基于柱状搜索的高阶依存句法分析[1]

李正华，车万翔，刘挺

哈工大计算机学院信息检索研究室 哈尔滨 150001

E-mail: {zhli, car, tliu}@ir.hit.edu.cn

摘 要：本文提出使用所有的孙子节点构成祖孙特征的高阶依存模型，并且使用柱状搜索策略限制搜索空间，最终找到近似最优依存树。另外，我们以较小的时间复杂度为代价，使用了丰富的依存关系特征，并且允许模型在解码的过程中进行依存关系选择。我们参加了 CoNLL 2009 年多语依存句法分析和语义角色标注国际评测，最终获得联合任务总成绩第一名，依存句法分析总成绩第三名。

关键词：柱状搜索；高阶特征；依存分析

Beam-search based High-Order Dependency Parser

Zhenghua Li, Wanxiang Che, Ting Liu

Information Retrieval Laboratory of Computer Science & Technology School, Harbin Institute of Technology, Harbin 150001

E-mail: {zhli, car, tliu}@ir.hit.edu.cn

Abstract: We propose a high-order parsing model which uses all grandchildren nodes to compose high-order features, and constraint the searching space basing on beam-search strategy, and find the approximately optimal dependency tree. In addition, We explore rich dependency label features and allow multiple relations for one arc during decoding. We attended CoNLL 2009 international evaluation task of multilingual syntactic and semantic dependency parsing and ranked the first in the joint task, and ranked the third in the syntactic parsing task.

Key words: Beam-search; High-order Model; Dependency Parsing

1 引言

依存分析是指给定句子 $x = w_1 w_2 ... w_n$，遵循某种依存语法体系，给出句子对应的依存树 y。依存句法相对于短语结构句法而言，其优点在于：（1）形式简洁，不增加额外的非终结标记，易于理解。（2）侧重于反映语义关系，可以很容易的和语义分析结合。（3）适合表示交叉关系，因此适用于大多数语言。（4）有利于实现线性时间的搜索算法。因此，依存句法分析受到国内外学者广泛的关注[1]。CoNLL 2006、2007 年连续两年评测多语依存分析任务。CoNLL 2008 评测英语依存语义分析任务。CoNLL 2009 评测多语依存语义分析任务。这些评测任务的开展，也促进了依存分析的发展。

本文内容组织为：第 2 部分介绍依存分析相关工作；第 3 部分介绍本文采用的基于柱状搜索的高阶依存分析方法；第四部分为实验；第 5 部分为结论及进一步工作。

2 相关工作

目前的依存分析的主流方法有两种，第一种是基于转移的方法，第二种是基于图的方法。

[1] **基金项目：**国家自然科学基金项目（60803093；60675034）；国家 863 项目（2008AA01Z144）

基于转移的方法将依存树的构建分解为一个动作序列，由分类器根据当前状态来决定下一个动作。Covington[2], Yamada[3], Nivre[4]等人采用了这种方法。

基于图的方法将依存分析看成有向图中最大生成树求解问题。对于输入的句子 $x = w_1 w_2 ... w_n$，可以构建一个有向图 $G = (V, E)$。$V = \{0,1,...,n\}$ 为节点集合，对应每一个词，0为增加的哑节点，用以表示依存树的根节点。$E = \{(i, j, l) \mid 0 \le i \le n, 1 \le j \le n, l \in L\}$ 为有向边集合。其中 L 表示依存关系集合。(i, j, l) 表示一条从 i 指向 j 的有向边，依存关系为 l。有向图 G 中每两个节点之间可以有多个同方向的但是不同依存关系的有向边。依存分析的目标是从图 G 中找到一颗权值最大的依存树。Eisner 设计出了基于 span 的算法，使得上述搜索过程可以在 $O(n^3)$ 时间内完成[5]。McDonald 提出了一阶依存分析模型，假设依存树中的弧相互独立，依存树的分值为所有弧权重的累加[6]。然后，McDonald 又将一阶模型扩展为二阶模型，假设依存树中一条弧的权值与它相邻的兄弟弧（核心节点与其前一个儿子节点构成的弧）相关[7]。Carreras 扩展了二阶模型，提出高阶模型。高阶模型假设一条弧的权值与其最左和最右的孙子弧（非核心节点与其儿子节点构成的弧）相关[8]。

3 基于图的依存分析方法

3.1 依存分析模型

我们扩展了 Carreras 的高阶模型，假设一条弧的权值与其所有的孙子弧相关。模型如公式(1)所示。

$$S(x, y) = \sum_{(h,c,l) \in y} S_{single}(h, c, l) + \sum_{\substack{(h,c_i) \in y \\ (h,c_{i+1}) \in y}} S_{sibling}(h, c_i, c_{i+1}) + \sum_{\substack{(h,c,l) \in y \\ (c,gc) \in y}} S_{grand}(h, c, gc, l) \tag{1}$$

$S(x,y)$ 表示输入句子 x 对应的依存树 y 的得分（分值），由三部分构成：每一条依存弧的权重，兄弟节点互相影响的权重，以及祖孙节点互相影响的权重。第二部分中，c_i 和 c_{i+1} 是相邻的兄弟节点，注意我们在计算兄弟之间互相影响的权重时，没有考虑依存关系。

3.2 解码算法

3.2.1 依存关系标注策略

McDonald 的方法在解码之前，利用一元特征为任意一条弧确定了唯一的依存关系，并且在解码过程中，直接使用一元特征对应的分值。这种做法虽然比较简单的解决了依存关系标注的问题，但是由于没有利用丰富的依存关系特征，因此很大程度上影响了依存关系准确率。为此，我们首先利用一元特征，为每一条弧确定 K_1 个可能的依存关系；然后利用二元特征，对这 K_1 个依存关系进行重排序，得到 K_2 个可能的依存关系，进一步缩小依存关系的数量（$K_2 \ll K_1 < |L|$）。L 表示训练语料中出现的依存关系类型集合。最后，在解码过程中针对每条弧我们仅使用选出的 K_2 个依存关系。整个过程如公式(2~3)所示。整个过程的时间复杂度为 $O(n^2 |L| \log K_1 + n^2 K_1 \log K_2)$。

$$L_1(h, c) = \arg\max_{l \in L}^{K_1} \left(\mathbf{w} \cdot \mathbf{f}_{uni}(h, c, l) \right) \tag{2}$$

$$L_2(h, c) = \arg\max_{l \in L_1(h,c)}^{K_2} \left(\mathbf{w} \cdot \left\{ \mathbf{f}_{uni}(h, c, l) \bigcup \mathbf{f}_{bi}(h, c, l) \right\} \right) \tag{3}$$

3.2.2 基于 span 的基本操作及分值计算

在 Eisner 提出基于 span 的算法之前，基于图的依存分析一般以组块（constituent）为基本单位。组块表示输入句子的一个片断对应的子树，即包括一个核心词和这个核心词的子孙节点。组

块中核心词的位置是任意的，可以是片段中的任意一个词。组块中除核心词外的所有节点均已找到它们的子孙节点。Eisner 算法以 span 作为解码的基本单位。Span 也表示输入句子的一个片断对应的子树。与组块不同的是，span 中的核心词必须位于片段首或尾，即 span 只包括了这个核心词的左边或右边的子孙节点。另外，除核心词外的另外一个片段首或尾词的修饰成分也可以是不完整的，即 span 没有包括这个词的左边的子孙节点或者右边的子孙节点。对于其他词，span 包括了它们所有的子孙节点。Span 的这种特性使得解码算法独立的确定一个词左边的修饰成分和右边的修饰成分，从而降低算法的复杂度。

Span 可以分为两种，完整 span 和不完整 span。完整 span 中除了核心词外其它词的所有修饰成分全部找到，使用$\leftarrow$或$\rightarrow$表示。不完整 span 除了核心词外，另外一个片段首或者尾词的修饰成分也没有全部找到，使用$|\leftarrow$或$\rightarrow|$表示。图 1 中包含了三个 span，分别记作 $sp_{s\rightarrow r}$， $sp_{r+1\leftarrow t}$ 和 $sp_{s\rightarrow|t}$。其中 $sp_{s\rightarrow r}$ 和 $sp_{r+1\leftarrow t}$ 表示了两个完整 span；而 $sp_{s\rightarrow|t}$ 表示一个不完整 span。$sp_{s\rightarrow r}$ 代表了以 w_s 为核心词的一颗子树，其他词 $w_{s+1,r}$ 都是 w_s 的右修饰成分，并且 $sp_{s\rightarrow r}$ 包括了 $w_{s+1,r}$ 完整的修饰成分。$sp_{r+1\leftarrow t}$ 则表示了以 w_t 为核心词的一颗子树。$sp_{s\rightarrow|t}$ 包括了 w_s 的右子孙节点，但是不包括 w_t 的右子孙节点。

我们采用柱状搜索的方法，扩展了 Carreras 针对高阶模型设计的解码算法。解码算法中包括两类操作。第一是将两个完整 span 合并为一个不完整 span，如图 1 所示。第二类操作将一个完整 span 和一个不完整 span 合并为一个完整 span，如图 2 所示。图 1 和图 2 中的操作得到新的 span 都是以最左边的词作为核心节点。得到以最右边词作为核心节点的 span 的操作是类似的。

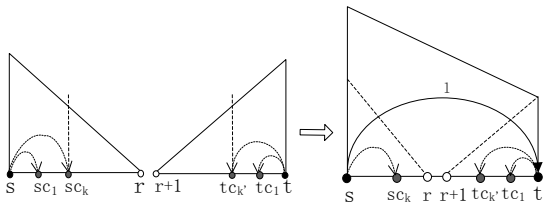

图 1 两个完整 span 合并为一个不完整 span

图 1 中将 $sp_{s\rightarrow r}$ 和 $sp_{r+1\leftarrow t}$ 合并为 $sp_{s\rightarrow|t}$。可以看到，$sp_{s\rightarrow|t}$ 除了包含 $sp_{s\rightarrow r}$ 和 $sp_{r+1\leftarrow t}$ 中的所有弧外，还增加了一条弧 $arc^{l}_{s\rightarrow t}$，即 w_s 为核心词，w_t 为修饰词，并且依存关系为 l。$sp_{s\rightarrow|t}$ 的分值如公式(4)所示，由 $sp_{s\rightarrow r}$ 的分值，$sp_{r+1\leftarrow t}$ 的分值以及增加弧 $arc^{l}_{s\rightarrow t}$ 的分值累加得到。其中，第三部分分值由增加弧引入的特征向量与特征权值向量点积得到，如公式(5)所示。特征向量由四部分构成，分别为一元特征，二元特征，兄弟特征和祖孙特征，如公式(6)所示。兄弟特征只考虑相邻的兄弟节点。祖孙特征考虑 t 所有左侧的儿子节点。Carreras 的高阶模型中仅考虑 t 最远的儿子节点，即 $tc_{k'}$。

$$S(sp_{s\rightarrow|t}) = S(sp_{s\rightarrow r}) + S(sp_{r+1\leftarrow t}) + S_{icp}(s,r,t,l) \tag{4}$$

$$S_{icp}(s,r,t,l) = \mathbf{w}\cdot\mathbf{f}_{icp}(s,r,t,l) \tag{5}$$

$$\mathbf{f}_{icp}(s,r,t,l) = \mathbf{f}_{uni}(s,t,l)\cup\mathbf{f}_{bi}(s,t,l)\cup\mathbf{f}_{sib}(s,sc_k,t)\cup\left\{\bigcup_{i=1}^{k'}\mathbf{f}_{grd}(s,t,tc_i,l)\right\} \tag{6}$$

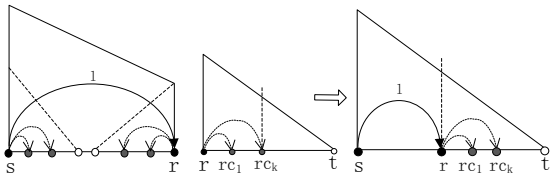

图 2 一个完整 span 和一个不完整 span 合并为一个完整 span

图 2 中 $sp_{s\to|r}$ 和 $sp_{r\to t}$ 合并为 $sp_{s\to t}$ 。这个操作不会增加弧，但是 $sp_{s\to t}$ 的分值仍由三部分构成。第三部分包含了由 r 的右侧儿子节点对应的祖孙特征贡献的分值，如公式(7~9)所示。同样，我们包括了所有 t 右侧儿子节点对应的祖孙特征，而 Carreras 的高阶模型中仅考虑 t 最远的儿子节点，即 tc_k 。

$$S(sp_{s\to t}) = S(sp_{s\to|r}) + S(sp_{r\to t}) + S_{cp}(s,r,t,l) \tag{7}$$

$$S_{cp}(s,r,t,l) = \mathbf{w}\cdot\mathbf{f}_{cp}(s,r,t,l) \tag{8}$$

$$\mathbf{f}_{cp}(s,r,t,l) = \bigcup_{i=1}^{k}\mathbf{f}_{grd}(s,r,rc_i,l) \tag{9}$$

3.2.3 基于柱状搜索的解码算法

由于高阶特征的引入，Eisner 算法不再具有动态规划算法要求的最优子结构特性。如图 2 中的合并操作，$sp_{s\to t}$ 的分值最大，并不意味着 $sp_{s\to|r}$ 和 $sp_{r\to t}$ 都分值最大，还取决于第三部分分值。而第三部分分值又和 $sp_{s\to|r}$ 与 $sp_{r\to t}$ 中包含的弧和依存关系相关。我们采用柱状搜索的方法来获得近似解。算法如图 3 所示。

```
1 initialization: C[s][s][c] = 0.0 ∀0 ≤ s ≤ n, c ∈ {cp, icp} # cp:complete; icp:incomplete
2 for j = 1..n
3   for s = 0..n
4       t = s + j
5       if t > n then break
6       C[s][t][icp] = max^K_{s≤r<t;1≤k1,k2≤K;l∈L2(s,t)} (C_{k1}[s][r][cp] + C_{k2}[t][r+1][cp] + S_{icp}(s,r,t,l))
7       C[t][s][icp] = max^K_{s≤r<t;1≤k1,k2≤K;l∈L2(t,s)} (C_{k1}[s][r][cp] + C_{k2}[t][r+1][cp] + S_{icp}(t,r,s,l))
8       C[s][t][cp] = max^K_{s<r≤t;1≤k1,k2≤K} (C_{k1}[s][r][icp] + C_{k2}[r][t][cp] + S_{cp}(s,r,t,l = C_{k1}[s][r][icp].lbl))
9       C[t][s][cp] = max^K_{s≤r<t;1≤k1,k2≤K} (C_{k1}[r][s][cp] + C_{k2}[t][r][icp] + S_{cp}(t,r,s,l = C_{k2}[t][r][icp].lbl))
10  end for
13 end for
```

图 3 基于柱状搜索的解码算法

我们在线图（chart）的每一个位置保留 K 个最优的 span，如 $C[s][t][icp]$ 保存了 $sp_{s\to|t}$ 类型的 K 个最优的 span。算法第 6 行的含义是，遍历 $s\le r<t$,合并 $C[s][r][cp]$ 和 $C[t][r+1][cp]$ 从而得到 $C[s][t][icp]$ 。由于 $C[s][r][cp]$ 和 $C[t][r+1][cp]$ 都包含 K 个 span，因此通过 k_1 和 k_2 进行两两组合。又由于在解码前根据一元特征和二元特征给确定了依存关系集合 $L_2(s,t)$ ，因此还需要尝试所有可能的依存关系。最终，我们将所有的组合根据分值得到 K 个最优的 span 作为 $C[s][t][icp]$ 的结果。

算法第 6 行和第 7 行的时间复杂度均为 $O(n^2K_2K^2\log K)$ ，其中 $K_2=|L_2|$ 。注意由于我们在获

取祖孙特征的时候，使用了所有的孙子节点，因此复杂度为$O(n)$。第 8 行和第 9 行复杂度为$O(n^2K^2\log K)$。因此整个算法复杂度为$O(n^4K_2K^2\log K)$。由于K和K_2一般很小且为常数，因此可以认为整个算法复杂度是$O(n^4)$。

4 实验

在 CoNLL 2009 多语语义依存分析评测任务中，我们采取将依存分析和语义角色标注级联的策略。CoNLL 2009 评测任务包括汉语、英语、日语、捷克语、德语、西班牙语、加泰罗尼亚语七种语言。其中英语、德语、捷克语还提供了跨领域（ood, out of domain）的测试语料，用以评价系统对不同领域语料的适应能力。

4.1 含有交叉弧语言的处理

CoNLL 2009 评测包含的七种语言中，捷克语、德语、英语都不同程度的含有交叉弧的依存树。我们的模型和算法无法处理含有交叉弧的情况。对于这三种语言，我们使用了 Nivre 的 pseudo-projective 方法[9]，先将含有交叉弧的句子转化为不含有交叉弧，用以训练我们的依存分析器。最后，我们将分析结果逆向转化，恢复为含有交叉弧的依存树。

4.2 特征集合

我们借鉴 McDonald 采用的一阶和二阶特征，以及 Carreras 采用的高阶特征，对我们去年参加 CoNLL 2008 评测任务的系统使用的特征[10]进行了扩充。

4.3 实验结果及分析

我们使用的机器配置是 2.5GHz Xeon CPU, 16G 内存。依存分析器内存使用峰值为 15G，训练时间最多约 60 个小时。由于 CoNLL 2009 年评测涉及语言比较多，并且我们的系统对内存等资源要求很高，我们没有足够的时间进行特征的选择和参数优化。训练模型过程中，针对不同语言，我们的参数设置为：依存关系数量$K_1=10, K_2=2$，最优 span 数量$K=10$，迭代次数$T=5\sim10$。测试阶段参数设置为：$K_1=20\sim50, K_2=2\sim3$，$K=10$。

我们的依存分析器在 CoNLL 2009 七种语言上的结果如表 1 所示。其中，LAS 表示依存关系准确率，UAS 表示依存弧准确率。在这次评测中，我们联合任务总成绩排名第一，依存分析总成绩排名第三，表 2 中给出了我们的系统和前两名系统的性能比较。其中带*表示对应项在整个评测中为第一名。在跨领域语料上测试时，依存分析性能下降比较大（4%~10%），因此如何增强依存分析模型领域适应性是一个很有意义的研究点。

表 1 CoNLL 2009 七种语言实验结果

	Labeled Accuracy Score (LAS)			Unlabeled Accuracy Score (UAS)		
	devel	test	ood	devel	test	ood
Catalan	86.65	86.56	—	90.31	90.33	—
Chinese	75.73	75.49	—	80.38	80.27	—
Czech	80.07	80.01	76.03	87.39	87.26	83.28
English	87.04	88.48	81.57	90.93	91.58	87.33
German	85.69	86.19	76.11	88.85	89.00	81.10
Japanese	92.55	92.57	—	93.55	93.37	—
Spanish	87.22	87.33	—	90.52	90.77	—

表 2 CoNLL 2009 评测依存分析依存关系准确率(LAS)比较

	Ca	Ch	Cz	Cz-ood	En	En-ood	Ge	Ge-ood	Ja	Sp
1	87.86*	76.11	80.38*	76.41*	88.79	80.84	87.29	76.77	92.34	87.64*
2	86.35	76.51	80.11	76.40	89.88*	82.64*	87.48*	77.34*	92.21	87.19
3**(ours)**	86.56 (-1.3)	75.49 (-0.6)	80.01 (-0.3)	76.03 (-0.4)	88.48 (-1.4)	81.57 (-1.1)	86.19 (-1.3)	76.11 (-1.2)	92.57* (—)	87.33 (-1.3)

5 结论及下一步工作

本文提出了一种基于柱状搜索，融合高阶特征的依存分析方法。这种高阶依存模型使用所有的孙子节点构成祖孙特征，扩展了 Carreras 的只是利用最左或者最右孙子节点的模型。由于融合高阶模型的搜索过程不再符合动态规划最优子结构原则，我们设计了柱状搜索策略限制搜索空间，最终找到近似最优解。另外，不同于以前的依存关系策略，我们以较小的时间复杂度为代价，使用了丰富的依存关系特征，并且允许模型在解码的过程中进行依存关系选择。我们参加了 CoNLL 2009 年多语语义依存分析国际评测，语义分析和句法分析联合任务总成绩第一名，依存分析总成绩第三名。

下一步我们需要进行特征选择、参数优化，并且和 Carreras 的模型进行实验对比。

参 考 文 献

[1] 李正华. 依存句法分析统计模型及树库转化研究. 硕士毕业论文, 哈尔滨工业大学. 2008.

[2] Michael A. Covington. 2001. A fundamental algorithm for dependency parsing. In Proceedings of the 39th Annual ACM Southeast Conference.

[3] Hiroyasu Yamada, Yuji Matsumoto. Statistical dependency analysis with support vector machines. In Proceedings of 8th International Workshop on Parsing Technologies. 2003:195-206.

[4] Joakim Nivre, Mario Scholz. Deterministic Dependency Parsing of English Text. In Proceedings of COLING. 2004:64~70.

[5] Jason Eisner. Bilexical grammars and a cubic-time probabilistic parser. Proceedings of the International Workshop on Parsing Technologies, MIT. 1997: 54-65

[6] Ryan McDonald, Koby Crammer and Fernando Pereira. Online Large-Margin Training of Dependency Parsers. Association for Computational Linguistics (ACL). 2005.

[7] Ryan McDonald and Fernando Pereira. Online Learning of Approximate Dependency Parsing Algorithms. European Association for Computational Linguistics (EACL). 2006.

[8] Xavier Carreras. Experiments with a high-order projective dependency parser. In Proceedings of the CoNLL 2007 Shared Task Session of EMNLP-CoNLL. 2007:957-961

[9] Joakim Nivre and J. Nilsson. Pseudo-Projective Dependency Parsing. Proc. of the 43rd Annual Meeting of the ACL. 2005: 99-106

[10] Wanxiang Che, Zhenghua Li, Yuxuan Hu, Yongqiang Li, Bing Qin, Ting Liu, Sheng Li. A Cascaded Syntactic and Semantic Dependency Parsing System. CoNLL 2008: Proceedings of the 12th Conference on Computational Natural Language Learning, pages 238–242