自然语言、形式语法与自动机

形式语法与自动机理论是自然语言的计算机处理的基础理论。

一、形式语法与自动机理论

1.形式语法

从数学的观点看，自然语言的表达式是一个个有穷长的符号串（其中的符号可以是子、词、音节等）。但并非每个串都是自然语言允许的合法的表达式。语法（grammar）的作用就是以某种方式在所有有穷串的集合中“选”出一个子集，该子集中的串是“合法”的。

有两类数学“装置”——自动机（automaton）和串重写系统（string rewriting system），可以用于从数学和计算机的角度考察语法。

一个形式语法（formal grammar），简称“语法”，实质上可以看作是一个基于公理和推演规则的推演系统。

定义1 形式语法

一个形式语法是一个四元组<*VT*, *VN*, *S*, *R*>，其中：

*VT*称终结符字母表（the terminal alphabet），直观上是语言中全部表层符号的有穷集合。

*VN*称非终结符字母表（the non-terminal alphabet），直观上是表示语法范畴的符号的有穷集合。

*VT*与*VN*相交为空。

*S*称初始符号（initial symbol），*S*∈*VN*。

*R*称规则集（the set of rules），直观上是一系列形如“ϕ→ψ”的规则的集合，其中ϕ和ψ是符号串。设Σ＝*VT*∪*VN*，Σ\*＝{*s*| *s*是有穷Σ串}，Σ\**VN*Σ\*＝{*a*︵*b*︵*c*| *a*, *c*∈Σ\*，*b*∈*VN*}，有穷集*R*⊆Σ\**VN*Σ\*×Σ\*。

定义2 派生（derivation）

设*G*＝<*VT*, *VN*, *S*, *R*>是一个语法。一个*G*的派生是一个符号串的序列*x*0, ..., *xn*，其中*x*0＝*S*，*xi*是由*xi*－1应用某条*R*中的规则得出的（2≤*i*≤*n*）。

定义3 生成（generate）

语法*G*生成串*s*（*s*∈*VT*\*），当且仅当，存在一个*G*的派生*x*0, ..., *xn*，*xn*＝*s*。

语法*G*生成的语言*L*(*G*)＝{*s*∈*VT*\*| *s*由*G*生成}。

例4 形式语法举例

1. *G*1＝<*VT*, *VN*, *S*, *R*>，其中*VT*＝{a, b}，*VN*＝{S, A, B}，*S*＝S，*R*如下：

1 S→ABS，

2 S→*e*（空串），

3 AB→BA，

4 BA→AB，

5 A→a，

6 B→b。

则*G*1如何生成串abba？

S ⇒ ABS ⇒ ABABS ⇒ ABAB ⇒ ABBA ⇒ ABbA ⇒ aBbA ⇒ abbA ⇒ abba。

不难发现*L*(*G*1)＝{*s*∈{a, b}\*| *s*中a与b个数相等}。

（2）*G*1＝<*VT*, *VN*, *S*, *R*>，其中*VT*＝{*P*, *f*, *c*, *x*0, ..., *xn*, ∧, ¬, ∀, (, )}，*VN*＝{Form, Term, Con, Var}，*S*＝Form，*R*如下：

1 Form → ¬ Form

2 Form → ( Form ∧ Form )

3 Form → ∀ Var Form

4 Form → *P* Term

5 Term → Con

6 Term → Var

7 Term → *f* Term

8 Con → *c*

9 Var → *x*0

 ……

9＋*n* Var → *xn*

不难发现*L*(*G*1)＝Form1st-order-logic({*P*, *f*, *c*})。

（3）一个“零”至“九十九”的汉语数字词的形式语法。

ZERO→零

DIG→一

DIG→二

……

DIG→九

TEN→十

DEC→TEN DIG

S→ZERO

S→DIG

S→TEN

S→DEC

S→TEN DIG

S→DEC DIG

语法生成串的过程以及串的语法结构可以通过（有序）树形图清晰地表现出来。树可以表现出：句子成分的层及分组信息、语法类型信息和顺序信息。

例5 根据例4（3），字符串“五十三”的树：

S

DEC

DIG

DIG

TEN

五

十

三

例6

一个Parser程序。

几个关于树的语法分析中经常用到的概念。

统御（dominance），直接统御；

属于（belong to），统制/命令（command），c-统制/命令（constituent-command）；

先行（precedence）。

1. 乔姆斯基层级（the Chomsky Hierarchy）

对所有形式语法进行的分类：按照重写规则的限制由弱增强，或者说语法生成能力的由强减弱，分为Type 0至Type 3。

定义7 Type 0至Type 3语法

设α, β, ψ是任意串，A, B是非终结符，x是终结符串。

Type 0（recursive enumerable grammar, unrestricted rewriting system）：由定义1所定义的语法。

Type 1（context sensitive）：每条规则都形如αAβ→αψβ，其中ψ≠*e*。（或：ψ→ω，ω不短于ψ。）

Type 2（context free）：每条规则都形如A→ψ。

Type 3（regular, right linear, finite state grammar）：每条规则都形如A→xB或A→x。

命题 8 几种语法间的包含关系

（1）Type 3⊆Type 2。

（2）不含空串的Type 2⊆Type 1（因为A→*e*不被Type 1允许）。

（3）Type 1⊆Type 0。

例9

例4中，*G*1是Type 0，Type 2；不是Type 1；不是Type 3。

*G*1表面上不满足Type 2的要求，但其规则可以重写为：

S→*e*，S→aB，S→bA，B→b，B→bS，A→a，A→aS，A→bAA，B→aBB，

因此是Type 2。

*G*2是Type 0，Type 1，Type 2；不是Type 3。

IL（Intensional Logic, by Montague and Gallion）是Type 3。

1. 自动机

自动机是一种理想化的数学计算装置。在自然语言处理中，可以想象自动机可以用来检验符号串的合法性——将一个符号串输入给自动机，由自动机依据形式语法判定其是否合法。如果合法，自动机给出结果“接受”，反之给出结果“拒绝”。

据判定规则和工作原理的不同，自动机根可分为多种。不同种类的自动机与不同类型的形式语法存在对应关系。

3.1有限状态自动机

有限状态自动机可用于判定正则语言（Type 3）。

定义10 有限状态自动机

一个有限状态自动机是一个五元组<*K*, Σ, δ, *q*0, *F*>。其中，

*K*是有穷集，称状态集；

Σ是有穷集，称字母表；

*q*0∈*K*，称初始状态；

*F*⊆*K*，称终止状态集；

Δ⊆*K*×Σ×*K*，称指令集（有穷）。

定义11 有限状态自动机的指令

设*M*＝<*K*, Σ, Δ, *q*0, *F*>是一个有限状态自动机，它的一个指令是一个三元组<*qi*, *x*, *qj*>∈Δ。其中，*qi*, *qj*∈*K*，*x*∈Σ。

有限状态自动机由控制器、读取头和输入带构成。输入带两端无限长，并被分成许多格子。输入符号串从左向右记在格子中，每个格子一个符号。没有符号的方格称空白方格，用符号“#”表示。工作时，自动机处于初始状态*q*0，然后读取头读入第一个符号*x*，然后根据程序指令<*q*0, *x*, *qj*>（如果存在的话），进入一个新状态*qj*（允许*q*0＝*qj*），然后读取头右移一格。重复进行以上操作，直至自动机不再有可执行的程序指令。此时，若读取头未读完字符串，则称自动机锁死，认为自动机拒绝符号串；若读完字符串，且自动机所在的状态恰好是预先定义好的终止状态之一，则自动机接受符号串；否则就拒绝符号串。

例12 有限状态自动机

一个用于识别“零”至“九十九”的汉语数字词的有限自动机图式

*q*0

*q*1

二, ...,九

*q*2

零, 一

十

*q*3

十

*q*4

一, ...,九

正则语言就是有限状态自动机语言。正则语言由正则语法生成，用有限状态自动机识别。

正则语法的缺陷：只能向右扩展。英语（自然语言）不是正则语言。

例13 例11自动机对应的形式语法（部分），有悖于直观

……

DIG→三

TEEN→十 DIG

S→五 TEEN

……

例14中，“五十三”其实是“五”+“十三”而非“五十”+“三”。

定理13 抽吸定（引）理

如果*L*是一无穷有限状态自动机语言，其字母表是Σ，那么存在串*x*, *y*, *z*∈Σ\*，使得*y*≠*e*且*xynz*∈*L*（对任意自然数*n*）。

例15

（1）IL：Ytt……YttZt。

（2）此定理用于否证一个语言是有限状态自动机语言。如{*anbn*| *n*≥0}。

3.2下推状态自动机（pushdown automata）

下推自动机可用于判定上下文无关语言。

下推自动机有一个遵循“先进后出”原则的符号暂存器，称为栈。

定义15 下推自动机

一个下推自动机是一个六元组<*K*, Σ, , Δ, *q*0, *F*>，其中

*K*是有穷集，称状态集；

Σ是有穷集，称（对象语言）字母表；

是有穷集，称栈字母表；

*q*0∈*K*，称初始状态；

*F*⊆*K*，称终止状态集；

Δ⊆*K*×Σ×\*×*K*×\*。

一个下推自动机指令形如<*qi*, *a*, *A*>→<*qj*, γ>，其中*qi*, *qj*是状态，*a*是对象语言字母表中的符号，*A*是栈字母表（可以不同于对象语言的字母表）中的符号，γ是基于栈字母表的符号串。当自动机在*qi*状态时，读取输入带子上的符号*a*，读取栈顶端的符号*A*，将*A*替换为γ并进入状态*qj*。若γ＝*e*，相当于将*A*抹去；若*A*＝*e*，相当于将γ加入栈。

下推自动机接受一个符号串，如果以下三个条件被同时满足：

1. 输入符号串被读取完；
2. 自动机处于最终状态；
3. 栈是空的。

例16 一个接受{*anbn*| *n*≥0}的下推自动机

*K*＝{*q*0, *q*1}，Σ＝{*a*, *b*}，＝{*A*}，初始状态*q*0，终止状态集{*q*0, *q*1}，操作：

<*q*0, *a*, *e*>→<*q*0, *A*>，

<*q*0, *b*, *A*>→<*q*1, *e*>，

<*q*1, *b*, *A*>→<*q*1, *e*>。

对应的语法（Type 2）：

S→*a*S*b*

S→*e*

上下文无关语言由上下文无关语法生成，可使用下推自动机进行识别。

一般认为，自然语言是一种上下文无关语言，因而可以用下推自动机来识别。但某些语言中存在着“交叉对应”现象，是无法被上下文无关语法描述的。因此，一些语言学家认为自然语言可能是一种介于上下文无关语言和上下文有关语言之间的语言，且多数情况下更接近于前者的语言。

3.3图灵机（Turing machine）

图灵机用于识别递归可枚举语言。

定义17 图灵机

一个图灵机是一个四元组<*K*, Σ, *s*, δ>，其中：

*K*是有穷集，称状态集；

Σ是有穷集，称字母表，#∈Σ；

是有穷集，称栈字母表；

*s*∈*K*，称初始状态；

*F*⊆*K*，称终止状态集；

部分函数δ：*K*×Σ→*K*×(Σ∪{**L**, **R**})，**L**, **R**不属于Σ。

图灵机由控制器、方格输入带和读写头构成。图灵机从初始状态*q*0开始，读写头处于最左边的非空白符号上。图灵机根据指令δ＝<<*qi*, *aj*>, <*qk*, *X*>>（<*qi*, *aj*>→<*qk*, *X*>）行事，即：如果图灵机处于*qi*状态且读入符号*aj*，就进入状态*qk*（允许*aj*＝*qk*），此时，若*X*是字母表中的一个符号*al*，则用*al*替换*aj*；若*X*是**L**或**R**，则令读写头向左或向右移动一格。被读取的符号可为#，因而图灵机的读写头原则上可到达带子的任一位置。若图灵机不再有可执行的指令，则图灵机停机，图灵机接受符号串。如果图灵机永不停机，则认为符号串被拒绝。

例18

（1）一个做*a*, *b*互换的图灵机，遇到右侧第一个#时停机。

*K*＝{*q*0, *q*1}，Σ＝{*a*, *b*, #}，初始状态*q*0，操作：

<*q*0, *a*>→<*q*1, *b*>

<*q*0, *b*>→<*q*1, *a*>

<*q*0, #>→<*q*1, #>

<*q*1, *a*>→<*q*0, **R**>

<*q*1, *b*>→<*q*0, **R**>

（2）一个图灵机，略过所有*b*，遇第一个*a*进入状态*q*1；若在*q*1状态遇到第一个#则停机，若在*q*0状态遇到第一个#则永不停机。

*K*＝{*q*0, *q*1}，Σ＝{*a*, *b*, #}，初始状态*q*0，操作：

<*q*0, *a*>→<*q*1, **R**>

<*q*0, *b*>→<*q*0, **R**>

<*q*0, #>→<*q*0, **R**>

<*q*1, *a*>→<*q*1, **R**>

<*q*1, *b*>→<*q*1, **R**>

例19 一个Type 0语言，字符串中包含数量相等的a, b, c

S→SABC

S→*e*

AB→BA

BA→AB

AC→CA

CA→AC

BC→CB

CB→BC

A→a

B→b

C→c

3.4线性有界自动机（linear bounded automata）

线性有界自动机用于识别上下文有关语言。线性有界自动机的输入带子有左右界限，读写头的运动不得超出这一界限。

自然语言存在的“交叉对应”现象表明自然语言可能是一种上下文有关语言，或至少介于上下文无关和上下文有关语言之间。与此相关的有Indexed Grammar理论（Hopcroft & Ullman）。

3.5自动机类型与形式语法和语言类型的对应关系

Chomsky与Kuroda：四种语法及其语言分别能与四种自动机一一对应：某语言能用某种类型的自动机识别，当且仅当，它能用与之对应的那一型语法生成。

这一结果对计算机的编译技术很有帮助，极大地推动了计算机的自然语言处理工作。

1. 应用举例

基于生成规则的形式语法理论在计算语言学中有较为广泛的应用。如现代汉语歧义格式分析。详见附录。

附录 现代汉语歧义格式分析报告（以AP VP NP为例）

**一、歧义与计算机**

歧义是指一种语言中的同一个符号串可能有多种不同的组合结构，进而可能表达不同的意义。歧义在自然语言中大量地存在。但是自然语言有丰富的排歧手段，如重音、语调、语义知识、语境等方面的途径，使许多语法形式上有歧义的语言结构在实际使用时只有一种可接受的理解方式，从而使歧义并不严重地影响自然语言的交际效率，某些歧义甚至有利于人类使用自然语言进行交际。

但是，计算机处理语言时，歧义会给计算机对语言结构的分析带来困难。

语法层面上，计算机分析自然语言的语法结构时，需要根据给定的语法规则构建短语或句子的语法生成树。有歧义的短语或句子可以合语法地构建结构不同的树。如果这些不同的树在人类看来并非都可取，那么计算机就需要通过某种手段排除那些合语法但不可取的树，即进行计算机的排歧。

语义层面上，按照弗雷格的组合性原则，一种“理想的”语言表达式的意义应由其组成部分的意义唯一地确定，或者说句子的意义是由句中词的意义经函数作用的结果。人工设计的形式语言，如一阶语言，满足这一特性，故好的形式语言的表达式具有“唯一可读性”，没有歧义的问题。但自然语言难以满足这一要求，大量地存在歧义，而歧义的存在破坏了组合性原则，使得计算机分析自然语言句子时带来不确定的语义结果。对人类来说，许多这样的不确定的结果可以通过语义知识或语境加以排除，但计算机要做到这一点就十分困难。

因此，无论是在语法还是语义层面，都给计算机处理语言带来了困难。

为了解决歧义问题，需要对各种歧义现象进行分类。从人的角度来看，自然语言歧义大致可分为结构层次歧义、结构关系歧义、语义关系歧义和语用歧义。而计算机首先面临的问题是结构层次歧义和结构关系歧义的处理，语义和语用方面的歧义处理更深入也更困难。

结构层面的歧义可根据顶层分析结果的不同分为外显型歧义和内含型歧义。两种分析结果的顶层节点性质相同为内含型歧义，不同则为外显型歧义。

另一方面，可以根据歧义结构间有无实质性或关键性的差异将歧义分为真歧义、准歧义和伪歧义。

对其以进行分类有助于为消歧制定不同策略。例如对于伪歧义，由于各歧义结构间没有实质性的差异，故只需在歧义结构或规则调用顺序间制订一个优先度即可；而真歧义和准歧义就需要用到实质性的消歧方法。再如，对于外显型歧义和内含型准歧义，就有可能在短语结构规则层面制订消歧策略；然而对于内含型真歧义来讲，就做不到这一点。总的来说，消歧主要依据语言结构中各级语言单位的功能特征，如在词典中记录每个词语占据句法结构位置的能力，通过规则的合一约束描述短语的功能特征，以及规定功能特征的取值如何发生动态变化等。

1. **歧义格式统计程序**

歧义格式统计程序是一种在结构层面量化考察语言中歧义格式的工具。设一个语法系统中共有n个非终结符，而一个具体格式由m个非终结符组合而成，那么对于nm种排列方式（全排列），歧义格式统计程序可给出其中哪些排列方式合语法，哪些不合语法；对于合语法的排列方式，程序可给出哪些有潜在的歧义格式，哪些只有唯一的格式；对于潜在有歧义的格式，程序可分辨其是外显型歧义还是内含型歧义，并统计有多少种歧义结构。

以本文的分析结果为例，本文考察了ap、vp、np三个非终结符组合成长度为3的串的情形，因而总共有33＝27种排列方式。27种排列全部合语法，并且全部有歧义。其中，只有“vp ap np”一种排列方式存在内含型歧义，歧义个数为5；其余排列呈现为外显型歧义，平均歧义个数为13.9。各种歧义结构中，以“vp vp vp”的歧义结构最多，达43种。本文考察的S序列为“ap vp np”，歧义格式共7种。

通过歧义格式统计研究，可以为评估一个格式的歧义程度提供一个可量化的标准，从而有助于评价和探讨非终结符设置的合理性，进而实现语法的优化和改良。同时，这种方法也可以反过来促进自然语言的研究，通过格式组合的演算来预估自然语言中歧义的种类、数量与表现。不过，歧义格式统计程序给出的歧义格式包含了一切数学上可能的组合，很多格式组合在自然语言中可能极少被用到，甚至根本没有实例。因此，运用歧义格式统计程序对自然语言歧义种类进行量化研究必须考虑到这种情形。某一格式的歧义种类多，并不代表该格式在自然语言中一定歧义多，也许许多的歧义形式在自然语言是不可能的；这些歧义格式至多只对计算机处理语言带来可能的障碍，而并不一定是自然语言的研究对象。

**三、“ap vp np”序列的歧义格式问题探讨**

在序列“ap vp np”的分析过程中，主要面临的问题是两种歧义结构

（1）(vp:状中(ap,vp:述宾(vp,np)))

（2）(vp:述宾(vp:状中(ap,vp),np))

是真歧义还是伪歧义的问题。考察例子：

1. 轻易地放弃坏习惯

轻易

放弃

坏习惯

ap

vp

np

vp

vp

轻易

放弃

坏习惯

ap

vp

np

np

vp

1. 若按（1）理解，“放弃坏习惯”先形成述宾结构，再由“轻易”作为状语加以修饰，因而整个短语是状中短语；若按（2）理解，“轻易”先修饰“放弃”，然后“坏习惯”再作为“轻易地放弃”的宾语，从而使整个短语形成述宾短语。从语义上看，这两种理解没有太大的实质性差异，因此应该属于伪歧义。

原则上说，伪歧义的消歧只需在各种歧义结构或规则的使用顺序上制订一个优先度即可。但具体问题中，究竟（1）的优先度更高还是（2）的优先度更高呢？这又会面临一些复杂的情况。比如：

（4）疯狂弹钢琴

（5）慢跑四百米

例（4）例（5）虽然按照（1）和（2）的模式都可以解释，但直觉上还是有某种倾向性：例（4）倾向于结构（1），即解释为“疯狂”修饰“弹钢琴”，而非“疯狂弹”接宾语“钢琴”；而例（5）倾向于结构（2），即解释为“慢跑”接“四百米”，而非“慢”修饰“跑四百米”（假设不把“慢跑”看作一个整体的话）。

慢

跑

四百米

ap

vp

np

vp

vp

疯狂

弹

钢琴

ap

vp

np

np

vp

为什么会出现这种差异呢？笔者认为可以从韵律和词汇两个方面加以解释。

从韵律的方面讲，（4）按（1）的方式解读，可形成“2＋3”的音步配置，比按（2）解读的“3＋2”更自然些。假如将（4）改为“（4’）狂弹钢琴”，那么按（2）解释为“狂弹”＋“钢琴”，形成“2＋2”，也未尝不可。同理，（5）按（2）解读会形成“2＋3”，比（1）的“1＋4”要自然得多。假如将（5）改为“（5’）慢慢跑百米”，那么按（1）的解释形成“2＋3”，相对于“3＋2”，似乎又更自然了。

慢慢

跑

百米

ap

vp

np

np

vp

狂

弹

钢琴

ap

vp

np

vp

vp

从词汇的方面讲，“弹钢琴”和“慢跑”相对于“疯狂弹”和“跑四百米”而言是更常用的词汇搭配，或者说，“弹”和“钢琴”之间的“亲和力”，相对于“疯狂”与“弹”之间的“亲和力”要大得多，因此人希望将“弹”和“钢琴”先组合成一个述宾结构，再由前面的“疯狂”修饰；“慢跑”的例子也同理。

这样看来，伪歧义结构的消歧也涉及到多种因素。计算机对伪歧义结构进行消歧时，应该考虑到韵律和词汇的因素。一方面，消歧的优选结果相对于被淘汰的结果而言应该更符合汉语的韵律模式；另一方面，更加常见或出现频率更高的词汇组合应该被优先考虑。这样，才能使伪歧义的消歧结果更加符合人对自然语言的直观。

**歧义格式分析报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S序列 | S序列可能的内部结构 | 功能类 | 实例1（外显型） | 实例2（外显型） | 实例3（外显型） |
| ap vp np | (dj:主谓(ap,vp:述宾(vp,np))) | dj |  | 公正惩治罪犯 |  |
| (dj:主谓(np:组合定中(ap,vp),np)) | dj | 轻易地放弃坏习惯（轻易地放弃是坏习惯） |  |  |
| (np:组合定中(ap,vp:述宾(vp,np))) | np |  |  | 高兴地打开礼物（在西方文化中是有礼貌的。） |
| (np:定中(np:组合定中(ap,vp),np)) | np |  | 公正惩治罪犯（在公正惩治方面犯罪的罪犯） |  |
| (np:联合(np:组合定中(ap,vp),np)) | np | （暂未发现实例） |
| (vp:状中(ap,vp:述宾(vp,np))) | vp | 轻易地放弃坏习惯\*\* | 公正惩治罪犯 | 高兴地打开礼物（他高兴地打开礼物。） |
| (vp:述宾(vp:状中(ap,vp),np)) | vp | 轻易地放弃坏习惯\*\*\* | 公正惩治罪犯 | 高兴地打开礼物 |

注释

\*\*与\*\*\*：属于伪歧义。实例2、3同。