一种有效的概率上下文无关文法分析算法*

朱胜火 周 明 刘 昕 黄昌宁

(清华大学计算机科学与技术系智能技术与系统国家重点实验室 北京 100084)

摘要 句法分析的研究是自然语言处理领域的一个重要组成部分.该文提出并实现了一种有效的概率上下文无关文法SCFG(stochastic context-free grammar)的分析算法.首先对原有的GLR分析表加以改进,以便能够利用分析过程的控制结构来计算有关的概率;然后对分析过程中的每个状态增设了下标,以区分不同的归约路径.通过上述手段,成功地引入了状态的前向(Forward)概率和内(Inner)概率.利用这两个概率可以计算输入句子的所有可能分析树的概率,用于选择最佳的分析结果.通过对大规模真实文本进行实验,结果表明,这种算法具有较高的分析效率.

关键词 SCFG(stochastic context-free grammar),分析算法,GLR算法,Earley算法,自然语言处理.

中图法分类号 TP301,TP18

概率的上下文无关文法SCFG(stochastic context-free grammar)或PCFG(probabilistic context-free grammar)在上下文无关文法CFG(context-free grammar)的基础上加入了概率模型,用于选择优先扩展的分析路径,或者筛选有歧义的分析结果.SCFG的分析算法通常是由原有的上下文无关文法的分析算法扩展而来的.例如,Fujisaki等人对CYK算法的扩展[1],Stolcke对Earley算法的扩展[2],Wright对GLR算法的扩展[3]等等.

本文提出并实现了一种有效的概率上下文无关文法(SCFG)的分析算法.第1节详细描述分析算法的设计,主要包括分析表的构造方法、分析控制结构、状态的前向概率和内概率以及概率化的分析过程,并给出一个分析实例;第2节叙述了对大规模真实文本的实验,并与相关的研究工作进行了比较.

1 分析算法的设计

GLR算法是一种比较有效的分析方法.[4]它采用了LR分析表、图栈(Graph-Structured Stack)和分析森林(Parse Forest)等措施,达到了较快的分析速度(GLR算法的时间复杂度与其他算法相同,是$O(n^3)$,但它具有较小的常系数).因此,本研究采用了类似于GLR算法的分析控制结构来计算分析树概率.

LR分析表最初被设计用于确定的上下文无关语言(如程序设计语言)的分析,在生成算法中采用合并项目集的方法避免可能的冲突.如果直接使用LR分析表来计算项目的前向概率和内概率,则因为在一个项目集中的不同项目可能具有不同的分析概率,所以难以计算项目集中每一个项目的前向概率和内概率.针对这个问题,我们在生成分析表时对某一个状态的相同符号的移进(或转移)动作产生的项目集不进行合并,用其核心项目的前向概率和内概率来定义该状态的前向概率和内概率.而该状态中的其他项目的前向概率和内概率则可通过其核心项目的前向概率和内概率来推定.

此外,在GLR算法的分析过程中,遇到两个(或多个)动作移进(或转移)到同一个项目集情况时,只产生一个新的状态.可是,经过若干步以后,该状态的后续状态可能要归约到两个(或多个)不同的状态,而归约后,转移产生两个(或多个)状态的概率一般说来是不同的,所以,前述遇到两个(或多个)动作移进(或转移)到同一个项目集情况时,只产生一个新的状态的做法,不能有效地利用分析控制结构来计算这些状态的概率.针对这个问题,我们通过对分析过程中的每个状态增设下标来表示分析的起始位置,以区分不同的归约路径,从而可以方便地计算每个归约后转移产生的状态的概率.

---

* 本文研究得到国家自然科学基金和航天预研基金资助.作者朱胜火,1974年生,硕士,主要研究领域为自然语言处理.周明,1964年生,博士后,副研究员,主要研究领域为机器翻译.刘昕,1974年生,硕士,主要研究领域为自然语言处理.黄昌宁,1937年生,教授,博士导师,主要研究领域为计算语言学.

本文通讯联系人:黄昌宁,北京 100084,清华大学计算机科学与技术系智能技术与系统国家重点实验室

本文1996-11-07收到原稿,1997-07-18收到修改稿

通过上述措施，既保留了原有 GLR 算法高效的特点，又可利用 Stolcke 计算前向概率和内概率的方法来优选分析路径和评价分析结果，从而有效地提高了分析效率.

1.1　分析表的构造算法

项目是一个带点的概率化的产生式，形如$(A\rightarrow\alpha\cdot\beta,p)$. 核心项目是$(\rightarrow\cdot S,p)$，或点不在最左边的项目$(A\rightarrow\alpha\cdot\beta,p)(|\alpha|>0)$. 项目集*由一个或多个项目组成，每个项目集中只有一个核心项目，每个项目集可以通过计算核心项目的所有预测项目得到，每个核心项目仅属于一个项目集. KERNER(I)表示项目集 I 的核心项目. 项目集的动作有移进(Shift)、归约(Reduce)和转移(Goto). 每个项目集对同一个终结符(或非终结符)可以有一个动作，也可以有多个动作. 分析表的构造方法如下：

(1) 起始项目集 I_0：KERNEL$(I_0)=(\rightarrow\cdot S,p)$；

(2) 对终结符 b，如果$(A\rightarrow\alpha\cdot b\beta,p)\in I_m$，$(A\rightarrow\alpha b\cdot\beta,p)\in I_n$，则$\langle$shift-to-$n\rangle\in$action$[m,b]$；

(3) 对非终结符 B，如果$(A\rightarrow\alpha\cdot B\beta,p)\in I_m$，$(A\rightarrow\alpha B\cdot\beta,p)\in I_n$，则$\langle$go-to-$n\rangle\in$action$[m,B]$；

(4) 如果$(\rightarrow S\cdot,p)\in I_m$，则$\langle$accept$\rangle\in$action$[m,\$]$；

(5) 如果 $I:\rightarrow\gamma\cdot,p)\in I_m$，则对每个 $b\in$FOLLOW(B)，有$\langle$reduce-$|\gamma|\rangle\in$action$[m,b]$.**

1.2　分析控制结构

为了引入概率化的分析，这里对 GLR 算法的分析控制结构做了一些修改.

我们将每个状态表示为$\langle m,i,k\rangle$，其中 m 是项目集 I_m 的标号，i 是下一个输入终结符的下标，简称输入下标，k 是 KERNEL(I_m)的起始下标.*** 分析过程的初始状态是$\langle0,0,0\rangle$.

如果 $v=\langle m,i,k\rangle$是活动状态，

(1) 如果$\langle$accept$\rangle\in$action$[m,x_i]$，则得到一个分析结果；

(2) 如果$\langle$shift-to-$n\rangle\in$action$[m,x_i]$，且 KERNEL(I_n)是 KERNEL(I_m)的下一个项目，则将 $w=\langle n,i+1,k\rangle$压入分析栈，位居 v 的上方；

(3) 如果$\langle$shift-to-$n\rangle\in$action$[m,x_i]$，且 KERNEL(I_n)不是 KERNEL(I_m)的下一个项目，则将 $w=\langle n,i+1,i\rangle$压入分析栈，位居 v 的上方；

(4) 如果$\langle$reduce-$l\rangle\in$action$[m,x_i]$，KERNEL$(I_m)=(Y\rightarrow\nu\cdot,p)$，对在栈中 v 之前所有第 l 个状态 $v''=\langle m'',k,k''\rangle$，KERNEL$(I_{m''})=(X\rightarrow\lambda\cdot Z\mu,p'')$，$\langle$go-to-$n\rangle\in$action$[m'',Y]$；

如果 KERNEL(I_n)是 KERNEL$(I_{m''})$的下一个项目，将 $w=\langle n,i,k''\rangle$压入分析栈，位居 v''的上方；

如果 KERNEL(I_n)不是 KERNEL$(I_{m''})$的下一个项目，将 $w=\langle n,i,k\rangle$压入分析栈，位居 v''的上方.

通过上述步骤可以得到分析的图栈，用类似于 GLR 算法中构造分析森林的方法[5]来得到分析森林.

1.3　状态的前向概率和内概率

Stolcke 曾经利用计算前缀概率的方法将 Earley 算法成功地扩展成 SCFG 的分析算法.[2]这里，我们借鉴了 Stolcke 计算前缀概率的方法，同时，为了适合我们的算法的分析控制结构，又做了一些调整. 为了引入概率化的分析过程，我们首先给出如下定义.

定义 1. (a) 路径是由状态构成的、由移进和转移连接的状态串.

(b) 如果一条路径中所有移进的终结符连接成串 x，则称之为生成串 x 的受限路径.

(c) 如果一条路径的第 1 个状态的核心项目与最后一个状态的核心项目具有相同产生式和起始下标，则称它为一条可归约的路径.

(d) 路径的长度是指路径中扫描状态的个数.

(e) 路径 $\mathscr{D}$ 的概率 $P(\mathscr{D})$是在 $\mathscr{D}$ 中所有用到的归约产生式的概率之积.

(f) 非终结符生成的子串概率 $P(X\Rightarrow x)$是开始于 X 的生成串 x 的所有可归约受限路径的概率之和. 当 $X=S$ 时，P 就是句子概率.

(g) 前缀概率 $P(S\Rightarrow_L x)$是开始于初始状态、终止于某个扫描状态且生成串 x 的所有可归约受限路径的概率

---

* 有时也叫作状态. 为了有所区分，将在分析表中的称为项目集，在分析过程中的称为状态.

** 这里使用了类似于 SLR(1)的分析表，为了简化，后面的例子仅使用类似于 LR(0) 的分析表.

*** 这里类似于 Earley 状态. 设 KERNEL(I_m)是 $X\rightarrow\lambda\cdot\mu$，则其对应的 Earley 状态是 $i:_kX\rightarrow\lambda\cdot\mu$.

之和.

通过在每个状态上定义以下两个概率,就可以计算子串概率和前缀概率.

定义 2. (a) 状态$\langle m,i,k\rangle$的前向概率$\alpha(\langle m,i,k\rangle)$是所有以该状态为结束状态、长度为$i$的受限路径的概率之和.

(b) 状态$\langle m,i,k\rangle$的内概率$\gamma(\langle m,i,k\rangle)$是所有以$\langle m,k,k\rangle$为起始状态、以$\langle m,i,k\rangle$为结束状态的、长度为$i-k$的路径的概率之和.

推论. 当输入串为$x(|x|=n)$时,

假定$S\Rightarrow_L x_{0\cdots k-1}X$是语法的最左推导,那么,非终结符$X$生成子串$x_{k\cdots i-1}$的概率是$P(X\Rightarrow x_{k\cdots i-1})=\Sigma\gamma(\langle m,i,k\rangle)$(对所有满足LEFT(KERNEL($I_m$))$=X$的$m$进行求和*);

$P(S\Rightarrow x)=\gamma(\langle m,n,0\rangle)=\alpha(\langle m,n,0\rangle)$,其中$m$满足$\langle\rightarrow S\cdot,p\rangle\in I_m$;

$P(S\Rightarrow_L x)=\Sigma\alpha(\langle m,n,k\rangle)$(对所有满足$\langle X\rightarrow\lambda x_{n-1}\cdot\mu,p\rangle\in I_m$的$m$进行求和).

由推论可知,通过计算前向概率和内概率可以得到子串和前缀概率.利用前缀概率可以优选分析路径,利用子串概率可以得到输入串的分析树概率.

1.4 概率化的分析过程

可将每个状态$\langle m,i,k\rangle$扩展为$\langle m,i,k,\alpha,\gamma\rangle$,$\alpha=\alpha(\langle m,i,k\rangle)$,$\gamma=\gamma(\langle m,i,k\rangle)$.分析过程的初始状态是$\langle 0,0,0,1,1\rangle$.

如果$v=\langle m,i,k,\alpha,\gamma\rangle$是活动状态,

(1) 如果$\langle$accept$\rangle\in$action$[m,x_i]$,则得到一个分析结果;

(2) 如果$\langle$shift-to-$n\rangle\in$action$[m,x_i]$,且KERNEL(I_n)是KERNEL(I_m)的下一个项目,则将$w=\langle n,i+1,k,\alpha',\gamma'\rangle$压入分析栈,位居$v$的上方.其中

$$\alpha'=\alpha\qquad\gamma'=\gamma$$

(3) 如果$\langle$shift-to-$n\rangle\in$action$[m,x_i]$,且KERNEL(I_n)不是KERNEL(I_m)的下一个项目,则将$w=\langle n,i+1,i,\alpha',\gamma'\rangle$压入分析栈,位居$v$的上方.其中

$$\alpha'+=\alpha R(Z\Rightarrow_L Y)P(Y\rightarrow v)$$

$$\gamma'=P(Y\rightarrow v)$$

(4) 如果$\langle$reduce-$l\rangle\in$action$[m,x_i]$,KERNEL(I_m)$=\langle Y\rightarrow v\cdot,p\rangle$,对在栈中$v$之前所有第$l$个状态$v''=\langle m'',k,k'',\alpha'',\gamma''\rangle$,KERNEL($I_{m'}$)$=\langle X\rightarrow\lambda\cdot Z\mu,p''\rangle$,$\langle$go-to-$n\rangle\in$action$[m'',Y]$:

如果KERNEL(I_n)是KERNEL($I_{m'}$)的下一个项目,将$w=\langle n,i,k'',\alpha',\gamma'\rangle$压入分析栈,位居$v''$的上方.其中

$$\alpha'+=\alpha''\gamma R(Z\Rightarrow Y)$$

$$\gamma'+=\gamma''\gamma R(Z\Rightarrow Y)$$

如果KERNEL(I_n)不是KERNEL($I_{m'}$)的下一个项目,将$w=\langle n,i,k,\alpha',\gamma'\rangle$压入分析栈,位居$v''$的上方.其中

$$\alpha'+=p\alpha''\gamma R(Z\Rightarrow Y)R(Z\Rightarrow_L W)$$

$$\gamma'+=p\gamma R(Z\Rightarrow Y)$$

$W=$LEFT(KERNEL(I_n)),Z是LEFT(KERNEL(I_m)),$Y\rightarrow v$是KERNEL(I_n)的产生式.

从分析森林中选取概率最大的分析树作为分析结果.具体算法见文献[6].

1.5 一个例子

利用图1的文法G分析句子I saw a man in the park with a scope.可以生成分析表,如图2所示.

注意,在分析表中每个项目集对同一个终结符(或非终结符)可以有1个动作,也可以有多个动作.例如,项目集0中对非终结符S有两个动作:$g1$和$g5$.

分析过程大致如下.起始状态是$\langle 0,0,0,1,1\rangle$,第1个输入的词是I,其词性是_n(名词),所以输入符号是_n.根据分析表,动作是s7,通过1.4节中分析过程步骤(3)得到新的状态是$\langle 7,0,1,0.5,0.333\ 333\rangle$.然后,查分析表知动作是r3,则根据第3条产生式进行规约,通过1.4节中分析过程步骤(4)得到状态$\langle 3,0,1,0.333\ 333,0.2\rangle$和$\langle 10,0,1,0.166\ 667,0.111\ 111\rangle$.以此类推得到句子的分析结果.图3表示了分析的前24个状态.图4表示了分析的全部状态

---

* LEFT(P)表示项目或产生式P的左边,如LEFT($X\rightarrow\lambda\cdot\mu$)$=X$.

** $+=$表示对右边的表达式求和.

(为清楚起见，省略了图 3 中一些没有扩展的状态).

1. S: NP VP 0.6
2. S: S PP 0.4
3. NP: _n 0.333
4. NP: _d _n 0.333
5. NP: NP PP 0.333
6. PP: _p NP 1
7. VP: _v NP 1

图1　文法G

	_n	_d	_p	_v	$	S	NP	PP	VP
0	s7	s8				g1/g5	g3/g10		
1					s2				
2			a						
3				s14					g4
4			r1						
5			s12					g6	
6			r2						
7			r3						
8	s9								
9			r4						
10			s12					g11	
11			r5						
12	s7	s8					g10/g13		
13			r6						
14	s7	s8					g15/g10		
15			r7						

图2　文法G的分析表

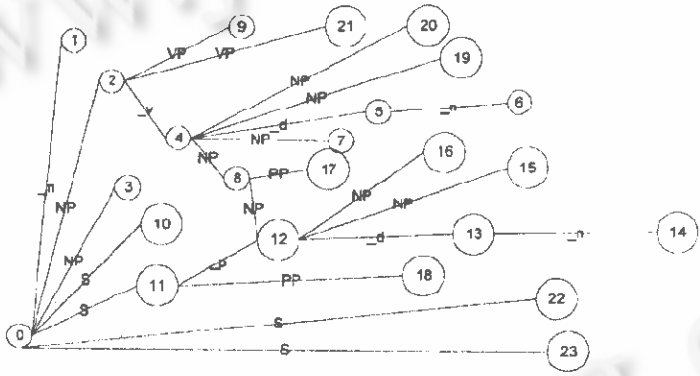

图 3　状态转移图(1)

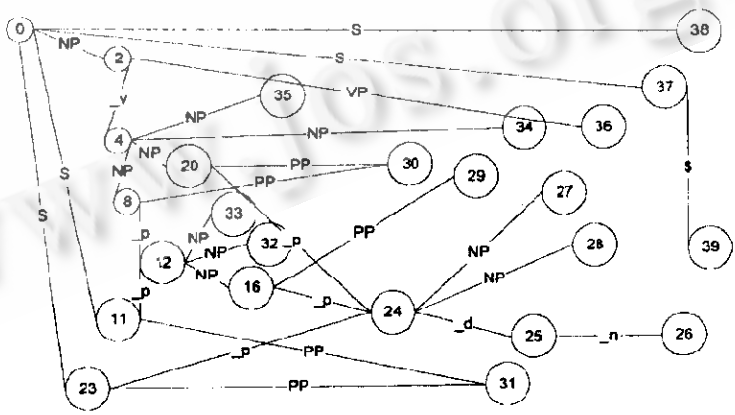

图 4　状态转移图(2)

state 0: ⟨0,0,0.1,1⟩
state 1: ⟨7,0,1,0.5,0.333 333⟩
state 2: ⟨3,0,1,0.333 333,0.2⟩
state 3: ⟨10,0,1,0.166 667,0.111 111⟩
state 4: ⟨14,1,2,0,1⟩
state 5: ⟨8,2,3,0,0.333 333⟩
state 6: ⟨9,2,4,0,0.333 333⟩
state 7: ⟨15,1,4,0,0.333 333⟩
state 8: ⟨10,2,4,0,0.111 111⟩
state 9: ⟨4,0,4,0.111 111,0.066 666 7⟩
state 10: ⟨1,0,4,0.066 666 7,0.066 666 7⟩
state 11: ⟨5,0,4,0.044 444 4,0.026 666 7⟩
state 12: ⟨12,4,5,0,1⟩
state 13: ⟨8,5,6,0,0.333 333⟩
state 14: ⟨9,5,7,0,0.333 333⟩
state 15: ⟨13,4,7,0,0.333 333⟩
state 16: ⟨10,5,7,0,0.111 111⟩
state 17: ⟨11,2,7,0,0.037 037⟩
state 18: ⟨6,0,7,0.014 814 8,0.008 888 89⟩
state 19: ⟨15,1,7,0,0.037 037⟩
state 20: ⟨10,2,7,0,0.012 345 7⟩
state 21: ⟨4,0,7,0.012 345 7,0.007 407 41⟩
state 22: ⟨1,0,7,0.016 296 3,0.016 296 3⟩
state 23: ⟨5,0,7,0.010 864 2,0.006 518 52⟩
state 24: ⟨12,7,8,0,1⟩
state 25: ⟨8,8,9,0,0.333 333⟩
state 26: ⟨9,8,10,0,0.333 333⟩
state 27: ⟨13,7,10,0,0.333 333⟩
state 28: ⟨10,8,10,0,0.111 111⟩
state 29: ⟨11,5,10,0,0.037 037⟩
state 30: ⟨11,2,10,0,0.008 230 45⟩
state 31: ⟨6,0,10,0.005 267 49,0.003 160 49⟩
state 32: ⟨13,4,10,0,0.037 037⟩
state 33: ⟨10,5,10,0,0.012 345 7⟩
state 34: ⟨15,1,10,0,0.016 460 9⟩
state 35: ⟨10,2,10,0,0.005 486 97⟩
state 36: ⟨4,0,10,0.005 486 97,0.003 292 18⟩
state 37: ⟨1,0,10,0.009 613 17,0.009 613 17⟩
state 38: ⟨5,0,10,0.006 408 78,0.003 845 27⟩
state 39: ⟨2,0,11,0.009 613 17,0.009 613 17⟩

2 讨 论

2.1 相关工作

对GLR算法的概率化扩展的其他工作有文献[3]的方法.该方法利用LR转移概率来近似SCFG概率,但是,由于LR转移概率与SCFG转移概率并不完全等价,所以,基于LR转移概率的分析模型并不完全等效于基于SCFG的概率分析模型.并且LR分析表的状态等同于项目(带点的产生式)的集合,任何项目的集合都是一个潜在的状态,所以在最坏情况下,状态的个数与项目个数成指数关系.

本研究实现了一个真正的SCFG的概率分析算法.而且,它的分析表的项目集(状态)与核心项目一一对应,所以项目集的个数仅与项目个数成线性关系.

本研究是在分析算法上的改进,原则上与被分析的具体语言无关.但由于语料库的限制,本文仅对英语句法分析进行实验.国内外某些作者已经应用SCFG分析汉语,详见文献[7,8].

2.2 分析的时间

为了验证本算法的有效性,我们分别利用本算法与Stockle扩展的Earley算法进行对比实验.实验采用了Susanne英语树库.[9]Susanne树库是一个面向自然语言处理的语法标注的语料库,它包含64个文件,共15万词.我们从中获取了723条带概率的产生式作为实验用的语法.[6]在Pentium/120的微机上对所有的5 671个句子进行了分析,其中选取50个句子的分析时间,如图5所示.从图中可以看出,本算法比Stockle扩展的Earley算法在分析时间上有明显的提高.

2.3 剪枝阈值的设定

为了提高分析速度,需要在分析过程中对分析图栈进行剪枝,然而剪枝影响到分析的通过率和正确率.这里分析的通过率定义为可以得到整个输入句子分析树的百分率;在能得到分析树的句子中,分析的正确率定义为在概率最大的分析树中正确分析的短语比率;如果剪枝阈值为r,则表示在具有相同输入下标的所有状态中,前向概率小于最大值的e^{-r}的状态时不进行移进动作.

我们对Susanne语料库N01中的113个句子进行分析,平均句长为16.5词,其正确率、通过率与剪枝阈值的关系如图6所示.从图中可以发现,通过率随剪枝阈值的增大而提高,当剪枝阈值达到10时,它基本保持不变;随剪枝阈值的增大,正确率稍微有所减小,在剪枝阈值达到10时,它也基本保持不变.所以,在实际分析时将剪枝阈值设为10,对Susanne英语树库中所有的5 671个句子进行了分析,通过率为75.3%,正确率为73.7%.

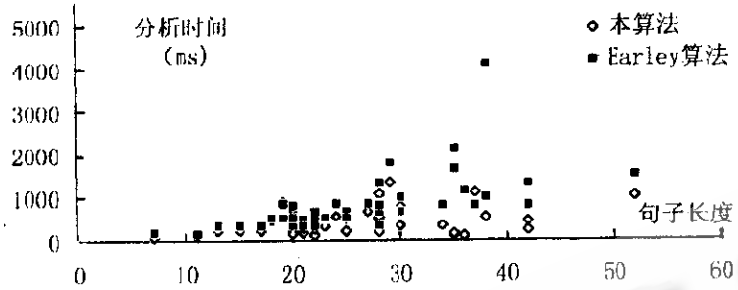

图 5　句子长度与分析时间的关系

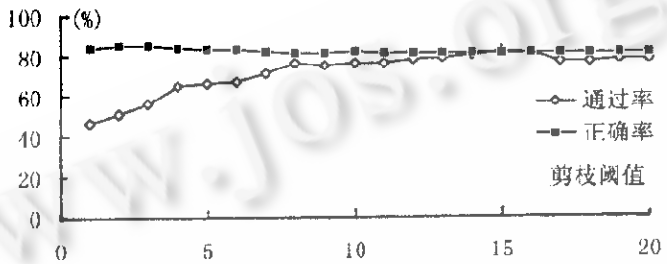

图 6　剪枝阈值与正确率、通过率的关系

参考文献

1 Fujisaki T, Jelinek F, Cocke J *et al*. A probabilistic parsing method for sentence disambiguation. In: Tomita M ed. Current Issues in Parsing Technology, International Workshop on Parsing Technologies. Pittsburgh, Boston: Kluwer Academic Publishers, 1991. 139～152

2 Stolcke A. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. Computational Linguistics, 1995,21(2):165～201

3 Wright J H. LR parsing of probabilistic grammars with input uncertainty for speech recognition. Computer Speech and Language, 1990,(4):297～323

4 Shann P. Experiments with GLR and chart parsing. In: Tomita M ed. Generalized LR Parsing, International Workshop on Parsing Technologies. Pittsburgh, Boston: Kluwer Academic Publishers, 1991. 17～34

5 Tomita M. Efficient Parsing for Natural Language. Boston: Kluwer Academic Publishers, 1986

6 朱胜火. 从双语对齐文本中获取翻译模板[硕士论文]. 清华大学,1997
(Zhu Sheng-huo. Learning translation patterns from bilingual aligned text [M.S. Thesis]. Tsinghua University, 1997)

7 张民. 基于弱限制随机上下文相关文法的汉语树库构造方法研究[博士论文]. 哈尔滨工业大学,1997
(Zhang Min. Research on algorithms of Chinese treeback construction based on weakly restricted stochastic context-sensitive grammars[Ph.D. Dissertation]. Harbin: Harbin Institute of Technology, 1997)

8 Lee L-S, Chien L-F, Lin L-J *et al*. An efficient natural language processing system specially designed for the Chinese language. Computational Linguistics, 1991,17(4):347～375

9 Sampson G. English for the Computer. Oxford: Oxford University Press, 1995

An Efficient Stochastic Context-Free Parsing Algorithm

ZHU Sheng-huo　ZHOU Ming　LIU Xin　HUANG Chang-ning

(State Key Laboratory of Intelligent Technology and Systems　Department of Computer Science and Technology
Tsinghua University　Beijing　100084)

Abstract　The research of parsing is important in the field of natural language processing. An efficient stochastic context-free parsing algorithm is described in this paper. In order to implement a stochastic context-free parser, the authors rebuild a GLR-algorithm-like parsing table so that derivation probabilities can be computed efficiently by making use of the parsing control structure, and add indices to each state in parsing period as identifiers of different parsing paths. Based on the techniques above, the forward and inner probabilities of states are introduced in this paper. With these two probabilities, the probabilities of all parsing trees of the input sentence can be computed to select an optimal parsing result. The experiment shows that the proposed algorithm is efficient.

Key words　Stochastic context-free grammar, parsing algorithm, GLR algorithm, Earley algorithm, natural language processing.