第3章 传统自然语言处理分析

目录

一、形式文法-乔姆斯基文法体系

3.1 形式文法-Chomsky文法体系

如果一个语言的词汇集是一个有限集V，对V中的元素毗连计算可以得到符号串集合V*，那V*就是由V构成的句子。

而通过毗连计算得到的字符串如果一个语言的词汇集是一并不一定都是某种语言中的句子。例如，“the man saw the ball”（“人看球”)在英语中是正确的,而由同样符号构成的“the saw the man ball”在英语中却是不正确的。我们把前者叫作成立的句子,后者叫作不成立的句子,而要区别一种语言中的成立的句子和

3.1 形式文法-Chomsky文法体系

不成立的句子，就必须采用某些办法把语言刻画出来，从而说明，在这一种语言中，什么样的句子是成立的什么样的句子是不成立的。我们可以采用三种办法来刻画语言。

（1）穷举法 — 把语言中全部成立的句子穷尽枚举出来。只适合句子数目有限的语言。

（2）语法描述 — 制定有限数量的规则来生成语言中无限个数的句子，这些句子是语言中合格的句子。这种能够刻画语言的有限个数的规则称为文法。记为G。

3.1 形式文法-Chomsky文法体系

（3）语言识别程序自动机 — 设计一种装置来检验输入符号串，来识别该符号串是不是语言L中成立的句子，如果是，这个装置就接收，如果不是语言中成立的句子，这个装置就不接收。

由此可见，刻画某类语言的有效手段是文法和自动机，文法用于生成语言，而自动机则用于识别语言。

美国著名语言学家乔姆斯基（N.Chomsky）将文法抽象成一个四元组，称为短语结构文法或短语结构语法。

3.1 形式文法-Chomsky文法体系

3.1.1短语结构语法理论

一种语言就是一个句子集，它包含了属于这种语言的全部句子，而语法是对这些句子的一种有限的形式化描述。可以利用一种基于产生式的形式化工具对某种语言的语法进行描述。

一部短语结构语法G可以用一个四元组来定义：

$$G=(Vt，Vn，P，S)$$

Vt： 终结符集合，终结符是指被定义的那个语言的词或符号；

Vn： 非终结符的集合，这些符号不能出现在最终生成的句子中，是专门用来描述语法的。Vt和Vn的并(∪)构成了符号集

3.1 形式文法-Chomsky文法体系

V，称为总词汇表，且Vt和Vn不相交，因此有：V=Vt∪Vn，Vt∩Vn=φ（φ表示空集）；

P： 有穷产生式集：　　α→β

式中α∈V*VnV*,β∈V*,*表示它前面的字符可以出现任意次；

S： 非终结符表Vn的一个元素，称为起始符。

下面就是采用短语结构语法对一个英语子集（受限英语）的语法的描述：

G=(Vt,Vn,P,S)

Vn={S, NP, VP, Det, N, V, Prep, PP}

Vt={the, girl, letter, pencil, write, with, a}

3.1 形式文法-Chomsky文法体系

S=S

P: S→ NP VP

NP→ Det N

VP→ V NP

VP→ VP PP

PP→ Prep NP

Det→ the | a

N→ girl | letter | pencil

V→ write

Prep→with

这一语法所描述的英语子集中，只有**the、girl、Letter、pencil、write、with、a**几个单词。

3.1 形式文法-Chomsky文法体系

形式文法的直观意义

形式文法是用来精确地描述语言（包括人工语言和自然语言）及其结构的手段。**形式语言学** 也称 **代数语言学**。

以重写规则 $\alpha \to \beta$ 的形式表示，其中， α，β 均为字符串。顾名思义：字符串 α 可以被改写成 β。一个初步的字符串通过不断地运用重写规则，就可以得到另一个字符串。通过选择不同的规则并以不同的顺序来运用这些规则，就可以得到不同的新字符串。

3.1 形式文法-Chomsky文法体系

3.1.2 约束的短语结构语法——乔姆斯基语法体系

短语结构语法是用于描述语言特性的一种形式体系。对于一种形式体系，如果它能定义的语言类型越多，就说它的描述能力越强。例如，如果形式体系T1可以定义5种语言，而形式体系T2可以定义10种语言，而且包含了所有可以被T1定义的5种语言，就说T2比T1具有更强的描述能力。

理论语言学家将语言分成两类：递归语言和可递归枚举语言。

对于一种语言，若能编写一部程序，使其能以某种顺序逐个地输出该语言的全部句子，就称该语言是可递归枚举的（生成）；

3.1 形式文法-Chomsky文法体系

如果能编一部程序，使其能在读入一个符号串后，可以判断该符号串是否为该语言的句子，就称该语言是递归的（可识别）。

一种语言可以是可递归枚举的，但却不一定是递归的，因为对给定的一个符号串，可能无法判断它是否是该语言的一个句子。

乔姆斯基语法体系是一组受限的短语结构语法，降低它的描述能力。他定义了四种语法：0型语法、1型语法、2型语法和3型语法。

3.1 形式文法-Chomsky文法体系

0型语法：是一种无约束的短语结构语法，也就是前面已经介绍的短语结构语法。

1型语法：也称做上下文有关语法，是一种满足下列约束条件的短语结构语法：

对于每一条形式为

$$x \rightarrow y$$

的产生式，符号串y中所包含的字符个数不少于字符串x中所包含的字符个数，而且$x,y \in V^*$。

3.1 形式文法-Chomsky文法体系

2型语法：也称做上下文无关语法，是一种满足下列约束条件的短语结构语法：

对于每一条形式为

$$A \rightarrow x$$

的产生式，其左侧必须是一个单独的非终结符，而右侧则是任意的符号串，即$A \in Vn, x \in V^*$。在这种语法中，由于产生式规则的应用不依赖于符号A所处的上下文，因此，称为上下文无关语法。

3.1 形式文法-Chomsky文法体系

3型语法：也称做正则语法，分左线性语法和右线性语法两种形式。在左线性语法中，每一条产生式的形式为

$$A \rightarrow Bt \quad 或 \quad A \rightarrow t$$

而在右线性语法中，每一条产生式的形式为

$$A \rightarrow tB \quad 或 \quad A \rightarrow t$$

这里，A和B都是单独的非终结符，t是单独的终结符，即$A, B \in Vn$，$t \in Vt$。

在这四种语法中，型号越高，所受到的约束就越多，其生成语言的能力就越弱，因而生成的语言集就越小，也更易于对其生成的语言进行计算机自动分析。

3.1 形式文法-Chomsky文法体系

3.1.3 句法分析树

在对一个句子进行分析的过程中，如果把分析句子各成份间关系的推导过程用树形图表示出来的话，那么，这种图称做句法分析树。

图3.1就是依据上述定义的语法对语句 The girl writes the letter with a pencil 进行句法分析时建立的句法分析树。

在句法分析树中，起始符总是出现在树的根上，终结符则出现在树的叶子上。

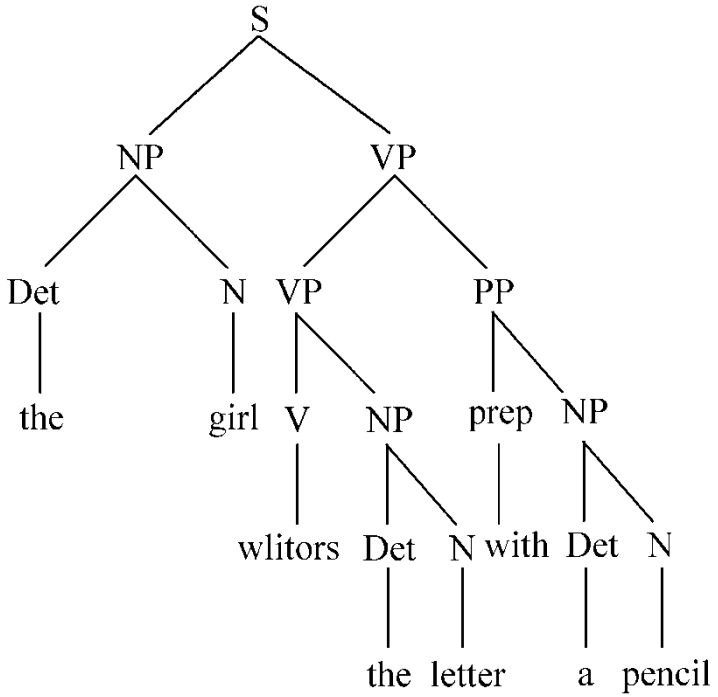


图3.1 句法分析树示例

二、句法分析方法

3.2 基于短语结构的句法分析

基于短语结构语法的自动分析算法主要有自顶向下回溯算法、自底向上并行算法、富田算法、左角分析算法和CYK算法等等。今天我们先介绍自顶向下回溯算法和自底向上并行算法。

3.2.1 自顶向下回溯算法

自顶向下分析算法的思想就是从起始符开始向着被分析的句子进行推导，推导过程的语法树建立从根节点开始，自上而下进行。每次推导只选择一种路径进行尝试，并保留其他可选择的路径，当推导失败时，进行回溯，尝试另一种推导路径。

3.2 基于短语结构的句法分析

例如，我们定义下面的一个语法：

G=(Vt，Vn，P，S)

Vn={S, NP, VP, Det, N, V, Prep, PP}

Vt={the, girl, letter, pencil, writes, with, a}

S=S

P: S→ NP VP	**(a)**
NP→ Det N	**(b)**
VP→ V NP	**(c)**
VP→ VP PP	**(d)**
PP→ Prep NP	**(e)**
Det→ the | a	**(f)**
N→ girl | letter | pencil	**(g)**
V→ writes	**(h)**
Prep→with	**(i)**

3.2 基于短语结构的句法分析

应用定义的语法对句子“the girl writes the letter with a pencil”的分析过程。

搜索步骤	搜索对象	所使用的规则	输入句子中遗留部分
(1)	S	(a)	the girl writes the letter with a pencil
(2)	NP VP	(b)	the girl writes the letter with a pencil
(3)	Det N VP	(f)	the girl writes the letter with a pencil
(4)	the N VP		the girl writes the letter with a pencil
(5)	N VP	(g)	girl writes the letter with a pencil
(6)	girl VP		girl writes the letter with a pencil
(7)	VP	(c)	writes the letter with a pencil
(8)	V NP	(h)	writes the letter with a pencil
(9)	writes NP		writes the letter with a pencil
(10)	NP	(b)	the letter with a pencil
(11)	Det N	(f)	the letter with a pencil
(12)	the N		the letter with a pencil
(13)	N	(g)	letter with a pencil
(14)	letter		letter with a pencil
(15)			with a pencil

3.2 基于短语结构的句法分析

这时，句子中还有遗留部分，但搜索对象中却已变空，分析过程已无法继续，只得回溯。回溯到第（7）步，看看是否还能利用别的规则进行分析。

(7′)	VP	(d)	writes the letter with a pencil
(16)	VP PP	(c)	writes the letter with a pencil
(17)	V NP PP	(h)	writes the letter with a pencil
(18)	writes NP PP		writes the letter with a pencil
(19)	NP PP	(b)	the letter with a pencil
(20)	Det N PP	(f)	the letter with a pencil
(21)	the N PP		the letter with a pencil
(22)	N PP	(g)	letter with a pencil
(23)	letter PP		letter with a pencil
(24)	PP	(e)	with a pencil
(25)	Prep NP	(i)	with a pencil
(26)	with NP		with a pencil

3.2 基于短语结构的句法分析

(27)	NP	(b)	a pencil
(28)	Det N	(f)	a pencil
(29)	a N		a pencil
(30)	N	(g)	pencil
(31)	pencil		pencil
(32)	NIL		NIL

在应用规则(f)和(g)对搜索对象进行替换时，由于规则的右边有多个单词可供选择，这时，可根据句子遗留部分的第一个单词确定。

3.2 基于短语结构的句法分析

3.2.2 自底向上并行算法

自底向上分析算法是从输入句子的句首开始依次取词向前移进，并应用合适的语法规则逐级向上归约（产生式倒过来用），直到构造出表示句子结构的整个推导树为止。换句话说，句法树的建立从树底部的叶节点（即词和词类）开始，直到根部。

本算法实际上分移进、归约两个步骤。所谓移进，就是把一个尚未处理过的符号移入栈顶，并等待更多的信息到来之后再做决定；所谓归约，就是对栈顶的那些与某一语法规则右边相匹配的符号，用该语法规则左边的符号来取代。

3.2 基于短语结构的句法分析

在移进-归约过程中，可能会出现有多条语法规则符合归结条件，这种情况称为“归约-归约”冲突；

也可能出现既符合移进条件又符合归约条件的情况，在这种情况下是移进还是归约呢？这称做“移进-归约”冲突。

解决这两种冲突是移进-归约算法的中心问题。

下面以对句子“**the girl writes the letter with a pencil**”的分析为例，说明采用移进-归约算法进行自底向上分析的过程。

3.2 基于短语结构的句法分析

步骤	栈	操作	输入句子中的遗留部分
(1)			the girl writes the letter with a pencil
(2)	the	移进	girl writes the letter with a pencil
(3)	Det	用规则(f)归约	girl writes the letter with a pencil
(4)	Det girl	移进	writes the letter with a pencil
(5)	Det N	用规则(g)归约	writes the letter with a pencil
(6)	NP	用规则(b)归约	writes the letter with a pencil
(7)	NP writes	移进	the letter with a pencil
(8)	NP V	用规则(h)归约	the letter with a pencil
(9)	NP V the	移进	letter with a pencil
(10)	NP V Det	用规则(f)归约	letter with a pencil
(11)	NP V Det letter	移进	with a pencil
(12)	NP V Det N	用规则(g)归约	with a pencil

3.2 基于短语结构的句法分析

(13)	NP V NP	用规则(b)归约	with a pencil
(14)	NP VP	用规则(c)归约	with a pencil
(15)	S	用规则(a)归约	with a pencil
(16)	S with	移进	a pencil
(17)	S Prep	用规则(i)归约	a pencil
(18)	S Prep a	移进	pencil
(19)	S Prep Det	用规则(f)归约	pencil
(20)	S Prep Det pencil	移进	
(21)	S Prep Det N	用规则(g)归约	
(22)	S Prep NP	用规则(b)归约	
(23)	S PP	用规则(e)归约	

3.2 基于短语结构的句法分析

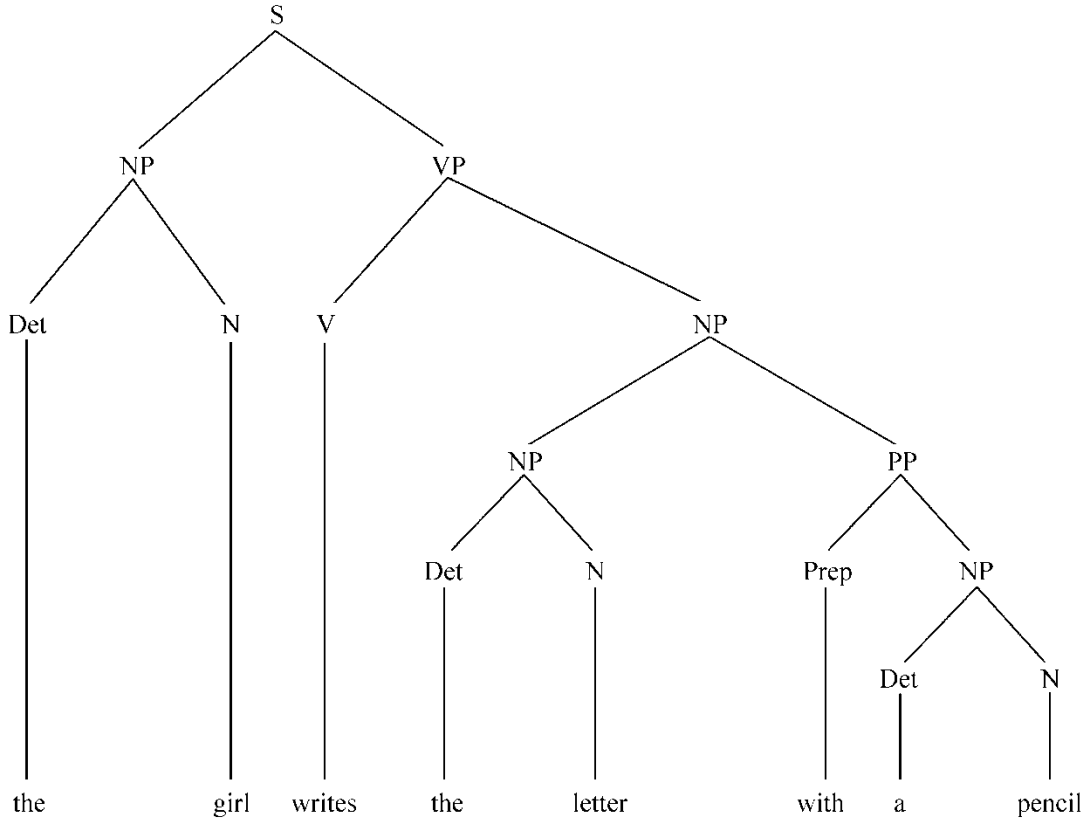
这时，输入句子串已空，但栈中却不是只有起始符S，况且语法中已无合适的规则可用来归约。进行回溯，返回到(14)，在这一步，先不采用规则(a)对其进行归约，而是移进下一个单词with，再使用规则(i)归约。

(14‘)	NP VP	回溯	with a pencil
(24)	NP VP with	移进	a pencil
(25)	NP VP Prep	用规则(i)归约	a pencil
(26)	NP VP Prep a	移进	pencil
(27)	NP VP Prep Det	用规则(f)归约	pencil
(28)	NP VP Prep Det pencil	移进	
(29)	NP VP Prep Det N	用规则(g)归约	
(30)	NP VP Prep NP	用规则(b)归约	
(31)	NP VP PP	用规则(e)归约	
(32)	NP VP	用规则(d)归约	
(33)	S	用规则(a)归约	

这时，输入句子串已空，且栈中只剩下起始符S，该句子被接受，分析成功。

3.2 基于短语结构的句法分析

利用自底向上分析算法对句子“the girl writes the letter with a pencil”的分析句法树如下图所示



3.3 递归转移网络与扩充转移网络

扩充转移网络（Augmented Transition NetWorks，ATN）属于一种增强的上下文无关语法，其基本思想是采用上下文无关语法来描写句子的成分结构，但对语法中的个别产生式增添了某些功能，主要是描写某些必要的语法限制，并建立句子的深层结构。

ATN是在递归转移网络（Recursive Transition NetWorks，RTN）上附加若干控制条件所形成的网络，而递归转移网络又是扩展的有限状态转移图（Transition NetWorks，TN）。所以本节先介绍有限状态转移网络和递归转移网络，最后再介绍扩充转移网络。

3.3 递归转移网络与扩充转移网络

3.3.1有限状态转移网络

有限状态转移网络（TN）只能用来生成和识别正则语言。

一个有限状态转移网络由一组状态（即结点）和一组弧组成：

（1）其中的一个状态被指定为起始状态。

（2）在每条弧上都标注着该语法的终结符（词或词类）。表明在句子分析和识别时状态转移的条件和转移的方向。必须在输入句子中找到符合该弧上标注的词，才可以进行这条弧所规定的转移。

（3）状态集中有一个名为结束状态的子集。如果输入句子的头从起始状态开始，经过一系列的转移，句尾恰好到达结束状态，就说这个句子被这个转移网络所接受（或识别）。

3.3 递归转移网络与扩充转移网络

TN的工作过程为：输入某一个句子（句子定义为终结符连接成的串），从起始状态出发，按有限状态转移网络中箭头所指方向，依次扫描输入词，观察所输入词与相应状态弧上的标记是否匹配，匹配的话即通过该弧，进入下一个状态。如果扫描到句子的终点，有限状态转移网络也进入了结束状态，就说这个句子被这个转移网络所接受（或识别）。

例1. 用转移网络来识别句子The small black ducks swallow flies的过程如表1.1（这里忽略了词法分析），转移网络如图3.1所示。

3.3 递归转移网络与扩充转移网络

词　　典

ducks	noun,verb（躲避、低下头、弯下腰）
flies	noun,verb
small	adj.
black	adj.,noun
swallow	noun,verb
the	det.

表 3.1　句子识别过程

词	当前状态	弧	新状态
the	a	a $\xrightarrow{\text{det}}$ b	b
small	b	b $\xrightarrow{\text{adj.}}$ b	b
black	b	b $\xrightarrow{\text{adj.}}$ b	b
ducks	b	b $\xrightarrow{\text{noun}}$ c	c
swallow	c	c $\xrightarrow{\text{verb}}$ e	e
flies	e	e $\xrightarrow{\text{noun}}$ f	f(识别)

3.3 递归转移网络与扩充转移网络

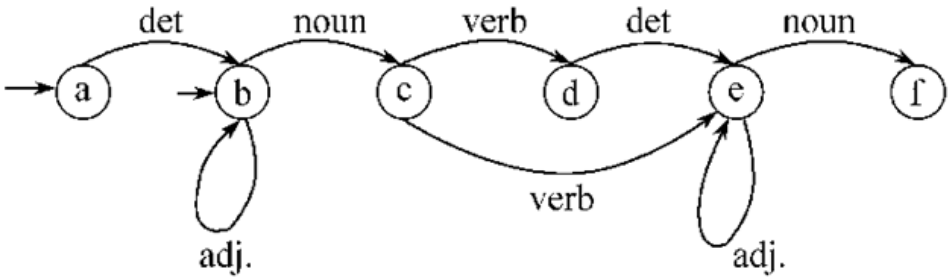


图 3.1　转移网络实例

识别过程到达 f 状态（终态），所以该句子被成功地识别了。分析结果如图 3.2 所示。

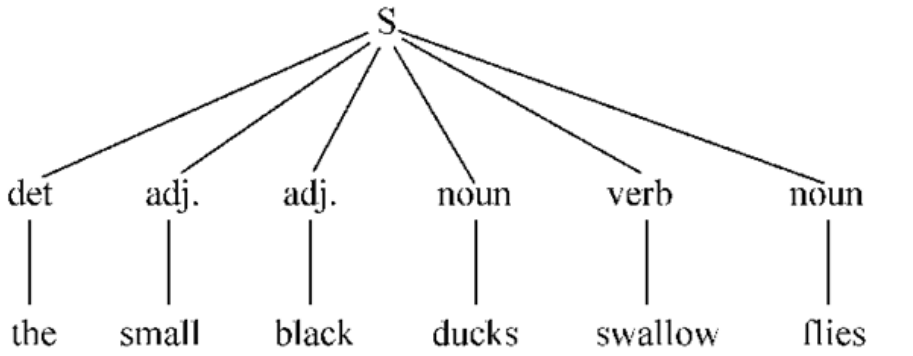


图 3.2　TN 分析树

3.3 递归转移网络与扩充转移网络

从上可以看出，这个句子还可以在网络中走其他弧，如词ducks也可以走弧c d，但接下来的swallow就找不到合适的弧了，对应于这个路径，该句子就被拒识了。由此看出，网络识别的过程中应找出各种可能的路径，因此算法要采用并行或回溯机制。

并行算法。并行算法的关键是在任何一个状态都要选择所有可以到达下一个状态的弧，同时进行试验。

回溯算法。在所有可以通过的弧中选一条往下走，并保留其他的可能性，以便必要时回过来选择之。这种方法需要一个堆栈结构。

TN只能识别正则语言，实际上任何一个有限状态转移网络都对应一部正则语法。所以，用有限状态转移网络表达自然语言是远远不够的。为提高TN的识别能力，提出了递归转移网络RTN。

3.3 递归转移网络与扩充转移网络

3.3.2递归转移网络

递归转移网络（Recursive Transition Networks，RTN）是对有限状态转移网络（TN）的一种扩展，在RTN中每条弧的标注不仅可以是一个终结符（词或词类）而且可以是一个用来指明另一个网络名字的非终结符。

例3.2下面是一部上下文无关语法：

$S \rightarrow NP\ \ V\ \ NP\ \ PP^*$

$NP \rightarrow T\ \ ADJ^*\ \ N\ \ PP^*$

$PP \rightarrow P\ \ NP$

其中X^*表示符号X可以出现零次或多次。这三条语法规则可以用图3.3所示的递归网络来表示。

3.3 递归转移网络与扩充转移网络

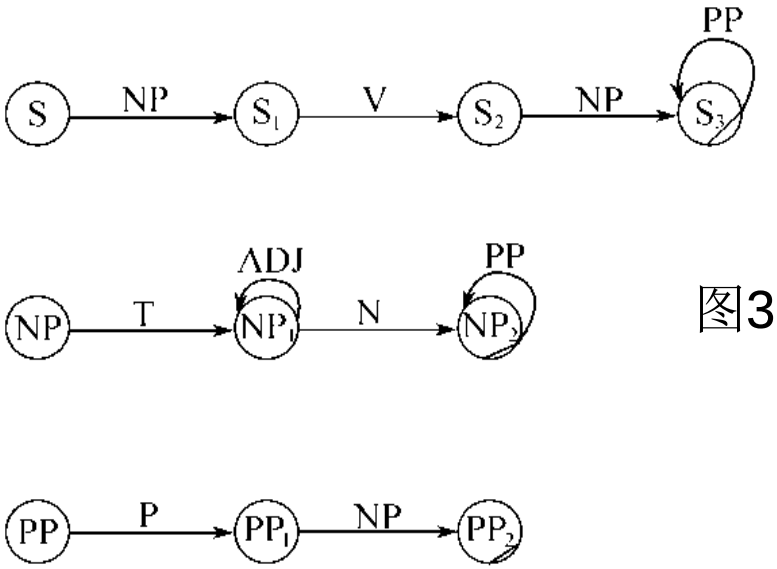


图3.3递归网络示例

在递归转移网络中，任何一个子网络都可以调用包括它自己在内的任何其他子网络。在图3.3中，表示名词短语NP的子网络中包含了介词短语PP，而在表示PP的子网络中又包括了NP。这种在NP的定义中包含了NP自身的定义叫做递归定义。相应的状态转移网络叫做递归转移网络。

3.3 递归转移网络与扩充转移网络

从生成能力上看，递归转移网络等价于上下文无关语法。但是要用它来分析自然语言，还必须在功能上予以增强，以便它可以描写各式各样的语法限制以及在识别过程中同时构造输入句子的句法结构。经过增强的递归转移网络就是下面要介绍的**扩充转移网络**。

3.3 递归转移网络与扩充转移网络

3.3.3扩充转移网络

扩充转移网络(Augmerted Transition Networks,简称ATN)是由一组网络构成的递归转移网络，每个网络都有一个网络名，它在以下三个方面对RTN进行了扩充：

（1）增加了一组寄存器，用以存储分析过程中得到的中间结果和有关信息。

（2）每条某些弧上除了用句法范畴（如词类和短语标记）来标注外，可以附加任意的测试，只有当弧上的这种测试成功之后才能通过这条弧。

（3）每条弧上还可以附加操作，当通过一条弧时，相应的动作便被依次执行，这些动作主要用来设置或修改寄存器的内容。

3.3 递归转移网络与扩充转移网络

ATN的每个寄存器由两部分构成：句法特征和句法功能寄存器。

（1）特征寄存器中，包含着许多维的特征，每一维特征都由一个特征名和一组特征值以及一个缺省值来表示。例如：

“数”：单数，复数。缺省:空）。

可以使用一维特征值来表示英语中动词的各种形式。

例如，对动词Work，可以使用下面的一维特征值来表示它的各种形式：

Work: present, past, present-participle, past-participle.

Default: present.

这里work就是特征名，present, past, present-participle等则是它的一组特征值。

3.3 递归转移网络与扩充转移网络

（2）**功能寄存器**则反映了句法成分之间的关系和功能。

分析树的每个结点都有一个寄存器，寄存器的上半部分是特征寄存器，下半部分是功能寄存器。

图3.4所示是一个简单的名词短语（NP）的扩充转移网络，网络中弧上的条件和操作如下：

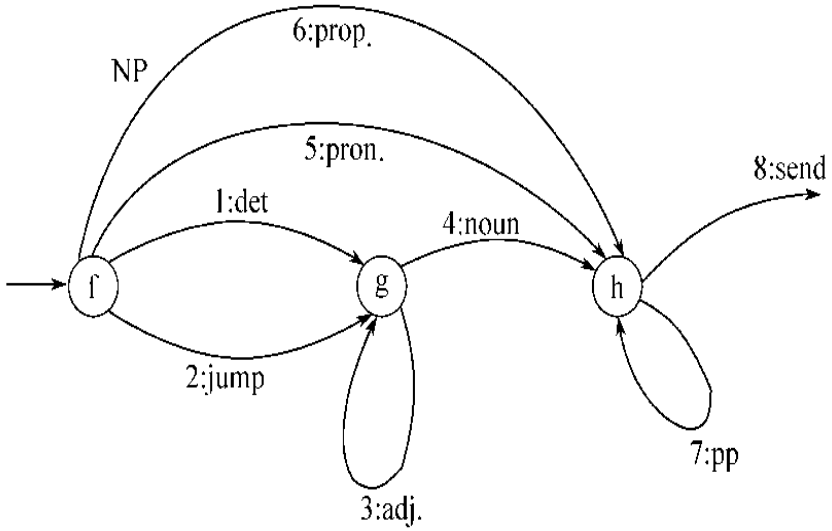


图3.4　名词短语（NP）的扩充转移网络

3.3 递归转移网络与扩充转移网络

NP-1： $f \xrightarrow{det} g$

A: Number ← *.Number

NP-4: $g \xrightarrow{Noun} h$

C: Number=*.Number or ϕ

A: Number ← *.Number

NP-5: $f \xrightarrow{pronoun} h$

A: Number ← *.Number

NP-6: $f \xrightarrow{proper} h$

A:Number ← *.Number

3.3 递归转移网络与扩充转移网络

上面的网络主要是用来检查NP中的数的一致值问题。

其中用到的特征是Number(数)，它有两个值：Singular(单数)和plural(复数)，缺省值是φ（空）。

C是弧上的条件，A是弧上的操作，*是当前词，proper是专用名词，Det是限定词，PP是介词短语，*.Number表示当前词的值。

NP是该扩充转移网络的网络名。网络NP可以是其他网络的子网络，也可包含其他网络，如其中的PP就是一个子网络，这就是网络的递归性。

3.3 递归转移网络与扩充转移网络

弧NP-1将当前词的Number放入当前NP的Number中，而弧NP-4则要求当前noun的Number与NP的Number是相同时，或者NP的Number为空时，将noun作为NP的Number，这就要求det的数和noun的数是一致的。因此，this book, the book, the books, these books 都可顺利通过这一网络，但是this books 或these book就无法通过。如果当前NP是一个代词（Pron.）或者专有名词（Proper），那么网络就从NP-5或NP-6通过，这时NP的数就是代词或专用名词的数。PP是一个修饰前面名词的介词短语，一旦到达PP弧就马上转入子网络PP。

3.3 递归转移网络与扩充转移网络

ATN方法是一个比较复杂的方法，尽管在自然语言理解的研究中得到了广泛的应用，但在实现过程中，还有许多问题，如非确定性分析、弧的顺序、非直接支配关系的处理等需要进一步的研究。

3.4 词汇功能语法

词汇功能语法是由J.Bresnan和R.M.Kaplan在1982年提出的，它是一种功能语法，但是更加强调词汇的作用。上面介绍的扩充转移网络（ATN语法）是有方向性的，也就是说，ATN语法的条件和操作要求语法的使用是有方向的，因为只有在寄存器被设置过之后才可被访问。而词汇功能语法（LFG）试图通过互不矛盾的多层描述来消除这种有序性限制，它利用一种结构来表达特征、功能、词汇和成份的顺序。

3.4 词汇功能语法

在LFG中，对句子的描述包括两部分：一个直接成分结构（C-structure）和一个功能结构（F-structure）。直接成分结构（C-structure）是由上下文无关语法产生的，用来描述表层句子的层次结构。功能结构（F-structure）则是通过附加到语法规则和词条定义上的功能方程来生成，其作用是表示句子的结构功能。

LFG采用了两种规则，一种是带有功能方程式的上下文无关语法规则，一种是词汇规则。表7.2给出了词汇功能语法（LFG）的语法规则，是带有功能方程式的上下文无关文法。

3.4 词汇功能语法

表 7.2 LFG 的语法规则

(1)	S→	NP	VP	
		(↑ Subject)=↓	↑=↓	
(2)	NP→	Determiner	Noun	
(3)	VP→	Verb	NP	NP
		↑=↓	(↑ Object)=↓	(↑ Object2)=↓

3.4 词汇功能语法

其中符号↑和↓称作元变量。↑表示当前成分的上一层次的直接成分，如规则中NP的↑就是S，VP的↑也是S；↓则表示当前成分。因此，规则（1）中的第一个方程式(↑Subject)=↓就可解释为把NP的属性传递给S的Subject特征。第二个方程式↑=↓表示将VP的所有属性传递给它的上一层成分S。

LFG的分析还依赖于句子中的词汇，词汇也带有功能方程式。例如，表7.3就是给出了一些词汇的LFG规则。

3.4 词汇功能语法

表 7.3 LFG 的词汇规则

handed	Verb	(↑ Tense)=Past
		(↑ Predicate) = ‘Hand<(↑ Subject),(↑ Object),(↑ Object2)>’
girl	Noun	(↑ Number)= Singular
		(↑ Predicate)=‘Girl’
baby	Noun	(↑ Number)= Singular
		(↑ Predicate)=‘Baby’
toys	Noun	(↑ Number)=Plural
		(↑ Predicate)= ‘Toy’
the	Determiner	(↑ Definiteness)=Definite
A	Determiner	(↑ Definiteness)=Indefinite
		(↑ Number)=Singular

3.4 词汇功能语法

其中，在动词的词条中，通过功能方程式定义了从语法功能到谓词—变元关系的映射。“<>”中表达的是句法模式，**Hand=<(↑Subject),(↑Object),(↑Object2)>**，表示谓语动词**hand**要有一个主语，一个直接宾语和一个间接宾语。

3.4 词汇功能语法

用LGF语法对句子进行分析的过程如下：

（1） 用上下文无关语法分析获得C-structure，不考虑语法中的功能方程式；该C-structure就是一棵直接成分树。

（2） 将各个非叶节点定义为变量，并用这些变量置换词汇规则和语法规则中功能方程式的元变量（↑或↓），建立功能描述，这一描述实际上就是一组功能方程式。

（3）对方程式作代数变换，求出各个变量，获得功能结构F-structure。

3.5 依存句法分析

- 依存句法理论

现代依存语法理论的创立者是法国语言学家Lucien Tesnière(1893-1954)。其思想主要反映在他1959年出版的《结构句法基础》。

3.5 依存句法分析

L. Tesnière 的理论认为：

一切结构句法现象可以概括为关联(connexion)、组合(jonction)和转位(tanslation)这三大核心。句法关联建立起词与词之间的从属关系，这种从属关系是由支配词和从属词联结而成；动词是句子的中心并支配别的成分，它本身不受其他任何成分支配。

3.5 依存句法分析

欧洲传统的语言学突出一个句子中主语的地位，句中其它成分称为“谓语”。依存语法打破了这种主谓关系，认为“谓语”中的动词是一个句子的中心，其他成分与动词直接或间接地产生联系。

Tesnière 还将化学中“价”的概念引入依存语法，一个动词所能支配的行动元（名词词组）的个数即为该动词的价数。

3.5 依存句法分析

依存语法：用词与词之间的依存关系来描述语言结构的框架被称为依存语法，又称从属关系语法。

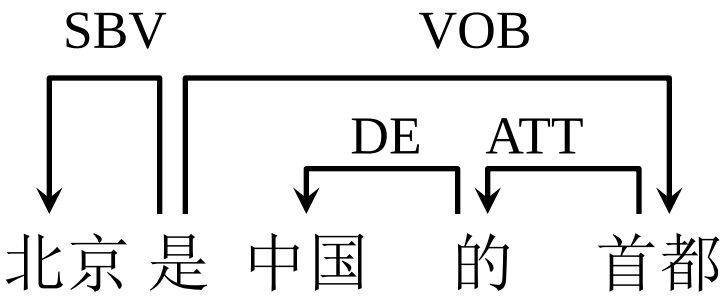
在依存语法理论中，“依存”就是指词与词之间支配与被支配的关系，这种关系不是对等的，而是有方向的。处于支配地位的成分称为支配者(governor, regent, head)，而处于被支配地位的成分称为从属者(modifier, subordinate, dependency)。

3.5 依存句法分析

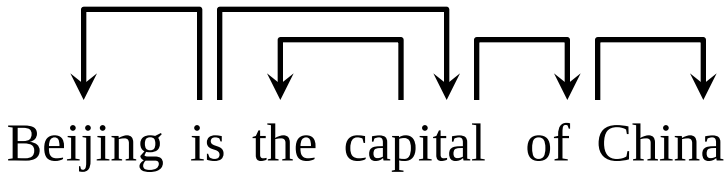
ATT 代表定中关系，即形容词与名词之间的修饰关系。

DE 代表“的”字结构。之前成分“的”作为支配词

SBV代表主谓关系，VOB代表动宾关系



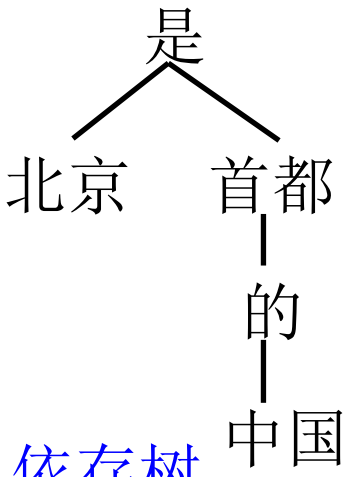
(e) 有向图-1



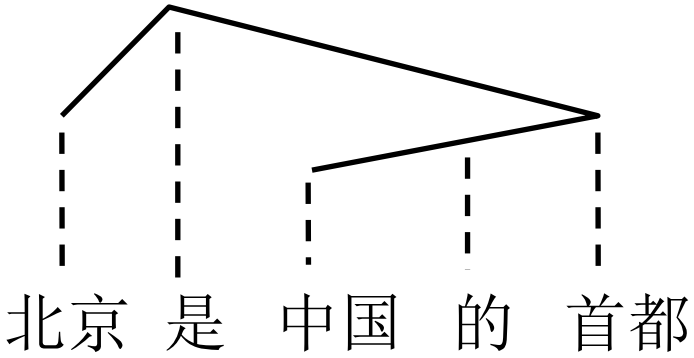
(e) 有向图-2

两个有向图用带有方向的弧(或称边，edge)来表示两个成分之间的依存关系，支配者在有向弧的发出端，被支配者在箭头端，我们通常说被支配者依存于支配者。

3.5 依存句法分析



(f) 依存树



(g) 依存投射树

图(f)是用树表示的依存结构，树中子节点依存于该节点的父节点。

图(g)是带有投射线的树结构,实线表示依存联结关系, 位置低的成份依存于位置高的成份，虚线为投射线。

3.5 依存句法分析

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：

(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。

3.5 依存句法分析

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)和可投射(projective)，并由此来保证句子的依存分析结果是一棵有“根(root)”的树结构。

3.5 依存句法分析

在处理中文信息的研究中，中国学者提出了依存关系的第五条公理，如下：

(5)中心成分左右两面的其它成分相互不发生关系。

3.5 依存句法分析

❑ 依存句法分析

建立一个依存句法分析器一般需要完成以下三部分工作：

(1) 依存句法结构描述

(2) 分析算法设计与实现

(3) 文法规则或参数学习

3.5 依存句法分析

目前依存句法结构描述一般采用有向图方法或依存树方法，所采用的句法分析算法可大致归为以下四类：

- 生成式的分析方法(Generative parsing)
- 判别式的分析方法(Discriminative parsing)
- 决策式的分析方法(Deterministic parsing)
- 基于约束满足的分析方法(Constraint satisfaction parsing)

3.5 依存句法分析

(1) 生成式的分析方法

生成式的句法分析方法采用联合概率模型生成一系列依存句法树并赋予其概率分值，然后采用相关算法找到概率打分最高的分析结果作为最后输出。这是一种完全句法分析方法，它搜索整个概率空间，得到整个句子的依存分析结果。

3.5 依存句法分析

- 二元文法的词汇关系模型
  (Bigram lexical affinities)

$$\Pr(words,tags,links)\approx\prod_{1\le i\le n}\Pr(tag(i)\,|\,tag(i+1),tag(i+2))\cdot\Pr(word(i)\,|\,tag(i))\cdot$$
$$\prod_{1\le i,j\le n}\Pr(L_{ij}\,|\,tword(i),tword(j))$$

其中，$tword(i)$ 表示符号i 的标记 ($tag(i)$) 和词本身 ($word(i)$)；L_{ij}是取值0或1的二值函数，L_{ij}=1表示i 和j 具有依存关系，L_{ij}=0表示i 和j 不具有依存关系；n是句子长度。

3.5 依存句法分析

一个标记序列(tags)由马尔柯夫(Markov)过程产生，某一个标记由该标记前面的两个标记决定，词由标记决定，观察每一对词(words)是否可以构成链接关系(link)的决策依赖于[tags, words]，即 link 对词汇是敏感的。最终生成words，tags，links 的联合概率模型。

3.5 依存句法分析

(2) 判别式的分析方法

判别式句法分析方法采用条件概率模型，避开了联合概率模型所要求的独立性假设。

- 最大跨度树模型

(Maximum Spanning Trees, Mst)

定义整棵句法树的打分是树中各条边打分的加权和：

$$s(x, y) = \sum_{(i,j)\in y} s(i, j) = \sum_{(i,j)\in y} \mathbf{w} \cdot \mathbf{f}(i, j)$$

3.5 依存句法分析

其中，s 表示打分值，y 是句子x 的一棵依存树，(x, y) 是y 中的结点对。f(•)是取值为1或0的高维二元特征函数向量，表示结点x_i 和 x_j 之间的依存关系，如果一棵依存树中两个词“打”和“球”存在依存关系，则：

$$f(i,j)=\begin{cases}1 & \text{如果 } x_i=\text{'打'} \; and \; x_j=\text{'球'} \\ 0 & \text{其他}\end{cases}$$

w 是特征 f(i,j) 的权值向量，w在确定了特征后由样本训练得到。

3.5 依存句法分析

该方法基本思想就是，在点和边组成的跨度树(spanning tree)中找到加权和分值最高的边的组合。跨度树中任意两个由词表示的节点之间都有边，根据特征和权值为每条边打分，求解最佳分析结果转化为搜索打分最高的最大跨度树问题。

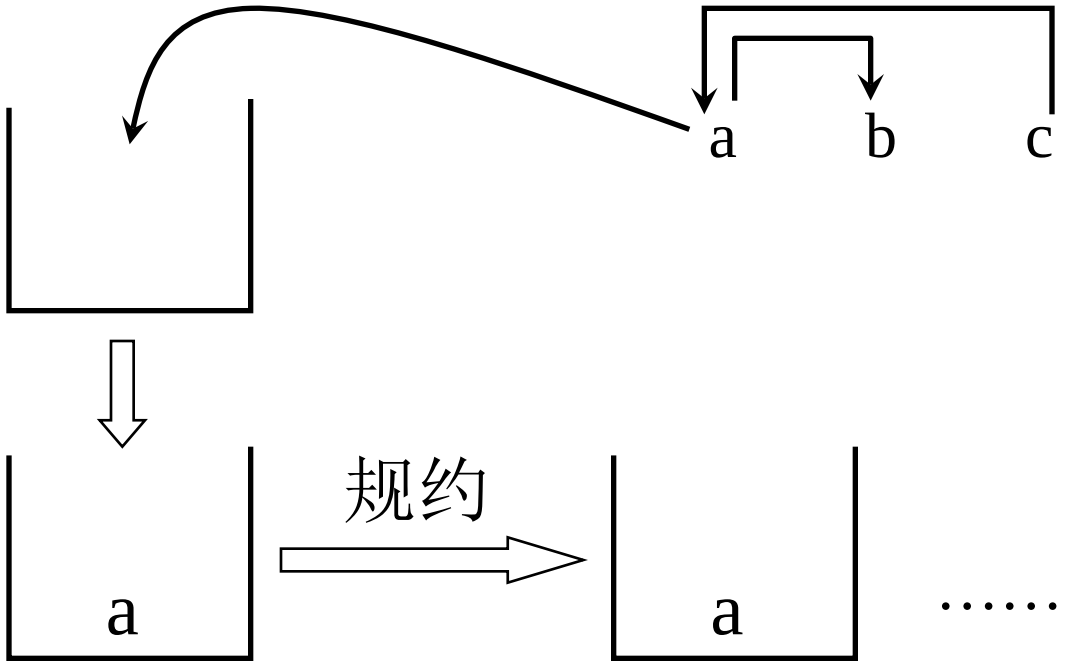
3.5 依存句法分析

(3) 决策式的分析方法

J. Nivre等(2003)提出的由左向右、自底向上的分析算法(移进－归约算法)：

分析结构是一个三元组<*S*, *I*, *A*>。*S*是栈(stack)，*I* 是待分析(剩余)的符号(token)序列，*A*是当前已有的依存关系集合。决策时有三种转换操作(transition)可供选择：Left-reduce，Right-reduce和Shift。分析器根据规则判断当前栈顶符号(token)与下一个输入符号(token)是否存在依存关系，如果存在，则将这一依存关系添加到集合*A*中，然后归约(Reduce)处于从属地位的符号，否则，移进Shift。

3.5 依存句法分析



将a、b依存关系添加到集合A中，……

3.5 依存句法分析

(4) 基于约束满足的分析方法

基于约束满足的依存句法分析方法采用约束依存语法(Constraint Dependency Grammar, CDG)，该方法将依存句法分析过程看作可以用约束满足问题(Constraint satisfaction problem, CSP)来描述的有限构造问题(finite configuration problem)。

3.5 依存句法分析

判别式方法将寻求最佳依存分析转化为最优路径搜索问题，使得诸多机器学习方法和运筹学的方法得以应用，在可计算性上具有优势，该方法的大部分精力放在如何降低算法复杂度上。

决策式方法的提出是为了提高依存句法分析的有效性即降低算法复杂度，分析的每一步都不需要保留多个可能的结果，而只给出一个确定的结果。这种算法属于贪婪(Greedy)算法，在准确率上没有优势，但算法复杂度一般是线性的。

3.6 格语法

格语法（Case Grammar）是美国语言学家菲尔墨（C.J.Fillmore）在60年代中期提出来的着重探讨句法结构与语义之间关系的一种语法理论和语义学理论。

3.6 格语法

❒ 格语法的来源

乔姆斯基在1957年出版的第一本书《句法结构》中提出了三大规则：短语结构规则、转换规则、语素音位规则。其短语结构规则（S → NP + VP；V + NP）的目标是生成所有的句子。结果，生成所有句子的目标虽然达到了，但是在生成正确句子（“约翰喝酒”）的同时，也生成出错误的句子（“洒喝约翰”）。这说明动词和名词之间要有一种语义限制。

3.6 格语法

乔姆斯基针对他第一本书存在的问题，于1965年出版了第二本书《语法理论的各方面》（The Aspects of the Theory of Yourself），主要是对第一本书的规则加以语义限制。但第二本书出版后不到一年又发现有新的问题。首先起来反对的是乔姆斯基的学生菲尔墨，他认为用各类格框架分析句法结构要比乔姆斯基的转换规则方便精密得多。

3.6 格语法

为了从语义的角度弥补转换生成语法的不足，菲尔墨1966年发表了《关于现代的格理论》（Toward a Modern Theory of Case），1968年发表了《格辨》（The Case for Case ），1971年发表了《格语法的某些问题》（Some Problem for Case Grammar)，1977年发表了《再论格辨》（The Case for Case Reopened)。其中的《格辨》是代表性论文。菲尔墨以上这些系列论文形成了一个语法学派，即所谓格语法，它实际上是转换生成语法发展出来的一个分支。

3.6 格语法

❒ 格的含义

在传统语法中，“格”是指某些屈折语法中用于表示词间语法关系的名词和代词的形态变化，这种格必定有显性的形态标记，即以表层的词形变化为依据。如德语的四格。在汉语中，名词和代词没有形态变化，所以没有格。

3.6 格语法

❒基本观点

C. J. Fillmore 指出：诸如主语、宾语等语法关系实际上都是表层结构上的概念，在语言的底层，所需要的不是这些表层的语法关系，而是用施事、受事、工具、受益等概念所表示的句法语义关系。这些句法语义关系，经各种变换之后才在表层结构中成为主语或宾语。

3.6 格语法

❒ 格的定义

格语法(Case Grammar)中的格是“深层格”，它是指句子中体词(名词、代词等)和谓词(动词、形容词等) 之间的及物性关系(transitivity)，如：动作和施事者的关系、动作和受事者的关系等，这些关系是语义关系，它是一切语言中普遍存在的现象。

3.6 格语法

这种格是在底层结构中依据名词与动词之间的句法语义关系确定的，这种关系一经确定就固定不变，不管经什么操作、在表层结构中处于什么位置、与动词形成什么语法关系，底层上的格与任何具体语言中的表层结构上的语法概念，如主语，宾语等，没有对应关系。

3.6 格语法

例如：(1) The door opened.

(2) The key opened the door.

(3) The boy opened the door.

(4) The door was opened by the boy.

(5) The boy opened the door with a key.

- the boy：施事格
- the door：客体格（受事格）
- the key：工具格

3.6 格语法

- 格语法的三条基本原则：

(1) S→M+P

句子 S 可以改写成情态(Modality)和命题(Proposition) 两大部分。

情态部分包括否定、时态、式（语气）、体（动作本身的状态，如是否完成或正在进行等）以及其他被理解为全句情态成分的状语。

命题牵涉到动词和名词短语、动词和内嵌小句之间的关系，动词是句子的中心，名词短语按其特定的格属关系依附于该动词。

3.6 格语法

(2) $P \rightarrow V+C_1+C_2+ \ldots C_n$

表示命题 P 都可以改写成一个动词 V 和若干个格 C 。动词是广义上的动词，包括：动词、形容词、甚至包括名词、副词和连词。

(3) $C \rightarrow K + NP$

K 为格标，是各种格范畴在底层结构中的标记，可以有各种标记形式，如：前置词、后缀词、词缀、零形式等。

3.6 格语法

❒ 格表

C. J. Fillmore 在1968年的论文中认为，命题中的格包括6种：

(1) 施事格(Agentive)：动作的发生者；

(2) 工具格(Instrumental)：对动作或状态而言作为某种因素而牵涉到的无生命的力量或客体。

(3) 承受格(Dative)：由动词确定的动作或状态所影响的有生物。如，He is tall.

3.6 格语法

(4) 使成格(Factitive)：由动词确定的动作或状态所形成的客体或有生物。或理解为：动词意义的一部分的客体或有生物。如：John dreamed a dream about Mary.

(5) 方位格(Locative)：由动词确定的动作或状态的处所或空间方位。如：He is in the house.

(6) 客体格(Objective)：由动词确定的动作或状态所影响的事物。如：He bought a book.

3.6 格语法

后来 Fillmore 在语言分析时又增加了一些格：

(7) 受益格(Benefactive)：由动词确定的动作为之服务的有生命的对象。

如：He sang a song for Mary.

(8) 源点格(Source)：由动词确定的动作所作用到的事物的来源或发生位置变化过程中的起始位置。

如：He bought a book from Mary.

3.6 格语法

(9) 终点格(Goal)：由动词确定的动作所作用到的事物的终点或发生位置变化过程中的终端位置。

如：I sold a car to Mary.

(10) 伴随格(Comitative)：由动词确定的与施事共同完成动作的伴随者。

如：He sang a song with Mary.

* 格的数目和名称并不是确定的。

3.6 格语法

❒ 用格语法分析语义：格框架约束分析

◆ 格框架表示

格框架中可有语法信息，也可有语义信息，语义信息是整个格框架的最基本的部分。

一个格框架可由一个主要概念和一组辅助概念组成，这些辅助概念以一种适当定义的方式与主要概念相联系。一般地，在实际应用中，主要概念可理解为动词，辅助概念理解为施事格、受事格、处所格、工具格等语义深层格。

3.6 格语法

例: In the room, he broke a window with a hammer.

[BREAK
[case-frame:
[agentive: HE
objective: WINDOW
instrumental: HAMMER
locative: ROOM]
[MODALs:
time: past
voice: active]]]

3.6 格语法

◆ 分析的基础

词典中记录动词的格框架和名词的语义信息。

对于动词：规定它们所属的必备格、可选格或禁用格，同时填充这些格的名词的语义条件。

如：《动词用法词典》把名词按其与动词格的关系分为14类：受事、结果、对象、工具、方式、处所、时间、目的、原因、致使、施事、同源、等同、杂类。

对于名词：填充语义信息，建立名词的语义分类体系。

3.6 格语法

◆ 分析步骤

(1) 判断待分析词序列中主要动词，在动词词典中找出该词的格框架；

(2) 识别必备格：如果格带有位置标志，则从指定位置查找格的填充物；如果格带有语法标志，则在这个分析的词序列中查找语法标志，进入相应的填充；如果格框架还需要其它必备格，查找其它名词的语义信息，按格框架的语义信息要求进行相应的填充。

3.6 格语法

(3) 识别可选格

(4) 判断句子的情态 Modal

格框架分析可以和句法分析结合起来：

(a) 句法分析：判断出句子的动词、名词短语、介词短语等；

(b) 查找动词的格框架与名词短语、介词短语的格关系，并进行相应的填充。

必须首先找到动词，从而获得格框架。

3.6 格语法

The young athlete will be running in Los Angeles next week.

从词典中查找 run 的格框架，如：

Verb: run

Case-Frame [

Neutral	-required （中性格）
Dative（与格）	-not allowed
Locative	-optional
Instrumental	-not allowed
Agentive	-required]

run 的中性格像一个物理实体或组织，如：

John ran the machine.

He ran the corporation.

3.6 格语法

CASE

[Agentive: the young athlete
Locative: Los Angeles
Neutral: the young athlete
[Modal
[Tense: Future
MOOD: Declarative
Time: next week]]]

3.6 格语法

❒ 格语法描写汉语的局限性

汉语的一些无动句、流水句、连动句、紧缩、动补、省略等结构，无法或不必用一个统率全句的模式来描述，其中连动句和兼语句尤为突出。

例如: (1) 他拿了书就上楼去了。

(2) 我们选他当班长。

3.7 概率上下文无关文法

- PCFG 规则

形式： $A \to \alpha, \quad P$

约束： $\sum_{\alpha} P(A \to \alpha) = 1$

例如：

$$\left.\begin{array}{l} \text{NP} \to \text{NN NN}, \ 0.60 \\ \text{NP} \to \text{NN CC NN}, \ 0.40 \end{array}\right\} \sum p = 1$$

$$\left.\begin{array}{l} \text{CD} \to \text{QP}, \ 0.99 \\ \text{CD} \to \text{LST}, \ 0.01 \end{array}\right\} \sum p = 1$$

3.7 概率上下文无关文法

◆例-1： S → NP VP, 1.00　　NP → NP PP, 0.40

NP → astronomers, 0.10

NP → ears, 0.18　　NP → saw, 0.04

NP → stars, 0.18　　NP → telescopes, 0.1

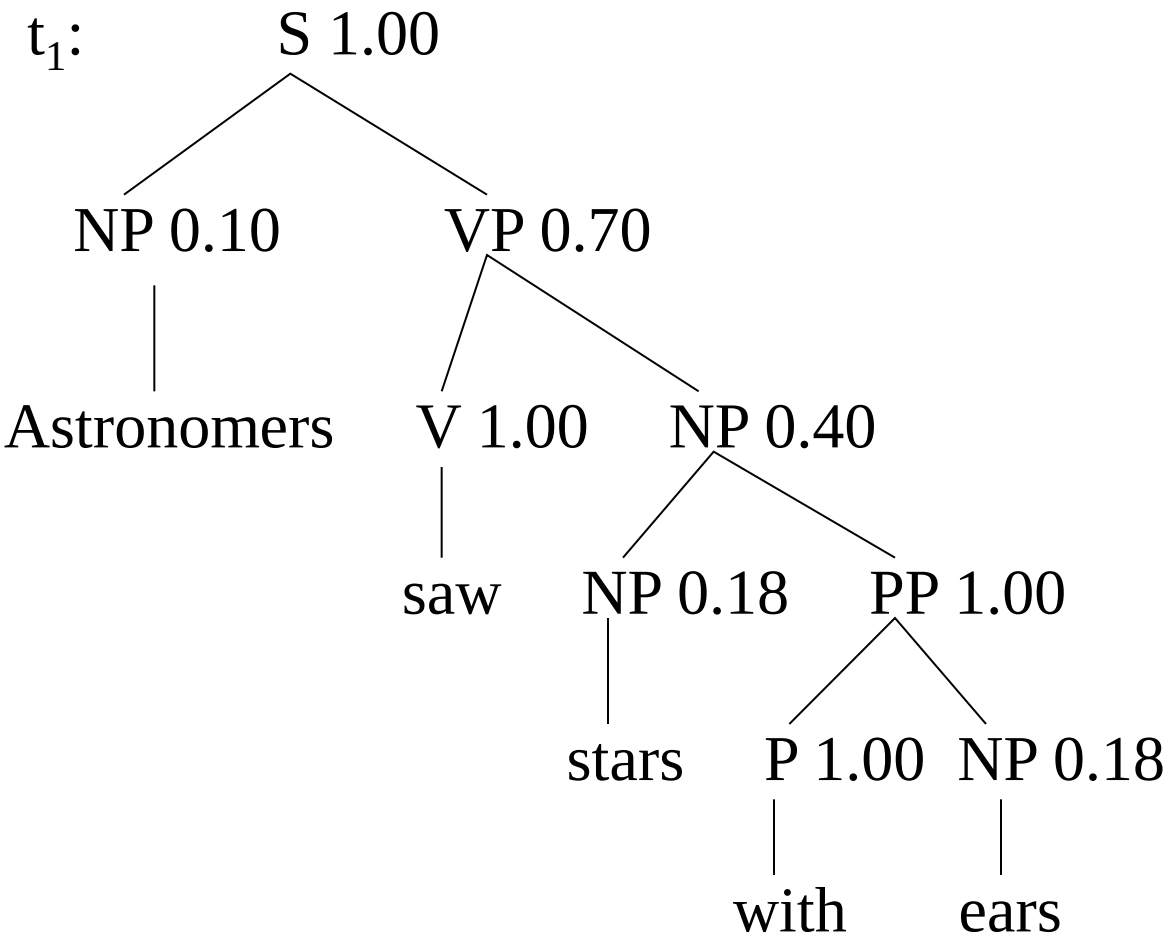
PP → P NP, 1.00　　P → with, 1.00

VP → V NP, 0.70　　VP → VP PP, 0.30

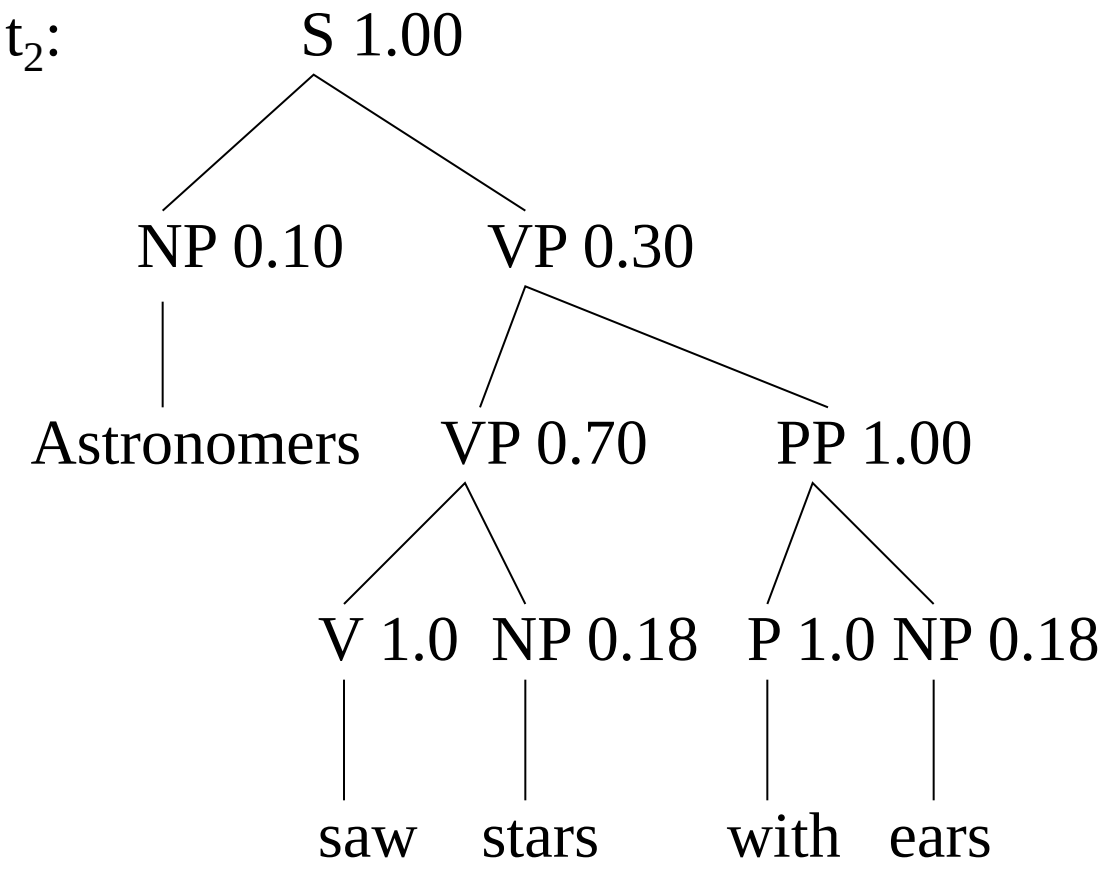
V → saw, 1.00

给定句子 S：*Astronomers saw stars with ears.*

3.7 概率上下文无关文法



3.7 概率上下文无关文法



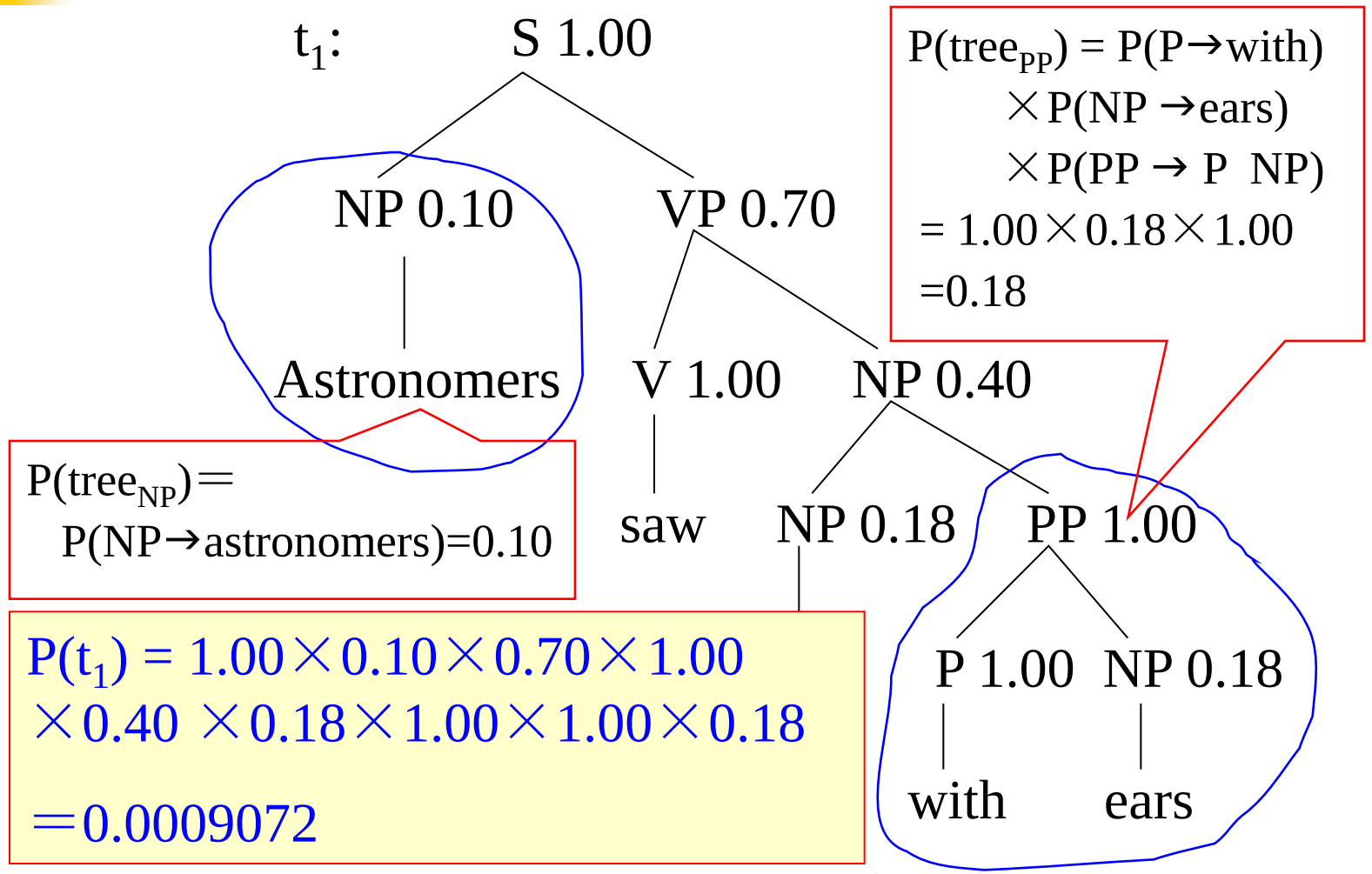
3.7 概率上下文无关文法

- 计算分析树概率的基本假设
  - 位置不变性：子树的概率与其管辖的词在整个句子中所处的位置无关，即对于任意的 k, $P(A_{k(k+C)} \rightarrow w)$ 一样。
  - 上下文无关性：子树的概率与子树管辖范围以外的词无关，即 $P(A_{kl} \rightarrow w|$任何超出$k\sim l$ 范围的上下文$) = P(A_{kl} \rightarrow w)$。

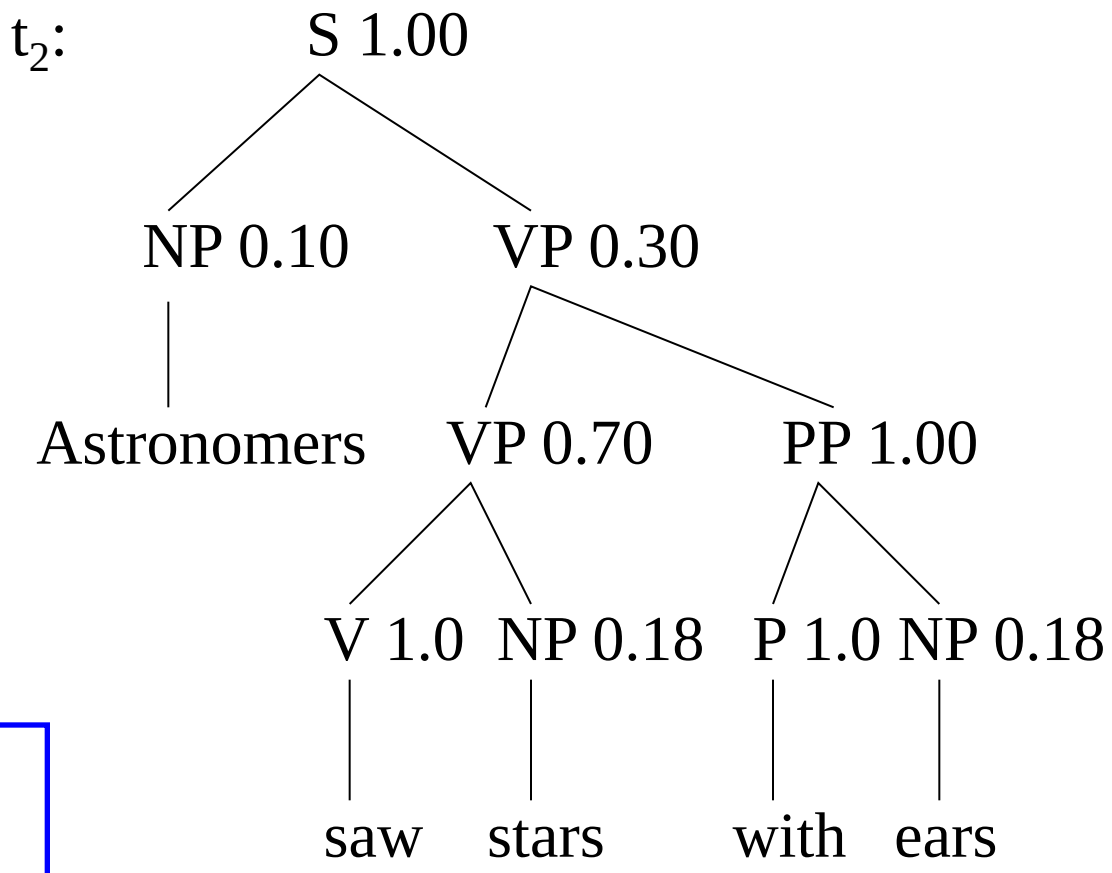
3.7 概率上下文无关文法

- 祖先无关性：子树的概率与推导出该子树的祖先结点无关，即 $\mathrm{P}(\mathrm{A}_{kl} \rightarrow w \mid$ 任何除 A 以外的祖先结点$) = \mathrm{P}(\mathrm{A}_{kl} \rightarrow w)$。

3.7 概率上下文无关文法



3.7 概率上下文无关文法

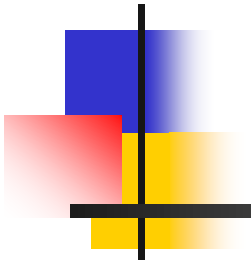


$P(t_2)=1.00\times0.10\times0.30\times0.70\times1.00\times0.18\times1.00\times1.00\times0.18$

$=0.0006804$

给定的句子 S：

$P(t_1) > P(t_2)$



三、词法分析方法

3.8 词法分析概述

词是自然语言中能够独立运用的最小单位，是自然语言处理的基本单位。

自动词法分析就是利用计算机对自然语言的形态 (morphology) 进行分析，判断词的结构和类别等。

词性或称词类（Part-of-Speech, POS）是词汇最重要的特性，是连接词汇到句法的桥梁。

3.8 词法分析概述

❑ 不同语言的词法分析

曲折语(如，英语、德语、俄语等)：用词的形态变化表示语法关系，一个形态成分可以表示若干种不同的语法意义，词根和词干与语词的附加成分结合紧密。

词法分析：词的形态分析(形态还原)。

分析语(孤立语)(如：汉语)：分词。

黏着语(如：日语等)：分词＋形态还原。

3.9 英语的形态分析

- 基本任务
  - 单词识别
  - 形态还原

3.9 英语的形态分析

❑ 英语单词的识别

例 (1) Mr. Green is a good English teacher.

(2) I'll see prof. Zhang home after the concert.

识别结果：

(1) Mr./ Green/ is/ a/ good/ English/ teacher/.

(2) I/ will/ see/ prof./ Zhang/ home/ after/ the/ concert/.

3.9 英语的形态分析

❑ 英语中常见的特殊形式的单词识别

(1) prof., Mr., Ms. Co., Oct. 等放入词典；

(2) Let’s / let’s => let + us

(3) I’m => I + am

(4) {it, that, this, there, what, where}’s =>

{it, that, this, there, what, where} + is

(5) can’t => can + not;

won’t => will + not

3.9 英语的形态分析

(6) {is, was, are, were, has, have, had}n't =>
{is, was, are, were, has, have, had} + not

(7) X've => X + have;

X'll=> X + will; X're => X + are

(8) he's => he + is / has => ?

she's => she + is / has => ?

(9) X'd Y => X + would (如果 Y 为单词原型)
=> X + had (如果 Y 为过去分词)

3.9 英语的形态分析

❑ 英语单词的形态还原

1. 有规律变化单词的形态还原

1) -ed 结尾的动词过去时，去掉ed；

*ed → * (e.g., worked → work)

*ed → *e (e.g., believed → believe)

*ied → *y (e.g., studied → study)

3.9 英语的形态分析

2) -ing 结尾的现在分词，

*ing → * (e.g., developing → develop)

*ing → *e (e.g., saving → save)

*ying →*ie (e.g., dying → die)

3) -s 结尾的动词单数第三人称；

*s → * (e.g., works → work)

*es → * (e.g., discusses → discuss)

*ies → *y (e.g., studies → study)

3.9 英语的形态分析

4) -ly 结尾的副词

*ly → *　　(e.g., hardly → hard)

…… ……

5) –er/est 结尾的形容词比较级、最高级

*er → *　　(e.g., colder → cold)

*ier → *y　　(e.g., easier → easy)

……

3.9 英语的形态分析

6) s/ses/xes/ches/shes/oes/ies/ves 结尾的名词复数，ies/ves 结尾的名词还原时做相应变化：

bodies → body, shelves → shelf,

boxes → box, etc.

7) 名词所有格 X’s, Xs’

3.9 英语的形态分析

2. 动词、名词、形容词、副词不规则变化单词的形态还原

一建立不规则变化词表

例：choose, chose, chosen

axis, axes

bad, worse, worst

3.9 英语的形态分析

3. 对于表示年代、时间、百分数、货币、序数词的数字形态还原

1) 1990s → 1990，标明时间名词；

2) 87th →去掉 th 后，记录该数字为序数词；

3) $20 → 去掉$，记录该数字为名词(20美圆)；

4) 98.5% → 98.5%作为一个数词。

3.9 英语的形态分析

4. 合成词的形态还原

1) 基数词和序数词合成的分数词， e.g., one-fourth 等。

2) 名词＋名词、形容词＋名词、动词＋名词等组成的合成名词， e.g., Human-computer, multi-engine, mixed-initiative, large-scale 等。

3.9 英语的形态分析

3) 形容词＋名词＋**ed**、形容词＋现在分词、副词＋现在分词、名词＋过去分词、名词＋形容词等组成的合成形容词，e.g., machine-readable, hand-coding, non-adjacent, context-free, rule-based, speaker-independent 等。

3.9 英语的形态分析

4) 名词＋动词、形容词＋动词、副词＋动词构成的合成动词，e.g., job-hunt 等。

5) 其他带连字符“-”的合成词，e.g., co-operate, 7-color, bi-directional, inter-lingua, Chinese-to-English, state-of-the-art, part-of-speech, OOV-words, spin-off, top-down, quick-and-dirty, text-to-speech, semi-automatically, *i*-th 等。

3.9 英语的形态分析

❑ 形态分析的一般方法

1) 查词典，如果词典中有该词，直接确定该词的原形；

2) 根据不同情况查找相应规则对单词进行还原处理，如果还原后在词典中找到该词，则得到该词的原形；如果找不到相应变换规则或者变换后词典中仍查不到该词，则作为未登录词处理。

3) 进入未登录词处理模块。

3.9 英语的形态分析

下面是英语词法分析的一个基本算法：

repeat
look for word in dictionary,
if not found ,
then modify the word.
until word is found or no further modification possible

其中word是一个变量，其初值就是当前词。

例 用上述算法分析catches ,ladies的过程如下：

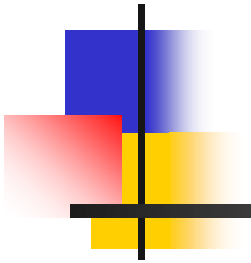
catches ladies ；词典中查不到。

catche ladie ；修改1，去掉"-s”。

#catch ladi ；修改2，去掉"-e”。

#lady ；修改3，变i为y。

上面修改2时就查到了catch，修改3时查到lady。当然更完整的词法分析还应当包括复合词的切分等，这里就不再进一步讨论了。



四、语义分析理论

3.10 语义理论简介

❑语义计算的任务：解释自然语言句子各个部分(词、词组及句子)的意义。

❑面临的困难：

(1) 自然语言句子中存在大量的歧义，涉及指代、同义/多义、量词的辖域、隐涵等；

(2) 同一句子不同人有不同的理解；

(3) 语义计算的理论、方法很不成熟。

3.10 语义理论简介

◆ 例子

(1) I bought a car with four wheels.

I bought a car with four dollars.

(2) These boys will be dedicated persons.

These boys will be denied license.

(3) 这件事情让我感到很头疼。

(4) 她说“你真恶心！”

3.10 语义理论简介

❒ 词的指称作为意义

该理论认为，词或词组的意义就是它们在现实世界上所指的事物。那么计算语义学的任务就是将词或词组与世界模型中的物体对应起来。

常用的现实世界模型假设世界上存在各种物体，包括人。

缺陷：对于复杂的问题这种定义无法处理。

启明星/暮星→金星；神仙？鬼？妖怪？

3.10 语义理论简介

❒ 心理图像、大脑图像或思想作为意义

该理论认为，词或词组的意义就是词或词组在人心理上或大脑中所产生的图像。

缺陷：在计算机中把心理图像有效地表示出来并不是一件容易的事情，而且，不一定所有的词义都有清晰的心理图像。

3.10 语义理论简介

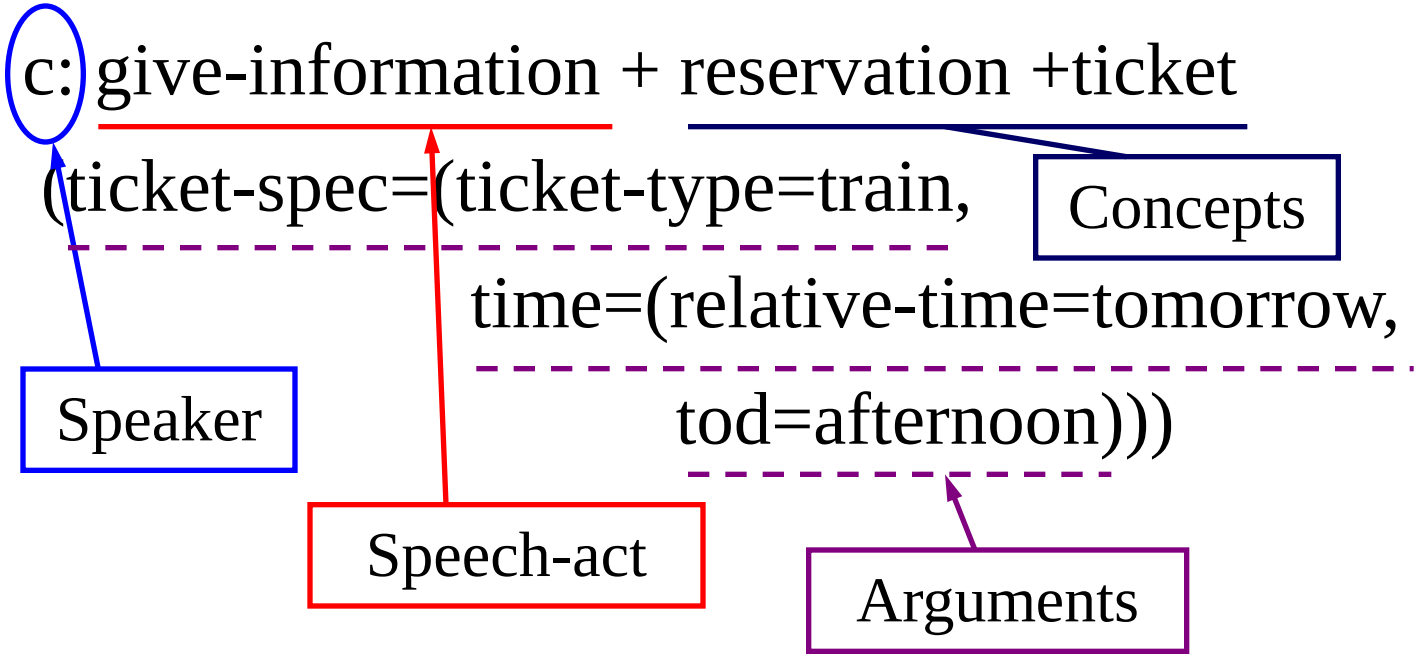
❒ 说话者的意图作为意义

该理论试图解释语言中一种被称为言语行为(Speech Acts) 的现象。

说话者把自己的话语当作行为希望听者理解、作出反应。这种意义被认为是独立于逻辑意义之外的。

3.10 语义理论简介

例如：我想预订明天下午的火车票。



缺陷：意图的定义和划分困难。

3.10 语义理论简介

❒ 过程语义

该理论认为，句子的语义定义为接受该句后所执行的程序或者所采取的某种动作。

优点：简单明了，对于计算机智能应用系统来说，这种定义在某种程度上是有效的。

缺陷：对于语言本身缺乏解释，且句子的语义常常和应用连接紧密，缺乏独立性。

3.10 语义理论简介

❒ 词汇分解学派

该理论把句子的语义基于它所含有的词和词组的意义之上，而词的意义则基于一组有限特征，这组特征通常称为语义基元。这样，只要给出一组语义基元和一些操作符，就可以把句子的语义描述出来。类似于化学中的元素学说。

缺陷：语义基元的定义、分解标准等不好把握，基元和组合操作的合理性直接影响句子语义描写的准确性。

3.10 语义理论简介

❒ 条件真理模型

该理论以谓词逻辑为基础，句子的语义定义为它所对应的命题或谓词在全体模型（或世界）中的真伪。

例如：“雪是白的”为真当且仅当在这个世界上雪是白的。

优点：对上下文无关部分的语义描写很有效。

缺陷：对时间、场景有关的语言现象不能很好地描述。不能很好地解释一句多义的问题。

3.10 语义理论简介

❒ 情景语义学

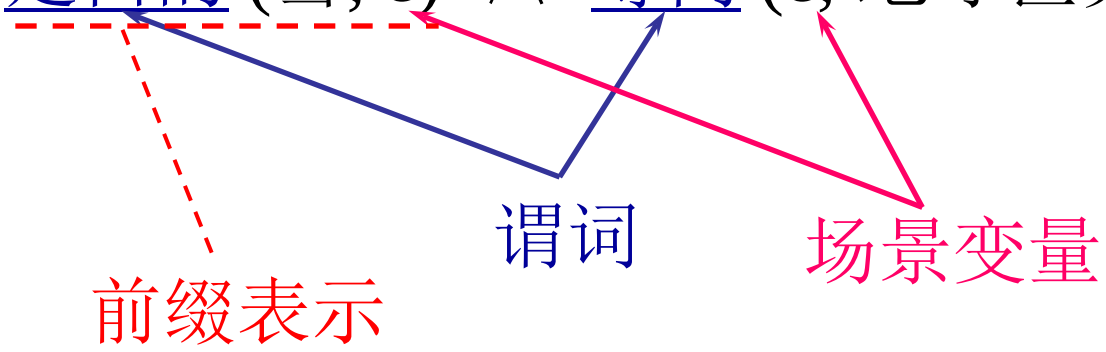
该理论认为句子的语义不仅和逻辑意义有关，而且和句子被使用的场景有关。

在语义表达式中引入一些与场景相关的变量，如事件变量、时间变量等，并用逻辑“与”算子对这些变量加以限制。

3.10 语义理论简介

例如：雪是白的：

是白的 (雪, e) ∧ 等同 (e, 地球世界)



3.10 语义理论简介

❒ 模态逻辑

起源于20世纪80年代初，AI。如：缺省逻辑、时态逻辑、真值维护系统等。

这类逻辑都是试图用一套公理系统来刻画现实世界和自然语言中常见的一些现象。这类现象从哲学上说就是一般性和特殊性的矛盾。

例如：鸟会飞　　企鹅不会飞

3.11 语义网络

❒ 背景

语义网络(semantic network)由美国心理学家M. R. Quilian 于1968年在研究人类联想记忆时提出。1977年美国 AI 学者 G. Hendrix 提出了分块语义网络的思想，把语义的逻辑表示与“格语法”结合起来，把复杂问题分解为几个较为简单的子问题，每个子问题用一个语义网络表示，把自然语言理解的研究向前推进了一步。

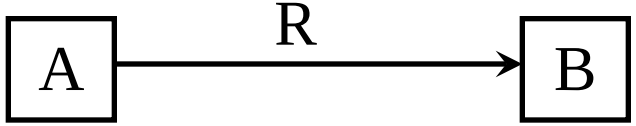
3.11 语义网络

❒ 语义网络的概念

语义网络通过由概念和语义关系组成的有向图来表达知识、描述语义。

- 有向图：图的结点表示概念，图的边表示概念之间的关系。
- 边的类型：(1)“是一种”：A到B的边表示“A是B的一种特例”；(2)“是部分”：A到B的边表示“A是B的一部分”；……

3.11 语义网络

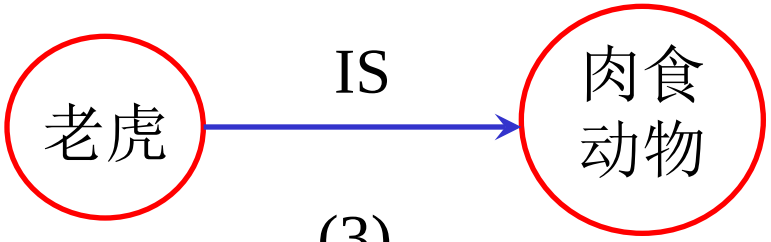


[在水中生活]

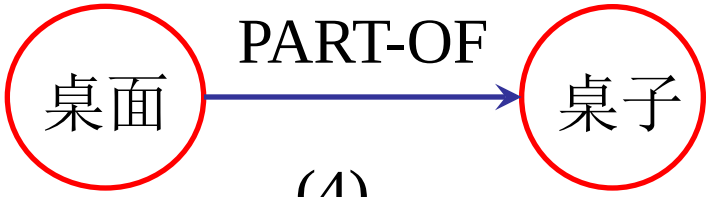
(1)

[有生命]
[吃食物]

(2)



(3)



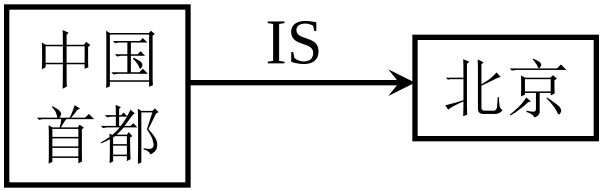
(4)

3.11 语义网络

❒ 语义网络的概念关系

语义网络各概念之间的关系，主要由 IS-A, PART-OF, IS, COMPOSED-OF, HAVE, BEFORE, LOCATED-ON 等谓词表示。

- IS-A：表示“具体—抽象”关系
- PART-OF：表示“整体—构件”关系
- IS：一个结点是另一个结点的属性



3.11 语义网络

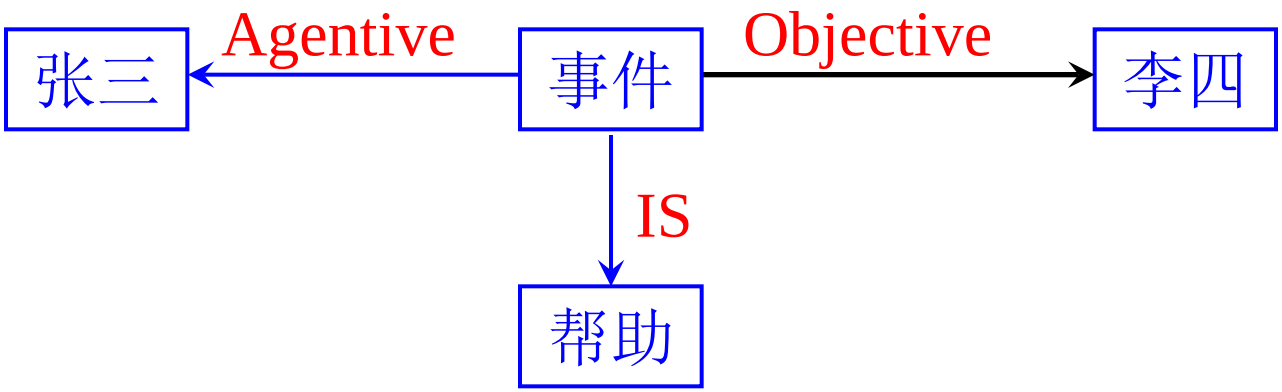
- HAVE：表示“占有、具有”关系
- BEFORE/AFTER/AT：表示事物间的次序关系
- LOCATED-ON/UNDER/AT：表示事物之间的位置关系

3.11 语义网络

❒ 事件的语义网络表示

当语义网络表示事件时，结点之间的关系可以是施事、受事、时间等。

例如：张三帮助李四。

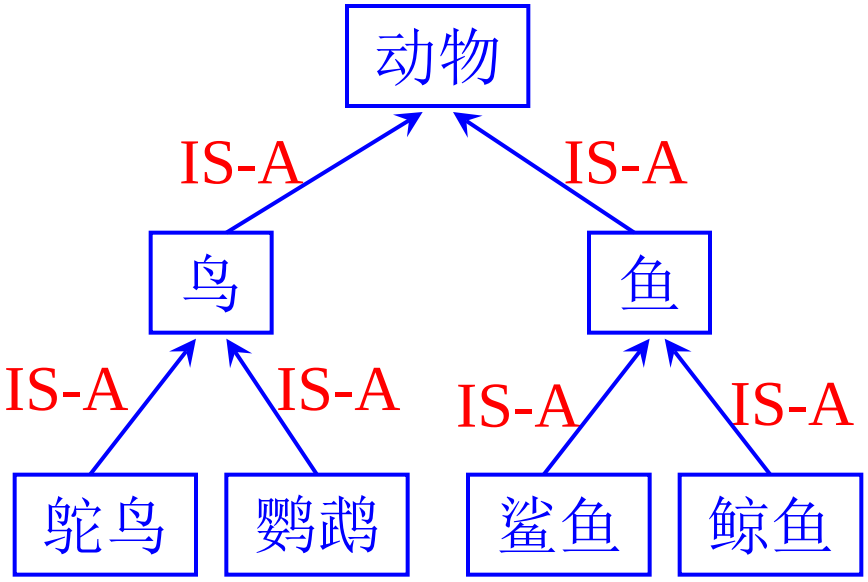


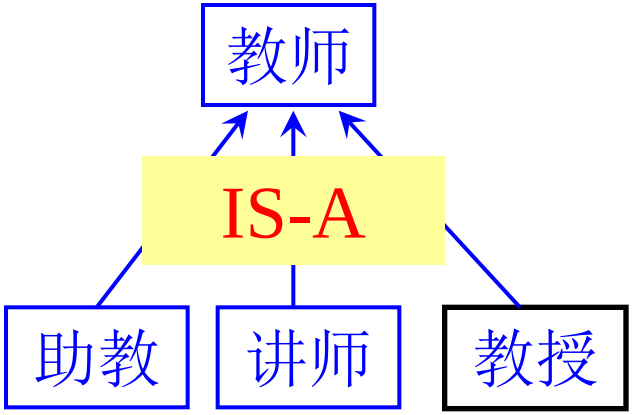
3.11 语义网络

- 事件的语义关系

(1)分类关系：事物之间的类属关系。

(2)聚焦关系：多个下位概念构成一个上位概念。

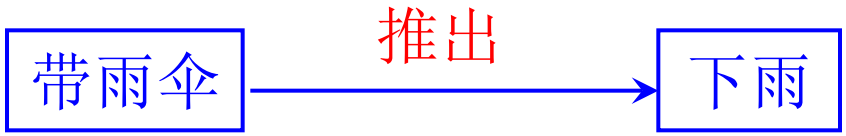




3.11 语义网络

(3) 推论关系：由一个概念推出另一个概念。

(4) 时间、位置关系：事实发生或存在的时间、位置。

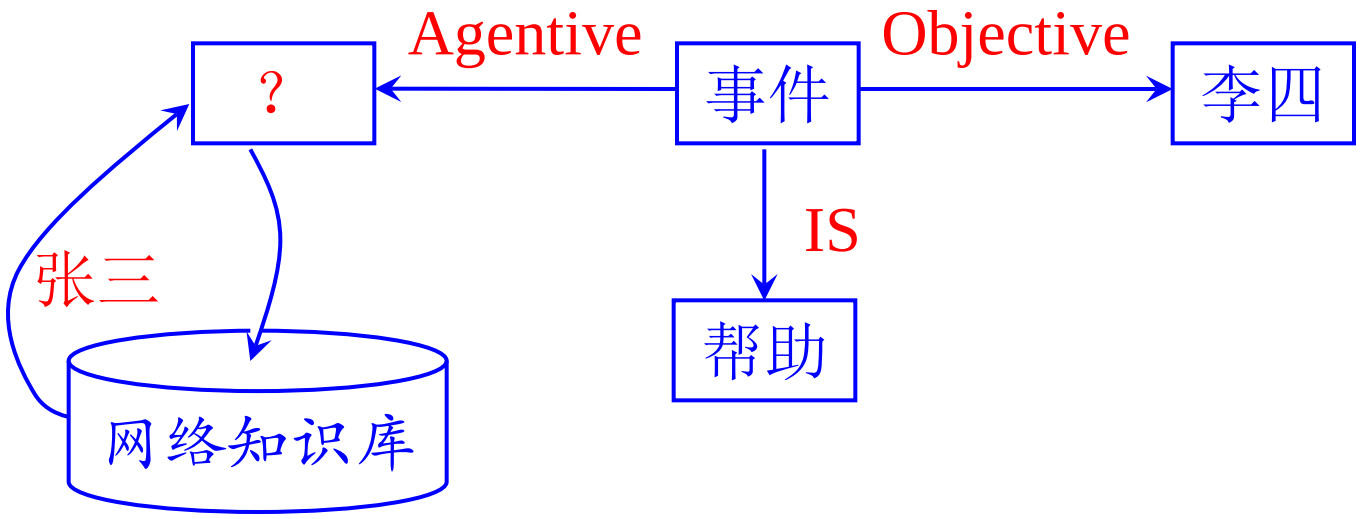


3.11 语义网络

- 基于语义网络的推理、分析

(1)根据提出的问题构成局部网络；

(2)用变量代表待求的客体。



3.11 语义网络

词义
- 内涵：词本身的意义，是对词代表的概念描述。
- 外延：词所指代的物体。

问题：如何在语义网络中表示和区分词的内涵和外延？

本章小结-句法分析部分

- 短语结构语法与乔姆斯基语法体系
  - 0型文法（短语结构）
  - 1型文法（上下文有关）
  - 2型文法（上下文无关）
  - 3型文法（正则文法）
- 基于短语结构语法的句法分析方法
  - 自顶向下的并行法
  - 自底向上的回溯法
- 递归转移网络与扩充转移网络
- 词汇功能语法
- 依存句法及分析方法
- 格语法

本章小结-词法分析部分

- 词法分析的任务（英语汉语有所不同）
- 英语形态分析
  - 单词识别
  - 形态还原
- 汉语自动分词
  - 汉语分词中的主要问题
  - 基本原则和辅助原则
  - 几种基本方法

    (MM、最少分词法、统计法等)

本章小结

- 未登录词识别
  - 人名、地名、机构名等
- 词性标注
  - 问题(兼类、标注集、规范)
  - 方法(规则方法、统计方法、综合方法)
- 分词与词性标注结果评测
  - 正确率、找回率、F-测度值

本章小结

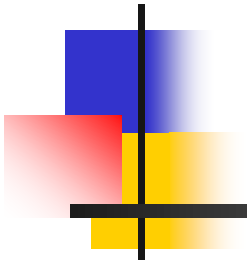
- 语义分析的基本任务及其面临的困难
- 语义计算研究概括及常见的语义理论

习题

1. 设计并实现算法用于还原英语动词。
2. 编写程序实现汉语逆向最大分词算法（可采用有限词表），并利用该程序对一段中文文本进行分词实验，校对切分结果，计算该程序分词的正确率、召回率及F-测度。
3. 设计并实现一个汉语未登录词（汉族人名）的识别算法(可限定条件)，并通过实验分析该算法的优缺点。

习题

4. 了解目前常见的几种汉语词性标注集，比较它们的差异，并阐述你个人的观点。
5. 掌握各种词性标注方法的要点，了解目前汉语词性标注的几种主要方法。
6. 试参考前人的工作，提出消除汉语自动分词中组合歧义的几点设想。
7. 阅读《信息处理用现代汉语分词规范》(中华人民共和国国家标准 GB13715)，了解规范的基本内容。



Thanks