	PU	_	_
23	大法官	大法官	NN	NN	_	31	agent	_	_
24	从	从	P	P	_	28	prep-depend	_	_
25	现有	现有	JJ	JJ	_	28	d-attribute	_	_
26	的	的	DEG	DEG	_	25	aux-depend	_	_
27	宪法	宪法	NN	NN	_	28	d-host	_	_
28	机制	机制	NN	NN	_	31	scope	_	_
29	中	中	LC	LC	_	28	aux-depend	_	_
30	，	，	PU	PU	_	31	PU	_	_
31	提示	提示	VV	VV	_	0	ROOT	_	_
32	了	了	AS	AS	_	31	aspect	_	_
33	两	两	CD	CD	_	34	d-quantity	_	_
34	院	院	NN	NN	_	31	patient	_	_
35	可	可	VV	VV	_	63	modal	_	_
36	在	在	P	P	_	63	prep-depend	_	_
37	行政	行政	NN	NN	_	38	d-restrictive	_	_
38	院长	院长	NN	NN	_	40	agent	_	_
39	自行	自行	AD	AD	_	40	manner	_	_
40	辞职	辞职	VV	VV	_	54	s-coordinate	_	_
41	、	、	PU	PU	_	54	PU	_	_
42	立法院	立法院	NN	NN	_	46	agent	_	_
43	对	对	P	P	_	45	prep-depend	_	_
44	行政	行政	NN	NN	_	45	d-restrictive	_	_
45	院长	院长	NN	NN	_	46	patient	_	_
46	提	提	VV	VV	_	49	s-or	_	_
47	不信任案	不信任案	NN	NN	_	46	content	_	_
48	或	或	CC	CC	_	49	aux-depend	_	_
49	解散	解散	VV	VV	_	54	s-coordinate	_	_
50	立法院	立法院	NN	NN	_	49	patient	_	_
51	、	、	PU	PU	_	54	PU	_	_
52	及	及	CC	CC	_	54	aux-depend	_	_
53	立法院	立法院	NN	NN	_	54	agent	_	_
54	通过	通过	VV	VV	_	63	d-content	_	_
55	兴建	兴建	VV	VV	_	58	d-content	_	_
56	电厂	电厂	NN	NN	_	55	content	_	_
57	相关	相关	JJ	JJ	_	58	d-attribute	_	_
58	法案	法案	NN	NN	_	54	content	_	_
59	等	等	ETC	ETC	_	54	aux-depend	_	_
60	三	三	CD	CD	_	61	d-quantity	_	_
61	种	种	M	M	_	63	d-quantity-p	_	_
62	解套	解套	NN	NN	_	63	d-restrictive	_	_
63	方案	方案	NN	NN	_	31	content	_	_
64	。	。	PU	PU	_	31	PU	_	_

依存句法分析-内容提要

11.3.2.2 依存分析算法

目前依存句法分析主要是在大规模训练语料的基础上用机器学习的方法（数据驱动方法）得到依存句法分析器。

数据驱动的依存句法分析方法主要有两种主流方法：

- **基于图（graph-based）的依存句法分析方法**

  基于图的方法将依存句法分析问题看成从完全有向图中寻找最大生成树的问题。其模型可以简单区分为一阶和高阶模型。 高阶模型可以使用更加复杂的子树特征，因此分析准确率更高，但是解码算法的效率也会下降。

- **基于转移（transition-based）的依存句法分析方法**

  基于转移的方法将依存树的构成过程建模为一个动作序列，将依存分析问题转化为寻找最优动作序列的问题，特征表示方面，基于转移的方法可以充分利用已形成的子树信息，从而形成丰富的特征。

11.3.2.2 依存分析算法

★ **基于图的生成式的分析方法 (generative parsing)**

基本思路：

1. 生成所有节点的完全有向图
2. 可用各种概率统计法（如最大似然估计）计算各边的概率
3. 取边的权值最大的作为唯一的边，加入有向图中。
4. 在有向图上使用Prim最大生成树算法，计算出最大生成树，格式化输出。

关键：依存边的计算方法

11.3.2.2 依存分析算法

例：分析句子“我吃米饭”的依存结构

解：1. 先进行分词与词性标注，得到：

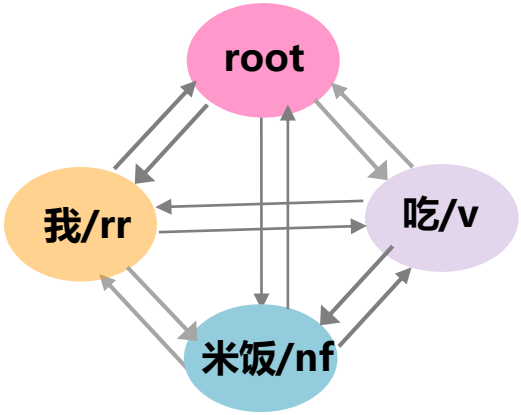
[我/rr, 吃/v, 米饭/nf]

2. 生成有向图

依存句法树中有虚根的存在，为其加入一个虚节点，这样一共有四个节点：

[##核心##/root,我/rr,吃/v,米饭/n]

每个节点都与另外三个构成一条有向边，一共4 * 3 = 12 条：



例引自：http://www.hankcs.com/nlp/to-achieve-a-simple-generative-dependency-parsing.html

11.3.2.2 依存分析算法

3.计算各边的权值

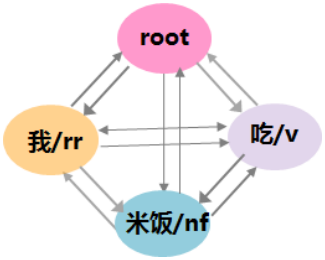
边概率计算方法

统计词语WordA与词语WordB构成依存关系DrC的频次，
词语WordA与词性TagB构成依存关系DrD的频次，词性TagA与词语WordB构成依存关系DrE的频次，词性TagA与词词性TagB构成依存关系DrF的频次。
为句子中词语i与词语j生成多条依存句法边，其权值为上述四种频次的综合（主要利用词-词频次，其余的作平滑处理用）

词A到词B依存边权值

WordA ： 词语A　　TagA ： 词语A的词性

WordB ： 词语B　　TagB ： 词语B的词性



词A到词B依存边权值 = -log(WordA到WordA依存关系数/总数)

词A到词B依存边权值 = -log(WordA到TagB依存关系数/总数) *10

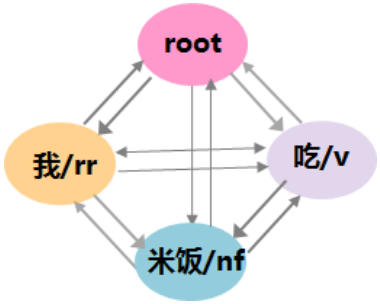
词A到词B依存边权值 = -log(TagA到WordB依存关系数/总数) *10

词A到词B依存边权值 = -log(TagA 到TagB依存关系数/总数) *100

11.3.2.2 依存分析算法

在统计语料中统计得：

```
1. ##核心##/root 到 我/rr : 未知 10000.0 ↵
2. ##核心##/root 到 吃/v : 未知 10000.0 ↵
3. ##核心##/root 到 米饭/n : 未知 10000.0 ↵
4. 我/rr 到 ##核心##/root : 核心成分 6.410175 ↵
5. 我/rr 到 吃/v : 施事 21.061098 经验者 28.54827 目标 33.656525 受事 37.021
   248 限定 43.307335 相伴体 48.00737 关系主体 53.115623 内容 53.115623 来
   源 64.101746 ↵
6. 我/rr 到 米饭/n : 限定 22.2052 施事 48.00737 受事 57.170277 目标 57.17027
   7 经验者 64.101746 连接依存 64.101746 ↵
7. 吃/v 到 ##核心##/root : 核心成分 1.7917595 ↵
8. 吃/v 到 我/rr : 连接依存 96.688614 介词依存 107.67474 施事 107.67474 ↵
9. 吃/v 到 米饭/n : 限定 24.849068 ↵
10.米饭/n 到 ##核心##/root : 核心成分 37.077995 ↵
11.米饭/n 到 我/rr : 连接依存 113.2556 ↵
12.米饭/n 到 吃/v : 受事 0.6931472↵
```



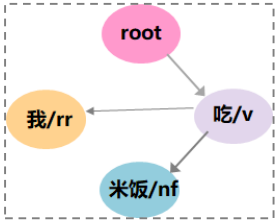
注：其中“未知”表示边不存在，“受事”“施事”表示依存关系，后面的小数表示权值。表中对概率取了负对数，所以接下来用加法求最小生成树即可。

11.3.2.2 依存分析算法

4.最小生成树

在有向图上使用Prim最小生成树算法，取边的权值最小的作为唯一的边，加入有向图中，得出最小生成树：

```
1. 2 0 核心成分
2. 3 2 受事
3. 1 2 施事
```

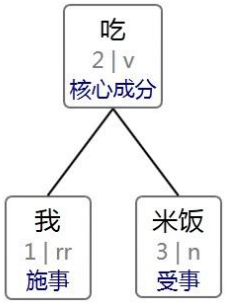


5.格式化输出

将其转为CoNLL格式输出：

```
1. 1 我    我    rr   rr   _   2   施事   _
   _
2. 2 吃    吃    v    v    _   0   核心成分 _
   _
3. 3 米饭   米饭   n    n    _   2   受事   _
   _
```

可视化



11.3.2.2 依存分析算法

生成方法评价

优点： 此类方法的准确率较高

弱点：

- 采用联合概率模型，在进行概率乘积分解时做了不尽合理的假设，不易加入语言特征；
- 因为采用全局搜索，算法的复杂度较高，一般为O (n^3) 或O (n^5)；
- 不易处理非投射现象。

11.3.2.2 依存分析算法

★ **基于转移的决策式(确定性)分析概率方法(deterministic parsing)**

基本思想：模仿人的认知过程，按照特定方向每次读入一个词。每读入一个词，都要根据当前状态做出决策(比如判断是否与前一个词发生依存关系)。一旦决策做出，将不再改变。所做决策即“采取什么样的分析动作(action)”。分析过程可以看作是一步步作用于输入句子之上的分析动作(action)的序列。

11.3.2.2 依存分析算法

1. 移进 - 归约算法

J. Nivre等(2003)提出的自左向右、自底向上的分析算法

算法思想：

用三元组**(S, I, A)**，表示分析状态格局(configuration)，仿照人类从左到右的阅读顺序，不断地读入单词，每读入一个单词便根据该单词特征和当前分析状态格局特征确定当前最佳动作，随着逐个单词读入，一步步“拼装”句法树。

该方法适用处理投射现象，处理非投射的动作体系有 Arc standard + swap (可以将栈顶两个元素位置交换) Nirve, ACL 2009

11.3.2.2 依存分析算法

三元组(S, I, A)

S：栈

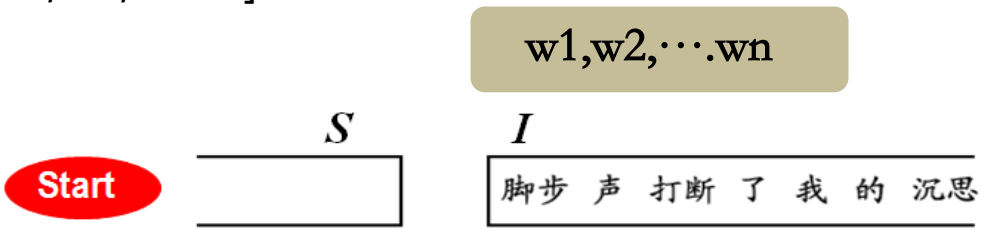
用来储存系统已经处理过的句法子树的根节点的，初始状态下 s=[Root]，

定义从栈顶数起的第i个元素为s_i。栈顶元素就s_1，s_1的下一个元素就是s_2.

s_2称为左焦点词， s_1 称为右焦点词。后记动作都是围绕着这两个焦点词展开的。

I：队列

用来存放未处理结点序列。初始状态下队列就是整个句子，且顺序不变
b=[w1,w2,....wn]



11.3.2.2 依存分析算法

A：依存弧集合

一条依存弧（在此可以看做一种变换）有两个信息：**动作类型+依存关系名称。**

- **依存关系名称** 取决于语料库中依存关系label
- 在Arc-eager 分析法中**动作类型**有4种：

Shift
(入栈)

$[\ldots]_S \quad [w_i,\ldots]_I \qquad \mathbf{push}(w_i) \qquad [\ldots w_i]_S \quad [\ldots]_I$

Left-Arc$_l$
(依存弧向左指)

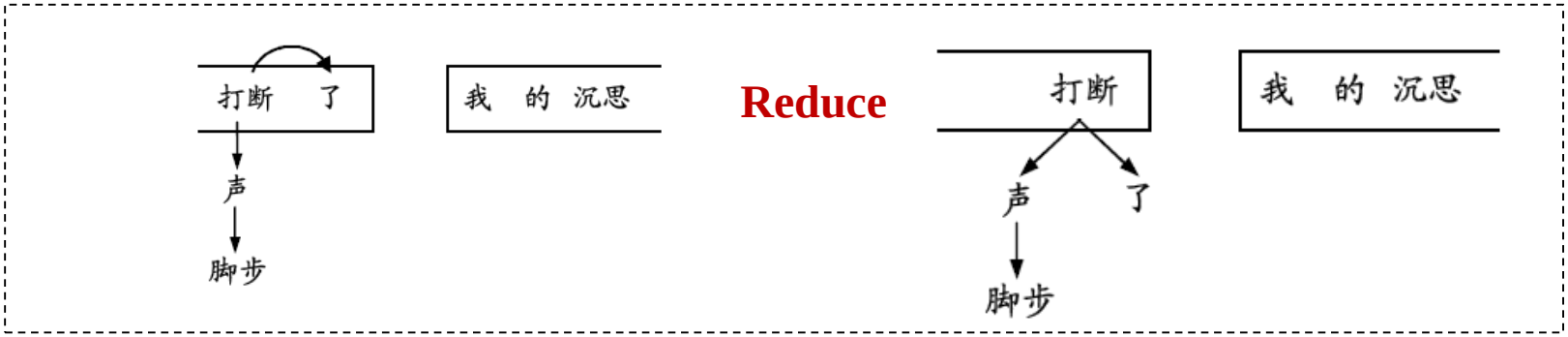
$[\ldots,w_i]_S \quad [w_j,\ldots]_I \qquad A \cup \{w_i \xleftarrow{l} w_j\}, \mathbf{pop}(w_i) \qquad [\ldots]_S \quad [w_j,\ldots]_I$

11.3.2.2 依存分析算法

Right-Arc$_l$

(依存弧向右指) $[\ldots,w_i]_S\ [w_j,\ldots]_I$ $A \cup \{w_i \xrightarrow{l} w_j\}, \mathbf{push}(w_j)$ $[\ldots w_i,w_j]_S\ [\ldots]_I$

Reduce $[\ldots w_i]_S\ [\ldots]_I$ $\mathbf{pop}(w_i)$ $[\ldots]_S\ [\ldots]_I$

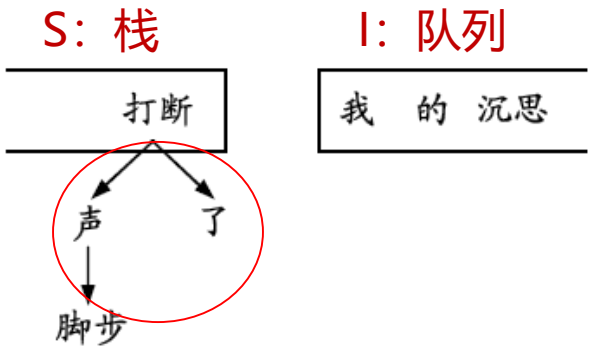


11.3.2.2 依存分析算法

分析状态格局(configuration)

当前分析步骤中的三元组(S, I, A)情况：

如：



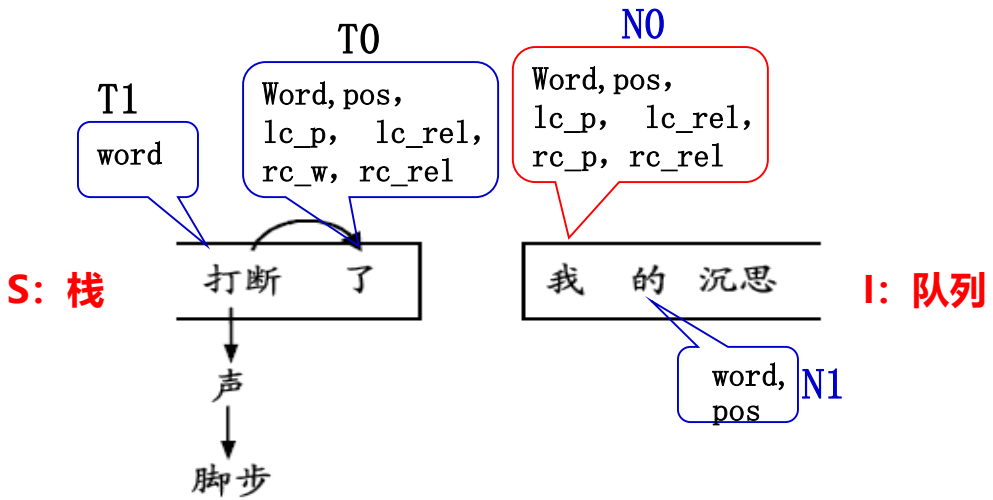
A：依存弧集合

每读入一个单词便根据该单词特征和当前分析状态格局特征确定当前最佳动作，逐个拼装”句法树。

如何根据输入单词特征和当前分析状态格局特征确定当前最佳动作？

11.3.2.2 依存分析算法

格局特征



	word	pos	lc_w	lc_p	lc_rel	rc_w	rc_p	rc_rel
T1	√							
T0	√	√		√	√	√		√
N0	√	√		√	√		√	√
N1	√	√						

T表示栈S里面的信息， T1指栈顶第二词（左焦点）T0指栈顶词(右焦点)

N表示队列I里面的信息， N0指待分析的第一个词， N1为待分析部分的第二个词

word表示词，pos表示词性，

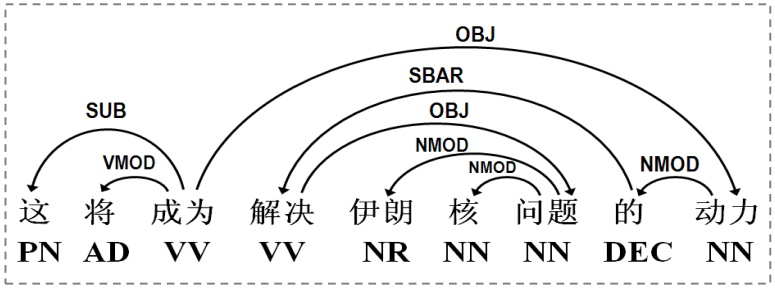
lc_w表示最左弧词，lc_p表示最左弧词性，lc_rel表示最左弧关系名；

rc_w表示最右弧词，rc_p表示最右弧词性，rc_rel表示最右弧关系名；

11.3.2.2 依存分析算法

训练语料处理

原语料库



特征

	word	pos	lc_w	lc_p	lc_rel	rc_w	rc_p	rc_rel
T1	√							
T0	√	√		√	√	√		√
N0	√	√		√	√		√	√
N1	√	√						

标注训练语料

动作 **特征**

Shift N0w:这 /N0p:PN /N1w:将N1p:AD

Shift T0w: 这/ T0p:PN /N0w:将 /N0p:AD/N1w:成为 /N1p:VV

L_A T1w:这/ T0w:将/ T0p:AD /N0w:成为/ N0p:VV/N1w:解决 /N1p:VV

L_A T0w:这 /T0p:PN /N0w:成为 /N0p:VV /N0lc_p:AD/ N0lc_rel: VMOD/N1w:解决/ N1p:VV

Shift T0w:成为/T0p:VV/T0lc_p:PN/T0lc_rel:SUB/T0rc_w:将/T0rc_rel:VMOD/N0w:解决/ N0p:VV N1w:伊朗/N1p:NR

Shift T1w:成为/T0w:解决/T0p:VV/N0w:伊朗/N0p:NR/N1w:核/N1p:NN

……

11.3.2.2 依存分析算法

模型：

统计方法：可用 CRF、最大熵、SVM分类器等建立预测模型

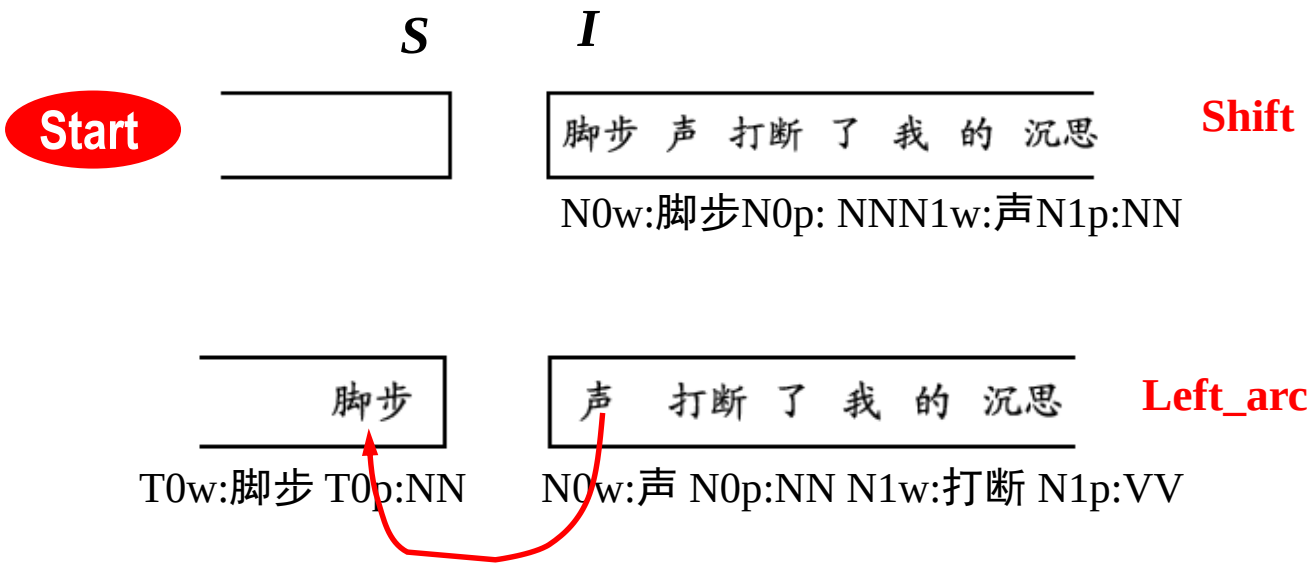
句法分析：

每走一步，根据该时刻的格局状态（特征信息）用训练好的模型，求的下一步动作，直至分析完成。

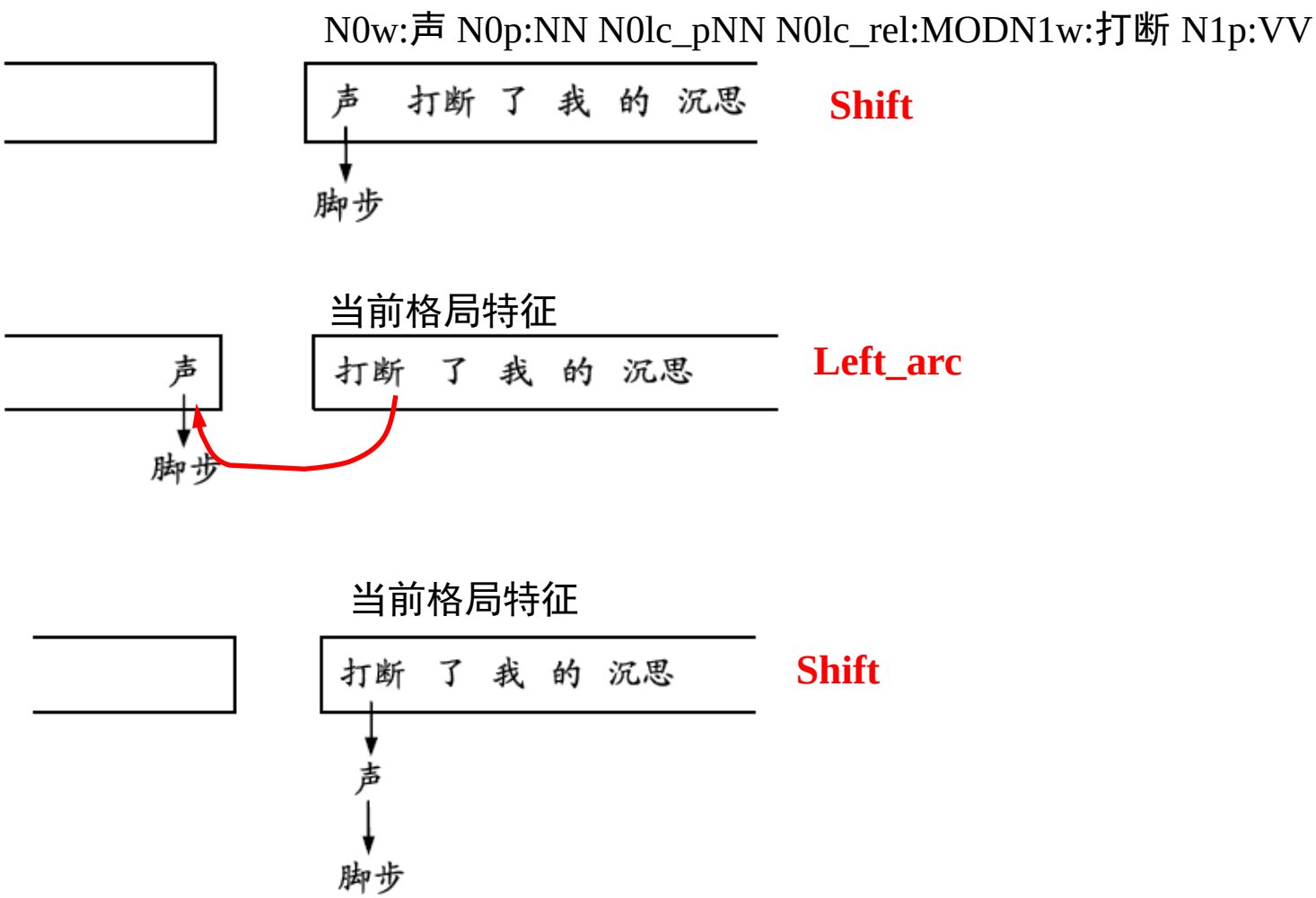
11.3.2.2 依存分析算法

例:

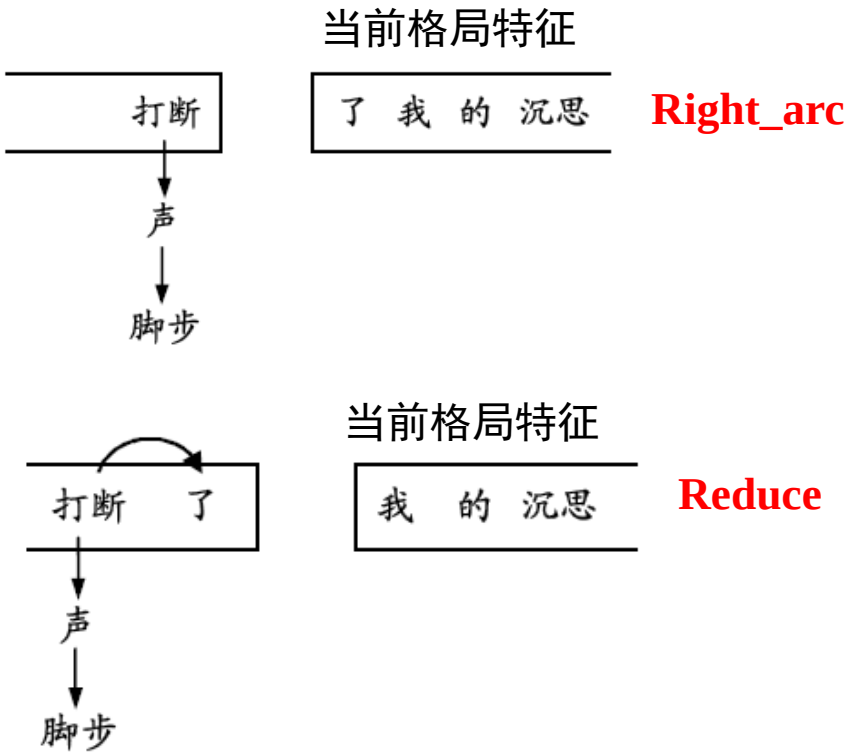
脚步	声	打断	了	我	的	沉思
NN	NN	VV	AS	PN	DEG	NN



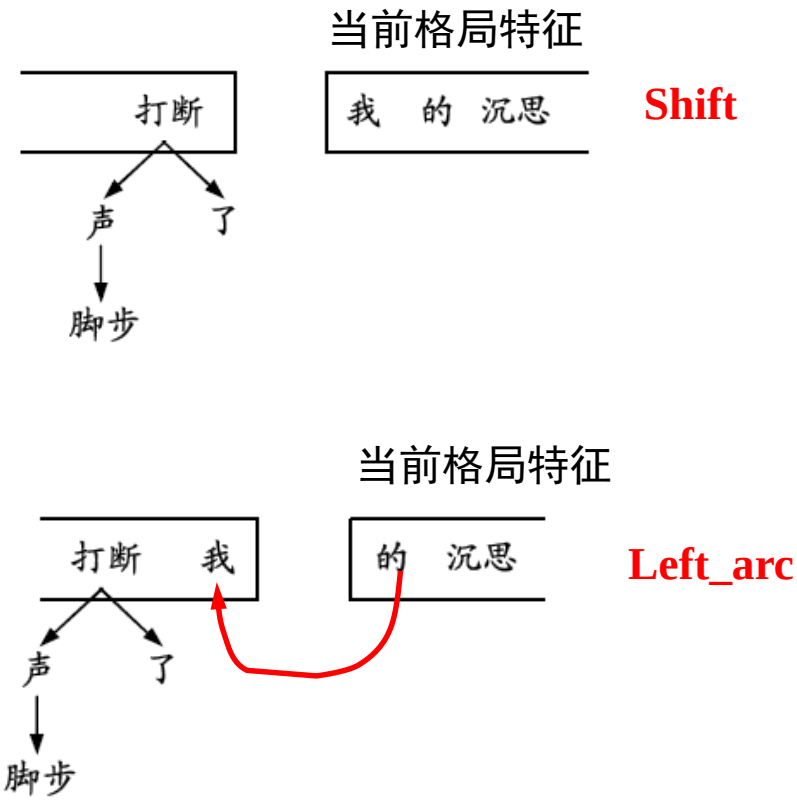
11.3.2.2 依存分析算法



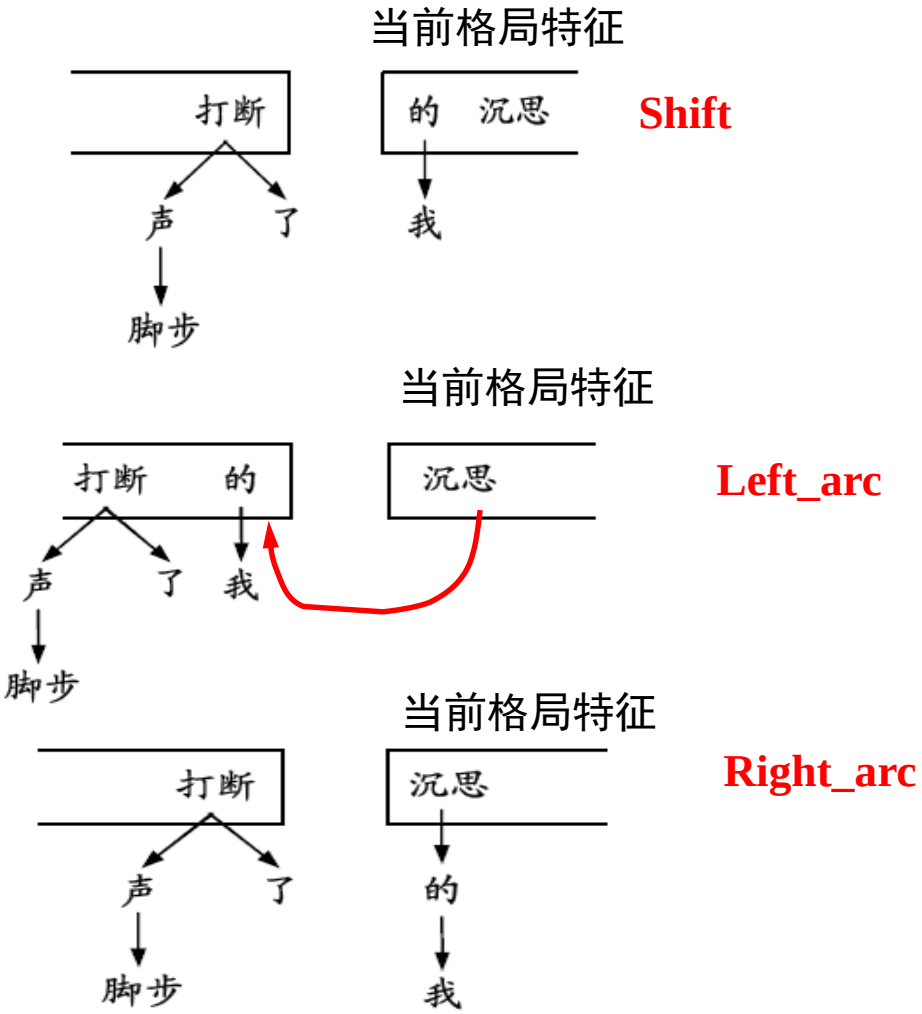
11.3.2.2 依存分析算法



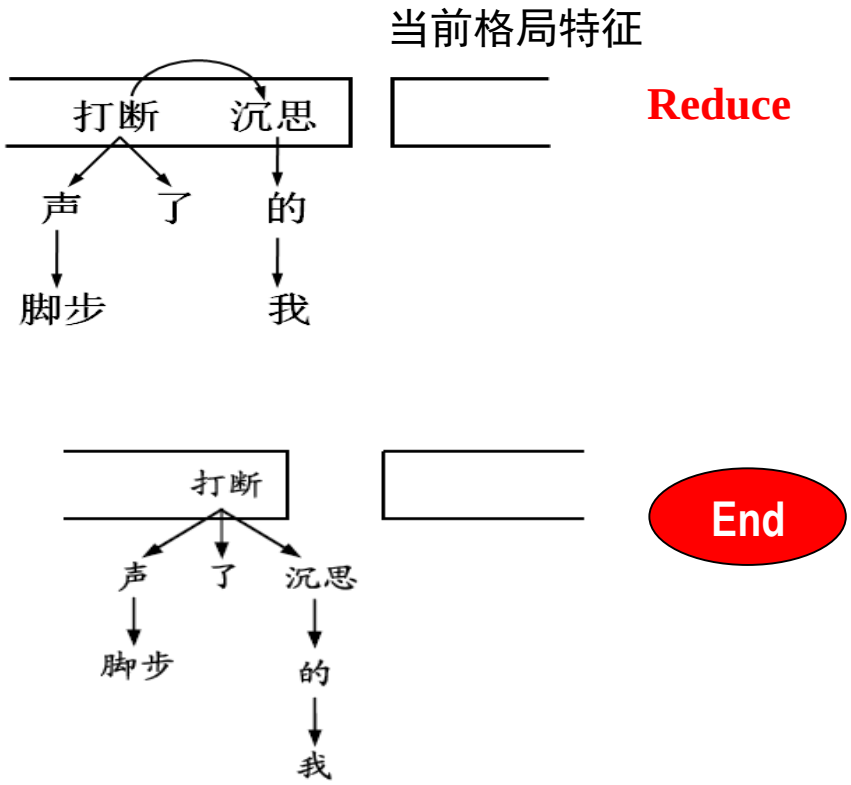
11.3.2.2 依存分析算法



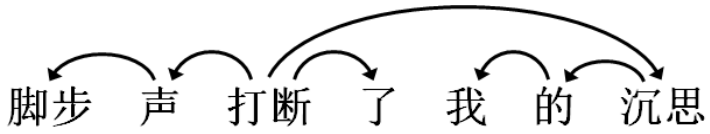
11.3.2.2 依存分析算法



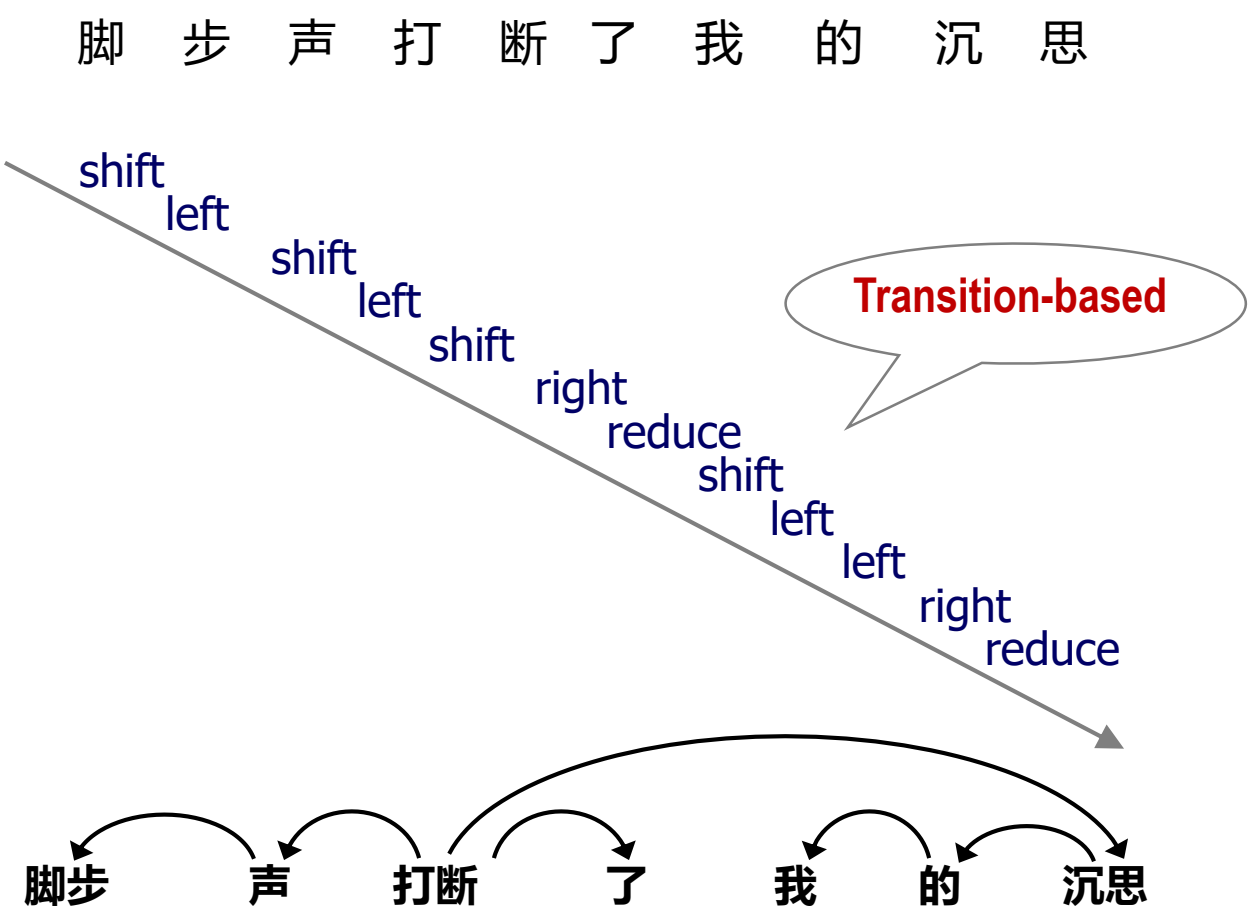
11.3.2.2 依存分析算法



分析结果



11.3.2.2 依存分析算法



11.3.2.2 依存分析算法

决策式方法评价

优点： 算法可以使用之前产生的所有句法结构作为特征；
可以达到线性复杂度：O(n)。

弱点： 以局部最优的加和代替全局最优，导致错误传递；
不可处理非投射现象，准确率稍逊于全局最优算法。

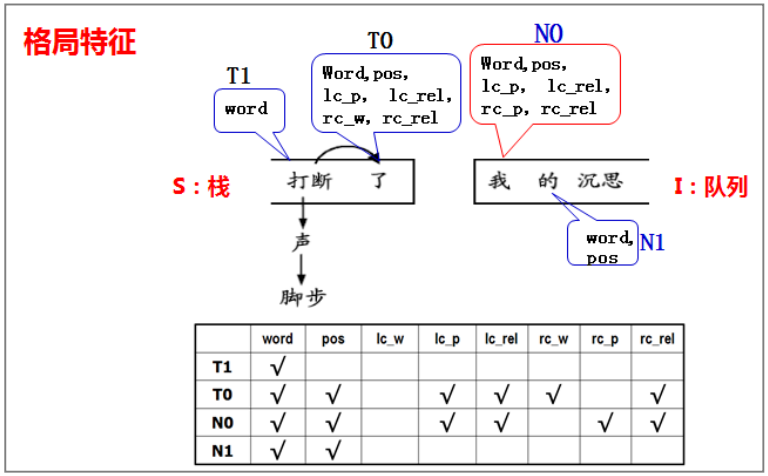
11.3.2.2 依存分析算法

★ 基于转移的决策式(确定性)分析神经网络方法

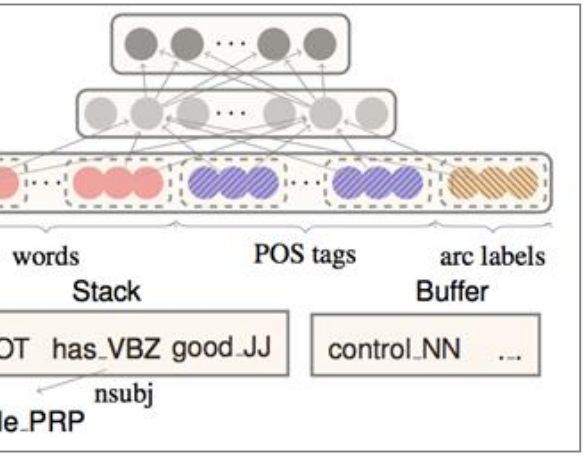
1. DNN 决策式分析 (Chen and Manning 2014)

基本思想:

将构成特征的信息项作为神经网的输入，由神经网自动进行特征提取和组合

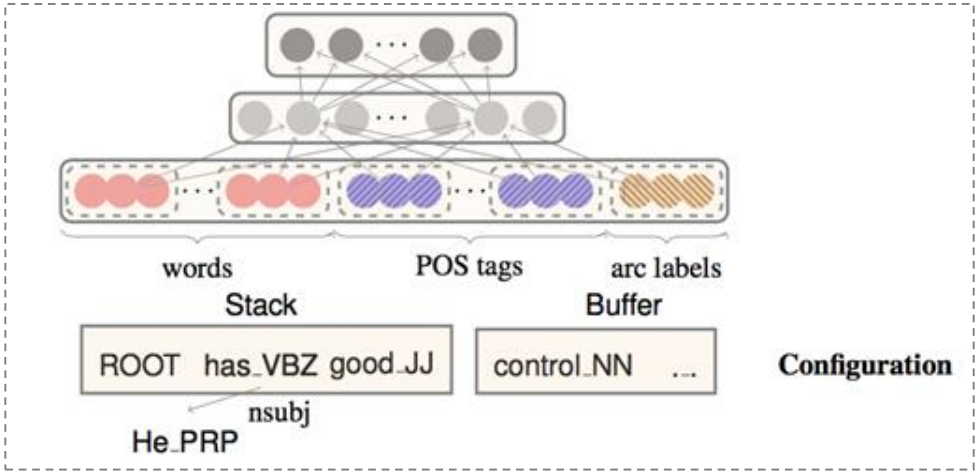


	word	pos	lc_w	lc_p	lc_rel	rc_w	rc_p	rc_rel
T1	√							
T0	√	√		√	√	√		√
N0	√	√		√	√		√	√
N1	√	√						

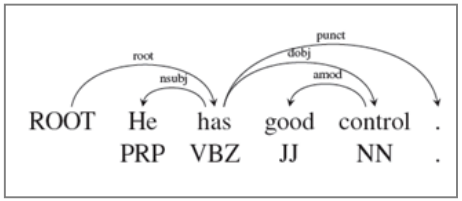


11.3.2.2 依存分析算法

■ **模型结构**



如：



输入： X^w： (1) S1, S2, S3, B1, B2, B3　　如：He, has, good, control , X , X

(2) $Lc_1(Si)$, $rc_1(Si)$, $Lc_2(Si)$, $rc_2(Si)$　i = 1,2

(3) $Lc_1(Lc_1(Si))$, $rc_1(rc_1(Si))$　i = 1,2　（X^w 共18 个词）

X^t： X^w 对应词的18 个POS

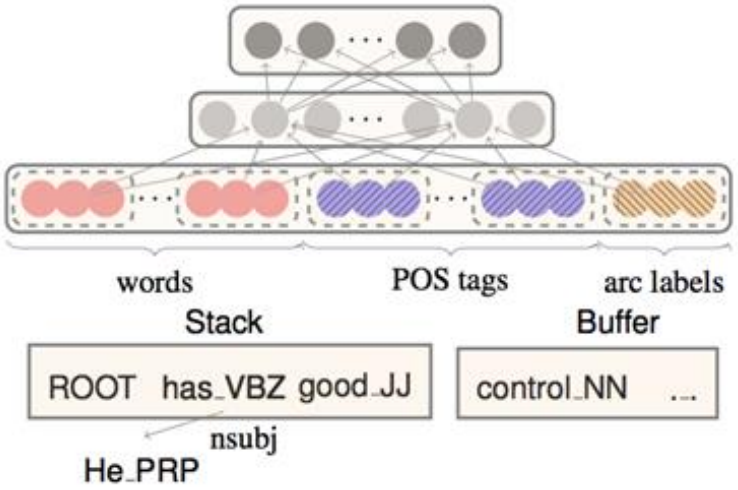
X^L： X^w 对应的12 条依存弧

输出： 动作（**Shift**、**Right-Arc$_l$**、**Left-Arc$_l$**、**Reduce**）的概率分布

参数： W_1^w、W_1^t、W_1^L、b1、W_2

11.3.2.2 依存分析算法

- **模型结构**



输出层： $p = \mathrm{softmax}(W_2 h)$

隐藏层： $h = (W_1^w x^w + W_1^t x^t + W_1^l x^l + b_1)^3$

输入层： $[x^w, x^t, x^l]$

输入： X^w, X^t, X^L

输出： 动作（Shift、Right-Arc_l、Left-Arc_l、Reduce）的概率分布

参数： W_1^w、W_1^t、W_1^L、b1、W_2

11.3.2.2 依存分析算法

■ 模型学习

训练实例 $\{(c_i, t_i)\}_{i=1}^{m}$

其中，C_i 是 一个格局， t_i 相应的动作

- **目标：** 最小化交叉熵损失

$$L(\theta) = -\sum_i \log p_{t_i} + \frac{\lambda}{2}\|\theta\|^2$$

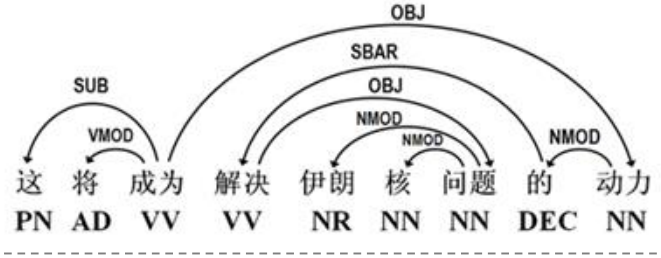
$$\theta = \{W_1^w, W_1^t, W_1^l, b_1, W_2, E^w, E^t, E^l\}$$

- **梯度下降可求 参数 θ**

11.3.2.2 依存分析算法

训练语料对比

依存树



统计法

动作 特征序列

	word	pos	lc_w	lc_p	lc_rel	rc_w	rc_p	rc_rel
T1	√							
T0	√	√		√	√	√		√
N0	√	√		√	√		√	√
N1	√	√						

Shift N0w:这 /N0p:PN /N1w:将N1p:AD

Shift T0w: 这/ T0p:PN /N0w:将 /N0p:AD/N1w:成为 /N1p:V

L_A T1w:这/ T0w:将/ T0p:AD /N0w:成为/ N0p:VV/N1w:解决 /N1p:VV

L_A T0w:这 /T0p:PN /N0w:成为 /N0p:VV /N0lc_p:AD/ N0lc_rel: VMOD/N1w:解决/ N1p:VV

Shift T0w:成为/T0p:VV/T0lc_p:PN/T0lc_rel:SUB/T0rc_w:将/T0rc_rel:VMOD/N0w:解决/ N0p:VV N1w:伊朗/N1p:NR

Shift T1w:成为/T0w:解决/T0p:VV/N0w:伊朗/N0p:NR/N1w:核/N1p:NN

.....

神经网络

X: 这 将 成为 解决 伊朗 核 问题 的 动力
PN AD VV VV NR NN NN DEC NN

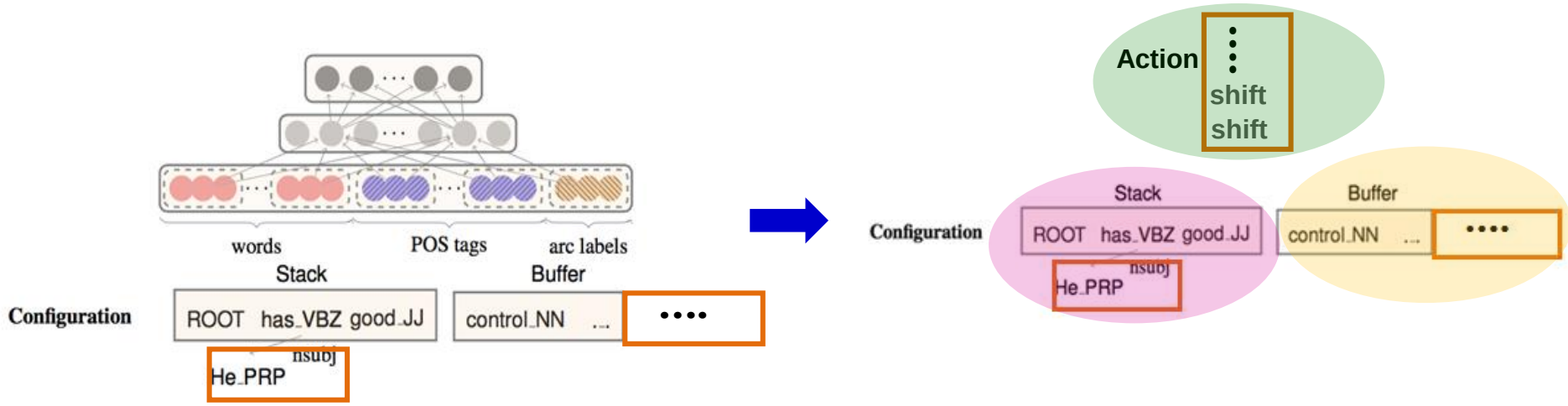
Y: Shift， Shift， Shift， L_A， L_A， Shift， Shift， Shift， Shift，
L_A， L_A， R_A， Shift， L_A， Shift， L_A， R_A

11.3.2.2 依存分析算法

★ 基于转移的决策式(确定性)分析神经网络方法

2. Stack LSTM: Dependency Parsing:

前一方法只使用了格局中部分结点的上下文信息进行依存分析；能否有某种方法使用全部信息（已生成的部分依存子树、输入队列子串、已有的分析动作序列）进行句法分析?



Dyer C etal .Transition-Based Dependency Parsing with Stack Long Short-Term Memory ,2015

11.3.2.2 依存分析算法

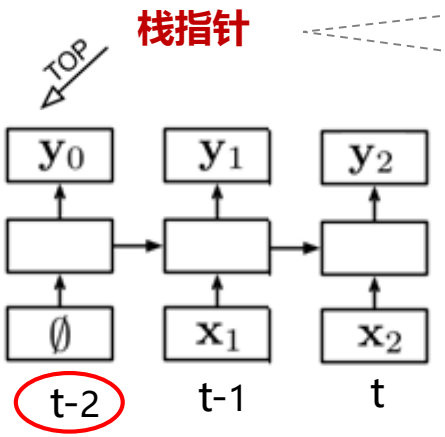
Stack LSTM: Dependency Parsing：

算法思想：将分析状态格局的 栈、队列和动作分别用三个stack-LSTM表示（分析格局信息为全部上下文信息），分析过程仍是每读入一个单词，根据该单词的当前格局和动作历史信息确定当前最佳动作，最后一步步拼装句法树。

11.3.2.2 依存分析算法

Stack LSTM Structure

LSTM



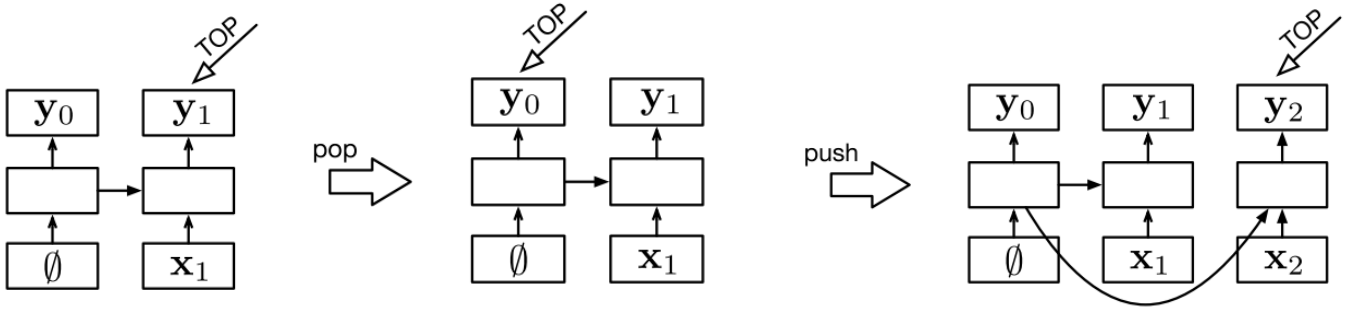
栈指针的位置决定了在计算新的cell上下文时LSTM中哪个cell 提供c_{t-1}和s_{t-1}

$$
\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_{ix}\mathbf{x}_t + \mathbf{W}_{ih}\mathbf{h}_{t-1} + \mathbf{W}_{ic}\mathbf{c}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_{fx}\mathbf{x}_t + \mathbf{W}_{fh}\mathbf{h}_{t-1} + \mathbf{W}_{fc}\mathbf{c}_{t-1} + \mathbf{b}_f) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \\
&\qquad \mathbf{i}_t \odot \tanh(\mathbf{W}_{cx}\mathbf{x}_t + \mathbf{W}_{ch}\mathbf{h}_{t-1} + \mathbf{b}_c), \\
\mathbf{o}_t &= \sigma(\mathbf{W}_{ox}\mathbf{x}_t + \mathbf{W}_{oh}\mathbf{h}_{t-1} + \mathbf{W}_{oc}\mathbf{c}_t + \mathbf{b}_o) \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t).
\end{aligned}
$$

$y_t = g(h_t)$　g是任意可微函数

11.3.2.2 依存分析算法

Stack LSTM Structure



栈操作：

- **pop** – 栈指针回到先前位置
- **push** – 输入加到栈顶

11.3.2.2 依存分析算法

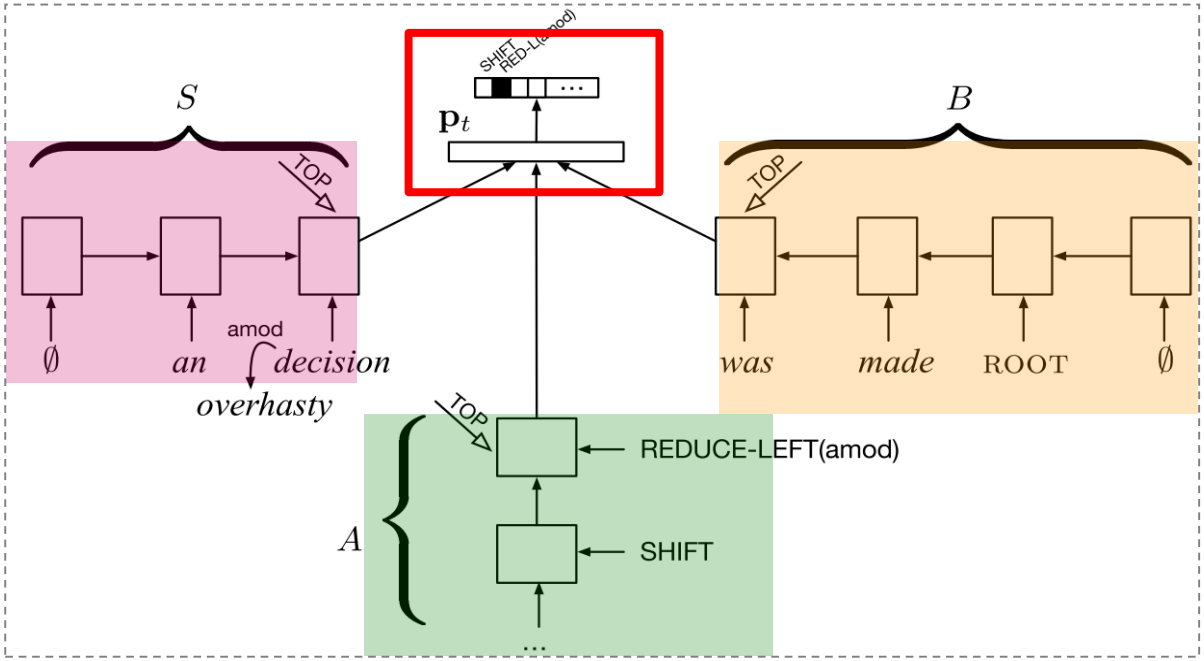
Stack LSTM: Dependency Parsing:

- **模型结构**

输出（动作）

队列：存放未处理的节点队列。用stack-LSTM 逆向存放

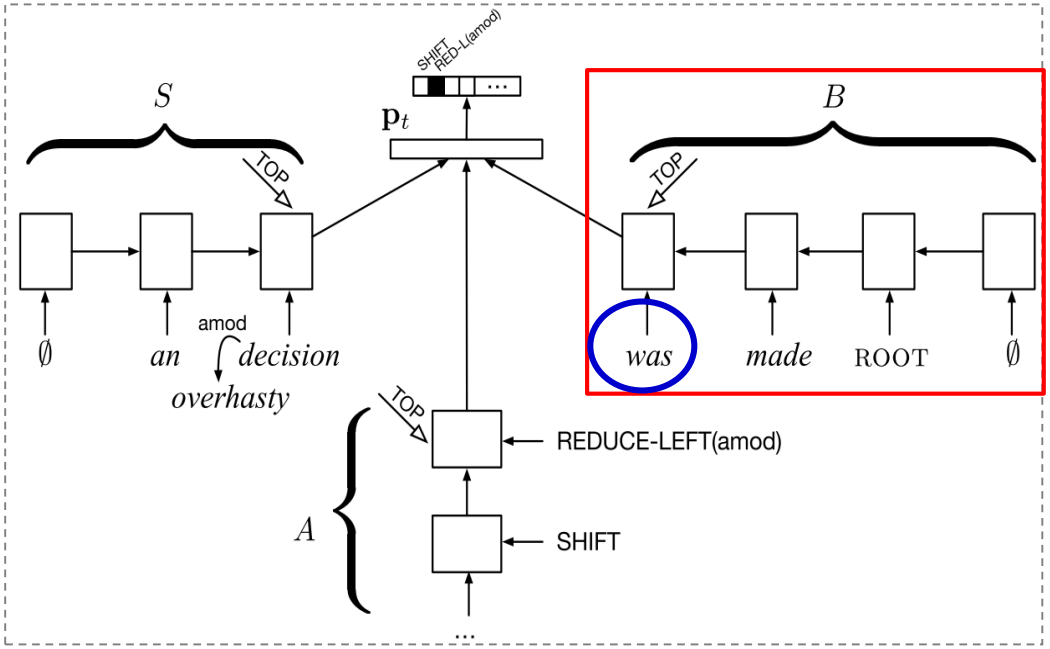
栈：存放已经处理过的片段的子树根。



栈A：存放分析过程动作序列集合

(Slide credits: Chris Dyer)

11.3.2.2 依存分析算法



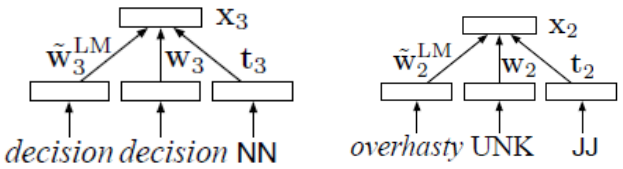
队列：存放未处理的节点队列。用stack-LSTM 逆向存放

输入：**X**

- 从神经语言模型中学到的向量表示($\tilde{w}_{LM}$)
- Word type的向量表示(w)
- POS的向量表示(t)

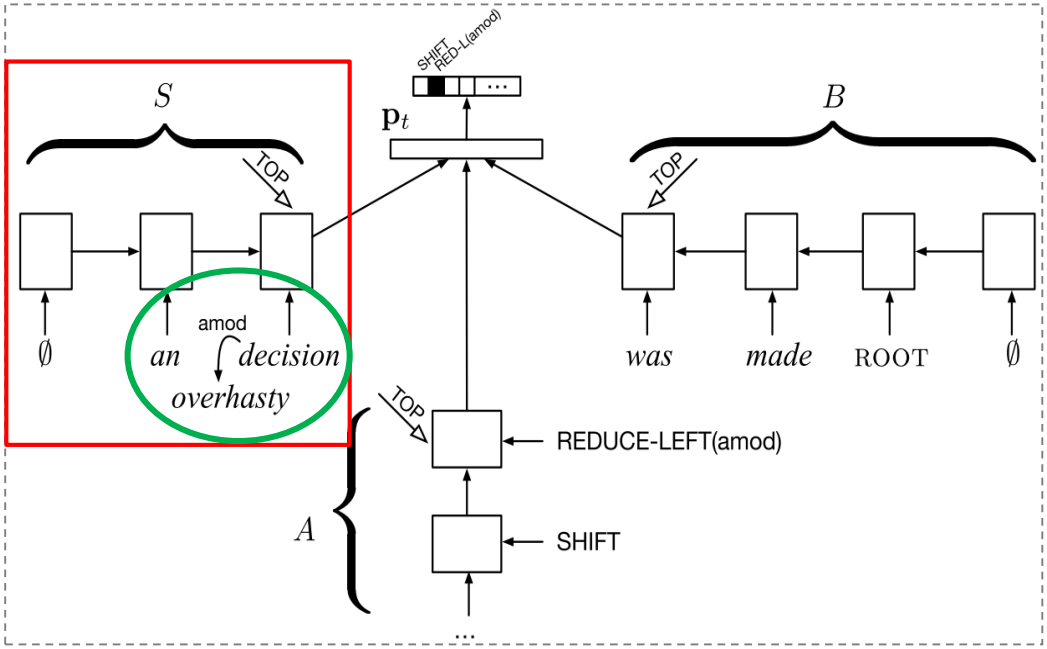
$$\mathbf{x} = \max\{\mathbf{0}, \mathbf{V}[\mathbf{w}; \tilde{\mathbf{w}}_{\mathrm{LM}}; \mathbf{t}] + \mathbf{b}\}$$

如：



11.3.2.2 依存分析算法

栈：存放已经处理过的片段的子树根节点

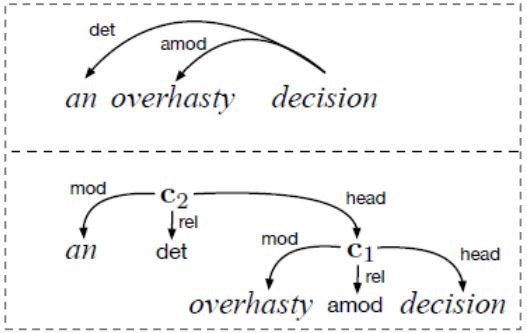


根节点：

- 节点可以是单个的单词

$$\mathbf{x} = \max\{\mathbf{0}, \mathbf{V}[\mathbf{w}; \tilde{\mathbf{w}}_{\mathrm{LM}}; \mathbf{t}] + \mathbf{b}\}$$

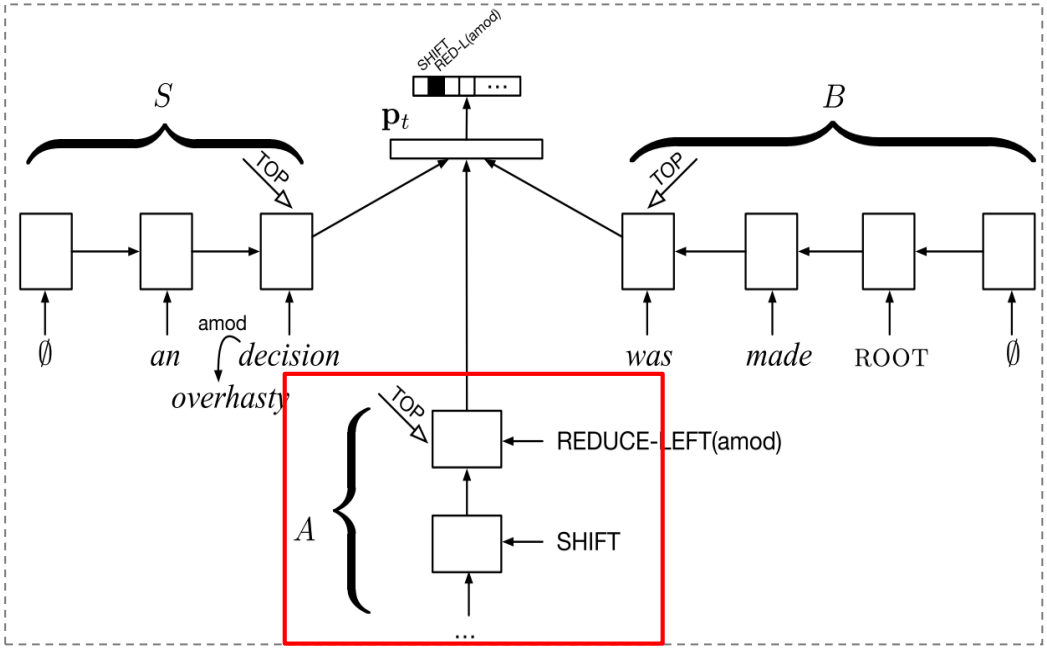
- 也可以是部分已经建立好的句法树片段



$$\mathbf{c} = \tanh(\mathbf{U}[\mathbf{h}; \mathbf{d}; \mathbf{r}] + \mathbf{e})$$

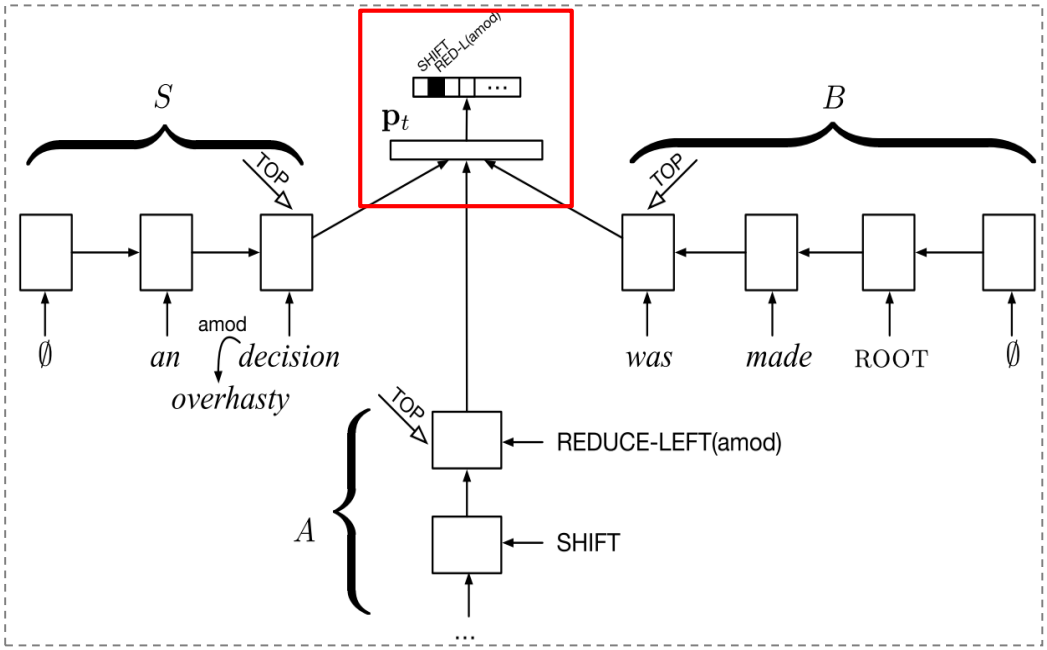
其中：h 为依存关系头，d为尾，r为依存关系；U、e 为参数

11.3.2.2 依存分析算法



栈A：存放分析过程动作序列集合（只做Push操作），每一个动作序列包括动作类型+依存关系名称。依存关系名称：取决于语料库中依存关系label

Stack_t	Buffer_t	Action	Stack_{t+1}	Buffer_{t+1}	Dependency
$(\mathbf{u}, u), (\mathbf{v}, v), S$	B	REDUCE-RIGHT(r)	$(g_r(\mathbf{u}, \mathbf{v}), u), S$	B	$u \xrightarrow{r} v$
$(\mathbf{u}, u), (\mathbf{v}, v), S$	B	REDUCE-LEFT(r)	$(g_r(\mathbf{v}, \mathbf{u}), v), S$	B	$u \xleftarrow{r} v$
S	$(\mathbf{u}, u), B$	SHIFT	$(\mathbf{u}, u), S$	B	—



输出： （动作）

$$\mathbf{p}_t = \max\{\mathbf{0}, \mathbf{W}[\mathbf{s}_t; \mathbf{b}_t; \mathbf{a}_t] + \mathbf{d}\}$$

$$p(z_t \mid \mathbf{p}_t) = \frac{\exp\left(\mathbf{g}_{z_t}^\top \mathbf{p}_t + q_{z_t}\right)}{\sum_{z' \in \mathcal{A}(S,B)} \exp\left(\mathbf{g}_{z'}^\top \mathbf{p}_t + q_{z'}\right)}$$

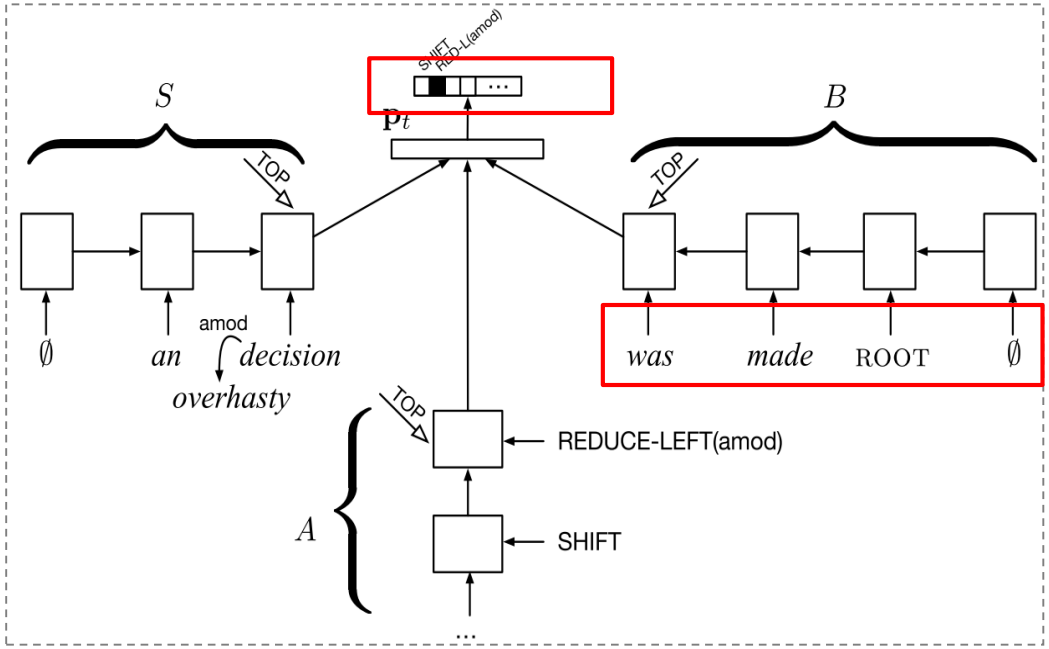
where

P_t:parser state embedding

g_z : the (output) embedding of the parser action z

q_z : bias term for action z.

11.3.2.2 依存分析算法



输入：词的表示序列

输出：动作序列

参数：

$$\mathbf{x} = \max\{\mathbf{0}, \mathbf{V}[\mathbf{w}; \tilde{\mathbf{w}}_{\mathrm{LM}}; \mathbf{t}] + \mathbf{b}\}$$

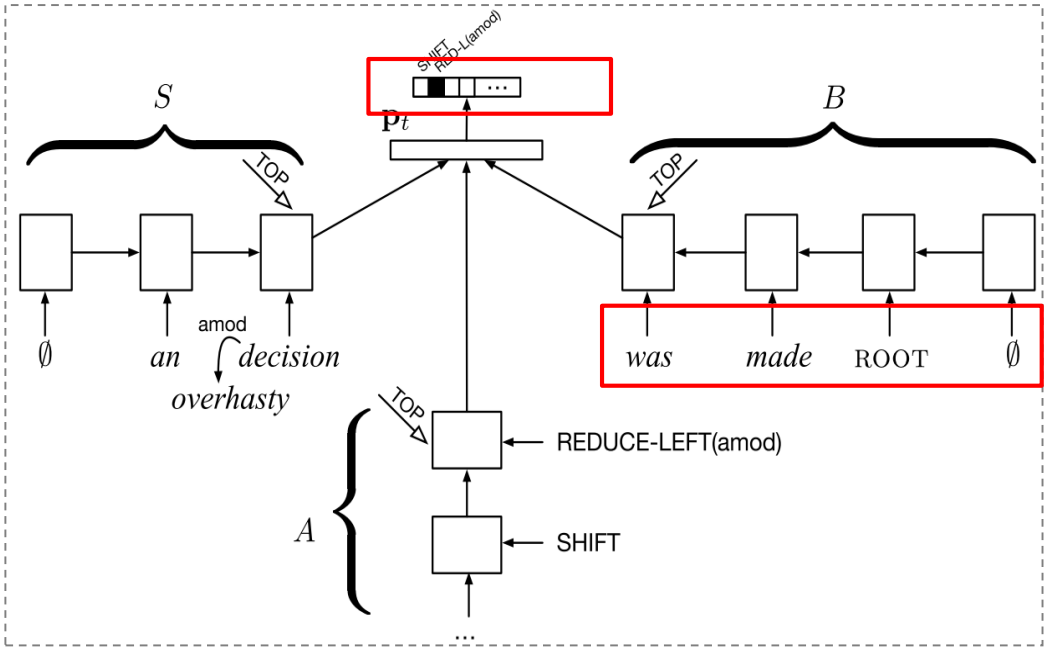
$$\mathbf{c} = \tanh\left(\mathbf{U}[\mathbf{h}; \mathbf{d}; \mathbf{r}] + \mathbf{e}\right)$$

$$\mathbf{p}_t = \max\{\mathbf{0}, \mathbf{W}[\mathbf{s}_t; \mathbf{b}_t; \mathbf{a}_t] + \mathbf{d}\}$$

$$p(z_t \mid \mathbf{p}_t) = \frac{\exp\left(\mathbf{g}_{z_t}^{\top}\mathbf{p}_t + q_{z_t}\right)}{\sum_{z' \in \mathcal{A}(S,B)} \exp\left(\mathbf{g}_{z'}^{\top}\mathbf{p}_t + q_{z'}\right)}$$

11.3.2.2 依存分析算法

■ 模型的训练



输入：词的表示序列

输出：动作序列

参数：

$$\mathbf{x} = \max\{\mathbf{0}, \mathbf{V}[\mathbf{w}; \tilde{\mathbf{w}}_{\mathrm{LM}}; \mathbf{t}] + \mathbf{b}\}$$

$$\mathbf{c} = \tanh\left(\mathbf{U}[\mathbf{h}; \mathbf{d}; \mathbf{r}] + \mathbf{e}\right)$$

$$\mathbf{p}_t = \max\{\mathbf{0}, \mathbf{W}[\mathbf{s}_t; \mathbf{b}_t; \mathbf{a}_t] + \mathbf{d}\}$$

$$p(z_t \mid \mathbf{p}_t) = \frac{\exp\left(\mathbf{g}_{z_t}^{\top}\mathbf{p}_t + q_{z_t}\right)}{\sum_{z' \in \mathcal{A}(S,B)} \exp\left(\mathbf{g}_{z'}^{\top}\mathbf{p}_t + q_{z'}\right)}$$

- **优化目标：**

$$p(\boldsymbol{z} \mid \boldsymbol{w}) = \prod_{t=1}^{|\boldsymbol{z}|} p(z_t \mid \mathbf{p}_t)$$

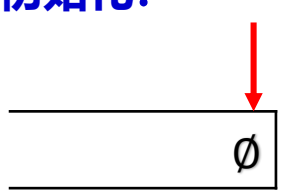
by minimizing the cross-entropy (maximizing the log-likelihood) of the correct transition $p(z_t|p_t)$ at each state along the path.

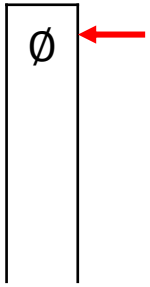
11.3.2.2 依存分析算法

- **分析过程：**

分析句子： 脚步 声 打断 了 我 的 沉思

初始化：

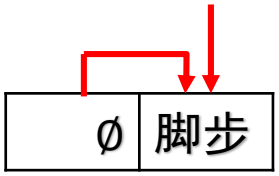


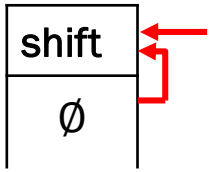


步骤	action

根据当前三个stack-lstm的状态计算，得到当卡最佳动作序列为shift

11.3.2.2 依存分析算法

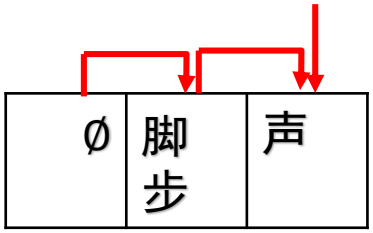


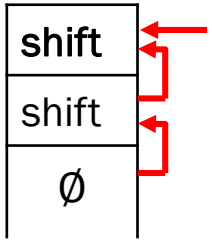


步骤	action
1	shift

根据当前三个stack-lstm的状态计算，得到当前最佳动作序列为shift

11.3.2.2 依存分析算法

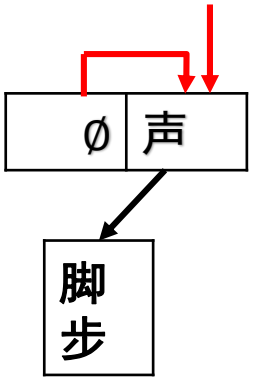




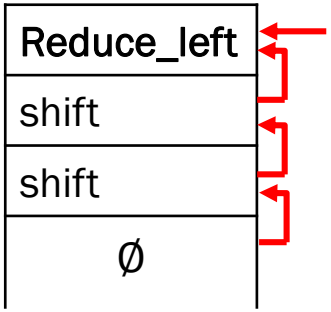
步骤	action
1	shift
2	shift

根据当前三个stack-lstm的状态计算，得到当卡最佳动作序列为reduce_left

11.3.2.2 依存分析算法



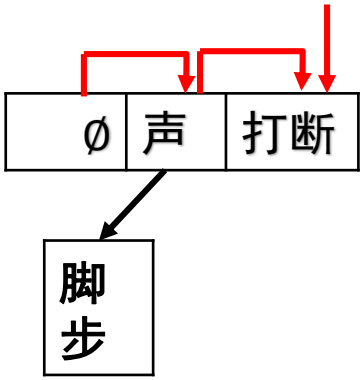
$$\mathbf{c} = \tanh\left(\mathbf{U}[\mathbf{h};\mathbf{d};\mathbf{r}] + \mathbf{e}\right).$$

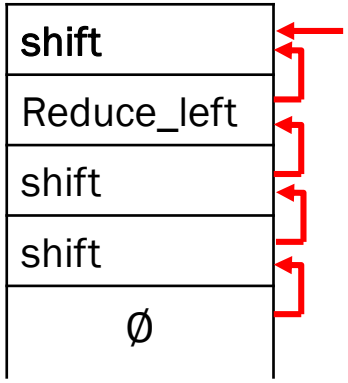


步骤	action
1	shift
2	shift
3	Reduce_left

根据当前三个stack-lstm的状态计算，得到当卡最佳动作序列为shift

11.3.2.2 依存分析算法

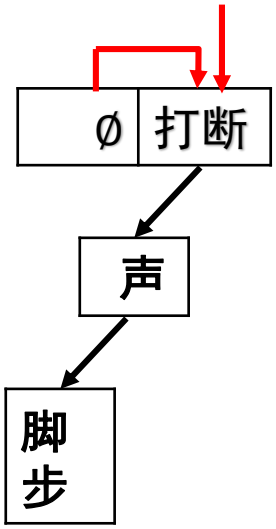


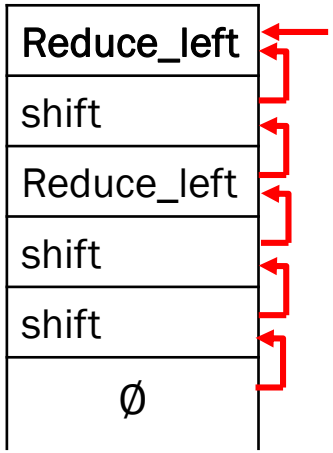


步骤	action
1	shift
2	shift
3	Reduce_left
4	shift

根据当前三个stack-lstm的状态计算，得到当卡最佳动作序列为reduce_left

11.3.2.2 依存分析算法

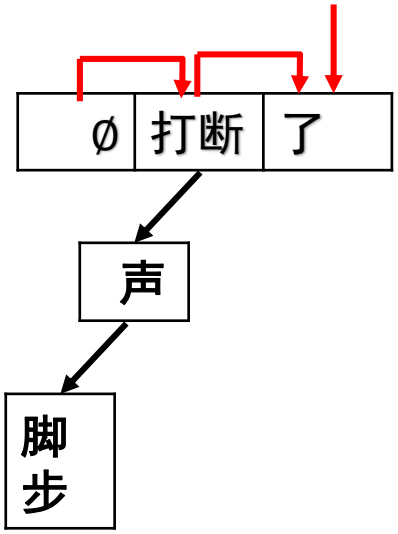


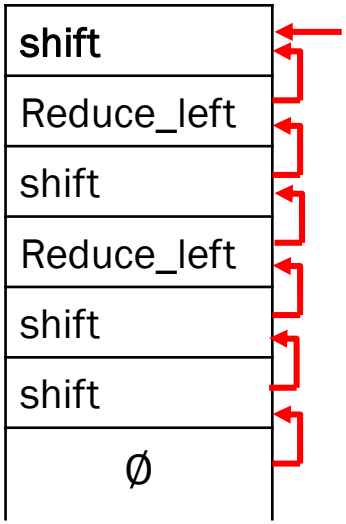


步骤	ction
1	shift
2	shift
3	Reduce_left
4	shift
5	Reduce_left

根据当前三个stack-lstm的状态计算，得到当卡最佳动作序列为shift

11.3.2.2 依存分析算法

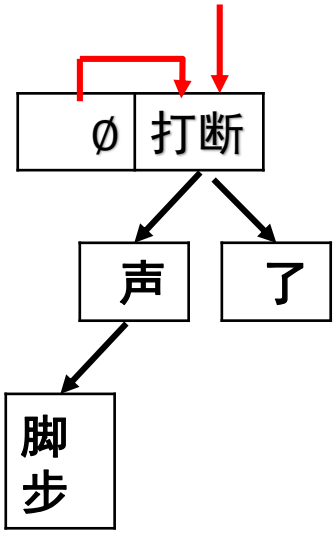


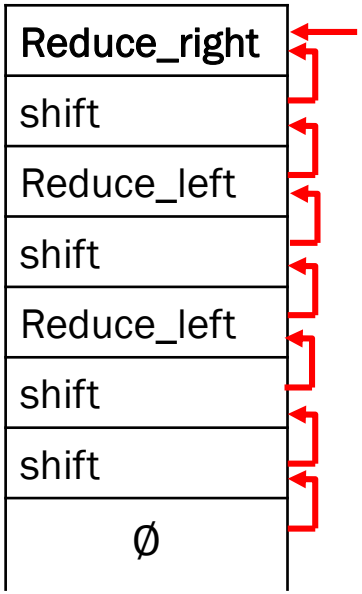


根据当前三个stack-lstm的状态计算，得到当卡最佳动作序列为reduce_right

步骤	ction
1	shift
2	shift
3	Reduce_left
4	shift
5	Reduce_left
6	shift

11.3.2.2 依存分析算法

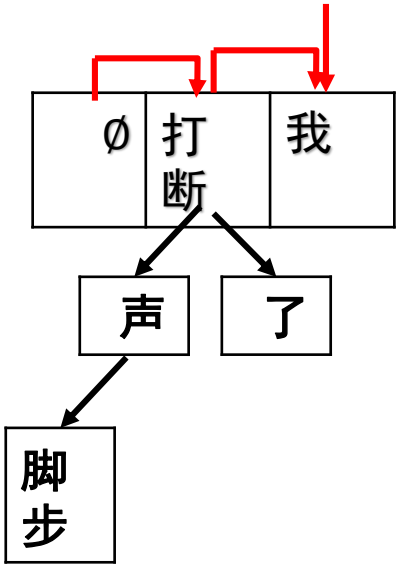


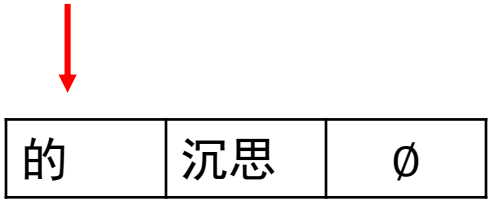


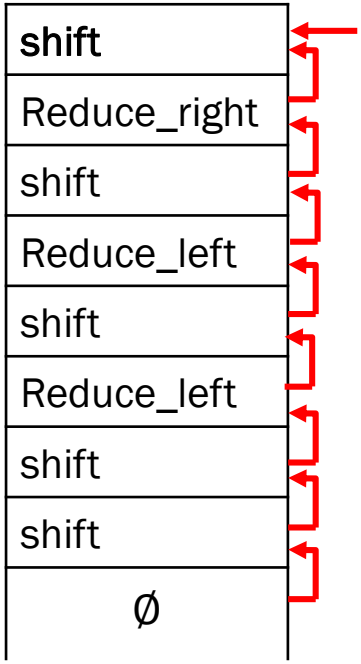
根据当前三个stack-lstm的状态计算，得到当卡最佳动作序列为shift

步骤	ction
1	shift
2	shift
3	Reduce_left
4	shift
5	Reduce_left
6	shift
7	Reduce_right

11.3.2.2 依存分析算法



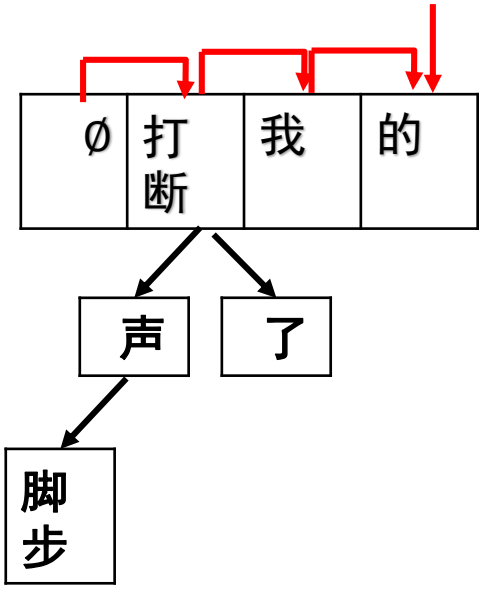


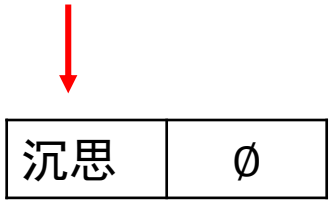


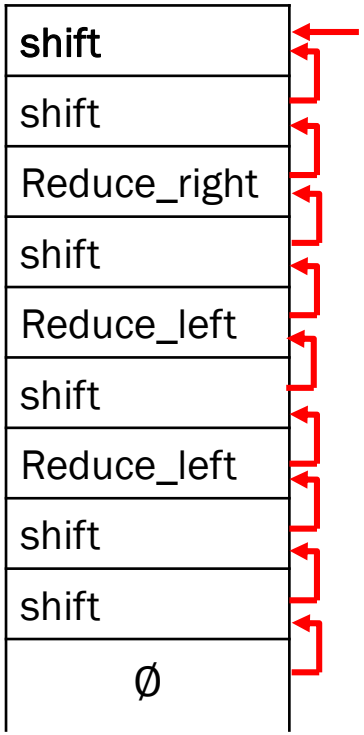
根据当前三个stack-lstm的状态计算，得到当卡最佳动作序列为shift

步骤	ction
1	shift
2	shift
3	Reduce_left
4	shift
5	Reduce_left
6	shift
7	Reduce_right
8	shift

11.3.2.2 依存分析算法



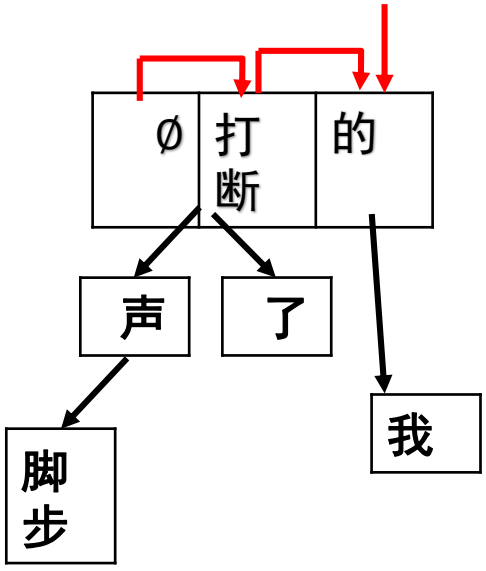


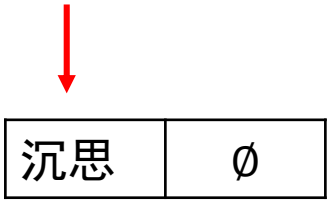


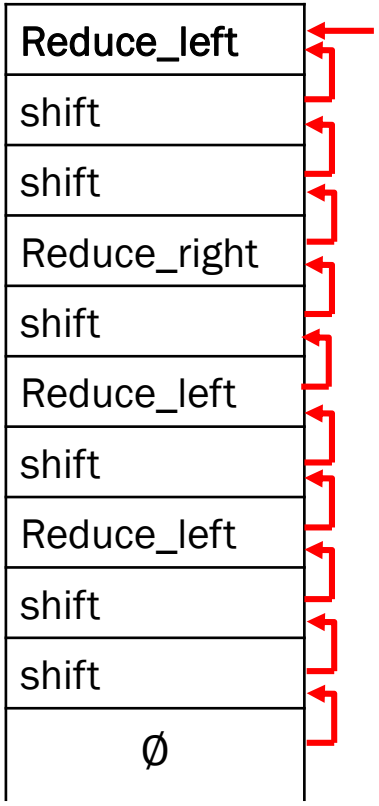
根据当前三个stack-lstm的状态计算，得到当卡最佳动作序列为reduce_left

步骤	ction
1	shift
2	shift
3	Reduce_left
4	shift
5	Reduce_left
6	shift
7	Reduce_right
8	shift
9	shift

11.3.2.2 依存分析算法



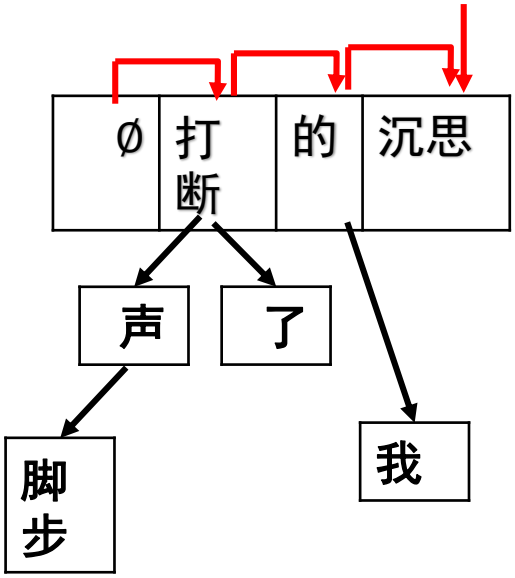


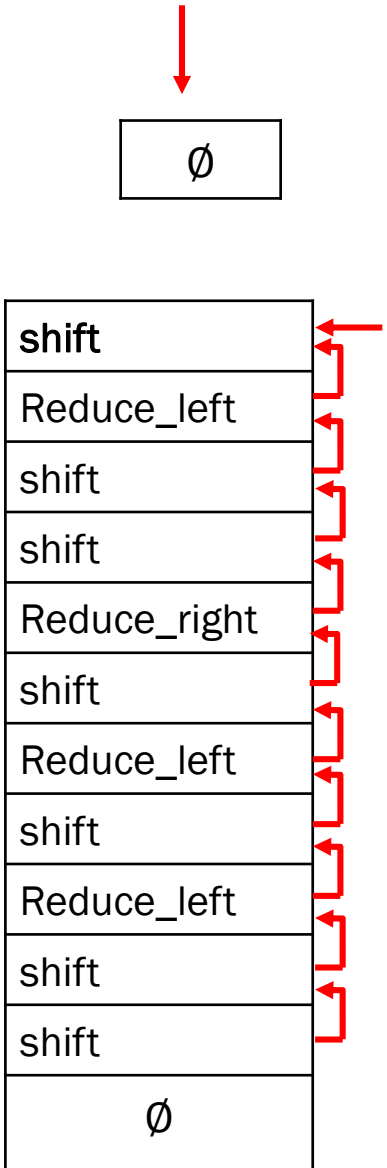


根据当前三个stack-lstm的状态计算，得到当卡最佳动作序列为shift

步骤	ction
1	shift
2	shift
3	Reduce_left
4	shift
5	Reduce_left
6	shift
7	Reduce_right
8	shift
9	shift
10	Reduce_left

11.3.2.2 依存分析算法

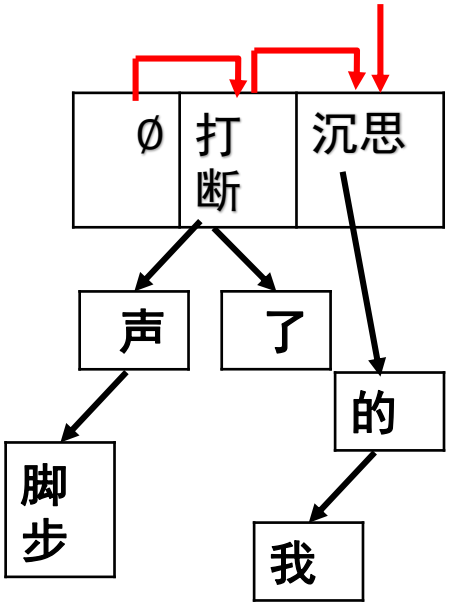


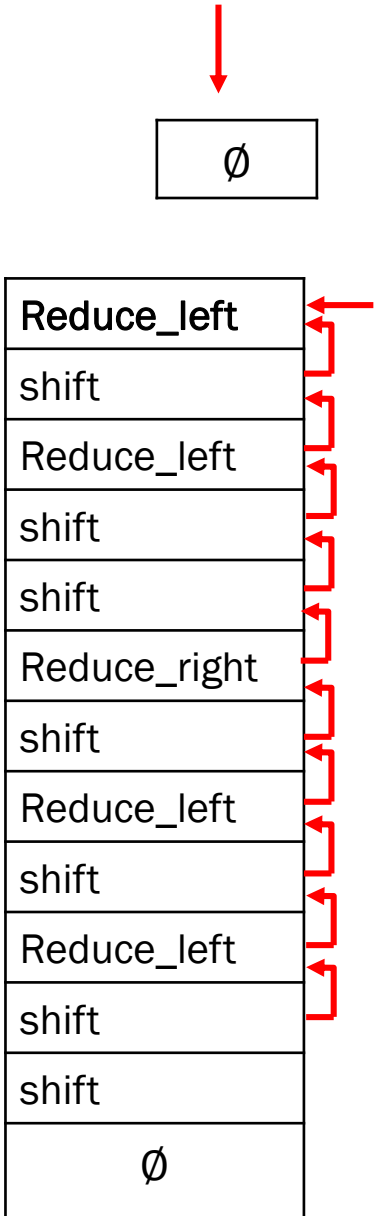


根据当前三个stack-lstm的状态计算，得到当卡最佳动作序列为reduce_left

步骤	ction
1	shift
2	shift
3	Reduce_left
4	shift
5	Reduce_left
6	shift
7	Reduce_right
8	shift
9	shift
10	Reduce_left
11	shift

11.3.2.2 依存分析算法

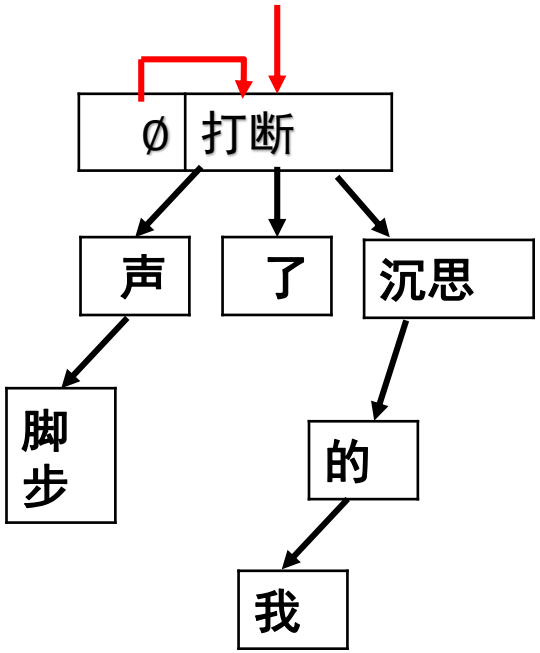


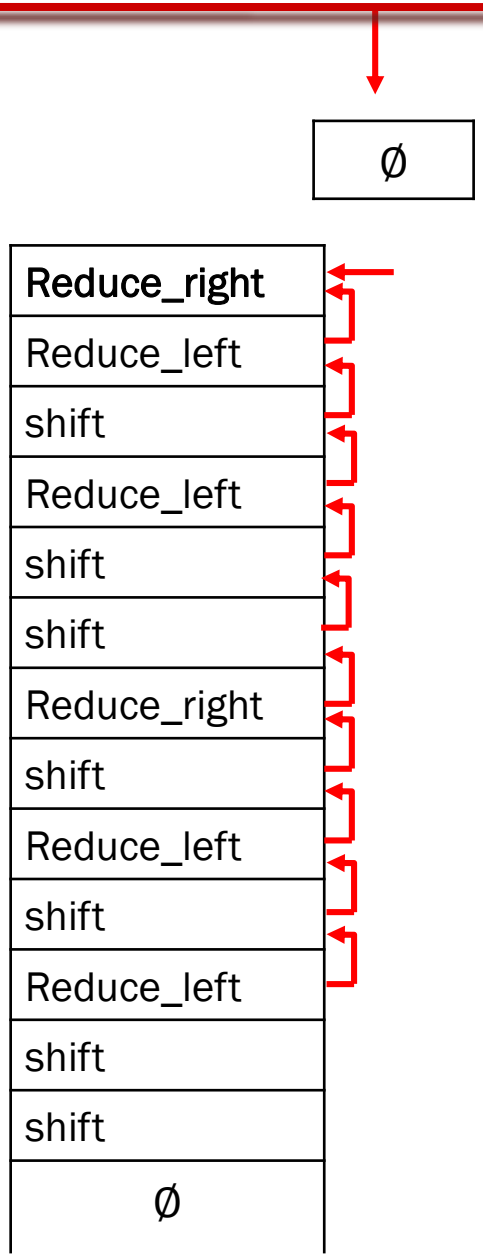


根据当前三个stack-lstm的状态计算，得到当卡最佳动作序列为reduce_right

步骤	ction
1	shift
2	shift
3	Reduce_left
4	shift
5	Reduce_left
6	shift
7	Reduce_right
8	shift
9	shift
10	Reduce_left
11	shift
12	Reduce_left

11.3.2.2 依存分析算法



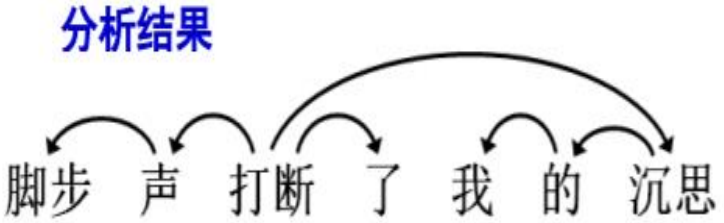


步骤	ction
1	shift
2	shift
3	Reduce_left
4	shift
5	Reduce_left
6	shift
7	Reduce_right
8	shift
9	shift
10	Reduce_left
11	shift
12	Reduce_left
13	Reduce_right

11.3.2.2 依存分析算法

Reduce_right
Reduce_left
shift
Reduce_left
shift
shift
Reduce_right
shift
Reduce_left
shift
Reduce_left
shift
shift
Ø

拼装句法树



依存句法分析-内容提要

11.3.1　依存文法

11.3.2　依存句法分析方法

11.3.3　依存句法分析评价

11.3.4　短语结构与依存结构关系

11.3.3 依存句法分析评价

依存句法分析器性能评测指标

- **无标记依存正确率**(unlabeled attachment score, UA)：所有词中找到其正确支配词的词所占的百分比，没有找到支配词的词(即根结点)也算在内。
- **带标记依存正确率**(labeled attachment score, LA)：所有词中找到其正确支配词并且依存关系类型也标注正确的词所占的百分比，根结点也算在内。
- **依存正确率**(dependency accuracy, DA)：所有非根结点词中找到其正确支配词的词所占的百分比。

11.3.3 依存句法分析评价

- **根正确率**(root accuracy, RA)：有两种定义方式：

  (1)正确根结点的个数与句子个数的比值；

  (2)另一种是所有句子中找到正确根结点的句子所占的百分比。

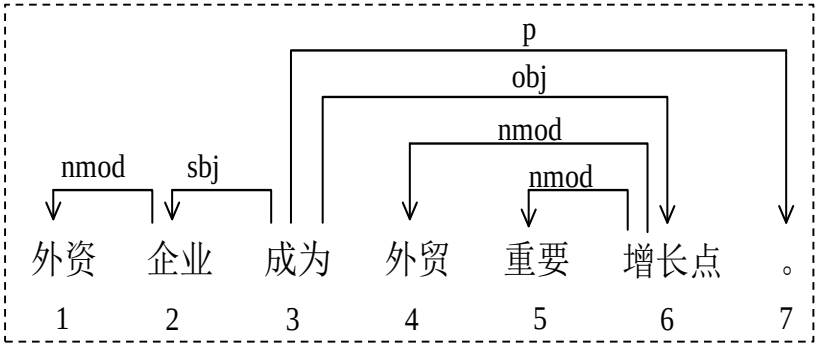
  对单根结点语言或句子来说，二者是等价的。

- **完全匹配率**(complete match, CM)：所有句子中无标记依存结构完全正确的句子所占的百分比。

11.3.3 依存句法分析评价

如：

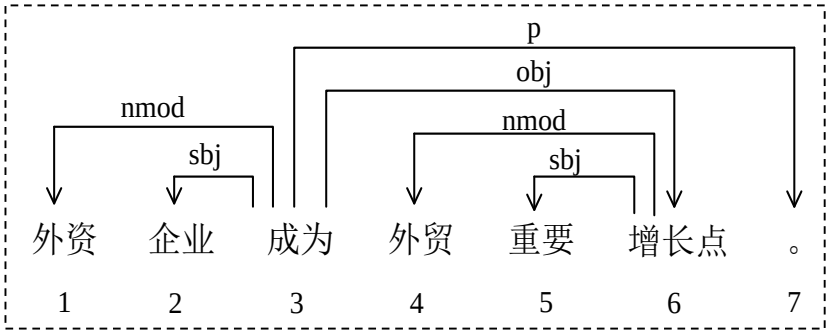
标准依存树



每个词对应的支配词及依存关系为：

外资(2,nmod), 企业(3,sbj), 成为(0,root),外贸(6,nmod), 重要(6,nmod), 增长点(3,obj), 。(3,p)

系统输出分析树



每个词对应的支配词及依存关系为：

外资(3,nmod)，企业(3,sbj)，成为(0,root)，外贸(6,nmod)，重要(6,sbj)，增长(3,obj)，。(3,p)

$$\text{无标记依存正确率（UA）} = \frac{6}{7} \times 100\% = 85.7\%$$

$$\text{带标记依存正确率（LA）} = \frac{5}{7} \times 100\% = 71.4\%$$

$$\text{依存正确率（DA）} = \frac{5}{6} \times 100\% = 83.33\%$$

依存句法分析-内容提要

11.3.1　依存文法

11.3.2　依存句法分析方法

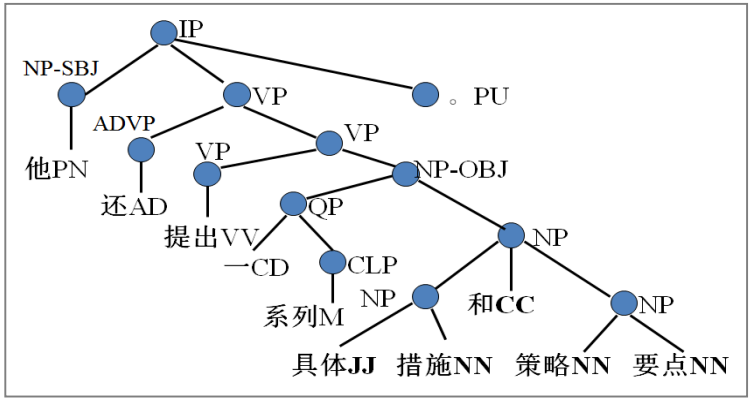
11.3.3　依存句法分析评价

11.3.4　短语结构与依存结构关系

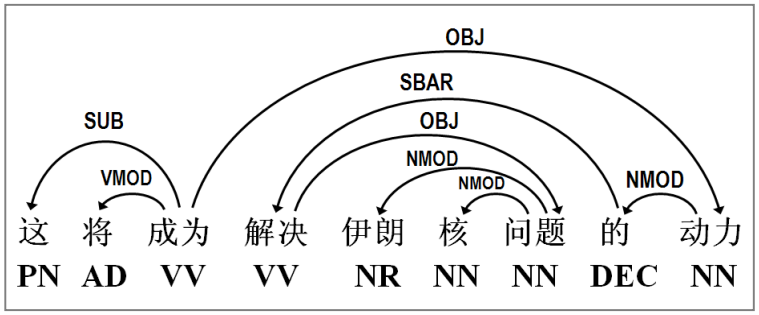
11.3.4 短语结构与依存结构关系

依存树与短语结构树的区别：

短语结构树



依存树



（1）依存树只有具体的词构成的终结结点，而短语树中即含 终结结点又含非终结结点。

（2）从分支上看，依存树中的父子关系表示相应的两个词之间的支配和被支配关系；而短语树上的分支表示子结点是父结点的组成成分， 依存树偏重关系结构，短语树偏重组成结构。

（3）对同一个句子，依存树层次不多，结点数目少，短语树结构树层次多，结点数目多。

（4）短语树从句子生成的角度分析，偏重结构；依存树的依存关系有词与词之间的语义关系，故语义分析中常用依存句法分析。

短语结构和依存结构是目前句法分析中研究最广泛的两类文法体系

11.3.4 短语结构与依存结构关系

短语结构可转换为依存结构

实现方法：

(1) 定义中心词抽取规则，产生中心词表;

(2) 根据中心词表，为句法树中每个节点选择中心子节点;

(3) 将非中心子节点的中心词依存到中心子节点的中心词上，得到相应的依存结构。

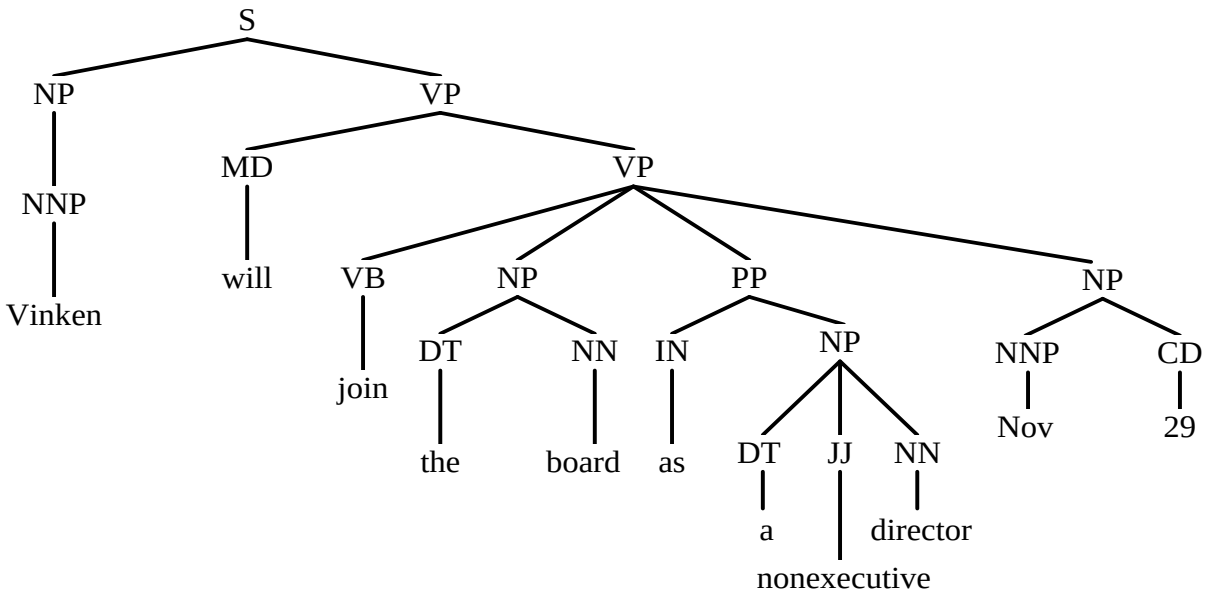
英语中心词提取规则：[Yamada and Matsumoto ,2003] 和 [Collins, 1999]

汉语中心词提取规则：[Sun and Jurafsky ,2004] 和 [Zhang and Clark, 2008]

11.3.4 短语结构与依存结构关系

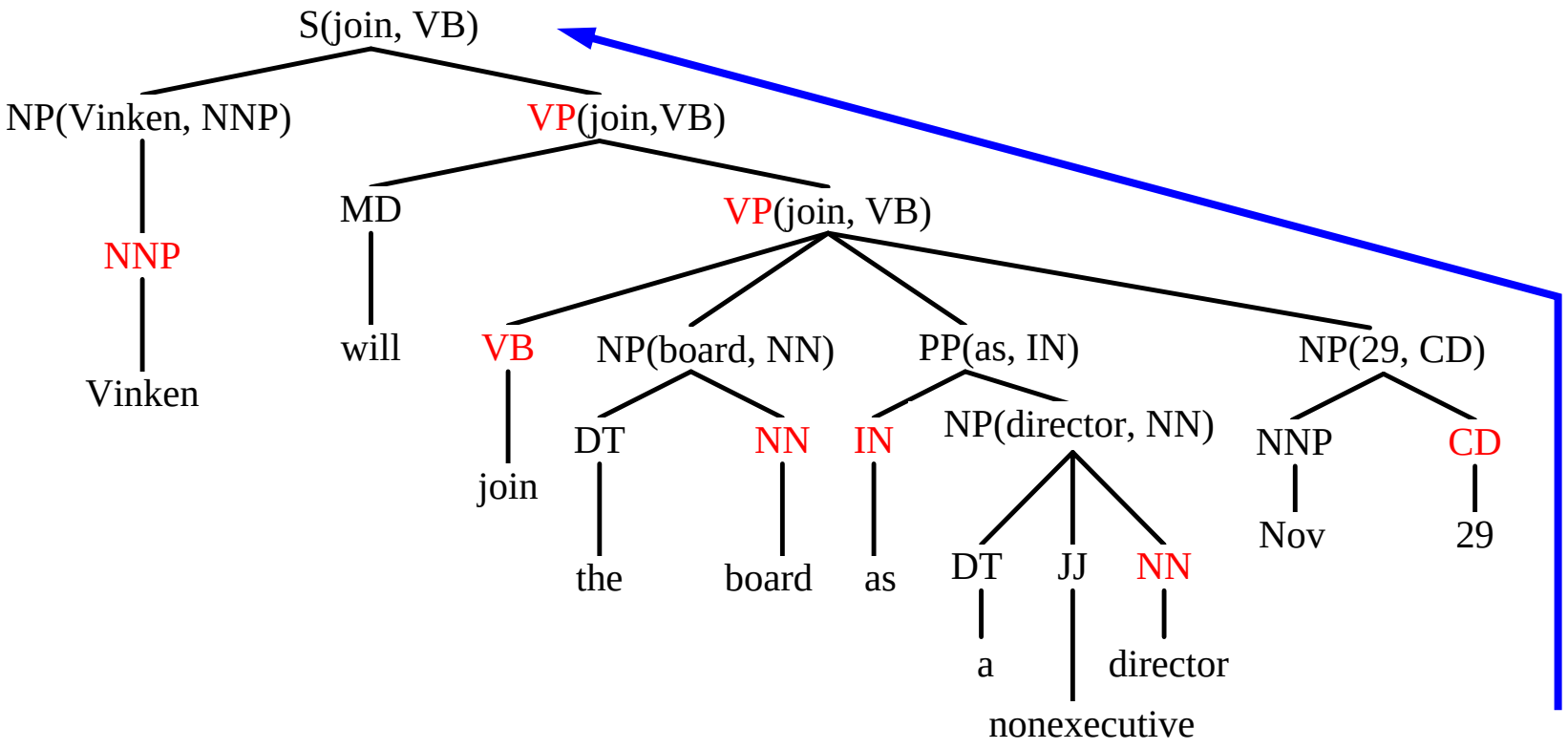
例如：给定如下短语结构树，转换成依存结构

Vinken will join the board as a nonexecutive director Now 29



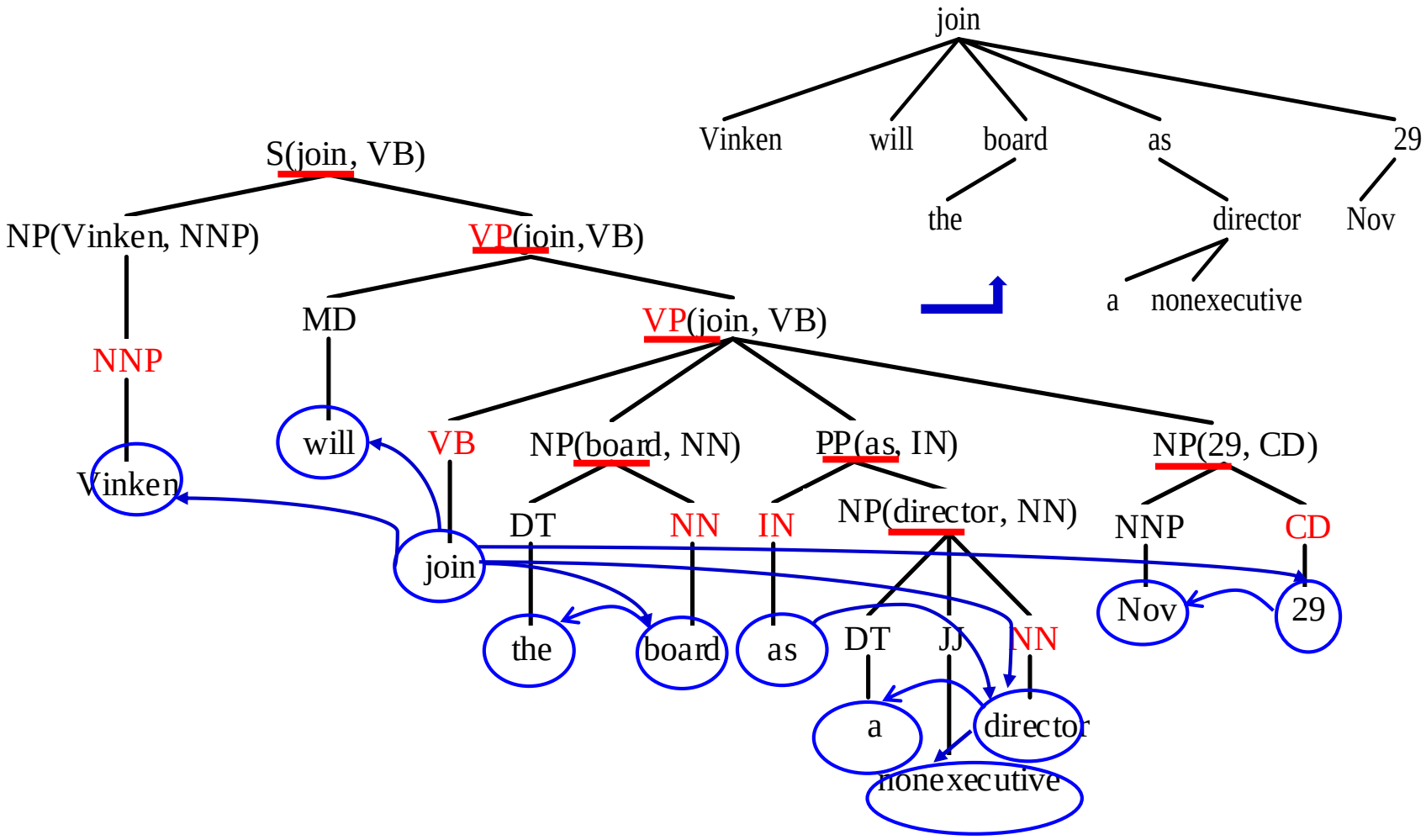
11.3.4 短语结构与依存结构关系

根据中心词表为每个节点选择中心子节点 (中心词通过自底向上传递得到)



11.3.4 短语结构与依存结构关系

将非中心子节点的中心词依存到中心子节点的中心词上



11.3.4 短语结构与依存结构关系

部分开源的依存句法分析器

- Stanford Parser

  http://nlp.stanford.edu/downloads/lex-parser.shtml

- MST Parser (Minimum-Spanning Tree Parser)

  http://www.seas.upenn.edu/~strctlrn/MSTParser/MSTParser.html

- MaltParser http://maltparser.org/index.html

- MINIPAR Parser (only for English)

  http://webdocs.cs.ualberta.ca/~lindek/minipar.htm

- Layer-based Dependency Parser (LDPar)

  http://www.openpr.org.cn/ (访问Download→ NLP Toolkit)

参考文献：

宗成庆，统计自然语言处理（第2版）课件

http://www.hankcs.com/nlp/corpus/chinese-treebank.html

http://www.hankcs.com/nlp/to-achieve-a-simple-generative-dependency-parsing.html

http://www.hankcs.com/nlp/parsing/crf-sequence-annotation-chinese-dependency-parser-implementation-based-on-java.html

Chris Dyer，Miguel Ballesteros，Transition-Based Dependency Parsing with Stack Long Short-Term Memory，ACL,2015

Danqi Chen，Christopher D. Manning，A Fast and Accurate Dependency Parser using Neural Networks

在 此 表 示 感 谢!

信息内容安全技术国家工程实验室
National Engineering Laboratory for Information Security Technologies

谢谢各位！