文章编号:1001 - 9081(2009)05 - 1251 - 03

基于期待类型的 Chart 句法分析算法

王 燚,李中志
(成都信息工程学院 网络工程系,成都 610225)
(wangyi1177@126.com)

摘 要:Chart 算法是目前句法分析中应用最广泛的算法之一,但该算法的计算效率仍有待提高。通过对两种常用 Chart 算法的分析,提出一种以由底向上的 Chart 算法为基础,结合自顶向下 Chart 算法的预测能力的算法。算法按严格从左到右、由底向上的方向进行,根据已有活动边的活动角色类型和句法规则,产生当前词位置上的期待类型表,并以此限制后续边的生成。对比实验的结果表明,分析速度较普通 Chart 算法提高了约 24%,同时也减少了一半以上因边池溢出而导致的分析失败的语句。

关键词:自然语言处理; 句法分析; Chart
中图分类号: TP18 **文献标志码**:A

Grammatical parsing algorithm based on expected categories

WANG Yi, LI Zhong-zhi
(*Department of Network Engineering, Chengdu University of Information Technology, Chengdu Sichuan* 610225, *China*)

Abstract: Chart algorithm is one of the most popular algorithms in grammatical parsing. Its computational efficiency, however, still needs to be improved. By analyzing two common used Chart algorithms, a new parsing algorithm was proposed based on the bottom-to-top Chart algorithm, combining with the predicating ability of the top-to-bottom Chart algorithm. The proposed algorithm executed exactly from left to right and bottom to top. At each word position in parsing, an expecting table was formed according to the active roles of existing active arcs and the grammatical rules, and was used to restrict the creating of arcs in the rest process of parsing. Experimental results show that the parsing speed improves about 24% compared with the standard Chart algorithm, meanwhile the number of the failed sentences reduces more than a half.

Key words: natural language processing; grammatical parsing; Chart

0 引言

句法分析将词性标注串分析为一棵句法树,作为进一步进行语义分析或语言翻译的基础。句法分析有多种方法,如:基于合一文法的句法分析、面向数据的句法分析(Data-Oriented Parsing,DOP)、基于规则的句法分析。基于规则的句法分析是目前汉语句法分析中最常用的一种,其具体算法也有多种,如:左角分析法、CYK 算法、Tomita 算法、Chart 算法等。各种句法分析理论和方法有两个共同的目标:高效率和高准确性。高准确性是句法分析结果能够被有效利用的基础,一直被人们广泛关注。然而高效性同样也是算法必须具备的一个特征。因为在许多自然语言处理的应用场合,如文本过滤、机器翻译、人机对话等,都需要具备一定的实时性。

Chart 算法是目前应用最为广泛的句法分析算法之一,是一种非常灵活的算法。通过修改其分析过程的一些策略,Chart 算法可以模拟多种其他句法分析算法,如:LR、CYK 算法等。但 Chart 算法的计算复杂度比较高。假设句子词长为 N,句法规则总长度为 G(以规则中的句法符号数计量),其计算复杂度为 $O(N^3G)$。对于稍长一些的句子(如词汇量在 30 词以上),标准 Chart 分析的速度在很多应用环境下是不可接受的。为了提高 Chart 算法的句法分析效率,人们已提出多种改进算法。如文献[1]中提出的基于局部优先的汉语句法分析方法,文献[2]中提出的"角色反演算法",文献[3]中提出的基于图算法的二元组合文法分析等。此外,人们也考虑了充分利用语言中提供的信息来提高句法分析效率的方法。如文献[4-5]中均提出利用句内标点符号的指示作用来提高句法分析的效率。

Chart 句法分析算法有许多实现方式,并各有其优缺点。其中由底向上的 Chart 算法是应用最为广泛的一种。由底向上的 Chart 算法的优点是:其生成边是依据被分析语句的词性序列逐步向上搭建而成的,因而可以避免使用大量在当前语句条件下无效的句法规则生成无效的边,其效率得到基本的保障。而自顶向下的 Chart 算法虽然在效率上不如由底向上的算法,但它仍然有自身的优势:可以保证生成的边在完整的语句句法结构中是有效的(即是完整句法树的一个组成部分),因为它是从语句类型(S)的规则开始向下逐步完成其组成部分的。通过综合两种方法的优点,本文提出了对 Chart 算法的一种旨在提高效率的改进:基于期待类型的 Chart 算法。实验表明,该算法较普通 Chart 算法在效率上提高了约 24%,同时也明显减少了因边池溢出而导致分析失败的语句数量,可以作为提高分析效率的措施之一被应用到 Chart 句法分析程序的实现中。

1 基于期待类型的 Chart 算法

1.1 Chart 算法简介

Chart 算法使用生成式句法规则,表示为:$N \rightarrow \xi$。其中 N

收稿日期:2008 - 11 - 13;**修回日期**:2009 - 01 - 31。
基金项目:国家自然科学基金资助项目(60474022);四川省教育厅科技基金资助项目(07ZC008)。
作者简介:王燚(1968 -),男,四川成都人,副教授,博士,主要研究方向:自然语言处理、文本数据挖掘、网络搜索; 李中志(1973 -),男,四川成都人,讲师,博士研究生,主要研究方向:信号处理、人工神经网络。

为非终结符，ξ 为由一个或多个终结符或非终结符组成的序列。序列中的每个句法类型称为规则的角色。

Chart 算法依赖于几个数据结构：边、边表（又称边图，Chart）和 Agenda 栈（又称日程表）。边定义为一个三元组 $\langle i, j, N \rightarrow \xi \rangle$，其中：$i$、$j$ 为边所跨越的词的范围，i 为边开始的词位置，j 为边结束的词位置；$N \rightarrow \xi$ 为生成该边所依据的句法规则，句法规则的左部 N 是边的句法类型。

边又可分为完成边（或称为非活动边）和活动边两类。完成边是指这样的边：其规则右部的所有角色均已“完成”，即使用句法类型与角色类型相同的完成边替代这些角色（与一条边的某个角色匹配的边称为其子边）。最基本的完成边为词性边，即直接由一个词的词性所生成的边。完成边形式上可以表示为 $\langle i, j, N \rightarrow \xi \rangle$。活动边是其规则中部分角色尚未完成的边。形式上活动边表示为 $\langle i, j, N \rightarrow \xi^1 . \xi^2 \rangle$，其中 ξ^1 表示规则中已完成的角色序列，ξ^2 表示尚未完成的角色序列。ξ^2 中的第一个角色称为该活动边的活动角色。点表示了活动角色的位置。

Chart 算法分析过程中生成的边存放在边表中。边表是一个二维表，其行列序号表示词位置，其 (i, j) 单元中存放从词位置 i 到 j 上所有边。

Agenda 是一个栈结构。Chart 分析中生成的边首先被推入 Agenda 栈中缓存。Agenda 栈中的边在适当的时机被弹出（不同的 Chart 算法实现中弹出的时机不同），并被用于“规则匹配”和“规则调用”的操作（见以下算法描述中的解释），用以完成已有活动边的活动角色或根据句法规则生成新的边。

Chart 算法有多种不同的实现方式，其中最常用的是从左到右、由底向上的 Chart 算法，描述如下。

输入：一个语句的词性序列，句法规则集；

输出：语句的句法结构。

1）设当前词汇位置 $j = 0$。

2）取当前词汇位置的词，根据其词性生成一条词性边，压入 Agenda 栈中。

3）从 Agenda 栈顶（FIFO）弹出一条完成边 $\langle j, k, N \rightarrow \xi \rangle$，根据其开始位置 j 和结束位置 k，存放在 Chart 表的 (j,k) 单元中。

4）规则匹配操作：将弹出边的类型 N 与 Chart 表中所有活动角色位置为 j 的活动边的活动角色进行匹配。如果活动边形如 $\langle i, j, N^1 \rightarrow \xi^1 . N \rangle$，即该边在 j 位置上的活动角色是其相应规则中的最后一个角色，操作产生的完成边 $\langle i, k, N^1 \rangle$，压入 Agenda 栈中；如果活动边形如 $\langle i, j, N^1 \rightarrow \xi^1 . N N^2 \xi^2 \rangle$，则形成新的活动边 $\langle i, k, N^1 \rightarrow \xi^1 N . N^2 \xi^2 \rangle$，将该边存放在 Chart 表的 (i, k) 单元中。

5）规则调用：使用弹出边对句法规则中所有形如：$N^1 \rightarrow N\xi$ 的规则，生成一条新的边。如果新生成的边是完成边 $\langle j, k, N^1 \rightarrow N \rangle$，压入 Agenda 栈中；如果是活动边 $\langle j, k, N^1 \rightarrow N . \xi \rangle$，则存放到 Chart 表的 (j, k) 单元中。

6）如果 Agenda 非空，转到第3）步；否则，转第7）步。

7）如果当前词汇位置小于语句的最后位置，则当前词汇位置加1，并转到第2）步；否则算法结束，返回 Chart 表 $(0, m)$ 单元中的 S 类型的完成边。其中，m 为语句的最后词位置。

从上述算法可以看出，在由底向上的 Chart 分析过程中，位置 j 上的“规则调用”时仅根据已有完成边和规则生成新的边，而没有考虑这些边在将来是否有用，这样可能产生大量“冗余边”——这些边的类型与所有到该位置的活动边的活动角色都不相同，不可能作为完整句法结构中的一部分，因而被称为是冗余的。这些边所造成的效率损失还不仅仅在于生成这些边，在后续分析中这些边的活动角色同样会被不断地完成，从而进一步生成许多无效边。从左到右、自顶向下的 Chart 算法如下所示。

输入：一个语句的词性序列，句法规则集；

输出：语句的句法结构。

1）设当前词汇位置 $j = 0$，并且根据规则库中所有形式为 $S \rightarrow \xi$ 的规则，生成形式为 $\langle 0, 0, S \rightarrow . \xi \rangle$ 的活动边，压入 Agenda 栈中。根据0位置上的词的词性，生成一条词性边，存入 Chart 表的 $(0, 1)$ 单元中。

2）从 Agenda 栈底顶（FIFO）弹出一条活动边 $\langle i, j, N \rightarrow \xi^1 . N^1 \xi^2 \rangle$，根据其开始位置 j 和结束位置 k，存放在 Chart 表的 (i,j) 单元中；

3）使用弹出的边进行规则匹配操作，即使用 Chart 表中形如 $\langle j, k, N^1 \rightarrow \xi \rangle$ 的完成边来完成该活动边的活动角色 N，生成一条新的边 $\langle i, k, N \rightarrow \xi^1 N^1 . \xi^2 \rangle$ 的边。如果 ξ^2 空，即该边为完成边，则存入 Chart 表的 (i, k) 单元中；否则，压入 Agenda 栈中。

4）使用该弹出的边进行规则调用操作，即对规则中所有形如 $N^1 \rightarrow \xi$ 的规则，生成一条新的活动边 $\langle j, j, N^1 \rightarrow . \xi \rangle$，并压入 Agenda 栈中。

5）如果 Agenda 非空，转到第3）步；如果 Agenda 为空，且当前节点位置小于语句的最后位置，则设置当前词汇位置为 $j + 1$，并取该位置上的词，根据其词性生成一条词性边，存放到 Chart 表的 $(j + 1, j + 2)$ 单元中；否则算法结束，返回 Chart 表 $(0, m)$ 单元中的 S 类型的完成边。其中，m 为语句的最后词位置。

从上述算法可以看出，自顶向下的 Chart 算法不会生成不能作为一个完整语句结构中一部分的边，从而也可以减少部分无效边的生成。但其生成的边中包含大量因没有适当词性支持而不能完成的边。

1.2 基于期待类型的 Chart 算法

从上述关于 Chart 算法的介绍中可以看出，Chart 算法中将产生的大量的冗余边。这些冗余边并不能成为一个在语法上有效完整句法结构的组成部分。减少这些冗余边是提高 Chart 算法效率的关键。

根据1.1节的分析，由底向上的 Chart 算法的优点是能够根据语句中现有的词和词性来生成边。这些边从词的构成来看都是现有语句可以支持的。但其缺点是有些边可能并不是最终合法的句法树所需要的。相反，由顶向下的 Chart 算法的优点是从开始符号 S 开始形成边，这些形成的边都是可能发展为一个完整句法结构中所需要的部分。但其缺点在于有些边在现有语句的词性组成的条件下不可能被最终完成。

由此可见，两个方向的 Chart 分析对其过程中生成边进行了不同的限制：自顶向下的方法可以根据已有活动边的活动角色对句法类型的要求，限制了后续边的生成；由底向上的方法则根据现有的词汇或词性限制了可以生成的边。实践证明，使用一个较大规模的句法规则集时，自顶向下产生的冗余边往往大大超过自底向上产生的冗余边。因而以由底向上的方法为基础，结合自顶向下方法中提供的额外约束，定义了基于期待类型的 Chart 算法，以期综合两者的优势，提高 Chart 算法的效率。

该方法的基本思路是：在从左到右、由底向上的 Chart 算法的规则调用时，只生成那些对完成已有活动边有效的边。所谓“有效”是指：待生成的边的类型是任意一个已有活动边的活动角色所直接或间接需要的。假设已存在一条形如 $\langle i, j, N^1 \rightarrow \xi^1 . N \xi^2 \rangle$ 的活动边，一种情况是：在位置 j 上根据规则

$N \rightarrow \xi$,有一条待生成的形如$\langle j, k, N \rightarrow \xi \rangle$ 的边。由于该待生成边的类型与活动边的活动角色类型都是 N,因而可以作为活动边的子边而被直接需要。另一种情况是:在位置 j 上根据规则 $N^1 \rightarrow \xi$,可能生成一条形如$\langle j, k, N^1 \rightarrow \xi \rangle$ 的边。虽然该边的类型 N^1 与 N 不同,但由于存在这样的规则:$N \rightarrow N^1\ \xi$,它使该待生成的边有希望在将来成为类型为 ξ 的边的一个组成部分,进而被用于完成活动边的活动角色 ξ,因而该边被活动边间接需要。如果以上两种情况都不能满足,那么待生成的边在将来不会有任何用途。通过避免生成这样的边,可以降低冗余边的数量,从而提高句法分析的效率。

基于以上思路,定义期待类型并提出基于期待类型的 Chart 算法。

定义 1　期待类型。对于一条活动边 $\langle i, j, N^1 \rightarrow \xi^1 . N^2\xi^2 \rangle$,其中活动角色 N^2 为 j 位置上的一个期待类型。j 位置上的所有期待类型形成一个集合,称为 j 位置的期待类型集。

定义 2　预测期待类型。如果 N 是位置 j 上的一个期待类型,而且 N 是非终结符,那么对于规则集中所有形如:$N \rightarrow N^E\xi$ 的规则,将 N^E 添加到 j 位置上的期待类型集中,这种操作称为在 j 位置上预测 N 的期待类型。

定义 2 中预测期待类型的操作需要递归进行,即当 N^E 被添加到期待类型集中之后,如果 N^E 不是一个终结符,则还需要进一步预测 N^E 的期待类型。在 j 位置上进行规则调用生成边之前,需要检查该边的类型是否出现在期待类型表中。如果没有,则不生成该边。

很显然,要保证在上述期待类型检查时不会遗漏有用的边,就必须保证 j 位置上的期待类型表是完备的。或者说,在分析 j 位置之后的边时,j 位置之前的所有活动边必须已全部分析出来,从而保证初始期待类型的完备性。这就要求算法必须严格按照从左到右的顺序执行。因而,基于期待类型的 Chart 算法是通过修改从左到右、由底向上的 Chart 算法得到的。具体算法介绍如下。

输入:一个语句的词性序列,句法规则集;

输出:语句的句法结构。

1) 设当前词汇位置 $k = 0$,并且将 S 作为 k 位置上的初始期待类型。

2) 对当前词汇位置 k 上所有期待类型进行预测,产生 k 位置上的期待类型集。

3)取当前词汇位置的词,根据其词性生成一条词性边,压入 Agenda 栈中。

4)从 Agenda 栈顶弹出一条边 $\langle j, k, N \rightarrow \xi \rangle$,根据其开始位置 j 和结束位置 k,存放在 Chart 表的 (j,k) 单元中。

5) 使用弹出的边,进行规则匹配操作,即将该边的类型 N 与到 j 位置的所有活动边的活动角色进行匹配。如果活动边形如$\langle i, j, N^1 \rightarrow \xi^1 . N \rangle$,即该边在 j 位置上的活动角色是其相应规则中的最后一个角色,操作产生的完成边$\langle i, k, N^1 \rangle$,压入 Agenda 栈中;如果活动边形如 $\langle i, j, N^1 \rightarrow \xi^1 . N\ \ N^2\xi^2 \rangle$,则形成新的活动边$\langle i, k, N^1 \rightarrow \xi^1\ N . N^2\xi^2 \rangle$,将该边存放在 Chart 表的 (i, k) 单元中,并将该边的活动角色的类型 N^2 存放到 k 位置上的期待类型表中。

6)使用该弹出的边进行规则调用操作,即对规则中所有形如:$N^3 \rightarrow N\,\xi$ 的规则,如果 N^3 不是 j 位置上的期待类型,则该规则被过滤;否则生成一条新的边。如果新生成的边是完成边$\langle j, k, N^3 \rightarrow N \rangle$,压入 Agenda 栈中;如果是活动边 $\langle j, k, N^3 \rightarrow N . N^4\xi \rangle$,则将 N^4 添加到位置 k 的期待类型表中。

7)如果 Agenda 非空,转到第 4)步;否则,转到第 8)步。

8)如果当前节点位置小于语句的最后位置,则当前此外位置加 1,并转到第 2)步;否则算法结束。

上述算法对标准的从左到右、由底向上的 Chart 算法进行了修改。在原算法的基础上增加了第 2)步操作,其作用是在 k 位置上进行句法分析前,首先根据 k 位置上的初始期待类型预测所有的期待类型;算法第 5)步对 Chart 算法的标准规则匹配操作进行了修改:将规则匹配中产生的活动边的活动角色类型添加到该边结束位置上的期待类型表中;算法第 6)步对 Chart 算法的标准规则调用操作进行了修改:过滤掉不符合该位置上期待类型要求的规则,从而减少了边的生成。

算法中预测期待类型的操作所增加的时间和空间复杂性都很低。设规则库中规则数量为 N,句子词长为 M,预测期待类型的时间复杂度为 $O(NM)$。设句法规则中的非终结符数量为 K,由于算法严格按从左到右的顺序执行,因而只需要保存一个 K 维的数组,其中各单元一个字节,以 0/1 表示相应句法类型是否被期待。这样,其空间复杂度为 $O(K)$。

2　实验分析

根据以上提出的算法,我们使用文献[5]作为实验语料进行了对比实验。TCT 树库共计包含 16 608 个已分析为句法树的语句,采用其中 60% 的语句作为训练语料提取句法规则,所得到的规则库共包含 4 632 条规则(从树库中提取规则的方法参见文献[6])。其余 40% 共计 6 643 个语句作为测试语料,首先还原为词性标注串,然后对这些词性标注串进行句法分析。实验对采用预测期待类型的 Chart 算法和普通从左到右、由底向上的 Chart 算法进行了速度和成功率的比较。在速度方面,普通 Chart 算法共耗时约 87 s,而采用预测期待类型的算法共耗时约 65 s,速度上提高了约 24%。在成功率方面,普通 Chart 算法因边池溢出而导致 113 个语句分析失败,而采用预测期待类型的算法的失败语句共有 47 句,失败的句子数量降低一半以上。这些实验结果证明了文本提出的算法对句法分析的效率和成功率具有比较明显的效果。

3　结语

句法分析中存在的大量冗余边是造成其效率低下的最主要的原因。本文提出的算法在一定程度上减少了冗余边的数量,在实验中较为明显地提高了句法分析的效率。但从根本上看,该算法中仍然存在相当数量的冗余边,需要采取更多的措施和标准来进一步过滤。该算法可以作为提高句法分析器效率的手段之一,与其他提高效率的方法(如文献[1]提出的角色反演算法)结合起来,以最大限度地提高句法分析的效率。

参考文献:

[1] 周强, 黄昌宁. 基于局部优先的汉语句法分析方法[J]. 软件学报, 1999, 10(1): 1 - 6.

[2] 白硕, 张浩. 角色反演算法[J]. 软件学报, 2003, 14(3): 328 - 333.

[3] 张玉艳, 杨潇, 黄国栋, 等. 基于图算法的二元组合文法分析[J]. 计算机应用, 2008, 28(7): 1668 - 1671.

[4] 李幸, 宗成庆. 引入标点处理的层次化汉语长句句法分析方法[J]. 中文信息学报, 2006, 20(4): 8 - 15.

[5] 毛奇, 连乐新, 周文翠, 等. 基于标点符号分割的汉语句法分析算法[J]. 中文信息学报, 2007, 21(2): 29 - 34.

[6] 周强, 张伟, 俞士汶. 汉语树库的构建[J]. 中文信息学报, 1997, 11(4): 42 - 51.

[7] 周强, 黄昌宁. 汉语概率型上下文无关语法的自动推导[J]. 计算机学报, 1998, 21(5), 385 - 392.