

基于高阶词汇依存的短语结构树重排序模型^{*}

王志国⁺, 宗成庆

(模式识别国家重点实验室(中国科学院 自动化研究所), 北京 100190)

Phrase Parses Reranking Based on Higher-Order Lexical Dependencies

WANG Zhi-Guo⁺, ZONG Cheng-Qing

(National Laboratory of Pattern Recognition (Institute of Automation, The Chinese Academy of Sciences), Beijing 100190, China)

+ Corresponding author: E-mail: zgwang@nlpr.ia.ac.cn

Wang ZG, Zong CQ. Phrase parses reranking based on higher-order lexical dependencies. *Journal of Software*, 2012, 23(10): 2628-2642 (in Chinese). <http://www.jos.org.cn/1000-9825/4192.htm>

Abstract: The existing works on parsing show that lexical dependencies are helpful for phrase tree parsing. However, only first-order lexical dependencies have been employed and investigated in previous research. This paper proposes a novel method for employing higher-order lexical dependencies for phrase tree evaluation. The method is based on a parse reranking framework, which provides a constrained search space (via N -best lists or parse forests) and enables the parser to employ relatively complicated lexical dependency features. The models are evaluated on the UPenn Chinese Treebank. The highest $F1$ score reaches 85.74% and has outperformed all previously reported state-of-the-art systems. The dependency accuracy of phrase trees generated by the parser has been significantly improved as well.

Key words: phrase structure; dependency structure; parse reranking; higher-order lexical dependencies; parse forest

摘要: 在句法分析中,已有研究工作表明,词汇依存信息对短语结构句法分析是有帮助的,但是已有的研究工作都仅局限于使用一阶的词汇依存信息.提出了一种使用高阶词汇依存信息对短语结构树进行重排序的模型,该模型首先为输入句子生成有约束的搜索空间(例如, N -best 句法分析树列表或者句法分析森林),然后在约束空间内获取高阶词汇依存特征,并利用这些特征对短语结构候选树进行重排序,最终选择出最优短语结构分析树.在宾州中文树库上的实验结果表明,该模型的最高 $F1$ 值达到了 85.74%,超过了目前在宾州中文树库上的最好结果.另外,在短语结构分析树的基础上生成的依存结构树的准确率也有了大幅提升.

关键词: 短语结构;依存结构;句法重排序;高阶词汇依存关系;句法森林

中图法分类号: TP391 文献标识码: A

句法分析是自然语言处理中的关键技术之一,其基本任务是确定自然语言句子的句法结构,为下一步的应用(机器翻译、信息抽取等)提供技术支持.近些年来,随着计算机性能的不不断提高以及大规模句法树库的建立,基于统计的句法分析技术已成为句法分析领域的主流方法.短语结构句法分析中最为常用的方法是基于概率

* 基金项目: 国家自然科学基金(60975053, 61003160); 中国科学院对外合作交流项目

收稿时间: 2011-05-13; 修改时间: 2011-09-02; 定稿时间: 2012-02-15

上下文无关文法(probabilistic context-free grammar,简称 CFG)的分析方法^[1],该方法使用上下文无关文法(CFG)对自然语言句子进行句法分析,每条 CFG 规则都赋予了相应的概率,最终,句法树的概率就是其中包含的所有规则的概率乘积.但是,文献[2]中指出,直接由句法树库评估得到的概率上下文无关文法并不能很好地对自然语言句子进行句法分析.究其原因是,PCFG 存在着两点主要缺陷^[3]:(1) 过于强烈的独立性假设使得规则之间缺乏结构依赖关系;(2) 没有对词汇建模,致使模型对词汇信息不敏感.针对 PCFG 的这些缺陷,过去的十几年中,研究人员提出了大量的针对 PCFG 的改进模型^[2,4-6].其中,词汇化的概率上下文无关文法(lexicalized PCFG,简称 LPCFG)^[7-9]是针对 PCFG 缺乏词汇信息这一缺陷而提出的代表性方法.在 LPCFG 中,所有的非终结符都用其中心词进行标注,每条 CFG 规则的概率都依据中心词进行估计.中心词信息的引入,使得 LPCFG 模型对词汇变得敏感,所以句法分析的准确率得以提升.但是,LPCFG 并没有改善 PCFG 缺乏结构依赖关系的缺陷,因此,为了进一步提高短语结构句法分析器的性能,仍需寻求其他信息.依存结构(dependency structure)是一个很好的选择,它描述了词与词之间的依赖关系,能够为 PCFG 带来结构依赖和词汇依赖关系.例如,给定短语“高科技项目”,其正确的短语结构分析树如图 1(a)所示,相应的依存树如图 1(c)所示.由 LPCFG 自动分析(使用 Charniak parser^[10])得到的短语结构树如图 1(b)所示,相应的依存树如图 1(d)所示.可以看出,LPCFG 未能给出正确的句法结构.此时,如果考虑依存关系,“高”应该依存到“科技”,而不应该依存到“项目”,这样就不会产生图 1(b)中错误的句法树了.

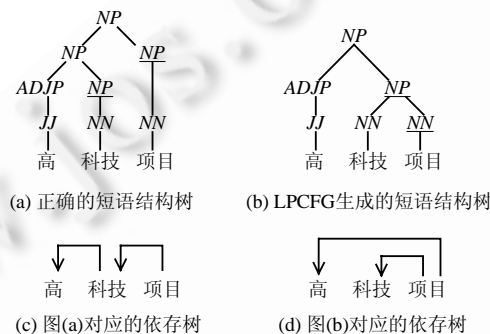


Fig.1 Phrase structure trees and their corresponding dependency trees

图 1 短语结构树及其对应的依存树

依存结构一般会被分割为一系列词汇依存结构(lexical dependency parts)的集合,每个词汇依存结构根据其中包含的依存边的个数来定义阶数.例如在图 2 中,dependency 是一阶词汇依存结构,sibling 和 grandchild 是二阶词汇依存结构,grand-sibling 和 tri-sibling 是三阶词汇依存结构.一般地,将依存边的个数大于 1 的词汇依存结构称为高阶词汇依存结构,例如图 2 中所示的 sibling,grandchild,grand-sibling 等.近些年来,高阶词汇依存信息已被成功地应用到了依存结构句法分析中^[11-13],但在短语结构句法分析中只应用了一阶词汇依存结构^[14-16].一阶词汇依存结构携带的词汇依存信息非常有限,而且丢掉了整个依存结构中的上下文信息.因此,为了提高短语结构句法分析器的性能,本文提出了将高阶的词汇依存信息考虑到短语结构句法分析过程中,利用高阶词汇依存信息对短语结构树进行评估.我们首先为输入句子生成有约束的搜索空间(例如 N-best 句法树列表或者句法分析森林),然后在约束空间内获取高阶词汇依存特征,并利用这些特征对短语结构候选树进行评估,最终选择出最优的短语结构分析树.在宾州中文树库上的实验结果表明,该模型的最高 F1 值达到了 85.74%,超过了目前在宾州中文树库上的最好结果.另外,在短语结构分析树的基础上生成的依存结构树的准确率也有了大幅提升.这表明,高阶词汇依存信息对于短语结构句法分析有很大的帮助.

本文第 1 节简单回顾国内外相关的研究工作.第 2 节描述利用高阶词汇依存信息进行短语结构树评估的模型.第 3 节描述利用高阶词汇依存信息进行重排序的方法,包括 N-best 重排序和基于森林的重排序方法.第 4 节对参数训练的方法进行讨论.第 5 节给出实验及结果分析.最后,第 6 节对全文工作进行总结和展望.

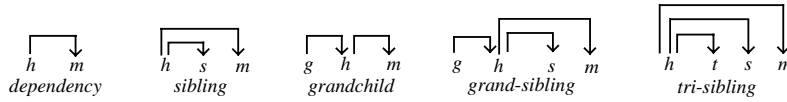


Fig.2 Lexical dependency types

图2 词汇依存结构类型

1 相关工作

句法重排序(*reranking*)模型是句法分析研究中的一种常用模型,该模型分为两个阶段:第1阶段利用一个生成式的句法分析器为输入句子生成一系列质量较高的候选句法树,第2阶段利用一个判别式模型并融入大量的特征对候选句法树进行重新排序.根据第1阶段生成的候选句法树形式的不同,句法重排序模型主要分为两类方法:

- 第1类方法称为 *N-best* 重排序^[15,17],该方法在第1阶段生成一个 *N-best* 候选句法树列表,其优点是在第2阶段引入任意特征,易于扩展;但其缺点是重排序空间受到 *N-best* 列表的限制,有些质量较好的候选可能被过滤掉;
- 针对这一缺点,文献[18]提出了另外一种重排序方法——基于森林的重排序.该方法首先为输入句子生成一个句法森林(不再是 *N-best* 列表),然后自底向上地为每个节点生成 *N-best* 子树,最后将根节点上最好的句法树作为最终的句法分析树输出.

近年来,已有一些利用依存结构信息进行短语结构句法分析的工作.总体上可以将这些工作分为如下3类:第1类方法称为依存驱动(*dependency-driven*)的短语结构句法分析方法^[19,20],该方法首先为输入句子(利用依存句法分析器)生成一个有标签的依存关系树(带有复杂的依存标签,方便从依存结构还原到短语结构),然后将得到的依存树转换为对应的短语结构树;第2类方法称为依存约束(*dependency-constrained*)的短语结构句法分析方法^[21-24],该方法首先为输入句子生成一个无标签的依存结构,然后利用该依存结构对短语结构句法分析的搜索空间进行剪枝,最后在剪枝后的搜索空间内进行短语结构句法分析;第3类方法称为基于词汇依存(*dependency-based*)的短语结构句法分析方法^[14,16],该方法利用依存结构中的词汇依存信息对短语结构树进行评估,进而选择出最优的短语结构分析树.

前面两类方法都是将依存结构作为硬约束(*hard constraint*)的方式来指导短语结构的分析,一旦依存结构存在错误,对短语结构分析的影响将无法挽回;而第3类方法将依存结构作为软约束(*soft constraint*),这样可以减弱错误的依存结构带来的负面影响.因此,第3类方法是一类很有潜力的方法,但是目前的研究都是仅仅使用了一阶词汇依存信息.正如前文指出的那样,一阶词汇依存关系提供的信息非常有限,它能够对短语结构句法分析带来的帮助也会受到很大的限制.

为了解决上述方法的缺陷,本文将高阶词汇依存信息以软约束的方式引入到短语结构句法重排序模型中.这样做的优点是:(1) 重排序模型提供了一个较小的、同时质量较高的搜索空间(*N-best* 列表或句法森林),这使得引入复杂的高阶词汇依存特征成为可能;(2) 高阶词汇依存结构为 PCFG 提供了更多的词汇依存信息和依存树的上下文信息,这有助于更好地对短语结构树进行评估.

2 基于高阶词汇依存特征的句法模型

本节首先介绍一种判别式句法分析模型,该模型可以方便地将高阶词汇依存信息以特征的形式融入到句法分析过程中;然后给出一种将短语结构树表示为带有标记的依存结构的方法,以方便词汇依存结构的抽取;最后定义一系列特征模板,将词汇依存结构映射成特征向量.

2.1 判别式句法分析模型

为了便于利用高阶词汇依存信息对短语结构树进行评估,我们引入一种判别式句法分析模型——全局线性模型(*global linear model*,简称 GLM)^[25,26].该模型的输入为待分析句子 $x \in X$,输出为短语结构树 $c \in C$.其他符号

约定如下:

- 训练样本 $(x_t, c_t), t=1, \dots, n$. 其中, $x_t \in X$ 为输入句子, $c_t \in C$ 是输入句子对应的正确的短语结构分析树;
- 函数 GEN , 用于为输入句子 x 枚举出一系列候选短语结构分析树 $GEN(x)$;
- 函数 Φ , 用于将每个输入输出对 $(x, c) \in X \times C$ 映射为词汇依存特征向量 $\Phi(x, y) \in \mathcal{H}^d$;
- 权值向量 $\bar{\alpha} \in \mathcal{H}^d$, 用于评价每维特征的重要程度.

对于输入句子 x , 其对应的短语结构树 c 可以通过下式进行评估:

$$Score(x, c) = \Phi(x, c) \cdot \bar{\alpha} \tag{1}$$

其中, $\Phi(x, c) \cdot \bar{\alpha}$ 为特征向量与权重向量的点积 $\sum_i \alpha_i \Phi_i(x, c)$. 输入句子 x 的句法分析结果 c^* 可以通过下式获得:

$$c^* = \arg \max_{c \in GEN(x)} Score(x, c) \tag{2}$$

2.2 将短语结构树表示为有标记的依存树

本节给出一种将短语结构树表示为依存结构树的方法. 虽然短语结构树和依存结构树在语言学上未必存在简单的映射关系, 例如对于依存结构是非投射(non-project)的情况就不存在映射关系, 但是对于汉语或英语这类语言, 其依存树几乎都是可投射的. 因此, 本文从实践的角度出发, 忽略非投射依存树的情况, 给出一种利用有标记的依存树(labeled dependency tree)来表示短语结构树的方法. 该方法的步骤如下:

(1) 将短语结构词汇化, 即为短语结构树中的每个节点标注上对应的中心词信息

首先, 利用中心词查找表(如文献[27])为句法树中的每个节点找到对应的中心子节点(head child), 例如在图 3(a)中, 节点 B 被识别为规则 $A \rightarrow B C D E$ 的中心子节点, F 被识别为规则 $D \rightarrow F G H$ 的中心子节点. 然后, 从句法树的叶子节点出发, 自底向上递归地将中心子节点的中心词(head word)向上传递给父节点. 例如在图 3(b)中, 节点 F 的中心词 w_2 向上传递给了其父节点 D , 节点 B 的中心词 w_0 向上传递给了其父节点 A . 图 3(a)中的短语结构树片段经过上述词汇化过程之后得到了图 3(b)所示的结果.

(2) 将词汇化的短语结构树转化为有标记的依存结构树

首先, 将每条规则中非中心子节点(non-head-child)的中心词依存到中心子节点的中心词上. 例如在图 3(b)中, 规则 $A \rightarrow B C D E$ 的非中心子节点 $(C D E)$ 的中心词 w_1, w_2 和 w_3 都要依存到中心子节点 (B) 的中心词 w_0 上. 此外, 为了将短语结构树中的句法标记信息也编码到依存结构上, 我们为每条依存边添加一个依存关系标记(dependency label), 标记的格式为 $N_h:P:N_m$, 其中, N_h 是中心子节点对应的句法标记, P 是父节点对应的句法标记, N_m 是非中心子节点对应的句法标记. 例如在图 3(c)中, w_1 依存到 w_0 这条依存边的依存关系是通过规则 $A \rightarrow B C D E$ 转换得到的, 其中, w_0 对应的节点为 B , 父节点为 A , w_1 对应的节点为 C , 因此, 依存关系为 $B:A:C$. 图 3(b)中, 词汇化的短语结构树经过上述转换过程之后得到的有标记的依存树如图 3(c)所示.

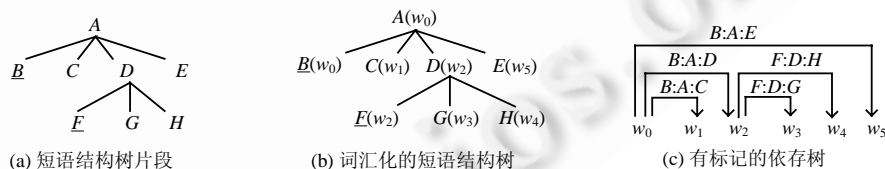


Fig.3 Represent phrase tree with labeled dependency tree

图 3 将短语结构树表示为有标记的依存树

2.3 词汇依存结构及特征向量表示

利用第 2.2 节中的方法将短语结构树表示为依存结构树后, 我们还需要将完整的依存树分解为各阶词汇依存结构的形式. 本文中, 我们采用图 2 中所示的 5 类词汇依存结构将完整的依存树进行分解.

在第 2.1 节中建立的判别式句法分析模型中, 词汇依存结构以特征向量的形式融入到模型中, 因此, 我们还需要将各阶词汇依存结构映射为特征向量的形式. 本文中我们采用表 1 所示的特征模板将每个词汇依存结构




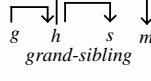
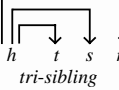
映射为特征向量,每维特征采用二元指示特征(binary indicator feature)的形式.即,如果某个特征在实例中出现,则取值为 1,否则取值为 0.因此,每个词汇依存结构都可以表示为一个高维的特征向量 ϕ , ϕ 中每维元素的取值均为 1 或 0,特征的维数由表 1 中特征模板实例化的特征数目决定.这样,短语结构树 C 的全局特征向量(global feature vector) Φ 可以从由其转换得到的完整的依存树 D 中抽取得到:

$$\Phi(C) = \sum_{d \in S(D)} \phi(d) \quad (3)$$

其中, $S(D)$ 表示由依存树 D 中抽取得到的所有词汇依存结构的集合, d 表示集合 $S(D)$ 中的一个词汇依存结构(本文中, d 仅限于表 1 中所示的几种词汇依存结构),函数 ϕ 用于按照表 1 所示的特征模板将词汇依存结构映射为特征向量.

Table 1 Feature templates of lexical dependencies

表 1 各阶词汇依存结构的特征模板

 dependency	元特征	 sibling	$POS(h), N(h), POS(s), N(s),$ $P(s), POS(m), N(m), P(m)$	
	$h, POS(h), N(h)$		$POS(h), N(h), N(s), P(s), N(m), P(m)$	
	$h, POS(h)$		$POS(h), N(h), POS(s), P(s), POS(m), P(m)$	
	$h, N(h)$		$POS(h), N(h), POS(s), N(s), POS(m), N(m)$	
	$m, POS(m), N(m)$		$POS(h), POS(s), POS(m)$	
	$m, POS(m)$		$N(h), N(s), N(m)$	
	$m, N(m)$		$N(h), P(s), P(m)$	
	二元特征		 grandchild	$POS(g), N(g), POS(h), N(h),$ $P(h), POS(m), N(m), P(m)$
	$P(m), h, POS(h), N(h), m, POS(m), N(m)$			$POS(g), N(g), N(h), P(h), N(m), P(m)$
	$h, POS(h), N(h), m, POS(m), N(m)$			$POS(g), N(g), POS(h), P(h), POS(m), P(m)$
	$P(m), POS(h), N(h), POS(m), N(m)$	$POS(g), N(g), POS(h), N(h), POS(m), N(m)$		
	$P(m), h, N(h), m, N(m)$	$POS(g), POS(h), POS(m)$		
	$P(m), h, POS(h), m, POS(m)$	$N(g), N(h), N(m)$		
	$P(m), h, m$	$N(g), P(h), P(m)$		
	$P(m), POS(h), POS(m)$			
	$P(m), N(h), N(m)$	$POS(g), POS(h), POS(s), POS(m)$		
	上下文词性特征	 grand-sibling		$N(g), N(h), N(s), N(m)$
	$P(m), N(h), POS(h), N(m),$ $POS(m), POS(h)+1, POS(m)-1$		$N(g), P(h), P(s), P(m)$	
	$P(m), N(h), POS(h), N(m),$ $POS(m), POS(h)-1, POS(m)-1$			
	$P(m), N(h), POS(h), N(m),$ $POS(m), POS(h)+1, POS(m)+1$		$POS(h), POS(t), POS(s), POS(m)$	
$P(m), N(h), POS(h), N(m),$ $POS(m), POS(h)-1, POS(m)+1$	$N(h), N(t), N(s), N(m)$			
$P(m), N(h), POS(h), N(m),$ $POS(m), POS(h)+1, POS(m)+1$	$N(h), P(t), P(s), P(m)$			
	 tri-sibling			

其中,小写字母 h, m, s, g 表示句子中的单词; $POS(x)$ 表示单词 x 的词性; $POS(x)+1$ 表示 x 右边的单词的词性; $POS(x)-1$ 表示 x 左边的单词的词性; $N(x)$ 表示编码在依存标签上的 x 对应的句法标记; $P(x)$ 表示编码在依存标签上的父节点的句法标记(具体含义见第 2.2 节).

3 基于高阶词汇依存的句法重排序模型

公式(2)中,特征向量的权重需要通过训练集学习得到,而参数学习的过程一般需要对训练集迭代多次.因此,推理算法的计算复杂度是一个不得不考虑的问题.在基于 PCFG 的短语结构分析中,常用的推理算法包括 CYK 算法、Chart 算法和 Early 算法等,这些算法的时间复杂度均为 $O(n^3G)$,其中, n 表示句子的长度, G 为语法常数.以宾州树库为例,句子的平均长度为 23 个词,语法常数 G 大于 1 000,这使得利用判别式方法精确地进行参数训练和句法分析推理变得几乎不可实现.因此,本文采用一种近似的句法分析模型——句法重排序模型.在句法重排序模型中,公式(2)中的 $GEN(x)$ 表示一个 N -best 句法树列表或者句法森林,它为重排序模型提供了一个较小的搜索空间,并且其中包含的候选句法树质量都比较高,这使得利用复杂的高阶词汇依存特征对短语结构树进行评估成为可能.

3.1 基于高阶词汇依存的N-best重排序模型

在 N-best 重排序模型中,GEN(x)表示由一个生成式句法分析器得到的 N-best 候选句法树列表,我们可以利用第 2.2 节中的方法将每个候选句法树转化为对应的依存树,然后按照第 2.3 节中的方法将该依存树映射成词汇依存特征向量的形式,接着利用公式(1)计算句法树得分,最后选取 N-best 候选中得分最高的句法树作为最终结果.

事实上,我们不难发现,按照第 2.2 节中的方法得到的依存树与原始的短语结构树之间存在着子树对应的关系,即短语结构树中的每个子树在依存树中都有一个与之对应的依存结构子树.例如,图 3(a)中以节点 D 为根节点的短语结构子树与图 3(c)中由 w₂,w₃ 和 w₄ 组成的以 w₂ 为根节点的依存结构子树相对应.因此,可以利用这一关系,将短语结构树表示为依存树的过程与词汇依存特征抽取的过程融合在一起,并据此设计一种递归的短语结构树评估算法.不失一般性,我们假定短语结构子树有如图 4(a)所示的结构,并为节点 P 定义一个四元组 (C_P,D_P,W_P,φ_P),其中,C_P表示以 P 为根节点的短语结构子树;D_P表示 C_P对应的依存结构子树;W_P表示节点 P 的中心词;φ_P表示节点 P 的局部特征向量(local feature vector),它表示由节点 P 上直接抽取得到的词汇依存特征向量.φ_P可以通过下式计算得到:

$$\phi_P = \sum_{d \in R(P)} \phi(d),$$

其中,R(P)表示每个由以 P 的中心词为根节点的词汇依存结构组成的集合.例如在图 3(c)中,我们可以抽取到所有以节点 A 的中心词 w₀ 为根节点的词汇依存结构集合 R(A),R(A)中的所有词汇依存结构如图 5 所示.

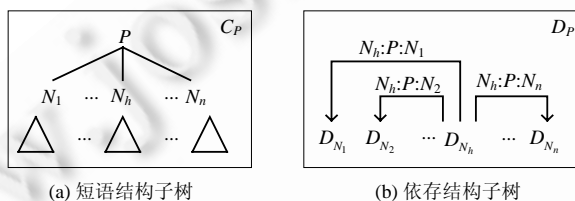


Fig.4 General structure of phrase sub-tree and dependency sub-tree

图 4 短语结构子树的一般形式及其对应的依存结构子树

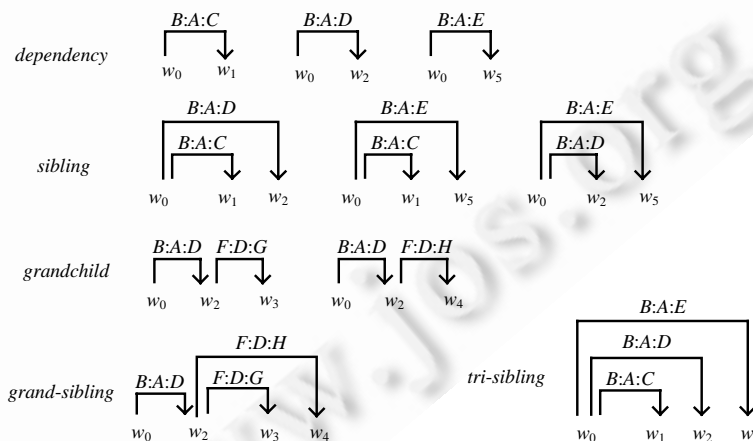


Fig.5 All the lexical dependencies for node A in Fig.3(a)

图 5 图 3(a)中节点 A 对应的所有的词汇依存结构

算法 1 列出了短语结构树评估算法的全部流程.该算法自底向上地为每个节点填充(C_P,D_P,W_P,φ_P)(算法 1 的第 2 行、第 3 行).在对子树进行评估时,首先找到节点 P 的中心子节点 N_h(第 8 行),然后将 N_h 的中心词传递

给 P (第 9 行). 接下来, 利用 D_{N_1}, \dots, D_{N_n} 为当前的子树建立对应的依存结构子树 D_P (第 11 行~第 14 行), 图 4(b) 给出了图 4(a) 对应的依存结构子树. 然后抽取节点 P 对应的所有的词汇依存结构, 并将其映射为局部特征向量 φ_P , 最后利用节点 P 的所有子节点递归地计算出 $Score(C_P)$:

$$Score(C_P) = \varphi_P \cdot \bar{\alpha} + \sum_{i=1}^n Score(C_{N_i}) \quad (4)$$

算法 1. 短语结构树评估算法 (phrase tree evaluating).

```

1: function Eval( $C$ )
2:   for  $P \in C$  in bottom up topological order do
3:     EvalSubTree ( $C_P$ )
4:   return Score( $C$ )
5:
6: procedure EvalSubTree( $C_P$ )
7:   //Assume the constituent is  $P \rightarrow N_1 \dots N_h \dots N_n$ 
8:    $N_h \leftarrow HeadChild(P \rightarrow N_1 \dots N_h \dots N_n)$ 
9:    $W_P \leftarrow W_{N_h}$ 
10:  //Building  $D_P$ 
11:  for  $N_i \in \{N_1, \dots, N_n\} \setminus N_h$  do
12:    Link  $D_{N_h}$  and  $D_{N_i}$  with a dependency arc
13:    Annotate the arc with label  $N_h:P:N_i$ 
14:    Make the root of  $D_{N_h}$  as  $D_P$ 's root
15:  Extract all lexical dependencies for  $P$  and map them into feature vector  $\varphi_P$ 
16:   $Score(C_P) = \varphi_P \cdot \bar{\alpha} + \sum_{i=1}^n Score(C_{N_i})$ 

```

3.2 基于森林的重排序模型

句法森林 (简称森林) 是多个句法树的压缩形式, 如图 6 所示的森林是图 1(a) 和图 1(b) 中的短语结构树的压缩形式. 森林可以表示为一个二元组 $\langle V, E \rangle$, 其中, V 是一个有限节点 (node) 集合, E 是超边 (hyperedge) 的集合. 森林中每个节点 $v \in V$ 可以表示为 $X_{i,j}$, 它表示非终结符 X 跨越输入句子的范围是 i 到 j , 如图 6 所示中的 $NP_{0,3}$. 森林中, 每个超边 $e \in E$ 可以表示为 $\langle tails(e), head(e) \rangle$, 其中, $head(e) \in V$ 表示规则的父节点, $tails(e) \in V^*$ 表示规则的所有子节点, 例如, 图 6 中的一个超边为 $\langle \{ADJP_{0,1}, NP_{1,2}\}, NP_{0,2} \rangle$.

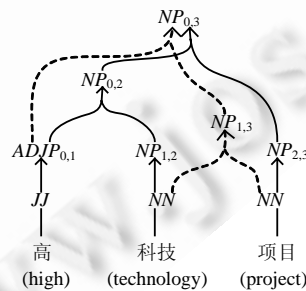


Fig.6 Parse forest

图 6 句法森林

为了获取森林, 文献 [13] 修改了 Charniak parser^[10], 使其直接输出森林. 受系统融合方法^[28,29]的启发, 本文中, 我们利用多个句法分析器输出的 N -best 候选句法树构造句法森林. 首先, 利用不同的句法分析器为输入句子分

别生成 N -best 句法树候选;然后,将这些句法树分割成上下文无关规则(context-free production),并且将规则中的每个句法标记标注上跨度信息;最后,利用得到的标有跨度信息的规则构造出句法森林.例如,我们可以将图 1(a)和图 1(b)中的句法树候选分割为 $\{NP_{0,3} \rightarrow ADJP_{0,1} NP_{1,3}, NP_{0,3} \rightarrow NP_{0,2} NP_{2,3}, \dots\}$,然后将这些规则融合成图 6 所示的句法森林.利用该方法得到的森林,可以超越 N -best 列表本身,构造出 N -best 列表中不包含的句法树,这一点可以在第 5.1 节中得以验证.

基于森林的重排序算法与算法 1 非常相似,不同之处在于:森林中每个节点对应不止一个超边.所以我们利用文献[30]中的 k -best 分析算法,自底向上地为每个节点建立 k 个最优的句法子树,最后选择根节点中得分最高的句法树作为结果.

4 参数训练

上节中介绍了两类重排序模型,本节中我们将探讨参数训练的方法.事实上,现有的很多模型都可以胜任此项工作,例如最大熵模型、条件随机域模型(CRF)和感知器模型(perceptron)等.本文中,我们采用在线学习(online-learning)算法进行参数训练,原因主要包括两点:首先,该算法是一种简单而且高效的算法,在获得相同的准确率的情况下它所需要的训练时间更少;第二,该算法在很多研究中都已被证明是有效的^[12,13,25,26].算法 2 描述了在线学习算法的流程,它对整个训练集迭代多次(第 3 行),每次迭代依次利用每个训练样本对模型参数进行更新(第 4 行~第 7 行),迭代过程结束后又会对权重取平均(第 8 行),这样做可以有效地避免过拟合(overfitting)现象的发生^[25].算法 2 中的第 5 行只是定性地给出了参数更新的方法,而具体的参数更新策略,需要我们根据实际任务来设计.针对本文的任务,我们设计了两个参数更新策略:感知器更新(perceptron update)和早期更新(early update).

算法 2. 在线学习算法(online learning algorithm).

- 1: **Input:** training data (x_t, c_t) for $t=1, \dots, T$;
- 2: $\bar{\alpha}^{(0)} \leftarrow 0$; $\nu \leftarrow 0$; $i \leftarrow 0$ //initial weights
- 3: **for** n in $1, \dots, N$ **do** // N iterations
- 4: **for** t in $1, \dots, T$ **do** // T training instances
- 5: $\bar{\alpha}^{(i+1)} \leftarrow$ update $\bar{\alpha}^{(i)}$ according to (x_t, c_t)
- 6: $\nu \leftarrow \nu + \bar{\alpha}^{(i+1)}$
- 7: $i \leftarrow i + 1$
- 8: $\bar{\alpha} \leftarrow \nu / (N * T)$ //averaging weights
- 9: **return** $\bar{\alpha}$

感知器更新(perceptron update)策略是一种常用的参数更新方法,该方法首先根据正确的句法树 c_t 获得 $F1$ 值最高的句法树 c_t^+ .

$$c_t^+ = \arg \max_{c \in GEN(x_t)} F1(c, c_t) \quad (5)$$

然后,利用当前参数 $\bar{\alpha}^{(i)}$ 计算出得分最高的句法树 \hat{c}_t .

$$\hat{c}_t = \arg \max_{c \in GEN(x_t)} \Phi(x_t, c) \cdot \bar{\alpha}^{(i)} \quad (6)$$

如果 \hat{c}_t 与 c_t^+ 不相同,则通过下式进行参数的更新:

$$\bar{\alpha}^{(i+1)} \leftarrow \bar{\alpha}^{(i)} + \Phi(c_t^+) - \Phi(\hat{c}_t) \quad (7)$$

否则,保持当前参数不变.

算法 3. 早期更新策略(early update strategy).

- 1: **function** *EarlyUpdate*($\bar{\alpha}, (x_t, c_t)$)
- 2: $\langle V, E \rangle \leftarrow$ *Forest*(x_t)


```

3:    $c_t^+ = \arg \max_{c \in \text{GEN}(\langle V, E \rangle)} F1(c, c_t)$  //oracle tree
4:   for  $v \in V$  in bottom up topological order do
5:      $Nbest_v \leftarrow NbestSubTrees(v, \bar{\alpha}, \langle V, E \rangle)$ 
6:     if  $v$  is in  $c_t^+$  then
7:        $\bar{\alpha} \leftarrow UpdateParam(\bar{\alpha}, v, c_t^+, Nbest_v)$ 
8:   return  $\bar{\alpha}$ 
9:
10: function  $UpdateParam(\bar{\alpha}, v, c_t^+, Nbest_v)$ 
11:   Get the oracle sub-tree  $s_v^+$  for node  $v$  from  $c_t^+$ 
12:   for  $j$  in  $1, \dots, L$  then
13:      $\hat{s}_v = \arg \max_{s_v \in Nbest_v} \Phi(s_v) \cdot \bar{\alpha}$ 
14:     if  $s_v^+ \neq \hat{s}_v$  then
15:        $\bar{\alpha} \leftarrow \bar{\alpha} + \Phi(s_v^+) - \Phi(\hat{s}_v)$ 
16:     else
17:       break
18:   if  $s_v^+ \neq \hat{s}_v$  then  $\hat{s}_v = s_v^+$ 
19:   return  $\bar{\alpha}$ 

```

虽然感知器更新策略在很多应用中都证明是有效的,但是参数更新要等到整个句法树都建立之后才能进行.我们认为,这样往往会错过参数更新的最佳时机.因为在分析一个句子时,如果已经发现了参数需要更新,还要用原来的参数继续分析当前句子未完成的部分,这将带来很大的噪声.因此,我们为基于森林的重排序方法设计了一种新的参数更新策略——早期更新(early update)策略.该策略的基本出发点是,将参数更新的过程融入到解码算法当中,针对每个子树进行参数的更新.早期更新策略的流程如算法 3 所示:首先,利用第 3.2 节中的方法为输入句子 x_t 生成句法森林 $\langle V, E \rangle$ (第 2 行),然后在森林中搜索出 $F1$ 值最大的句法树 c_t^+ (第 3 行);接下来,自底向上地对森林中的每个节点进行遍历(第 4 行~第 7 行).假定当前处于节点 v ,首先利用参数 $\bar{\alpha}$ 为其生成 k 个最优的子树 $Nbest_v$ (第 5 行),然后判断节点 v 是否在 c_t^+ 中(第 6 行),如果条件满足,则进行参数更新(第 7 行).这样做的目的是,我们希望更新后的参数能够从森林中将 c_t^+ 抽取出来.因此,我们按照 c_t^+ 来进行参数更新,而对于其他节点则不予考虑.第 10 行~第 19 行给出了参数更新的流程:首先,从 c_t^+ 中找出以节点 v 为根节点的 $F1$ 值最高的子树 s_v^+ (第 11 行);然后,利用当前参数从 $Nbest_v$ 中计算出得分最高的子树 \hat{s}_v (第 13 行).如果 s_v^+ 和 \hat{s}_v 不相同,则进行参数更新 $\bar{\alpha} \leftarrow \bar{\alpha} + \Phi(s_v^+) - \Phi(\hat{s}_v)$ (第 15 行).我们希望参数更新之后, s_v^+ 和 \hat{s}_v 应该相同,因为只有 $s_v^+ = \hat{s}_v$ 才能保证后面节点(森林中上层的节点)依然能够根据 c_t^+ 进行参数更新.所以,第 12 行~第 17 行中进行了多次的参数更新,直到 $s_v^+ = \hat{s}_v$ 相同为止.如果经过最大迭代次数 L 之后 s_v^+ 和 \hat{s}_v 仍然不同,我们会通过第 18 行的操作来强制其相同.实验中我们发现,大多数情况下,不超过 5 次迭代就能使得 $s_v^+ = \hat{s}_v$,所以我们设定 $L=5$.

早期更新策略与感知器更新策略相比:感知器更新要等到整个句法树构建完成后再进行参数更新,而早期更新策略是当发现某个子树与正确子树不相同时就进行参数的更新(这也是“早期更新”这一名称的由来).因此,早期更新策略中参数更新得更为及时,引入的噪声也会大为降低,这一点将会在后面的实验中得以验证.

5 实验与结果

本节中,我们通过实验来验证本文提出方法的有效性,所有实验均在宾州中文树库(Penn Chinese Treebank, 简称 CTB)5.0 版上进行.

为了方便与其他研究结果进行对比,实验中我们采用标准的树库划分方式,即 Art.301-325 作为开发集,

Art.271-300 作为测试集,其他剩余部分作为训练集^[5].实验中,采用 $F1$ 值作为系统的评测指标,所有的 $F1$ 值都采用 EVALB^[31]评估得到, $F1$ 值的计算公式如下所示^[1]:

$$Precision = \frac{\text{分析得到的正确短语个数}}{\text{分析得到的短语总数}} \times 100\%,$$

$$Recall = \frac{\text{分析得到的正确短语个数}}{\text{标准树库中的短语个数}} \times 100\%,$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \times 100\%.$$

5.1 N -best列表和句法森林的获取

在本实验中,首先利用已有的句法分析器生成 N -best 句法树列表,然后利用第 3.2 节中介绍的方法,根据 N -best 列表构造句法森林.我们使用交叉验证(cross validation)的方式为训练集中的每个句子生成 N -best 列表.首先将整个训练集划分为 20 等份,然后利用 Berkeley parser^[32]和 Charniak parser^[10]依次为每个等份生成 50-best 句法树列表,其中,对每个等份进行句法分析时使用的模型利用除该等份外剩余的 19 等份进行训练.开发集和测试集的 N -best 列表由全部训练集训练的模型分析得到.

表 2 列出了在测试集上 N -best 句法树列表和句法森林的最大 $F1$ 值.其中,“Berkeley(50)”表示由 Berkeley parser 生成的 50-best 句法树列表;“Charniak(50)”表示由 Charniak parser 生成的 50-best 句法树列表;“Comb(100)”表示将上述两个列表合并在一起之后的句法树列表;“Nbest”标识的一行数据表示对应 N -best 句法树列表的最大 $F1$ 值;“Forest”标识的一行数据表示利用相应 N -best 列表构建的句法森林的最大 $F1$ 值(该值使用文献[18]中提出的 forest oracle algorithm 评估得到).从表 2 中可以发现,句法森林的 $F1$ 值要好于对应的 N -best 列表的 $F1$ 值.这证明,利用 N -best 列表构造句法森林是有效的,而且获得的森林中含有了原 N -best 列表中不包含的句法树.

Table 2 Oracle $F1$ (%) of N -best lists and forests on test set

表 2 测试集上 N -best 列表和句法森林的最大 $F1$ 值(%)

	Berkley (50)	Charniak (50)	Comb (100)
N best	89.13	89.20	91.61
Forest	90.22	90.38	94.05

5.2 基于开发集的参数调试

虽然第 4 节中介绍的参数训练方法可以对特征权重进行训练,但是其中还有一些参数需要手动地进行调节,例如在线学习算法中的迭代次数 N ,基于森林重排序模型中 k -best 分析算法中的搜索宽度 k 等.本节通过在开发集上进行一系列实验来调试这些参数.

图 7 绘制的曲线表达了在线学习算法中迭代次数对 $F1$ 值的影响,虽然曲线稍有波动,但是可以发现, $F1$ 值是随着迭代次数的增加呈上升趋势的,而且平均后的模型可以进一步提高 $F1$ 值.为了避免过拟合(overfit)现象,在后面的实验中,我们都设定迭代次数为 $N=10$.

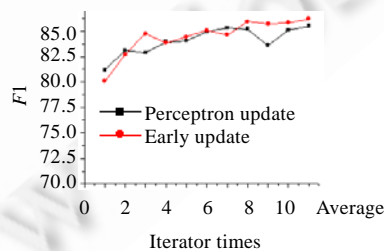


Fig.7 $F1$ curves varying with iteration times in Algorithm 2

图 7 算法 2 中迭代次数与 $F1$ 值的关系曲线

图 8 中绘制了 k -best 分析算法中搜索宽度 k 对 $F1$ 值影响的曲线,从中可以发现,当搜索宽度大于 5 之后, $F1$

值基本趋于稳定,因此在后面的实验中,我们都设定搜索宽度为 $k=5$.另外,从图 8 中我们还可以发现,利用早期更新策略进行参数更新,可以获得更好的句法分析性能.

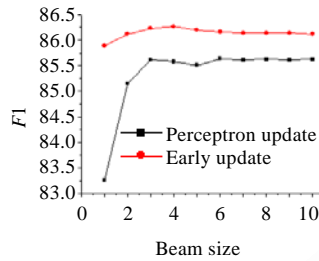


Fig.8 F1 curves varying with beam size

图 8 搜索宽度与 F1 的关系曲线

5.3 基于测试集的模型性能评估

本节通过在测试集上进行一系列的实验来验证本文提出方法的有效性.我们使用 Berkeley parser^[32]和 Charniak parser^[10]作为基线(baseline)系统,并根据前面描述的方法建立 3 个基于高阶词汇依存特征的句法分析系统.为了表述方便,我们为每个系统约定如下简称:

- (1) Berkeley: Berkeley parser;
- (2) Charniak: Charniak parser;
- (3) NbestRerank: 利用感知机更新策略进行参数训练的 N -best 重排序系统;
- (4) ForestRerank: 利用感知机更新策略进行参数训练的基于森林的重排序系统;
- (5) EarlyUpdate: 利用早期更新策略进行参数训练的基于森林的重排序系统.

使用第 5.2 节中得到的参数配置信息,我们在测试集上对上述系统进行了测试,相应的 F1 值列于表 3 中.

Table 3 F1 scores on test set

表 3 各系统在测试集上的 F1 值

	Berkeley (50)	Charniak (50)	Comb (100)
Berkeley	83.13	—	—
Charniak	—	82.41	—
NbestRerank	84.68	83.29	84.68
ForestRerank	84.31	83.11	85.72
EarlyUpdate	85.06	83.32	85.74

与基线系统相比,使用高阶词汇依存特征的系统的 F1 值都有所上升.无论是使用哪个 N -best 列表进行训练和测试,EarlyUpdate 系统都获得了最高的 F1 值,但是 F1 值上升的幅度却与 N -best 列表有很大的关系.当使用由 Berkeley parser 生成的 50-best 列表 Berkeley(50)时,F1 值上升了 1.93 个百分点;但在使用 Charniak parser 生成的 Charniak(50)时,F1 值却仅仅上升了 0.91 个百分点.我们认为其中的原因是,Charniak parser 本身就使用了词汇化信息,而 Berkeley parser 是一个完全非词汇化的句法分析器,因此,Berkeley parser 生成的列表对高阶词汇依存信息的敏感度会更高.

另外,还可以发现,在使用 Berkeley(50)和 Charniak(50)的列表进行训练和测试时,ForestRerank 系统的 F1 值比 NbestRerank 还要低一些.究其原因是,利用感知机更新策略错过了参数更新的最佳时机,所以基于森林的重排序系统(ForestRerank)的性能反而比基于 N -best 的重排序系统(NbestRerank)还要低.但在采用早期更新策略之后,基于森林的系统(EarlyUpdate)的 F1 值就超过了基于 N -best 的系统(NbestRerank),这一结果验证了早期更新策略的有效性.当使用 Comb(100)列表进行训练和测试时,NbestRerank 比 Berkeley 提高了 1.55%,ForestRerank 又比 NbestRerank 提高了 1.04%,而 EarlyUpdate 最终将 F1 提高到 85.74%.

从统计的角度出发,我们对这 5 个系统进行了统计假设检验,其中,NbestRerank,ForestRerank 和 EarlyUpdate 均采用 Comb(100)列表进行训练和测试.使用文献[33]中的方法,我们计算了每两个系统之间的 95%置信区间,结果列于表 4 中.

与表 3 中 *F1* 值显示的情况相似,NbestRerank 与 Charniak 和 Berkeley 相比是统计显著的,ForestRerank 和 EarlyUpdate 比 NbestRerank 是统计显著的.虽然 EarlyUpdate 在 *F1* 值上略高于 ForestRerank,但是它们是统计不显著的.

Table 4 Statistical significance test results among various systems

表 4 各系统之间的统计显著性检验

	Charniak	Berkeley	NbestRerank	ForestRerank	EarlyUpdate
Charniak		~	<	<	<
Berkeley	~		<	<	<
NbestRerank	>	>		<	<
ForestRerank	>	>	>		~
EarlyUpdate	>	>	>	~	

其中,“>”表示 A 系统明显优于 B 系统,“<”表示 B 系统明显优于 A 系统,“~”表示系统 A 和系统 B 是统计不显著的.

直观上讲,我们在句法分析过程中利用了大量的高阶词汇依存信息,因此,短语结构树对应的依存准确率也应该有所改进.为了验证这一点,我们将系统产生的短语结构树转换为相应的依存树,然后与准确率较高的依存句法分析系统(MSTParser^[34])进行比较.考虑到现有的依存句法分析器所采用的依存关系标记与本文的依存标记不同,因此为了便于比较,我们不考虑依存关系标记,采用无标记的依存准确率(unlabeled dependency accuracy,简称 UA)进行评估.为了体现系统的有效性,我们训练了一个一阶的 MSTParser(MST 1-ord)和一个二阶的 MSTParser(MST 2-ord).然后,分别利用这两个依存句法分析系统分析带有完全正确的(gold-standard)词性和自动标注(automatically)词性(词性标注准确率为 95.17%)的测试集数据,详细结果列于表 5 中.

Table 5 Unlabeled dependency accuracy (UA)

表 5 各系统的无标记依存准确率

Parsers	UA (%)
Charniak	82.31
Berkeley	84.05
NbestRerank	85.89
ForestRerank	85.69
EarlyUpdate	86.26
MST 1-ord (automatic POS)	79.62
MST 2-ord (automatic POS)	80.24
MST 1-ord (gold-standard POS)	85.23
MST 2-ord (gold-standard POS)	86.66

从表 5 中可以看出,我们的系统得到的句法树的 UA 值优于 Charniak 和 Berkeley.虽然我们的系统没有使用完全正确的词性标签,但其 UA 依然超过了使用了完全正确词性标签的 MST 1-ord.其中,EarlyUpdate 系统甚至与使用了完全正确词性标签的 MST 2-ord 相差无几.

表 3~表 5 中的数据说明,使用高阶词汇依存信息不但可以提高短语结构树的 *F1* 值,而且可以改善短语结构树的依存准确率(UA).

5.4 各阶词汇依存特征测试

前面的实验都使用了表 1 中列出的全部词汇依存特征,结果显示,这对句法分析性能的改善有很大的帮助.本节通过实验来检验:是否各阶词汇依存结构对句法分析性能的提升都有所帮助.实验中使用 EarlyUpdate 系统,并利用 Comb(100)列表进行训练和测试.首先,利用一阶词汇依存特征(dependency)对句法森林进行重排序;然后,将二阶词汇依存特征(sibling 和 grandchild)添加进重排序系统;最后,加入三阶词汇依存特征(grand-sibling

和 *tri-sibling*).在开发集上的测试结果列于表 6 中,从表中可以看出,依次加入各阶词汇依存特征之后系统的 *F1* 值逐步上升,这证明,各阶词汇依存特征对系统性能的提升都是有帮助的.

Table 6 *F1* (%) score on development set using different lexical dependency types

表 6 依次加入各阶词汇依存特征后,开发集上的 *F1* 值

	<i>F1</i> (%)
Baseline	84.59
+ <i>dependency</i> (一阶)	85.46
+ <i>sibling & grandchild</i> (二阶)	86.20
+ <i>grand-sibling & tri-sibling</i> (三阶)	86.37

5.5 相关系统的性能比较

表 7 列出了当前最好的句法分析系统以及本文提出的系统在标准测试集上的实验结果.“Charniak & Johnson Reranker”是一个 *N*-best 重排序系统,与其相比,本文的 *NbestRerank* 系统使用了高阶词汇依存特征,因此获得了较高的 *F1* 值.表 7 中,(Zhang *et al.*,2009)系统利用系统融合的方法提高句法分析器性能,它利用 Berkeley parser 和 Charniak parser 对 *N*-best 列表分别进行打分,然后利用这两个分数并结合大量的上下文无关文法(CFG)特征对 *N*-best 列表进行重排序.本文中的 *EarlyUpdate* 系统没有使用第 1 阶段句法分析器的分数,也没有引入 CFG 特征,仅仅使用了高阶词汇依存特征性能就超过了该系统.(Burkett and Klein,2008)系统、(Huang and Harper,2009)系统以及(Niu,*et al.*,2009)系统在模型训练过程中都使用了除宾州中文树库(CTB)之外的其他资源,而本文的 *EarlyUpdate* 系统在没有使用额外资源的情况下,依然获得了更高的 *F1* 值,*EarlyUpdate* 系统甚至超过了目前最好的句法分析系统,这证明,高阶词汇依存信息非常有助于短语结构句法分析.

Table 7 *F1* (%) scores of state-of-the-art methods compared with ours on the Penn Chinese Treebank

表 7 在宾州中文树库上目前最好的系统与本文系统的比较

单系统	<i>F1</i> (%)
(Petrov and Klein, 2007) ^[6]	83.32
(Huang and Harper, 2009) ^[35]	84.15
<i>N</i> -best 重排序系统	
Charniak & Johnson Reranker ^[17]	83.30
本文的 <i>NbestRerank</i> 系统	84.68
系统融合	
(Zhang <i>et al.</i> , 2009) ^[56]	85.45
使用了额外资源的系统	
(Burkett and Klein, 2008) ^[37]	84.24
(Huang and Harper, 2009) ^[35]	85.18
(Niu <i>et al.</i> , 2009) ^[38]	85.20
利用高阶词汇依存特征的系统	
本文的 <i>EarlyUpdate</i> 系统	85.74

6 总结与展望

本文提出了一种利用高阶词汇依存特征对短语结构树进行评估的方法.在重排序模型的框架下,利用高阶词汇依存特征对 *N*-best 列表和句法森林进行了重排序实验.在宾州中文树库上的实验结果显示,本文的方法超过了目前最好的句法分析系统,这证明,利用高阶词汇依存特征进行短语结构句法分析是十分有效的.

虽然本文中所有的实验都是在中文树库上进行的,但从理论上讲,文中的方法与具体语言无关,它可以应用到任何(可以将短语结构树转换为有标记的依存树)的语言上,这需要我们在将来的实验中加以验证.另外,本文中使用的重排序模型,其句法分析的上限会受到第 1 阶段句法分析器性能的约束.如果能够使用一种高效的推理算法(如 beam search 算法),将高阶词汇依存特征直接应用到句法分析推理过程中,那么句法分析的性能可能会得到进一步的提高,这需要我们在将来的工作中进行深入的研究.

References:

- [1] Zong CQ. Statistical Natural Language Processing. Beijing: Tsinghua University Press, 2008. 147–189.
- [2] Klein D, Manning CD. Accurate unlexicalized parsing. In: Proc. of the ACL 2003. Association for Computational Linguistics, 2003. 423–430. <http://aclweb.org/anthology-new/P/P03/> [doi: 10.3115/1075096.1075150]
- [3] Jurafsky D, Martin JH. Speech and Language Processing: An Introduction to Natural Language Processing. 2nd ed., Prentice Hall, 2008. <http://www.cs.colorado.edu/~martin/slp.html>
- [4] Matsuzaki T, Miyao Y, Tsujii J. Probabilistic CFG with latent annotations. In: Proc. of the ACL 2005. Ann Arbor: Association for Computational Linguistics, 2005. 75–82. <http://aclweb.org/anthology-new/P/P05/> [doi: 10.3115/1219840.1219850]
- [5] Petrov S, Barrett L, Thibaux R, Klein D. Learning accurate, compact, and interpretable tree annotation. In: Proc. of the COLING-ACL 2006. Sydney: Association for Computational Linguistics, 2006. 433–440. <http://aclweb.org/anthology-new/P/P06/> [doi: 10.3115/1220175.1220230]
- [6] Petrov S, Klein D. Improved inference for unlexicalized parsing. In: Proc. of the NAACL-HLT 2007. Rochester: Association for Computational Linguistics, 2007. 404–411. <http://aclweb.org/anthology-new/N/N07/>
- [7] Bikel DM. Intricacies of Collins' parsing model. Computational Linguistics, 2004,30(4):479–511. [doi: 10.1162/0891201042544929]
- [8] Charniak E. A maximum-entropy-inspired parser. In: Proc. of the NAACL 2000. Association for Computational Linguistics, 2000. 132–139. <http://aclweb.org/anthology-new/A/A00/>
- [9] Collins M. Head-Driven statistical models for natural language parsing [Ph.D. Thesis]. Philadelphia: University of Pennsylvania, 1999.
- [10] Charniak parser. <http://blip.cs.brown.edu/download/reranking-parserAug06.tar.gz>
- [11] Koo T, Collins M. Efficient third-order dependency parsers. In: Proc. of the ACL 2010. Uppsala: Association for Computational Linguistics, 2010. 1–11. <http://aclweb.org/anthology-new/P/P10/>
- [12] McDonald R, Crammer K, Pereira F. Online large-margin training of dependency parsers. In: Proc. of the ACL 2005. Ann Arbor: Association for Computational Linguistics, 2005. 91–98. <http://aclweb.org/anthology-new/P/P05/> [doi: 10.3115/1219840.1219852]
- [13] McDonald R, Pereira F. Online learning of approximate dependency parsing algorithms. In: Proc. of the EACL 2006. Association for Computational Linguistics, 2006. 81–88. <http://aclweb.org/anthology-new/E/E06/>
- [14] Collins M, Roark B. Incremental parsing with the perceptron algorithm. In: Proc. of the ACL 2004. Association for Computational Linguistics, 2004. 184–191. <http://aclweb.org/anthology-new/P/P04/> [doi: 10.3115/1218955.1218970]
- [15] Collins M, Koo T. Discriminative reranking for natural language parsing. Computational Linguistics, 2005,31(1):25–70. [doi: 10.1162/0891201053630273]
- [16] Klein D, Manning CD. Fast exact inference with a factored model for natural language parsing. In: Proc. of the NIPS 2002. Cambridge: MIT Press, 2002. 3–10. <http://books.nips.cc/nips15.html>
- [17] Charniak E, Johnson M. Coarse-to-Fine *n*-best parsing and MaxEnt discriminative reranking. In: Proc. of the ACL 2005. Ann Arbor: Association for Computational Linguistics, 2005. 173–180. <http://aclweb.org/anthology-new/P/P05/> [doi: 10.3115/1219840.1219862]
- [18] Huang L. Forest reranking: Discriminative parsing with non-local features. In: Proc. of the ACL 2008. Columbus: Association for Computational Linguistics, 2008. 586–594. <http://aclweb.org/anthology-new/P/P08/>
- [19] Hall J, Nivre J. A dependency-driven parser for German dependency and constituency representations. In: Proc. of the PaGe 2008. Columbus: Association for Computational Linguistics, 2008. 47–54. <http://aclweb.org/anthology-new/W/W08/#1000>
- [20] Hall J, Nivre J, Nilsson J. A hybrid constituency-dependency parser for Swedish. In: Proc. of the NODALIDA 2007. 2007. 284–287. <http://math.ut.ee/nodalida2007/>
- [21] Wang R, Zhang Y. Hybrid constituent and dependency parsing with Tsinghua Chinese Treebank. In: Proc. of the LREC 2010. 2010. 1950–1954. <http://www.lrec-conf.org/lrec2010/>
- [22] Wang ZG, Zong CQ. Phrase structure parsing with dependency structure. In: Proc. of the Coling 2010. Beijing: Int'l Committee on Computational Linguistics, 2010. 1292–1300. <http://aclweb.org/anthology-new/C/C10/>

- [23] Xia F, Palmer M. Converting dependency structures to phrase structures. In: Proc. of the HLT 2001. Association for Computational Linguistics, 2001. <http://aclweb.org/anthology-new/H/H01/> [doi: 10.3115/1072133.1072147]
- [24] Xia F, Rambow O, Bhatt R, Palmer M, Sharma DM. Towards a multi-representational treebank. In: Proc. of the TLT-7. Association for Computational Linguistics, 2009. 159–170. <http://aclweb.org/anthology-new/N/N09/>
- [25] Collins M. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In: Proc. of the EMNLP 2002. Philadelphia: Association for Computational Linguistics, 2002. 1–8. <http://www.aclweb.org/anthology-new/W/W02/#1000> [doi: 10.3115/1118693.1118694]
- [26] Collins M. A new statistical parser based on bigram lexical dependencies. In: Proc. of the ACL'96. Association for Computational Linguistics, 1996. 184–191. <http://aclweb.org/anthology-new/P/P96/> [doi: 10.3115/981863.981888]
- [27] Yamada H, Matsumoto Y. Statistical dependency analysis with support vector machines. In: Proc. of the IWPT 2003. Association for Computational Linguistics, 2003. http://aclweb.org/anthology-new/sigparse#2003_0
- [28] Fossum V, Knight K. Combining constituent parsers. In: Proc. of the NAACL 2009. Boulder: Association for Computational Linguistics, 2009. 253–256. <http://aclweb.org/anthology-new/N/N09/>
- [29] Sagae K, Lavie A. Parser combination by reparsing. In: Proc. of the NAACL 2006. New York: Association for Computational Linguistics, 2006. 129–132. <http://aclweb.org/anthology-new/N/N06/>
- [30] Huang L, Chiang D. Better k -best parsing. In: Proc. of the IWPT 2005. Vancouver: Association for Computational Linguistics, 2005. 53–64. http://aclweb.org/anthology-new/sigparse#2005_0
- [31] Blaheta D, Sekine S. Evalb tool. 2008. <http://nlp.cs.nyu.edu/evalb/>
- [32] Berkeley parser. <http://code.google.com/p/berkeleyparser/>
- [33] Zhang Y, Vogel S, Waibel A. Interpreting BLEU/NIST scores: How much improvement do we need to have a better system. In: Proc. of the LREC 2004. 2004. 2051–2054. <http://www.lrec-conf.org/lrec2004/>
- [34] MSTParser. <http://sourceforge.net/projects/mstparser/>
- [35] Huang ZQ, Harper M. Self-Training PCFG grammars with latent annotations across languages. In: Proc. of the EMNLP 2009. Singapore: Association for Computational Linguistics, 2009. 832–841. <http://aclweb.org/anthology-new/D/D09/>
- [36] Zhang H, Zhang M, Tan CL, Li HZ. K -Best combination of syntactic parsers. In: Proc. of the EMNLP 2009. Singapore: Association for Computational Linguistics, 2009. 1552–1560. <http://aclweb.org/anthology-new/D/D09/>
- [37] Burkett D, Klein D. Two languages are better than one (for syntactic parsing). In: Proc. of the EMNLP 2008. Honolulu: Association for Computational Linguistics, 2008. 877–886. <http://aclweb.org/anthology-new/D/D08/>
- [38] Niu ZY, Wang HF, Wu H. Exploiting heterogeneous treebanks for parsing. In: Proc. of the ACL-IJCNLP 2009. Suntec: Association for Computational Linguistics, 2009. 46–54. <http://aclweb.org/anthology-new/P/P09/>

附中文参考文献:

- [1] 宗成庆. 统计自然语言处理. 北京: 清华大学出版社, 2008. 147–189.



王志国(1985—),男,辽宁盘锦人,博士生,主要研究领域为句法分析,机器翻译.



宗成庆(1963—),男,博士,研究员,博士生导师,CCF 会员,主要研究领域为机器翻译,文本分类,口语信息处理.