第 6 卷第 6 期
1995 年 6 月

软 件 学 报
JOURNAL OF SOFTWARE

Vol. 6, No. 6
Jun. 1995

自然语言句法分析的有界深度控制和早期剪枝*

万建成

(山东工业大学计算机系,济南 250014)

摘要 由于词汇量大、句法结构复杂,很难或无法采用列表、LL、LR 等建立扫描表方法,提高自然语言句法分析的效率. 在传统的自顶向下规则扫描和 ATN 网络分析算法基础上,本文提出了有界深度控制早期剪枝的分析技术. 该技术可有效地限制分析的搜索范围,减少回溯,提高句法分析效率.

关键词 自然语言处理,句法分析,ATN 网络,RTN 网络.

在自然语言理解、机器翻译、应用系统的智能接口设计、汉语自动分词、语音识别的后处理、拼音智能汉字输入等研究中,普遍关心的基本问题是汉语自然语言的句法分析. 自然语言的句法结构远比机器语言结构和语义复杂得多、词汇量大得多. 这就使自然语言句法分析的实现在复杂性上大大高于,而运算速度上远远低于机器语言,从而影响了自然语言处理技术的实用性. 因此,能否大幅度提高句法分析算法的效率,就成了自然语言处理研究的重要方面[1].

重写规则、RTN 状态转移、特别是 ATN 扩充状态转移网络,由于其表达复杂上下文相关句法和语义分析的能力,在自然语言处理研究中得到了广泛的应用[2,3]. 基于这些句法表达的句法分析过程,可以抽象为句法树的不确定性搜索问题. 造成这些句法分析效率低的关键原因是回溯. 当句法树的某一分支与输入不匹配时,就放弃该分支而递归试验兄弟分支,直到该结点的所有分支都试验过.

如此,只要找到一个测度,以此避免递归试验或提前结束实验注定失败的分支,就可以减少回溯次数,而使分析效率得以提高.

在说明了句法分析的效率问题之后,本文给出了几种改进效率的算法,并指出了这些算法对于处理自然语言的局限性,和进一步改进的必要性. 在定义了控制回溯的测度——句型长度之后,本文设计提出了减少回溯、加快分析速度的有界深度搜索和早期剪枝算法.

1 句法分析的效率问题和改进方法

句法分析理论和实现研究,大都是围绕上下文无关文法展开的[4]. 实现上下文无关文法

* 本文 1993-06-14 收到,1993-10-08 定稿

作者**万建成**,1949 年生,副教授,主要研究领域为人工智能,自然语言处理,专家系统,汉字输入,程序语言.

本文通讯联系人:万建成,济南 250014,山东工业大学计算机系

分析最简单、最直观的方法就是回溯算法. 它模拟非确定性文法，计算复杂度(空间和时间效率)被看成是待分析输入符号串长度的函数. 回溯算法需要线性的空间，但可能花费指数的时间，当句法规则的数目增多，句法结构复杂化时，句法分析的时间消耗往往使人无法忍受[1]. ATN[2]网络是规则之外的另一句法表达方法，它在自然语言处理研究中得到广泛使用，其分析复杂度也是指数级的.

文献[4]提到的 Cocke—Younger—Kasami 的列表分析方法，它们的时间复杂度大幅度降为$O(n^3)$，n 是输入符号串的长度. 在 ATN 分析方面，Kay[5]等提出了良定子串(Well Formed Substring)的表分析方法，使基于 ATN(除去语义分析实质上是 RTN)分析的时间复杂度从$O(2^n)$降到$O(n^3)$. 由于这些改进都是针对上下文无关文法进行的，它们不适用于自然语言处理的上下文有关文法[6].

实现上下文无关文法的一大类无回溯分析算法 LL(k)、LR(k)和优先文法等，可以有线性的时间复杂度$O(n)$. 它们的实现要建立包含全部终止和非终止符在内的剖析矩阵. 自然语言巨大的词汇符号量和复杂的句法关系，使得剖析矩阵占用太多的存储空间，加之这些算法只适用上下文无关文法，因而限制了对于自然语言处理的可用性.

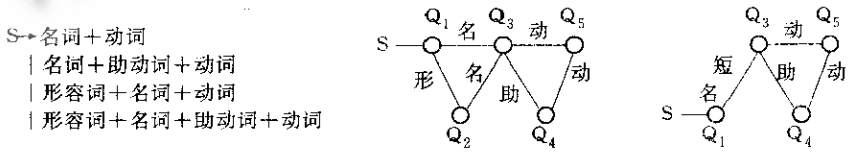
潘恩[7]提出并实现了“通过向前看状态”和自顶向下和自底向上结合的分析方法，在研究减少回溯方面，也作过尝试. 但由于本质上自顶向下和自底向上分析的复杂度相同，仅“向前看”有限的字符，并不适应自然语言复杂的句法结构，效率的提高也非常有限.

总而言之，自然语言处理中词汇量极大，需要的是上下文相关的多义性文法，而且对一切满足句法关系的多个分析都感兴趣[4]. 以往的各改进算法都在这些重要方面受到了限制，所以不得不采用效率低的带回溯的分析方法.

本文从自然语言复杂句法结构特征(上下文有关)和穷尽分析搜索(产生一切合法分析)的需要出发，借鉴人工智能研究中有界深度搜索和搜索的剪枝技术，提出了一种基于句法的树结构、依靠句型对输入符号串长度要求来限定搜索空间，减少回溯的句法分析方法. 该方法可有效地提高搜索的效率，因为仅使用与上下文无关的句型长度进行分析控制，因而适用于 RTN、ATN 和生成规则表达的上下文相关自然语言处理的句法分析.

2　句法树搜索：规则和 ATN 网络抽象

一个 FSTN(Finite State Transition Network)状态转移网络表达了一个句法结构[3]. 相反，一个不含非终止符和递归的简单句法结构可以用一个 FSTN 表示. 如图 1 所示，Q1…Q5 是状态，Q1 是起始状态，Q5 是终止状态. S 是该网络表示的句法规则.



(a) 规则　　(b) FSTN　　(c) RTN

图1 同一句法的规则和FSTN、RTN的对应

如果允许FSTN的弧的标志为句法的非终止符,例如图1(c),就形成了递归状态转移网络RTN(Recursive Transition Network).根据扩充状态转移网络ATN(Augmented Transition Network)的定义可知,ATN不过是包含了上下文信息和状态跳转的RTN网络.于是,可以证明如下命题.

命题1. 如果不考虑上下文有关信息,从句法结构上ATN与RTN是等价的,而且与句法规则表达也是等价的.(证明略)

一个不存在循环和自环的RTN网络,可以转换为一棵有向树(以下'树'都指有向树).方法很简单,只要把含多个入射弧的结点,分裂成多个单一入射弧、但仍保留原射出弧的结点.同理,一个不存在循环和自环的ATN网络,也可以转换为一棵树.类似地可以得到,一个不存在循环但可有自环的RTN、ATN网络,可以转换为允许同名结点无限重复(或允许存在自环)的广义树(见以下定义).

定义1. 广义(有向)树允许任一树枝及其指向的结点以同名方式无限连续重复出现的(有向)树,或简而言之,允许结点有自环的树,称为广义(有向)树.

定义2. 广义(有向)树森林一个或多个广义(有向)树的集合称为广义(有向)树森林.

推论1. 在不考虑语义或上下文相关性的条件下,一个不含循环的ATN网络存在一个与其语法上等价的广义树(森林),和一个与其等价的规则集.(由命题1正确性可证).

推论2. 在不考虑语义或上下文相关性的条件下,一个不含循环的ATN网络上的句法分析过程,存在一个从句法分析功能上与其等价的广义树(森林)上的句法分析过程.(由推论1的等价性可证)

为叙述方便,以下提到的FSTN、RTN、ATN网络都指不含回路的.

3 句型长度

在一个不含自环的FSTN所对应的树上,任何从根结点到达某一叶结点(终止结点)所形成的路径描述了一个句型.例如在图2(a)上,从S到Q7的路径对应的句型是〈名+助+动〉.因为FSTN不含非终止符,显然,满足这一句型要求的输入符号串长度只能是3.当输入串的长度小于或大于3时,该句型的匹配是不可能成功的.

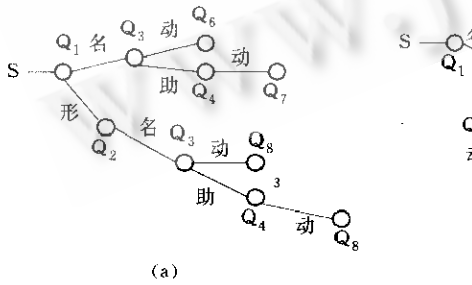


图2 RTN、ATN到广义树的转换

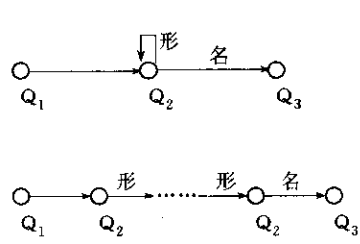


图3 广义树的自环及其无限扩展

这一简单事实说明,任何句型要与输入符号串匹配成功,其必要条件是句型的终止符号个数等于输入符号串的长度.对一个不含自环的FSTN,其转换树上从根结点到任何叶结

点的路径长度是确定的.可以利用一特点限制句法分析的搜索过程.例如,若输入串的长度是 2,对于图 2(a),只有从 S 到 Q_6 的路径可能成功,因为其他任何到叶结点的路径长度都大于 2.这就是本文提出有界深度控制和早期剪枝以加快分析速度、减少回溯的基本思想.

问题并非这么简单,当着 FSTN 包含自环时,在含非终止符或自环的 RTN 网络上,句型长度(从根或任意结点到某叶结点路径长度)就不那么直观和好计算了(见图 3).

对于仅含自环不含非终止符的路径,由于存在可能的无限扩展,句型长度也就可能趋向无穷大(当然,对于自然语言趋向无穷大是不可能的).对于 0 扩展,句型长度就是不含自环时的路径长度.对于含 m 个自环、每个自环最多发生 n 次扩展的句型,路径的最大长度就限制在不含自环长度与 mn 的和.这样,就出现了最少和最大路径长度(或最少和最大句型长度)的约束要求.

对于不含自环但含非终止符的路径,句型长度计算就复杂多了.这是因为,非终止符在句法分析时要产生扩展,其扩展长度与非终止符代表的子句法结构的复杂程度有关.由于子句法对应的广义树可有多个从根到叶结点的路径,所以非终止符所代表的子句型长度不能仅使用一个数值标示,从而也就无法准确计算含非终止符句型的长度.但可以采取类似自环的扩展,计算最小和最大句型长度.

对于既含终止符又含自环的路径,其句型长度计算则是以上两种情况的组合.

总上,得到 RTN(或 ATN 或规则)广义树的句型长度及其计算的定义.

定义 3. 子句型　设 T 是一句法规则集(或 RTN、ATN 网络)的广义树,d 是 T 上任意一结点.定义从 d 通过有向树枝到达某一叶结点(终止结点)的路径为整个句法的一子句型.或简称为句型.

据上说明,任一(子)句型对应的路径可包含终止符、非终止符以及由两者形成的自环.

定义 4. (子)句型长度　设 T 是一句法的广义树,L 是树 T 上某一结点 d 起始的(子)句型.该句型 L 的长度计算如下:

L 的长度=非自环终止符树枝数+Σ非自环非终止符扩展长度+Σ自环终止符扩展长度+Σ自环非终止符扩展长度.

对于包含非终止符和自环的句型,其长度可进一步定义为最小和最大长度.

定义 5. 最小(大)句型长度　句型 L 的最小(大)句型长度,定义为将 L 的自环和非终止符完全扩展后,可包含的最少(多)终止符的个数.

定义 6. 静态句型长度　广义树上定义的句型长度称为静态句型长度.

定义 7. 动态句型长度　非终止符总是对应一个独立的广义树,其根结点汇集着长度不等的多个子句型.设 N 是句型 K 中的一个非终止符.将 N 对应的某一子句型 Li 长度替代 N 的句型长度,而得到的 N 所在句型 K 的新的长度,称为句型 K 的动态长度.

任一动态句型长度必定包含在其静态长度的范围内.动态长度的最小值可大于静态长度最小值,其最大值可小于静态长度的最大值.简言之,满足句型静态长度的输入串不一定也满足该句型的动态长度要求.可见,动态长度更精确地描述了句型对长度的限制要求.

4　有界深度搜索与早期剪枝

将句法分析抽象为广义树上的搜索问题,并定义了句型长度概念,就可以应用优化树搜

索提高句法分析的效率了.有两种提高效率的方法:有界深度搜索和早期剪枝.

已经知道,句型长度限制了一个句型成功时对输入符号串的长度要求.有界深度搜索,就是当着搜索在广义树上某结点的位置(树的深度)已经超过输入串的长度时,就立即停止搜索,并宣告失败返回的搜索方法.这正是在一般树搜索中限制深度的有界深度搜索思想.这是在已经进入某一分支的搜索,且在未到达搜索的终了时已提前确认失败,而避免多余无效操作的方法.

该有界搜索在深度优先或宽度优先的算法中都可以使用.

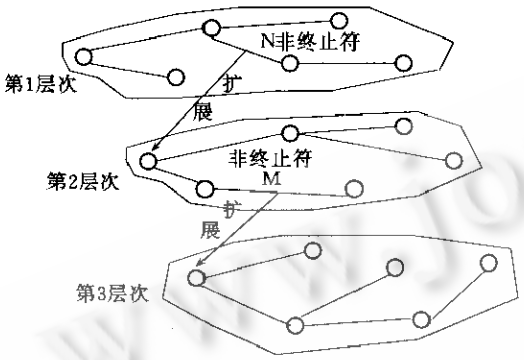


图4　非终止符的扩展层次

早期剪枝不同于有界深度搜索.它是在进入某一结点前,首先计算和检查该结点句型长度.如果最小的句型长度小于输入串长度,或最大的句型长度大于输入串长度,则未经匹配就放弃该结点的搜索.这样,就在未进入搜索前,剪掉了部分注定失败的结点及其子树,从而减少了回溯,提高了句法分析速度.

为了正确实现含非终止符复杂句法的分析,必须处理好非终止符扩展时动态句型长度的计算和有关问题.这一方面是分析算法的要求,另一方面是为了尽可能大的提高分析效率.这是个涉及非终止符调用时的扩展层次问题,见图 4.

5　减少回溯的有界深度控制与早期剪枝算法

该算法有以下特点:①采用广义树的有界深度优先递归搜索;②广义树中除自环外,不存在回路;③产生一切输入串满足的句法规则,即搜索在广义树上是穷尽的.

设 T 是某句法的广义树,结点 N∈T 有 n 个树枝(n≥0).每个树枝 Li 表示为 Li=〈Ai,Ni,Ci,mini,maxi〉,其中 Ai∈终止符+非终止符,Ni 是当 Ai 与输入串匹配成功时继续搜索而转移到的下一个结点,Ci,i∈0..n,Ci≠0 表示有自环,mini、maxi 是沿 Li 到达终止结点的最小和最大句型长度.当 N 没有树枝(n=0),或仅有自环时(Ci=n),N 是终止结点.结点 N=〈min,max〉,min 和 max 是从 N 到达其所有终止结点的最小和最大句型长度,也可以表示成 N.min 和 N.max.注意,非终止符对应子广义树根结点的 min 和 max 值,只能比较粗糙地起到限制搜索的作用.实际搜索中要使用其树枝对应的句型长度作为动态句型长度来尽可能准确地限制搜索范围.

设 X=〈X1,…,XL〉为输入串,L≥1 是输入串的长度.P 为当前输入串位置,Xp 为当前待处理的输入符号.Succ 是全程变量,指示句法分析是否有过全局成功.Layer 是全程变量,指示当前非终止符扩展层次.算法中略去了对 Layer 的处理.

本算法是一个递归函数,全局或局部分析成功时它返回当时匹配成功的下一个字符位置,返回 0 说明没有成功.

为简化描述,以突出有界深度和剪枝处理,算法中未描述成功匹配时句型结构的产生.

整个算法命名为 LPDepth，描述如下.

算法：

```
函数 LPDepth(P:输入串字符位置;
             N:句法树结点;
             Min,Max:动态句型长度):输入串字符位置;
全程常量:    L:输入串长度;
全程变量:Layer :非终止符扩展深度,初始值=1;
         Succ  :有否分析成功的句型,初始值=false;
局部变量:ToEnd :输入串字符位置;OnceSucc:分析有否局部成功;
         Q     :句法树结点;    m       :输入串字符位置;
begin of LPDepth;{算法操作开始}
Rep:if P>L
    then if N∈终止结点 or N是空结点
         then if Layer=1
              then Succ=true; return P;{全局成功}
              else return 0;
if (N∈终止结点 and N没有子树枝) or N是空结点
  then return P;{局部成功,对非终止符处理有意义}
ToEnd:=L-P+1;
if (ToEnd<Min) or (Max<ToEnd) then return 0;
                {输入串长度超过句型长度限制,停止对子树的搜索}
for each 树枝 Li∈N do {Li=(Ai,Ni,Ci,mini,maxi)}
  case Ai of
  终止符:if (Min-N.min+mini)<ToEnd<(Max-N.max+maxi)
         then
           if Ai=Xp      {Xp是当前处理的输入符号}
           then if Ci>0  {Li是N的自环}
                then P=P+1;Min=Min-1;Max=Max-1;
                     goto Rep;
                else return LPDepth(P+1,Ni,mini-1,maxi-1);
         else if Ci>0    {Li是N的自环}
              then return LPDepth(P,Ni,Min,Max);
  非终止符:取Ai网络及其Ai.min,Ai.max;
           OnceSucc=false; Q=Ai; {Q为暂存}
           for each 树枝 Kj∈Ai do
             使Kj成为结点Q的唯一树枝;
             m=LPDepth(P,Q,Min-Ai.min+Kj.min,Max-Ai.max+Kj.max);
             if m≠0
             then OnceSucc=true;
                  if Ci>0 {Li是N的自环}
                  then P=m; Min=Min-Ai.min; Max=Max-Ai.max;
                       goto Rep;
                  else return LPDepth(m,Ni,Min-Ai.min,Max-Ai.max);
             else if Ci>0 {Li是N的自环}
                  then return LPDepth(P,Ni,Min,Max);
```

```
            if not OnceSucc {N 的所有分枝都未成功}
              then if Ci>0 {Li 是 N 的自环}
                     then return LPDepth(P,Ni,Min,Max);
    end;
end of LPDepth; {算法操作结束}
```

6 算法特性与实验结果

本文提出的减少回溯的 LPDepth 算法，是作者在研究基于句法和有限语义分析的汉语同音词识别系统 FPY[8,9]时设计并实现的. 研究实验表明，使用本算法处理 FPY 的 170 余条含大量非终止符扩展和递归的 20 余类汉语句法规则，使典型短语句型的运算时间由从无深度控制和早期剪枝的 10 分钟以上，下降到低于 1 秒钟(386 微机). 可见算法在提高句法分析效率方面是十分有效的.

综合以上，算法 LPDepth 和本文提出的思想归纳如下：

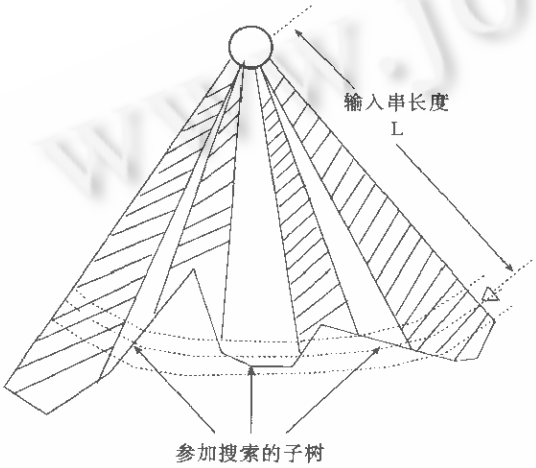


图5 算法的剪枝效果

① 减少回溯是通过有界深度控制和早期剪枝实现的. 有界深度控制使搜索超过输入串长度时，就停止进一步的匹配和扩展. 早期剪枝使一切小于和大于输入串长度临近值的句型树和分枝都被剪除掉，从而限制了参与进行搜索的句型范围. 动态句型长度的计算使得临近范围压缩到可能的最小值，增强了剪枝效果. 如果 D 为一句型集广义句型树的根，在本算法的作用下，只有句型的定义(最小和最大)长度在输入串 L 长度△临近的路径才参加搜索，其他都被剪除或放弃(如图 5 所示).

② LPDepth 是应用上述思想实现的深度优先搜索算法. 该思想还可以在宽度优先搜索中实现，方法与 LPDepth 相似. 每当扩展一结点或进入某分枝时，就检查其句型长度. 不满足静态、动态长度要求的就立即停止搜索.

③ 实现本思想，要求对句法的广义句法树进行句型长度标定. 如果能在 RTN、ATN 树甚至规则集上直接标定句型长度，并适当修改算法，仍可以实现本文提出的减少回溯的思想.

④ 依据句法树结点和树枝的个数估计本算法对效率的改进程度是困难的，因为有界深度控制和剪枝对于效率的作用取决于具体句法树的结构. 对于所有终止结点都在同一深度的句法树，当输入串长度小于或大于该深度的某一临近值(由句法定义决定的)时，算法不发生任何符号匹配而立即失败终止，即拒绝接受所有句型. 当然，如果输入串长度与所有终止结点深度临近时，算法的效率不会有任何提高，此时算法就退化为一般的低效搜索. 假如句法树的各终止结点的深度不一致，那么只有临近输入串长度的终止结点对应的路径才被搜索.

当着最短和最大句型长度差别较大时，有界搜索的效果受到了很大的影响. 然而，这一般发生在刚刚进入非终止符扩展树的根结点时，随着动态句型长度的计算，搜索的范围仍然受到了限制.

⑤ 本文的思想和 LPDepth 算法是基于穷尽式树搜索的，所以在理论上的计算复杂度仍然是指数级的，但由于大量减少了搜索范围和回溯，因而分析效率得到了很大的提高.

⑥ 存在问题. 本算法和思想的关键是准确标定句法广义树的句型长度. 不存在递归和自环时，这是很容易使用深度优先的后根处理方法而自动实现的(相应算法略). 存在递归时，如何合理和精确地标定是一个待研究的问题. 目前采用的是手工标定方法，取最少和最大允许递归次数，以计算递归扩展的最小和最大句型长度. 如④所述，动态句型长度的计算，此时不仅起到了限制搜索的作用，而且确保了左递归存在时算法的终止.

参考文献

1 郭进. 统计语言模型及汉语音字转换的一些新结果. 中文信息学报，1993，7(1)：18－27.

2 Woods W A. An experimental parsing system for transition network grammars. In: Rustin R ed. Natural Language Processing, New York: Algorithmics Press, 1972. 113－154.

3 刘开瑛，郭炳炎. 自然语言处理. 北京：科学出版社，1991. 23－31.

4 石青云. 形式语言及其句法分析. 北京：科学出版社，1987.

5 Kay Martin. Algorithm schemata and data structures in syntactic processing. Proceedings of the Symposium on Text Processing, Nobel Academy, 1980.

6 Charniak E, McDermott D. Introduction to artificial intelligence. Addison－Wesley Publishing Company, 1985. 194－244.

7 潘恩，陈沐天. 减少回溯的自然语言理解. 计算机研究与发展，1988，**25**(4)：13－17.

8 万建成. 汉语同音词的上下文相关识别. Comm COLIPS，1992，**2**(1)：33－39.

9 万建成. FPY 中同音词智能识别方法. 中文信息学报，1993，**7**(2)：27－35.

DEPTH LIMITED SEARCH AND PRUNING IN SYNTAX ANALYSIS OF NATURAL LANGUAGE PROCESSING

Wan Jiancheng

(*Department of Computer Science, Shandong University of Technology, Jinan* 250014)

Abstract　Because of large vocabulary and high complexity, it is difficult and almost impossible to make use of LL, LR and Well Formed Substring Table techniques to speed up syntactic analysis in Natural Language Processing. Based on the traditional Top－Down analysis of rules and ATN network, a depth limited search and early pruning technique is proposed and discussed in this paper, which greatly improves the efficiency of natural language parser, by restricting the search scope and reducing backtracking.

Key words　Natural language processing, syntax analysis, ATN network, RTN network.