学校代码：10285
学　　号：20184227035

硕士学位论文

（学术学位）

基于树形条件随机场的高阶句法分析

TreeCRF-based High-Order Syntactic Parsing

研 究 生 姓 名　张宇
指导教师姓名　李正华
专　业　名　称　计算机科学与技术
研　究　方　向　自然语言处理
所　在　院　部　计算机科学与技术学院
论文提交日期　2021 年 6 月

苏州大学学位论文独创性声明

本人郑重声明：所提交的学位论文是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含其他个人或集体已经发表或撰写过的研究成果，也不含为获得苏州大学或其它教育机构的学位证书而使用过的材料。对本文的研究作出重要贡献的个人和集体，均已在文中以明确方式标明。本人承担本声明的法律责任。

论文作者签名：张宇　　日　期：2021.6.8

苏州大学学位论文使用授权声明

本人完全了解苏州大学关于收集、保存和使用学位论文的规定，即：学位论文著作权归属苏州大学。本学位论文电子文档的内容和纸质论文的内容相一致。苏州大学有权向国家图书馆、中国社科院文献信息情报中心、中国科学技术信息研究所（含万方数据电子出版社）、中国学术期刊（光盘版）电子杂志社送交本学位论文的复印件和电子文档，允许论文被查阅和借阅，可以采用影印、缩印或其他复制手段保存和汇编学位论文，可以将学位论文的全部或部分内容编入有关数据库进行检索。

涉密论文□

本学位论文属　　在_____年___月解密后适用本规定。

非涉密论文☑

论文作者签名：张宇　　日　期：2021.6.8

导师签名：李正华　　日　期：2021.6.8

基于树形条件随机场的高阶句法分析

摘 要

句法分析任务是句子理解的重要中间过程之一. 其中，概率估计一直是句法分析领域的一个核心问题. 然而，无论是神经网络方法还是深度学习时代以前的方法，采用基于全局概率模型的句法分析工作都非常少，主要的原因在于树形条件随机场（TreeCRF）推断的高复杂度. 在本文中，我们提出将 TreeCRF 应用到依存句法和成分句法这两个主要的句法分析任务. 为了解决 TreeCRF 的低效问题，关键的想法是批次化树结构的推断算法，并且用基于自动求导的反向传播代替 Outside 算法. 目前句法模型被不断简化，采用局部损失目标是当前句法分析方法的一个趋势，我们则进一步在一阶 TreeCRF 的基础上采用了高阶拓展. 高阶 TreeCRF 进一步增加了算法复杂度，为此，我们还提出利用基于平均场变分推断的近似推断算法代替精确推断的 TreeCRF 方法，从而增加了解析效率.

具体而言，本文的研究内容主要包含三个方面：

(1) 基于 TreeCRF 的高阶依存句法分析方法

本文提出将 TreeCRF 方法应用到神经依存句法分析器当中，并进一步提出了一个二阶 TreeCRF 的扩展. 导致 TreeCRF 低效的主要瓶颈在于 Inside-Outside 算法，尤其是 Outside 算法的计算. 为了解决这个问题，一方面，我们提出对 Inside 算法进行批次化，从而利用 GPU 的并行计算能力来加速，将算法复杂度从 $O(n^3)$ 降低到了 $O(n^2)$. 另一方面，我们还提出将复杂的 Outside 算法用高效的反向传播代替，显著提升了效率，使得一阶和二阶模型的速度分别达到了 500 和 400 句每秒. 我们在 13 个语言的 27 个数据集上进行了详细实验，结果表明了 TreeCRF 和高阶建模的有效性.

(2) 基于 TreeCRF 的高阶成分句法分析方法

本文提出将高阶 TreeCRF 应用到成分句法分析中. 为了解决效率问题，我们应用了和依存模型中一致的批次化技术和反向传播来加速. 此外，我们提出一个简单的两阶段解析方法，和前人的一阶段解析相比结果相当，但是更加高效. 我们还参考了依存句法的模型架构和参数设置，提出用双仿射打分机制替换传统打分方法，发现在双向 LSTM 编码器中引入的诸如 Dropout 的策略改进可以极大提升解析的性能. 在中英

文三个基准数据集上的实验结果表明，我们提出的模型结果显著超越了现有方法，并且一阶和二阶模型的速度分别达到了 1,092 和 598 句每秒. 我们的模型在使用 BERT 之后达到了现有最好的结果.

(3) 基于变分推断的高效句法分析方法

为了解决精确推断的 TreeCRF 方法高复杂度的问题，本文提出在依存句法和成分句法分析中引入基于平均场变分推断的近似方法. 相比于高阶 TreeCRF 方法，变分推断将算法在 GPU 上的复杂度从 $O(n^2)$ 降低到了 $O(n)$，大大提升了模型效率. 在中英文共五个数据集上的实验结果表明，我们的二阶变分推断方法在性能上显著超越了一阶模型，达到了和二阶 TreeCRF 模型可比较的水平，与此同时在依存句法和成分句法上的解析速度分别达到了 1,126 句每秒和 905 句每秒，大大超越了精确推断的二阶 TreeCRF. 此外，使用 BERT 之后，我们的变分推断方法的结果达到或接近了现有的最佳结果.

综上，我们在依存和成分句法这两种句法分析任务上提出应用 TreeCRF 以及一个二阶 TreeCRF 拓展，显著提升了句法分析器的性能. 我们采用批次化以及反向传播等加速技术，解决了 TreeCRF 的效率问题. 本文同样还探究了变分法等近似方法对解析效率的影响. 我们发现变分法在保持高阶模型的性能的同时，大大加快了解析速度.

关键词：句法分析，依存句法分析，成分句法分析，树形条件随机场，变分推断

作 者：张 宇

指导老师：李正华

TreeCRF-based High-Order Syntactic Parsing

Abstract

Syntactic parsing is one of the most important intermediate processes in sentence comprehension, and probability estimation has always been a core problem in the parsing field. However, in either deep learning (DL) era or pre-DL era, there exist very few works based on global probabilistic modeling, mainly due to the high complexity of tree-structure CRF (TreeCRF) inference. This thesis proposes to apply TreeCRF to both dependency parsing and constituency parsing. The key idea to solve the inefficiency issue is to batchify the inference algorithm for tree structures, and meanwhile avoid the complex Outside algorithm via back-propagation. Currently, parsing models are greatly simplified, and it's a trend to adopt local loss for syntactic parsing. In contrast, we propose a high-order extension to first-order models. While high-order modeling further increases the algorithm complexity, we also try to apply mean field variational inference (MFVI) as an alternative to exact inference of TreeCRF method, which greatly improves the parsing efficiency.

Specifically, the main research content of this thesis includes three parts:

(1) TreeCRF-based high-order dependency parsing.

This thesis proposes to apply TreeCRF-based method to neural dependency parsing, and further presents a second-order extension. The main bottleneck leading to the inefficiency of TreeCRF lies in the Inside-Outside algorithm, especially the calculation of the Outside pass. To overcome this, we propose to batchify the Inside algorithm, and reduce the time complexity from $O(n^3)$ to $O(n^2)$ by utilizing the power of GPU parallel computation. In addition, we replace the complex Outside algorithm with back-propagation equipped with auto-differentiation, which significantly improves the efficiency and speeds up the model to 400 sentences/s. We conduct extensive experiments on 27 datasets in 13 languages, and the results reveal the effectiveness of TreeCRF and high-order modeling.

(2) TreeCRF-based high-order constituency parsing.

This thesis proposes to apply high-order TreeCRF to constituency parsing. To solve the

efficiency issue, we apply batchification techniques and back-propagation consistent with the dependency model to accelerate. Moreover, we propose a simple two-stage parsing approach, which has comparable results with previous one-stage methods, but is more efficient. We also refer to the model architecture and parameter settings of dependency models, and propose to replace the traditional scoring method with a biaffine scoring mechanism. We find that the parsing performance can be largely improved via better encoder settings like Dropout configuration, leading to similar results with current state-of-the-art Transformer encoder. Experimental results on three Chinese and English benchmark datasets show that our proposed models significantly surpass existing methods. In terms of parsing speed, our first-order and second-order models can parse over 1,092/598 sentences/s. After using BERT, our models achieve new state-of-the-art performance on all datasets.

(3) Efficient syntactic parsing based on variational inference.

In order to deal with the high complexity of the TreeCRF method for exact inference, this thesis proposes to introduce an approximate method based on mean field variational inference for dependency and constituency parsing. Compared with high-order TreeCRF, variational inference reduces the time complexity on GPU from $O(n^2)$ to $O(n)$, improving the model efficiency greatly. Experimental results on five Chinese and English datasets show that our second-order variational inference method significantly outperforms the first-order model, and achieves comparable results with second-order TreeCRF models. Meanwhile, our models can parse over 1,126 and 905 sentences/s on dependency parsing and constituency parsing respectively, greatly surpassing the exact inference of second-order TreeCRF methods . Moreover, after using BERT, our variational inference method achieves or approaches the performance of current state-of-the-art models.

In summary, this thesis proposes to apply TreeCRF and further presents a second-order extension for both neural dependency and constituency parsing, achieving the current state-of-the-art performance. To tackle the inefficiency issue, we apply batchification techniques and back-propagation to reduce the algorithm complexity. This thesis also studies the impact of approximate methods like variational inference on parsing efficiency. We find it greatly improves the parsing speed while has a similar performance to exact high-order modeling.

Key words: Syntactic Parsing, Dependency Parsing, Constituency Parsing, TreeCRF, Variational Inference

Written by Yu Zhang

Supervised by Zhenghua Li

目　　录

第一章 绪论

1.1 研究背景和意义

自然语言处理（Natural Language Processing, NLP）是目前人工智能方兴未艾的领域之一. 一个完整的自然语言处理句子分析流程主要分为三个部分：1）词法分析；2）句法分析；3）语义分析[1]. 其中词法分析包含了词性标注（Part-of-Speech Tagging）、命名实体识别（Named Entity Recognition, NER）以及消歧（Disambiguation）等子任务，中文中由于词语之间没有天然边界，还需要额外进行中文分词. 句法分析的目的是以句法树的形式刻画句子结构，主要包含了依存句法（Dependency Parsing）和成分句法（Constituency Parsing）分析这两种范式. 语义分析则是为了理解句子的内含语义，包含语义角色标注（Semantic Role Labeling, SRL）、语义依存分析（Semantic Dependency Parsing, SDP）和抽象语义表示（Abstract Meaning Representation, AMR）等子任务.

上述的三个流程通常以管道的方式进行, 而句法分析作为连接词法和语义分析的中间步骤，具备十分重要的研究价值. 目前存在着多种句法分析文法，例如组合范畴文法（Combinatorial Categorical Grammars, CCGs），成分文法（Constituency Grammars or Context Free Grammars, CFGs），以及依存文法（Dependency Grammars）. 其中依存文法对应的依存句法分析，和成分文法对应的成分句法分析是目前最常见的句法分析范式，也是本文主要的研究对象.

依存句法分析的目的是识别句子中词语的修饰关系. 如图1-1所示，给定一个句子 $\boldsymbol{x} = w_0, w_1, \cdots, w_n$，一棵依存树被定义为 $\boldsymbol{t} = \{(i \rightarrow j, l) \mid 0 \leq i \leq n, 0 < j \leq n, l \in \mathcal{L}\}$，其中 $(i \rightarrow j, l)$ 是一条从头（head）w_i 到修饰词（modifier）w_j 的弧，弧的标签为 $l \in \mathcal{L}$. 有标签树 $\boldsymbol{t}$ 可以进一步被分解为 $(\boldsymbol{y}, \boldsymbol{l})$，即一棵无标签树 $\boldsymbol{y}$ 和树上所有标签组成的序

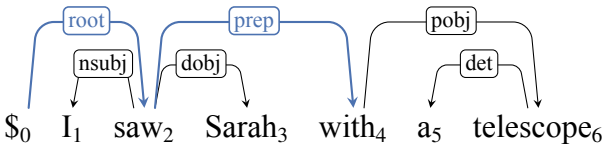
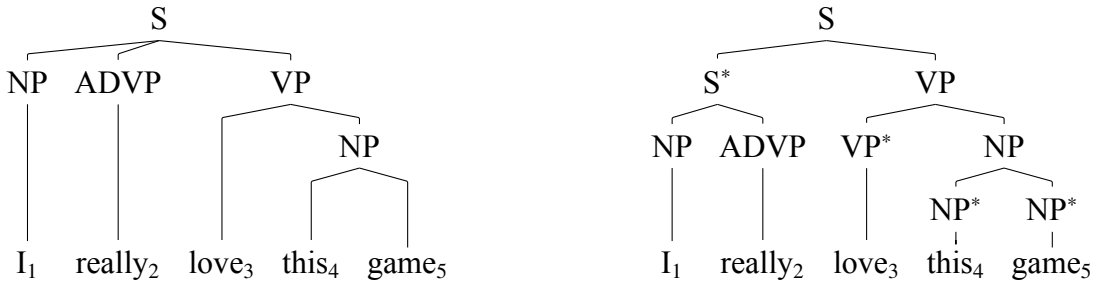


图 1-1 一个完整依存树的例子. 对于局部标注的场景，仅有一部分的弧被标注，例如图中两个粗蓝弧.



(a) **原始句法树**　　(b) **遵循乔姆斯基范式的左二叉化句法树**

图 1-2　成分句法树的例子. 其中词性在这里被忽略

列 $\boldsymbol{l}$. 得益于基准数据集宾州书库（Penn Treebank, PTB）的发布，以及深度学习技术的发展，目前依存句法分析技术得到了广泛研究[2]. 其中计算自然语言学习会议（Conference on Computational Natural Language Learning, CoNLL）连续多年针对依存句法任务发布了评测任务，尤其是近年来依托通用依存（Universal Dependencies, UD）(Nivre 等[3]) 项目发布的多语言评测比赛，大大推动了依存句法分析技术的进步. 由于结构简单、形式直观的特点，依存句法分析一直被广泛的应用在多个其他任务中，例如机器翻译 (Zhang 等[4])、关系抽取 (Song 等[5])、意见挖掘 (Zhang 等[6]) 等等.

成分句法则旨在构建一个层次化的树结构. 如图1-2，其中每个叶子结点是输入句子的每个词，而非终端结点作为区块（Constituents），如 $\text{VP}_{3,5}$. 正式地，给定一个由 n 个词组成的句子 $\boldsymbol{x} = w_1, \ldots w_n$，如图 1-2a所示，一棵成分句法树可以表示为 $\boldsymbol{t} = \{((i,j),l) \mid 1 \le i \le n, i \le j \le n, l \in \mathcal{L}\}$，其中 $((i,j),l) \in \boldsymbol{t}$ 是一个包含 $w_i ... w_j$ 的区块，对应的句法标签为 $l \in \mathcal{L}$. 本文中，为了方便模型处理，我们还将原始的树转换为了符合乔姆斯基范式（Chomsky Normal Form, CNF）的二叉树形式，如图 1-2b所示. 成分句法分析相比于依存句法而言，研究历史更加悠久，尽管目前没有依存句法技术那么流行，但是以其分析技术为基础衍生出了一系列在其他任务上的解析方法，例如 UCCA 语义分析 (Jiang 等[7])、嵌套命名实体识别 (Fu 等[8]) 等等.

目前，针对句法分析任务，已经有大量的句法分析范式被提出，主要的有生成式方法、判别式方法，以及基于转移的方法、基于图的方法等等. 其中，基于图的方法尝试对每棵可能的句法树进行打分，然后选择一棵分值最高的树作为输出. 最简单的一阶方法将句法树分值分解为每个独立变量的分值之和，极大简化了模型建模. 与其他方法相比，基于图的方法无需设计复杂的转移系统，容易集成丰富的特征，可以达到很高的准确率，一直以来都是最流行的句法分析方法之一，因此也是本文所主要关注的句法分析方法.

传统的句法分析方法十分依赖于离散特征的人工设计和结构化建模. 由于缺乏对输入句子上下文的捕获能力，传统方法不得不手动设计诸如前缀、后缀、词性等特征，作为词输入的补充信息. 此外，显式的的树约束也是不可或缺的，模型依赖于 Max Margin 或 TreeCRF 等训练方法来全局最大化一棵树的分值/概率.

最近，由于深度神经网络的强大上下文编码能力，句法分析器的方法愈来愈有简单化的趋势. 作为依存句法和成分句法领域各自最有代表性的模型，Dozat 等[9] 的依存句法分析器 Biaffine Parser 和 Stern 等[10] 的成分句法分析器正是符合这样的潮流. 其中 Biaffine Parser：1）采用了一个诸如双向 LSTM 或 Transformer (Vaswani 等[11]) 这样强大的编码器；2）采取了简单的头选择训练损失函数. 类似地，Stern 等[10] 在双向 LSTM 编码器的基础上采取了 Max Margin 训练目标最大化正确树的分值，而 Gaddy 等[12] 更进一步，直接采取二分类方法判断短语树的每个组块是否存在. 这些方法由于建模简单，速度很快，因此一直是当前最为流行的句法分析方法.

有鉴于此，我们尝试在本文中将传统方法与现代的句法分析器相结合. 首先，我们在章节 二和章节 三中采用了 TreeCRF 代替了当前流行的局部训练损失目标或者 Max Margin 方法. 与这些方法相比，TreeCRF 的主要优势在于可以获得树/子树概率. 不同于没有上下界的分值，树/子树概率是评估解析置信度的更好指标，此外也可以作为一种软特征方便地作为下游任务的输入 (Zhang 等[4,6]). 主要限制 TreeCRF 应用的原因在于 Inside-Outside 算法的高复杂度问题，尤其是 Outside 过程，通常而言 Outside 算法要两倍慢于 Inside. 对此，我们主要从两个方面尝试解决：1）我们为 Inside 算法设计了精巧的批次化方法，使得 TreeCRF 能够利用 GPU 的并行计算能力大大加速；2）得益于集成了自动求导机制的深度学习库的出现，我们无需和前人一样需要在 CPU 上进行完整的 Inside-Outside 过程得到梯度，而是将 Outside 过程由高效的反向传播机制代替.

此外，已经有很多工作证明，在模型中引入包含更多变量交互的高阶特征可以极大提升句法分析的准确率 (McDonald 等[13], Chen 等[14], Ji 等[15]). 因此在章节 二和章节 三中，我们在一阶模型的基础上引入了一个高阶 TreeCRF 的拓展. 为了能够对二阶子树特征打分，我们还参考 Biaffine 打分器为其设计了一个 Triaffine 机制. 结果表明，高阶 TreeCRF 在大多数数据集上都带来了显著的提升.

尽管可以应用批次化技术，高阶 TreeCRF 仍不可避免地加剧了推断算法效率低下的问题. 在机器学习社区中，研究者在针对推断算法高复杂度或者不可精确推断问

表 1-1 依存句法分析数据集的数据统计，包含句子数和标签数.

	#Train	#Dev	#Test	#labels
PTB	39,832	1,700	2,416	45
CoNLL09	22,071	1,762	2,556	41
NLPCC19	29,991	4,098	8,295	22

题时，一个通常的做法是尝试采用近似推断算法. 因此在章节 四中，我们还尝试了利用近似的平均场变分推断（Mean Field Variational Inference, MFVI）方法来代替高阶 TreeCRF. 和其他同样在句法分析中广泛应用的近似推断算法，例如循环置信传播 (Smith 等[16]) 和对偶分解 (Martins 等[17])，相比，MFVI 的效率更高，收敛更快，并且通常可以达到更好的结果 (Wang 等[18]). 此外，MFVI 能够方便的得到树/子树的概率，这保留了 TreeCRF 的优势. 我们在依存句法和成分句法分析两种范式中分别设计了两种不同的因子图以及变分推断迭代算法，结构表明 MFVI 在达到和高阶 TreeCRF 相接近的性能的同时，显著提升了句法模型的解析速度.

1.2 数据集和评价指标

数据. 对于依存句法分析，我们在 13 个语言的 27 个数据集上进行了实验和分析，包含两个广泛使用的数据集：英语的斯坦福依存规范 (Chen 等[14]) 的宾州树库（Penn Treebank, PTB）和中文的 CoNLL09 数据 (Hajič 等[19]). 我们还采用了公开于 NLPCC19 跨领域句法分析任务的中文数据集 (Peng 等[20])，其中一共包含了四个源领域和三个目标领域. 方便起见，我们直接合并了四个领域的 Train/Dev/Test 数据到更大的数据集. 这些数据的一个特征是大部分句子都是利用基于主动学习的局部标注得到的. 表 1-1列出了相关数据的统计信息. 最后，遵循 Ji 等[15] 和 Zhang 等[21]，我们在 Universal Dependencies (UD) v2.2 和 v2.3 上进行了实验. 我们采用了 Zeman 等[22] 使用的 300 维多语言预训练词向量，并采用 CharLSTM 表示作为输入. 对于 UD2.2，为了和 Ji 等[15] 公平比较，我们和 CoNLL18 任务一样 (Zeman 等[22])，使用了毛文本，并且直接使用了他们的句子分割和符号化的结果. 对于 UD2.3，为了与 Zhang 等[21] 比较，我们报告的结果使用了正确词性.

对于成分句法分析，我们主要在三个中文和英文的数据集上进行实验. 前两个数据集，即 PTB 和 CTB5.1，是句法分析社区中比较常用的两个数据集. 我们遵循了传

表 1-2 成分句法分析数据集的数据统计，包含句子数和标签数. 对于 “#labels”，我们列出了原始树和 CNF 树对应的标签数.

	#Train	#Dev	#Test	#labels	
				original	CNF
PTB	39,832	1,700	2,416	26	138
CTB5.1	18,104	352	348	26	162
CTB7	46,572	2,079	2,796	28	265

统的 Train/Dev/Test 数据分割. 考虑到 CTB5.1 的 Dev/Test 都只有大约 350 句，为了得到更加稳定一致的结果，我们还在更大的 CTB7 数据上进行了实验，相关的数据分割设置参考了官方手册建议. 表 1-2列出了相关数据的统计信息，其中 UD 的数据较多，因此不一一列举. 可以看到 CNF 转换引入了很多新的区块标签，其中大部分（大约 75%）都源于连续单链的折叠过程.

评价指标. 对于依存句法分析，我们使用无标签/有标签附着分值（Unlabeled/Labeled Attachment Score, UAS/LAS）作为主要的评价指标，其中 UAS 指头正确的词数占总词数之比，LAS 则进一步还要求词的头正确的同时弧上的标签也正确. 评价时，PTB 中的词性会被忽略掉. 对于局部标注的 NLPCC19 数据，我们采用了官方的评价脚本，直接忽略了没有正确头标注的词. 我们采用了 Dan Bikel 的随机解析评价比较器来进行显著性检验.

对于成分句法分析，在解析之后，我们将最佳的 CNF 树转化为 *n*-ary 树再进行评价. 这里有必要提及一些有用的细节. 由于解码算法没有相应的约束，预测的最佳 CNF 树可能包含很多不合法的产生式. 以图 1-2b为例，模型可能输出 $\mathrm{VP}_{3,5} \rightarrow \mathrm{PP}^{*}_{3,3}\ \mathrm{NP}_{4,5}$，其中 VP 和 PP* 是不兼容的. 在 *n*-ary 后处理过程中，我们直接忽略了 “*” 符号之前具体的字符串 PP. 有鉴于此，如果解码的时候增加一定的约束，结果有可能进一步提高，这里我们留待作为后续的工作. 我们使用了标准的区块级别的准确率、召回率和 F_1 值（P/R/F_1）作为评价指标，并使用 EVALB 工具[1]来评价. 特别地，一个例如 $\mathrm{VP}_{3,5}$ 的预测区块如果出现在了正确树中，那就被认为是正确的.[2]

[1] https://nlp.cs.nyu.edu/evalb

[2] 由于一些研究者可能会实现他们自己的评价脚本，为了比较的公平，需要澄清一些细节：1）一些诸如 {-NONE-} 的空区块在预处理的时候被移除了；2）评价的时候作为根结点的区块（英语里是 {TOP，S1}，中文里是空字符串) 被忽略了；3）包含一些例如 {:，“，”，.，?，!} 这些标点的区块也被忽略了；请注意中文标点会作为正常的字符被评价；4）一些在同一集合中的标签，例如 {ADVP，PRT}，被认为是等价的.

1.3 相关工作

1.3.1 依存句法分析

目前依存句法分析的两种主流方法分别是基于图的方法和基于转移的方法. **基于图的方法**旨在从一个有向完全图中找到一棵概率（分值）最大的树，**基于转移的方法**将句法树的构建转化为一个移进和归约的动作序列，并寻求得到最优序列.

McDonald 等[23] 首先提出了基于图的训练方法. 他们提出采用基于动态规划的 Eisner 算法从有向完全图中得到分值最大的投影树，然后用 Max Margin 方法来训练. 后来 McDonald 等[24] 将方法拓展到了非投影树解析领域，使用了 ChuLiu/Edmods 算法 (Chu 等[25]) 得到分值最高的树. 由于学习和解码算法的复杂度很高，基于图的方法在给树打分时需要很强的独立性假设，其中最简单的弧分解假设认为树的每条弧相互独立（称为一阶模型），相应的树分值为所有弧的分值之和. 后来有研究者提出放宽独立性假设，将邻接兄弟、祖父、共同父亲等子结构的分值也引入树的打分中，基于此，他们提出了一系列的二阶 (McDonald 等[13], Carreras[26])、三阶 (Koo 等[27]) 模型，并在依存句法上得到了可观的准确率提升. 解码时一阶方法通常采用 Eisner 和 MST 算法来进行投影/非投影解码，高阶方法在使用一阶算法解码时一般需要结合最小贝叶斯风险（Minimum Bayesian Risk, MBR ）解码 (Smith 等[28]). 值得一提的是 McDonald 等[13] 发现二阶非投影解析是 NP-hard 问题，无法设计相应的动态规划结构融入高阶分值，因此基于图的高阶方法仅限于投影解析.

基于转移的方法则旨在将寻找句法树归约为搜索一个最优动作序列的问题. 一个转移系统一般由一个栈以及一个队列组成，分别储存部分解码的序列和未处理的词，接着系统定义了一系列的动作进行入栈/出栈、合并子树等操作，最终消耗完毕所有词，得到一棵完整句法树. 常见的转移系统有 Arc-standard 和 Arc-eager 等. 原始的基于转移的方法采用了局部的贪心算法来逐步预测每个动作，这可能导致错误传播（error propagation）的问题. 后来 Zhang 等[29], Huang 等[30] 引入了全局训练方法，并在解码时采用集束搜索（beam search），显著提升了准确率，达到和基于图方法相近的性能.

进入到深度学习时代，得益于神经网络编码器上下文编码能力的进步，有大量的工作尝试了不同的神经网络模型，Chen 等[14] 首先将前馈神经网络引入到基于转移的方法中，Kiperwasser 等[31], Wang 等[32] 在基于图的模型中引入了双向 LSTM 编码器，

Li 等[33] 则首次将 Transformer 用于依存句法分析模型，依存句法分析的性能得到了飞跃. 其中，Dozat 等[9] 在 3 层双向 LSTM 编码器的基础上，采用了一个简单的基于头选择的训练目标，首次将 PTB 的 LAS 性能推进到了 94.08. 这些模型大都采用了简单的一阶基于图的模型，而基于转移的方法渐渐式微，其中 Ma 等[34] 是最近的基于转移的工作之一，他们不再采用经典的转移系统，而是用 Seq2Seq 框架结合 Pointer Network，达到了和基于图模型相近的性能.

近期开始有研究者探索将传统的结构化学习和高阶建模技术应用到在神经网络中. Zhang 等[21] 探索了在一阶 Biaffine Parser 上结构化学习的作用. 他们在多语言数据上比较了局部头选择损失、全局 Max Margin 损失和 TreeCRF 损失的性能. 他们表明全局的 TreeCRF 损失总体上要稍微好于 Max Margin 损失，并且对大多数语言而言，结构化学习带来了虽然不多但是显著的 LAS 提升. 他们表明结构化学习，特别是 TreeCRF，极大地提升了句子级匹配的准确率，这与我们的观察相仿.

Falenska 等[35] 则提出了一个在依存句法上的分析性工作. 通过将 Kiperwasser 等[31] 的一阶基于图的方法扩展到二阶，他们尝试探究有多少结构化上下文被双向 LSTM 编码器捕捉. 他们拼接 3 层 LSTM 的在 i,k,j 位置的输出向量来给邻接兄弟子树打分，并采用了 Max Margin 训练损失和二阶 Eisner 解码算法 (McDonald 等[13]). 基于他们相对负面的结果和分析，他们认为高阶建模是多余的，因为双向 LSTM 已经可以有效的隐式捕捉足够的结构化上下文. 在我们的工作中，我们使用了一个更强的基线模型，并且得到了相比于他们的工作更显著的 UAS/LAS 提升. 特别地，我们进行了深入的分析，表明显式建模高阶信息可以帮助句法模型，因此与双向 LSTM 编码器是互补的.

Ji 等[15] 引入了高阶结构信息，并在 Biaffine Parser (Dozat 等[9]) 上用图神经网络来隐式建模. 他们使用了多层的图注意力网络（Graph Attention Network，GAT）(Veličković 等[36]) 作为组块，每层 GAT 的每个节点通过集聚邻居节点信息产生新的表示. 通过多次迭代，一个节点渐渐的收集到多跳的高阶信息作为单条弧打分的全局证据. 训练时他们遵循了原始的头选择训练损失. 与此相对的，我们的工作采用的全局的 TreeCRF 损失并显式地将高阶分值引入到了 Biaffine Parser.

1.3.2 成分句法分析

成分句法分析领域最简单的**生成式模型**为概率上下文无关文法（Probabilistic Context-Free Grammars, PCFGs）. PCFGs 将一棵句法树的概率分解为所有产生式 $A \rightarrow B_1 B_2 \ldots B_n$ 的概率之积，其中 A 代表非终端结点，B 代表一个终端/非终端结点，并寻求最大化该概率. Collins[37] 在基于简单的头发现规则（head-finding rules）下为每个非终端结点 A 引入词汇化标记得到 $A(w)$，将概率上下文无关文法拓展为了词汇化的概率上下文无关文法（Lexicalized PCFGs）. 这推动了生成式模型的广泛流行. Matsuzaki 等[38] 和 Petrov 等[39]（Berkeley Parser）利用隐变量对非终端结点进行自动标记，并通过 EM 算法来训练产生细粒度的上下文无关规则，该方法首次超越了 Lexicalized PCFGs 方法. 最近的工作中，Dyer 等[40] 提出了循环神经网络文法（Recurrent Neural Network Grammars, RNNGs），在一个自顶向下的转移系统的基础上额外附加了一个生成动作 $\textsc{Gen}(\cdot)$，最终模型建模句子和句法树的联合概率 $p(\boldsymbol{x}, \boldsymbol{y})$. Kuncoro 等[41] 对 RNNGs 进行了详细的消融性分析，发现转移系统中的栈结构最为重要，相应的他们的 Stack-only RNNGs 达到了当时成分句法分析的最佳性能.

与生成式模型相对的为**判别式模型**，其中包含两种代表性方法，即基于转移的方法和基于图的方法. 和依存句法类似，基于转移的方法包含了一个栈以及一个缓存来存放已有句法树和未处理词，转移系统根据当前动作逐步处理词汇，并根据当前状态预测下一个动作，最终归约出一棵句法树. 根据树的迭代和子树归约的次序不同，存在三类转移系统，分别为基于后序（post-order）遍历（或自底向上）、基于前序（pre-order）遍历（或自顶向下）以及基于中序（in-order）遍历 (Liu 等[42]) 的转移系统. 解码时通常都采用集束搜索. 基于转移的方法通常会遇到两个问题：1）曝光偏置（*exposure bias*），即训练时模型通常面对的都是正确的动作，而解析时可能得到错误预测，根据错误动作继续进行预测时结果可能会越来越差；2）损失函数不匹配（*loss mismatch*），转移系统的训练采用了局部动作级别损失而忽视了全局的结构指标. 对于第一个问题，通常的解决办法为动态指导（dynamic oracle），即在预测偏离正确树之后指导模型向着错误相对更少的状态转移，例如在标签预测时，将不存在于正确树的区块的标签的答案设计为 $\emptyset$，Cross 等[43] 应用了动态指导并给出了详细证明. 对于第二个问题，Fried 等[44] 采取了策略梯度（policy gradient）的方法解决，策略梯度定义了一个风险函数（通常为负 F_1 值），训练时模型用期望风险最小化的训练目标代替了局部动作损失.

我们的成分句法分析器基于判别式的高性能基于双向 LSTM 编码器的现代神经网络分析器 (Stern 等[10])，在基于图的模型上利用了 minus feature (Cross 等[43]) 进行区块的打分. 最近有很多工作都参考了 Stern 等[10]. Gaddy 等[12] 尝试分析什么样的，以及有多少上下文被双向 LSTM 隐式编码. Kitaev 等[45] 将 2 层双向 LSTM 替换为自注意力层，并通过分离上下文和位置注意力，发现了可观的提升. 与此对应的，我们的工作表明通过合理的设置双向 LSTM，例如 Dropout 策略，Stern 等[10] 的分析器可以超过 Kitaev 等[45] 的结果. 请注意 Kitaev 等[45] 的工作中使用了很大的词向量.

与此同时，近期有许多工作没有显式考虑结构化约束或者 CKY 解码，极大简化了成分句法分析任务. Gómez-Rodríguez 等[46] 通过对每个词设计复杂的标签编码树信息，尝试采用序列标注方法解决成分句法分析任务. Vilares 等[47] 通过一系列的增强技术，例如多任务学习和策略梯度，进一步增强了序列标注方法. Shen 等[48] 提出对于正确树中的每个邻居词对预测一个数值距离，并应用自底向上的贪婪解码找到一棵最优的树. 然而，所有上面的工作仍然在解析性能上极大的落后于主流的方法.

1.3.3 TreeCRF 加速方法

TreeCRF 方法在计算句法树概率时需要在全局空间上对句法树的分值进行归一化，这带来了效率低下的问题，因此目前关于 TreeCRF 的句法分析的工作相对较少. Finkel 等[49] 首先提出了一个基于非神经网络的引入丰富特征的 TreeCRF 成分句法分析器. Durrett 等[50] 拓展了 Finkel 等[49] 的工作，使用一个带非线性激活的前馈神经网络来打分产生式. 这两个工作都在 CPU 上显式进行了 Inside-Outside 计算，带来了严重的效率问题.

与此对应的，和 TreeCRF 类似进行结构化学习，但是无需全局归一化的 Max Margin 方法相对要更加流行. Taskar 等[51] 首次在依存句法上提出 Max Margin 解析，训练时最大化正确句法树和其他可能句法树的分值间隔，后来 McDonald 等[23,23] 都采用了这种学习目标，而近期的代表性工作中，Kiperwasser 等[31], Wang 等[32] 均采用了 Max Margin 训练. 当前流行的成分句法模型 (Stern 等[10], Kitaev 等[45], Zhou 等[52]) 通常也采用了 Max Margin 训练方法.

最近也有工作提出对 TreeCRF 等结构化学习算法进行加速. Zhang 等[53], Jiang 等[54], Li 等[55] 作为最近的基于 TreeCRF 的工作，采用了 Cython Programming 并结合多线程，在 CPU 上加快了 Inside-Outside 的计算. 类似地，Kitaev 等[45] 将 Cython

Programming 应用到了 Max Margin 训练上. 而在本文中，我们从两个方面来加速 TreeCRF 算法：1）批次化技术；2）用基于自动求导的反向传播代替 Outside 算法.

对于序列标注任务而言，批次化很直接，并且已经被很好的解决，比如 NCRF++ 的实现[3]，但是在树形结构中这个要复杂得多. 我们首次提出分别在依存和成分句法上批次化树形结构的 Inside 和 Viterbi（Eisner/CKY）计算，以便于利用 GPU 加速.

Eisner[56] 在成分句法分析上提出了一个关于 Outside 算法和反向传播机制等价性的理论证明，并同样讨论了其他类似于依存文法的范式. 我们在本文的工作中借鉴了他们的理论性工作，在依存和成分文法中用反向传播完全取代了 Outside 算法，显著提升了解析效率. 作为一个经验性分析，我们相信我们的尝试能够让 TreeCRF 在现实系统中应用起来.

Torch-Struct[4] (Rush[57]) 是我们的方法之外一个同期独立完成的工作，他们同样为依存句法和成分句法实现了批次化的 Eisner/CKY 算法，并且和我们一样，将 Outside 算法用 Inside 过程以及基于自动求导的反向传播代替. 然而，Torch-Struct 旨在实现通用目的的结构化预测算法. 与此不同的是，我们着力于复杂的解析器模型，并力求达到依存句法分析和成分句法分析在准确率和效率上的最佳性能. 此外，Torch-Struct 关注于一阶算法，而我们分别还在两种句法分析范式中引入了二阶的 Eisner/CKY 算法，并做了大量的针对性实验和分析.

1.3.4 近似推断方法

针对句法分析的推断算法，我们关注于两个问题：1）最大后验概率推断（Maximun *a posteriori* inference, MAP inference），即得到后验概率最大化的句法树；2）边缘概率推断（Marginal inference），即得到句法树中每个变量的边缘概率. 尽管为推断算法引入高阶特征，能够帮助模型准确率的提升 (McDonald 等[13], Carreras[26], Koo 等[27], Ma 等[58])，然而这种通常导致两种情况：1）相应的精确推断算法有很高的复杂度；2）无法得到一个可精确推断的动态规划算法来计算目标结构的概率. 对于前者，我们可以考虑批次化方法和自动求导等技术，利用 GPU 并行计算的能力，降低算法复杂度. 而对于后一种情况，通常我们不得不考虑近似方法. 由于不可精确推断问题的广泛存在，因此一直以来近似推断算法在 NLP 社区都有很多都应用.

[3]https://github.com/jiesutd/NCRFpp

[4]https://github.com/harvardnlp/pytorch-struct

Martins 等[17] 将依存句法分析问题近似为整数线性规划（Integer Linear Programming, ILP）问题. 与传统算法相比，他们利用了大量的局部和全局的丰富特征，例如兄弟、祖父特征和价键特征，作为线性约束，用 ILP 来求解，保证输出是一棵合法的依存树. Koo 等[59] 将非投影依存句法分析形式化为对偶分解（Dual Decomposition, DD）问题，将一个难以求解的问题分解为若干个可求解的子问题，例如可以将输出树的问题分解到单条弧、邻接兄弟、双头等子结构上 (Martins 等[60]).

另一个和本文采用的平均场变分推断高度相关的方法是循环置信传播（Loopy Belief Propagation, LBP）. LBP 定义了一个包含多种局部和全局特征的由变量和因子构成的因子图，接着算法迭代式地让因子（变量）收集邻居变量（因子）的信息，最终收敛后得到后验概率. Smith 等[16], Gormley 等[61] 在依存句法分析上引入了 LBP. 除了上面提到的兄弟（Sib）、祖父（Grand）因子，他们还引入了丰富的全局因子，例如 Exactly1 约束每个词有且仅有一个头，Tree 要求所有词必须组成一棵树等等. Naradowsky 等[62] 将 LBP 引入到了成分句法分析，也采用了类似的高阶因子（如 Tree）在成分句法上的变体. 与精确算法相比，LBP 可以用可接受的复杂度引入大量全局因子，并为模型带来了可观的性能收益.

还有一些工作同时使用了上述的近似算法. Auli 等[63] 在组合范畴文法中同时采用了 LBP (Smith 等[16]) 以及 DD (Koo 等[59])，取得了最佳性能，并对两种方法做了经验性比较. Martins 等[64] 提出了在非投影依存句法分析中同时利用了 LBP (Smith 等[16]) 以及 ILP (Martins 等[17]) 方法，表明两种方法采用了一致的因子图，并在局部近似的目标函数的理念上是相一致的.

近期在神经网络模型上，涌现出了大量的关于近似推断在语义角色标注 (Li 等[65])、序列标注 (Wang 等[66]) 等结构化预测任务等上的应用. Wang 等[18] 首次在基于图的语义依存分析上提出了二阶模型，并采用 LBP 和 MFVI 作为近似算法，观察到了两种推断算法相似的性能提升. Wang 等[67] 首次在依存句法分析中引入了 MFVI，他们采用了兄弟、组合和共同父亲这三种二阶特征. 我们的依存模型中采用的基于头选择的 MFVI 参考了 Wang 等[67] 的更新方法，但是为了和精确推断算法公平比较，本文中我们只保留了兄弟特征. 上述工作引入了近似推断算法之后都达到了和精确推断可比较的结果，同时速度上大大提升，这和我们的发现是一致的.

1.4　章节和内容安排

本文共分为五个章节，各章节具体安排如下:

第一章：绪论. 本章介绍本文中每个工作的任务背景和意义，并阐述一下与工作相关的有关文献的进展和内容，以及与我们方法的对比.

第二章：基于树形条件随机场的高阶依存句法分析方法. 本章在当前最佳的依存句法分析器的基础上，提出了一个二阶树形条件随机场的拓展，进一步提升了句法分析的性能. 为了解决效率问题，我们还提出了批次化技术在 GPU 上对训练和解码算法进行加速，以及用高效的反向传播代替 Outside 算法.

第三章：基于树形条件随机场的高阶成分句法分析方法. 本章将树形条件随机场的应用到了神经成分句法分析器当中. 我们应用了和依存模型一样的加速方法解决了树形条件随机场效率低下的问题. 我们提出了一个简单的两阶段解析方法，来进一步提升分析器的效率. 参考依存句法的模型架构和参数设置，我们还为解析器引入了新的打分方法，以及有效的 Dropout 策略，使得解析器达到了新的最佳水平.

第四章：基于变分推断的高效句法分析方法. 本章在前面章节提出的高阶句法分析器的基础上，提出利用平均场变分推断方法来近似推断后验概率. 与精确推断的高阶 TreeCRF 方法相比，变分推断显著降低了复杂度，并达到了和高阶 TreeCRF 接近或相当的结果.

第五章总结与展望. 本章总结本文的主要内容，并展望后续可能的句法分析方法.

第二章 基于 TreeCRF 的高阶依存句法分析

本章深入详细的比较了与当前最佳的 Biaffine Parser 所采用的局部训练相比，基于全局 TreeCRF 训练的方法的效果，并首次提出了在神经依存句法分析器上应用一个二阶 TreeCRF 的拓展. 由于高复杂度和低效一直以来都是困扰 TreeCRF 的广泛应用的一个重要原因，因此我们提出了一个能够高效批次化 Inside 算法和 Eisner 算法的方法，来支持 TreeCRF 在 GPU 上的大规模张量并行计算. 并且，我们还通过反向传播机制，避免了复杂 Outside 算法的计算. 我们在 13 个语言的 27 个数据集上进行了实验，实验结果和分析表明，在深度学习时代之前的技术，诸如结构化学习（全局 TreeCRF 损失函数）和高阶建模仍然是有益的，并且可以进一步在最好的 Biaffine Parser 的基础上提升性能，尤其是在局部标注的场景下.

2.1 引言

依存句法分析任务是 NLP 领域的一个基础性任务，由于其简洁性，以及可以方便地在多语言上获得句法和语义信息的特性，目前在这一任务上已经有了大量的研究. 目前在依存句法分析任务上有两个主流方法，分别是基于转移（transition-based）的方法，和基于图（graph-based）的方法，这里我们的方法主要关注于基于图的解析方式.

在深度学习时代之前，基于图的解析依赖于很多手工特征的设计，比如词性、前缀、后缀等等. 与神经网络方法相比，以前的方法有两个显著的不同. 首先，对于非神经网络方法而言，结构化学习（structured learning）是不可或缺的，即训练时需要显式地建模树结构的约束. 通常此类方法采用的是 Max Margin 训练算法，首先用当前模型训练一棵分值最高的树，然后更新模型参数，以保证正确的树的分值要高于预测树.

第二个显著的区别在于高阶特征的使用. 高阶特征为模型带来了显著的提升. 基础的一阶模型将句法树的分值分解为若干条独立的弧的分值 (McDonald 等[23]). 后续的工作进一步引入了二阶依存弧对的分值，比如邻接兄弟 (McDonald 等[13]) 和祖父-父亲-孩子这样的弧对 (Carreras[26], Koo 等[27])，这些高阶扩展都带来了模型性能

的显著提升[1]. 但是这些高阶模型需要引入更复杂的解码算法，导致了模型更加低效.

相比之下，基于图的神经网络依存句法分析器的发展呈现出相反的趋势. Pei 等[68] 提出利用前馈神经网络来自动学习 Chen 等[14] 的若干特征组合，并计算子树得分. 他们的工作表明引入二阶邻接兄弟子树的分值显著提高了性能. 随后，Wang 等[32] 和 Kiperwasser 等[31] 都建议使用双向 LSTM 作为编码器，以及在一阶模型中利用 minus feature 来对单条弧打分. 这三个代表性方法都采用了全局的 Max Margin 方法. Dozat 等[9] 提出了一种强大而高效的 Biaffine Parser，并在各种数据集和语言上获得了最先进的精度. Biaffine Parser 也是一阶的，通过对每个词进行局部头选择（head selection）的方式 (Zhang 等[69])，采用了更简单、更有效的非结构化训练方法.

基于这些对比，我们尝试在基于图的分析器的基础上，将前深度学习时代的一些方法与神经网络模型做一下连接. 这里要解决的**第一个问题**是：以前的一些技术，比如结构化学习和高阶建模，能够进一步提升当前最佳的分析器 *Biaffine Parser* 的性能吗[2]，如果可以，他们在哪些方面是有用的?

对于结构化学习而言，相比 Max Margin 方法，我们采用更复杂且更不常用的 TreeCRF. 主要原因有两方面. 首先，概率分布估计一直是当前数据驱动的 NLP 方法的一个核心的问题 (Le 等[70]). 如果将分析器的输出应用到更高层的任务，一棵句法树的概率 $p(\boldsymbol{y} \mid \boldsymbol{x})$ 比没有上下边界的分值 $s(\boldsymbol{x}, \boldsymbol{y})$ 一般而言要更加有用. 其次，边缘概率是一种理论上比较可靠的方法来评估模型输出子树的置信度，可以用于最小贝叶斯风险（MBR）解码 (Smith 等[28])，并且已经被证明了对于词级别基于局部标注句法树的主动学习（active learning）(Li 等[71]) 很有用.

尽管很有用，但是 TreeCRF 不如 Max Margin 那么流行，其中一个原因是由于 Inside-Outside 算法的高复杂度，尤其是 Outside 算法. 据我们所知，所有现存的模型都是在 CPU 上运算 Inside-Outside 算法. 而由于 CPU/GPU 巨大的效率差异，这一低效的问题在深度学习时代变得更加严重. 这就引发了**第二个问题**: 我们是否能够批次化 *Inside-Outside* 算法，并且直接在 *GPU* 上进行计算？如果这样的话，我们就能够利用诸如 PyTorch 这样的高效深度学习张量库来进行计算，并且将高效的 TreeCRF 应用到更多的场景 (Le 等[70], Cai 等[72]).

总体而言，针对上面的两个问题，我们在本章节的贡献如下：

[1] 三阶和四阶模型的提升不大，这可能是由于特征稀疏的问题导致的 (Koo 等[27], Ma 等[58]).

[2] 尽管最近的一些工作汇报了相比 Biaffin Parser 更高的性能，但是都引入了一些外部资源，比如大规模语言模型的上下文词表示. 在相同网络和相同实验设置的场景下，这些工作的结果都是相近的.

- 我们第一次提出将二阶 TreeCRF 应用到神经依存句法分析中. 我们还提出了一个高效的 Triaffine 结构来对于二阶子树打分.
- 我们提出通过 GPU 上大规模的并行张量计算来批次化 Inside 算法，来进行更高效的 TreeCRF 损失函数的计算. 我们表明复杂的 Outside 算法对于梯度和边缘概率的计算而言已不再必须，相应的可以用高效的反向传播代替.
- 我们在 13 个语言的 27 个树库上进行了实验. 结果和分析都表明，在深度学习时代，结构化学习和高阶建模在许多方面对当前最好的 Biaffine Parser 仍然是有用的.

2.2 基线模型

我们复现了当前最好的 Biaffine Parser (Dozat 等[9])，并在两方面对其做了修改：1) 用 CharLSTM 代替原来的词性 embedding，2) 用一阶 Eisner 算法 (Eisner[73]) 代替原来的 MST 算法来进行投影树解码.

2.2.1 打分方法

图 2-1展示了我们的打分方法，一共由四个部分组成.

输入向量. 第 i 个输入向量可以分为两部分：词向量，以及词 w_i 的 CharLSTM 表示向量.

$$\mathbf{e}_i = \text{emb}(w_i) \oplus \text{CharLSTM}(w_i) \tag{2.1}$$

其中 CharLSTM(w_i) 是由先将 w_i 输入到一个双向 LSTM，然后拼接两个最后的隐向量获得 (Lample 等[74]). 我们发现用 CharLSTM(w_i) 替换原有的词性 embedding 会带来稳定的提升，并且也简化了实验流程，尤其是对于多语言实验而言，无需再用到词性标注器来额外生成输入句子的词性.

双向 LSTM 编码器. 为了编码句子级别的上下文，分析器对输入 $\mathbf{e}_0 \dots \mathbf{e}_n$ 应用了三层双向 LSTM. 顶层的第 i 个词的输出向量被表示为 $\mathbf{h}_i$.

MLP 特征提取层. 两个共享的 MLP 层被应用到了 $\mathbf{h}_i$ 上来获取相应的低维向量，通过这种方式只保留句法相关的信息

$$\mathbf{r}_i^h; \mathbf{r}_i^m = \text{MLP}^{h/m}(\mathbf{h}_i) \tag{2.2}$$

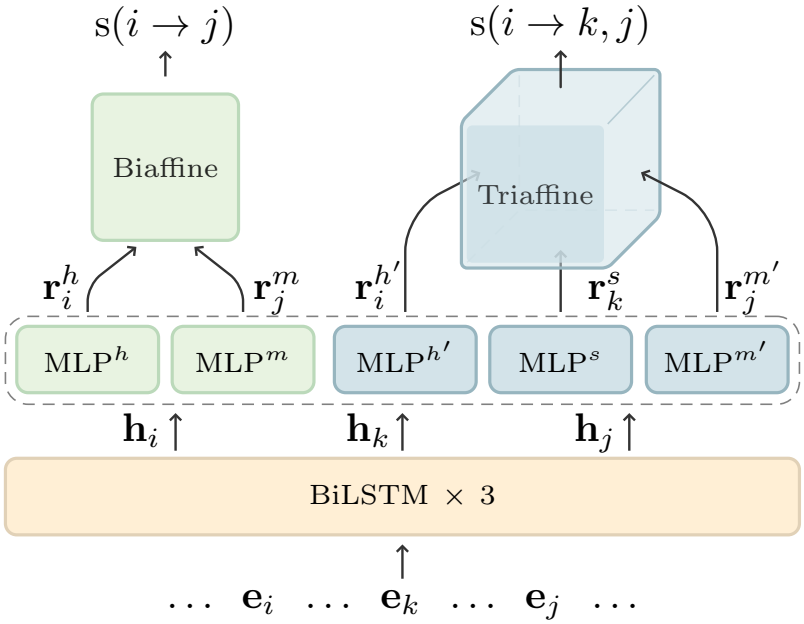


图 2-1　带二阶扩展的打分方法.

其中 $\mathbf{r}_i^h$ 和 $\mathbf{r}_i^m$ 是词 w_i 分别作为一条弧的头和修饰词的表示.

Biaffine 打分器. Dozat 等[9] 首次提出通过 biaffine attention 计算一条依存弧 $i \to j$ 的分值

$$\mathrm{s}(i \to j) = \begin{bmatrix} \mathbf{r}_j^m \\ 1 \end{bmatrix}^{\mathrm{T}} \mathbf{W}^{biaffine} \mathbf{r}_i^h \tag{2.3}$$

其中 $\mathbf{W}^{biaffine} \in \mathbb{R}^{(d+1)\times(d)}$. 这种计算方式在 GPU 上进行尤其高效.

2.2.2　局部头选择训练损失

Biaffine Parser 采用了一个简单的词级别非结构化损失函数，试着对每个词独立最大化正确头的局部概率. 在一个训练实例中，对于一个正确的头-依赖对 (w_i, w_j)，其对应的交叉熵损失为

$$L(i \to j) = -\log \frac{\exp(\mathrm{s}(i \to j))}{\sum_{0 \le i' \le n} \exp(\mathrm{s}(i' \to j))} \tag{2.4}$$

换句话说，模型的训练是基于一个简单的头选择目标，最终所有词的损失都被累加起来，而没有考虑任何树结构的约束.

(a) 单弧 (b) 邻接兄弟

图 2-2 两种打分的子树结构.

2.2.3 解码方法

Biaffine Parser 将骨干树的搜索和依存弧标签标注视为两个独立的任务. 这里为了方便我们采取了相同的两阶段解析策略. 首先，给定所有依存弧的分值，我们采用复杂度为 $O(n^3)$ 的一阶 Eisner 算法来寻找最优的骨干句法树.

$$\boldsymbol{y}^* = \arg\max_{\boldsymbol{y}} \mathrm{s}(\boldsymbol{x}, \boldsymbol{y}) \equiv \sum_{i \to j \in \boldsymbol{y}} \mathrm{s}(i \to j) \tag{2.5}$$

其次，给定最优的无标签骨干树，我们采用局部贪婪分类的方法为树中的每条弧分配一个标签，得到一棵最优的有标签树，具体细节可以参考 Dozat 等[9].

2.3 二阶 TreeCRF

我们从两方面极大拓展了 Biaffine Parser：使用概率 TreeCRF 来进行结构化训练，以及显式引入高阶子树的分值.

具体地，我们在基本的一阶模型的基础上引入了二阶邻接兄弟子树的分值:[3]

$$\mathrm{s}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i \to j \in \boldsymbol{y}} \mathrm{s}(i \to j) + \sum_{i \to \{k,j\} \in \boldsymbol{y}} \mathrm{s}(i \to \{k, j\}) \tag{2.6}$$

其中 k 和 j 是 i 的两个相邻的孩子，并且满足 $i < k < j$ 或者 $j < k < i$. 图2-2展示了两种我们要打分的子树结构.

作为一个概率模型，TreeCRF 以下面的方式计算一棵树的条件概率

$$p(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{\exp(\mathrm{s}(\boldsymbol{x}, \boldsymbol{y}))}{Z(\boldsymbol{x}) \equiv \sum_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x})} \exp(\mathrm{s}(\boldsymbol{x}, \boldsymbol{y}'))} \tag{2.7}$$

[3]我们还可以进一步扩展，引入祖父-父亲-孩子的子树分值，然后基于 Koo 等[27] 提出的 $O(n^4)$ 的 Viterbi 算法解码. 这里我们留待作为未来的工作.

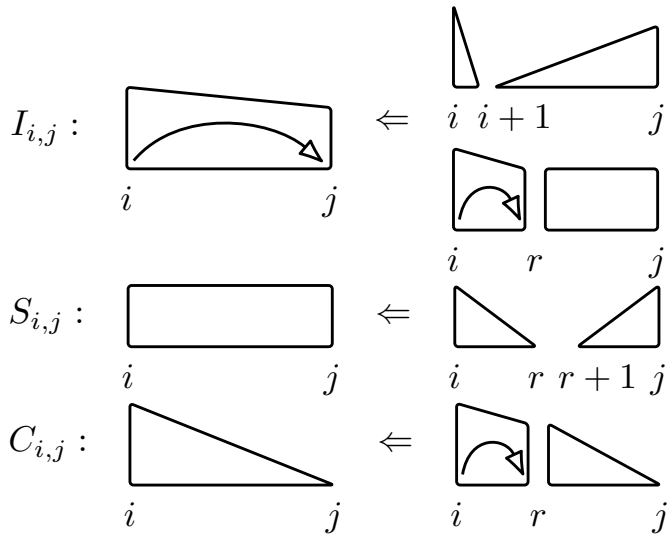


图 2-3　基于自底向上动态规划的二阶 Inside 算法图示.

其中 $\mathcal{Y}(\boldsymbol{x})$ 是输入句子 $\boldsymbol{x}$ 对应所有合法的句法树，$Z(\boldsymbol{x})$ 通常被称为配分项（partition term）.

训练时，TreeCRF 应用了下面的损失函数来最大化给定 $\boldsymbol{x}$ 正确句法树 $\boldsymbol{y}$ 的条件概率.

$$\begin{aligned} L(\boldsymbol{x}, \boldsymbol{y}) &= -\log p(\boldsymbol{y} \mid \boldsymbol{x}) \\ &= -\mathrm{s}(\boldsymbol{x}, \boldsymbol{y}) + \log Z(\boldsymbol{x}) \end{aligned} \tag{2.8}$$

2.3.1　基于 Triaffine 的二阶子树打分

为了避免对原有打分方法有大的改动，我们采用了一个直接的扩展来获得邻接子树的分值. 首先，我们用三个额外的 MLP 层来进行和 Biaffine 所需要的类似的特征提取

$$\mathbf{r}_i^{h'}; \mathbf{r}_i^{s}; \mathbf{r}_i^{m'} = \mathrm{MLP}^{h'/s/m'}(\mathbf{h}_i) \tag{2.9}$$

$\mathbf{r}_i^{h'}; \mathbf{r}_i^{s}; \mathbf{r}_i^{m'}$ 是词 w_i 分别作为头，兄弟，和依赖对应的表示.[4]

然后，我们提出了一个对 Biaffine 自然的扩展，Triaffine 结构，来通过上述的三

[4]另一种方式是重用一阶部分头和依赖的表示，但是我们的前置实验表明这种方法的结果要更差一点

算法 1 二阶 Inside 算法.

define: $I, S, C \in \mathbb{R}^{n \times n \times B}$ ▷ B is #sents in a batch
initialize: $C_{i,i} = 0, 0 \le i \le n$
for $w = 1$ **to** n **do** ▷ span width
  Batchify: $0 \le i;\ j = i + w \le n$
  $I_{i,j} = \log\left(\exp\left(C_{i,i} + C_{j,i+1}\right) + \sum_{i<r<j} \exp\left(I_{i,r} + S_{r,j} + \mathrm{s}(i \to \{r, j\})\right)\right) + \mathrm{s}(i \to j)$
  $S_{i,j} = \log \sum_{i \le r < j} \exp\left(C_{i,r} + C_{j,r+1}\right)$
  $C_{i,j} = \log \sum_{i < r \le j} \exp\left(I_{i,r} + C_{r,j}\right)$
end for ▷ refer to Fig. 2-3
return $C_{0,n} \equiv \log Z(\boldsymbol{x})$

个向量计算分值[5]

$$\mathrm{s}(i \to \{k, j\}) = \begin{bmatrix} \mathbf{r}_k^s \\ 1 \end{bmatrix}^{\mathrm{T}} \mathbf{r}_i^{h'\mathrm{T}} \mathbf{W}^{triaffine} \begin{bmatrix} \mathbf{r}_j^{m'} \\ 1 \end{bmatrix} \tag{2.10}$$

其中 $\mathbf{W}^{triaffine} \in \mathbb{R}^{d' \times (d'+1) \times (d'+1)}$ 是一个三维的张量. Triaffine 可以方便的通过 einsum 函数在 GPU 上高效计算.

2.3.2 高效的 TreeCRF 计算方法

如公式 2.8所示，计算 TreeCRF 损失的关键在于如何计算配分项 $Z(\boldsymbol{x})$. 这个问题在前深度学习时代的非神经网络模型上已经被很好的解决了. 我们可以在 Viterbi 算法的基础上，直接用 sum product 操作取代 max product 操作，这样就可以用相同的多项式时间的复杂度得到 $Z(\boldsymbol{x})$. 然而，对非神经网络模型而言，由于缺乏自动求导机制，单独的计算 Inside 算法仍然是不够的. 为了获取边缘概率，然后得到特征权重的梯度，我们还需要实现更加复杂的 Outside 算法，而这通常要两倍慢于 Inside 算法. 这可能是 TreeCRF 在非神经网络上不如 Max Margin 流行的主要原因.

据我们所知，所有以前的神经网络上 TreeCRF 解析的工作都显式地实现了 Inside-Outside 算法来进行梯度的计算 (Zhang 等[21], Jiang 等[54]). 为了提升效率，这些计算通常被从 GPU 迁移到 CPU，用 Cython 加速.

这里我们表明 Inside 算法可以被高效的批次化，从而充分利用 GPU 并行计算的能力. 图 2-3和算法 1展示了批次化版本的二阶 Inside 算法，这是 McDonald 等[13] 的

[5]我们同样尝试了 Wang 等[18] 的方法，他们用了三个 biaffine 操作来模拟三个输入向量的交互，但是这种方法的结果相对要差一点. 篇幅限制我们省略了相应结果的汇报.

二阶 Eisner 算法的一个直接扩展，将其中的 max product 操作用 sum product 取代. 为了简便，我们忽略了自右边位置 j 向左边位置 i 归并生成 incomplete/complete/sibling 区块的描述.

具体地，对一个批次中 B 个句子，我们首先将不同位置但是相同宽度的区块 (i, j) 的分值打包成为一个大的张量. 接着，我们在 GPU 上通过高效的大规模张量并行操作来同时进行计算和归并. 我们还可以用类似的方式来批次化解码算法，这里省略细节.

值得注意的是，这里描述的技术同样可以在其他形式的文法里应用，比如 CKY 算法形式的成分句法分析 (Finkel 等[49], Drozdov 等[75]). 具体在章节 三中描述.

2.3.3　通过反向传播完成的 Outside 算法

Eisner[56] 以成分句法分析（短语结构）为例，对反向传播机制和 Outside 算法的等价性进行了理论证明. 这里我们同样在依存句法上验证了其等价性.

进一步地，我们同样发现边缘概率 $p(i \rightarrow j \mid \boldsymbol{x})$ 直接与对 $\log Z(\boldsymbol{x})$ 关于分值 $\mathrm{s}(i \rightarrow j)$ 的偏导相关，这可以很方便的证明

$$
\begin{aligned}
\frac{\partial \log Z}{\partial \mathrm{s}(i \rightarrow j)} &= \frac{\partial \log Z}{\partial Z} \cdot \frac{\partial Z}{\partial \mathrm{s}(i \rightarrow j)} \\
&= \frac{1}{Z} \cdot \frac{\partial \sum_{\boldsymbol{y}} \exp\left(\mathrm{s}(\boldsymbol{x}, \boldsymbol{y})\right)}{\partial \mathrm{s}(i \rightarrow j)} \\
&= \frac{1}{Z} \cdot \frac{\partial \sum_{\boldsymbol{y}: i \rightarrow j \in \boldsymbol{y}} \exp\left(\mathrm{s}(\boldsymbol{x}, \boldsymbol{y})\right)}{\partial \mathrm{s}(i \rightarrow j)} \\
&= \frac{1}{Z} \cdot \frac{\sum_{\boldsymbol{y}: i \rightarrow j \in \boldsymbol{y}} \exp\left(\mathrm{s}(\boldsymbol{y}) - \mathrm{s}(i \rightarrow j)\right) \cdot \partial \exp(\mathrm{s}(i \rightarrow j))}{\partial \mathrm{s}(i \rightarrow j)} \\
&= \sum_{\boldsymbol{y}: i \rightarrow j \in \boldsymbol{y}} \frac{\exp\left(\mathrm{s}(\boldsymbol{y})\right)}{Z} \\
&= \sum_{\boldsymbol{y}: i \rightarrow j \in \boldsymbol{y}} p(\boldsymbol{y} \mid \boldsymbol{x}) \\
&= p(i \rightarrow j \mid \boldsymbol{x})
\end{aligned}
\tag{2.11}
$$

因此偏导数等价于所有包含 $i \rightarrow j$ 的弧的句法树的概率之和，即边缘概率 $p(i \rightarrow j \mid \boldsymbol{x})$. 对于 TreeCRF 分析器，我们可以通过将解码算法中的分值用边缘概率代替，进行**最小贝叶斯风险**（Minimum Bayesian Risk, MBR）解码 (Smith 等[28])，从而带来一个稳定的提升，相关推导见附录 A.

2.3.4 局部标注处理

作为一个有吸引力的研究方向，很多研究证明构建或者收集局部标注数据很有效 (Nivre 等[76], Hwa[77], Pereira 等[78])，在依存句法分析中，一个句子可以关联一个局部标注树. 当结合主动学习（active learning）时，局部标注可以发挥更大的作用，由于只需要标注部分较难的子树结构，这样可以极大减轻标注者的标注负担. Li 等[71] 中有关于这部分的详细调研. 此外，Peng 等[20] 最近基于这方面的研究公开了一个局部标注的多领域中文树库.

那么问题就变成了如何利用局部标注数据来训练模型. Li 等[71] 基于这个目的提出扩展 TreeCRF 到局部标注场景，并在非神经网络模型上取得了不错的效果. 这里我们借鉴他们的研究到了神经网络模型中. 我们尤其关注于在结构化学习和高阶建模上利用局部标注数据.

对基本的基于一阶局部训练方法的 Biaffine Parser 来说，利用局部标注最直接的方式是忽略未标注的词. 与此相对的是，树结构约束允许标注弧影响未标注词的概率分布，并且高阶建模进一步促进了词之间的交互. 因此，直觉上结构化学习和高阶建模都是非常有用的.

对于局部标注，我们和 Li 等[71] 一样，定义训练的目标函数如下:

$$
\begin{aligned}
L(\boldsymbol{x}, \boldsymbol{y}^p) &= -\log \sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}); \boldsymbol{y} \supseteq \boldsymbol{y}^p} p(\boldsymbol{y} \mid \boldsymbol{x}) \\
&= -\log \frac{Z(\boldsymbol{x}, \boldsymbol{y}^p) \equiv \sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}); \boldsymbol{y} \supseteq \boldsymbol{y}^p} \exp(\mathrm{s}(\boldsymbol{x}, \boldsymbol{y}))}{Z(\boldsymbol{x})}
\end{aligned}
\tag{2.12}
$$

其中 $Z(\boldsymbol{x}, \boldsymbol{y}^p)$ 只包含所有与给定局部标注树兼容的句法树，并且可以用和 $Z(\boldsymbol{x})$ 类似的方式进行高效的计算.

2.4 实验结果及分析

参数设置. 我们直接采用了 Dozat 等[9] 的大部分参数设置，包含 Dropout 和初始化策略. 对于 CharLSTM，和 Lample 等[74] 一样，输入字向量的维度是 50，输出向量的维度是 100. 对于二阶模型，我们设置 $\mathbf{r}_i^{h'/s/m'}$ 的维度为 100（进一步提升到 300 带来的提升很小）. 我们训练模型最多 1,000 次迭代，并且如果在 dev 上的性能连续 100 次不提升那就停止训练.

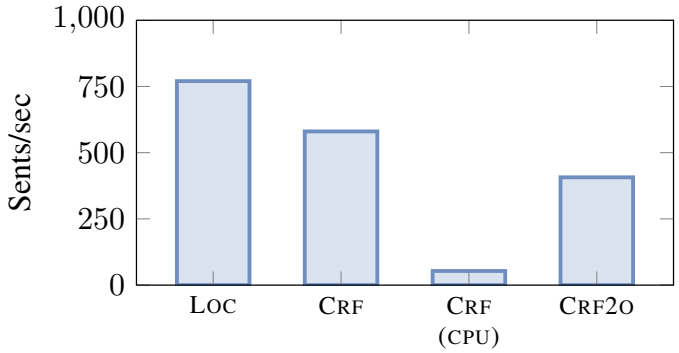


图 2-4　PTB 的 test 数据上的速度比较.

模型定义. LOC 使用了局部交叉熵训练损失函数，并且解码时使用 Eisner 算法来获取最优的投影树. CRF 和 CRF2O 各自代表一阶和二阶 TreeCRF 模型. LOC^{MST} 代表基本的基线模型，但是解码时采用和 Dozat 等[9] 一样的 MST 算法输出非投影树.

2.4.1　效率比较

图 2-4在 PTB 的测试集上比较了不同模型的解析速度. 为了能够公平比较，我们在相同的机器上运行所有的模型，模型的配置为 Intel Xeon CPU (E5-2650v4, 2.20GHz) 以及 GeForce GTX 1080 Ti GPU. "CRF (CPU)" 表示直接在 CPU 运算 Inside-Outside 算法上，并利用 Cython 加速的模型. 由于句子间是相互独立的，因此这里还利用的多线程. 我们一共用了 4 个线程，并且进一步增加没有更多的提升速度.

可以看到通过批次化 Inside 算法以及用自动求导机制的反向传播代替 Outside 算法这两方面的改进，TreeCRF 效率被极大的提升了. 对于 CRF 模型，我们的实现可以解析超过 500 句每秒，大约十倍快于多线程的 "CRF (CPU)". 对于二阶的 CRF2O，我们的分析器达到了 400 句每秒的速度，达到了实时系统的需求.

2.4.2　主要结果

表 2-1列出了 Dev 和 Test 数据上的结果. 两个数据上的结果趋势大部分是一致的. 为了能够与前人的结果公平比较，我们只比较了不利用外部资源的工作，例如 ELMo (Peters 等[79]) 或 BERT (Devlin 等[80]). 可以看到我们的基线模型 LOC 在 PTB 和 CoNLL09 相较于前人达到了最好的性能

在 PTB 上，CRF 和 CRF2O 没有能够进一步提升解析准确率，这部分是由于这两个数据上的结果已经很高了. 然而，正如章节2.4.3所示，结构化学习和高阶建模仍然

表 2-1 主要结果. 我们在 test 上基于 Loc 做了显著性检验，其中 “†” 代表 $p < 0.05$，“‡” 代表 $p < 0.005$.

		Dev		Test	
		UAS	LAS	UAS	LAS
PTB	Dozat 等[9]	-	-	95.74	94.08
	Falenska 等[35]	-	-	-	91.59
	Li 等[33]	95.76	93.97	95.93	94.19
	Ji 等[15]	95.88	93.94	95.97	94.31
	Zhang 等[21]	-	-	-	93.96
	Loc	95.82	93.99	96.08	94.47
	Crf w/o MBR	95.74	93.96	96.04	94.34
	Crf	95.76	93.99	96.02	94.33
	Crf2o w/o MBR	**95.92**	**94.16**	**96.14**	**94.49**
	Crf2o	95.90	94.12	96.11	94.46
CoNLL09	Dozat 等[9]	-	-	88.90	85.38
	Li 等[33]	88.68	85.47	88.77	85.58
	Loc	89.07	86.10	89.15	85.98
	Crf w/o MBR	89.04	86.04	89.14	86.06
	Crf	89.12	86.12	89.28	86.18†
	Crf2o w/o MBR	89.29	86.24	89.49	86.39
	Crf2o	**89.44**	**86.37**	**89.63**‡	**86.52**‡
NLPCC19	Loc	77.01	71.14	76.92	71.04
	Crf w/o MBR	77.40	71.65	77.17	71.58
	Crf	77.34	71.62	77.53‡	71.89‡
	Crf2o w/o MBR	77.58	71.92	77.89	72.25
	Crf2o	**78.08**	**72.32**	**78.02**‡	**72.33**‡

带来了一些正面的影响.

Crf 在 CoNLL09 上显著超越了 Loc，并且 Crf2o 可以进一步提升性能.

在局部标注数据 NLPCC19 上，Crf 极大的超过了 Loc，这表明了结构化学习在局部标注场景下的有效性. Crf2o 通过显式地建模二阶子树的特征，进一步的提升了解析性能. 这些结果验证了我们在章节 2.3.4的直觉性讨论. 需要注意的是，局部标注的结果看起来比较低，这里的原因是局部标注的词通常来说对于模型都是比较难的.

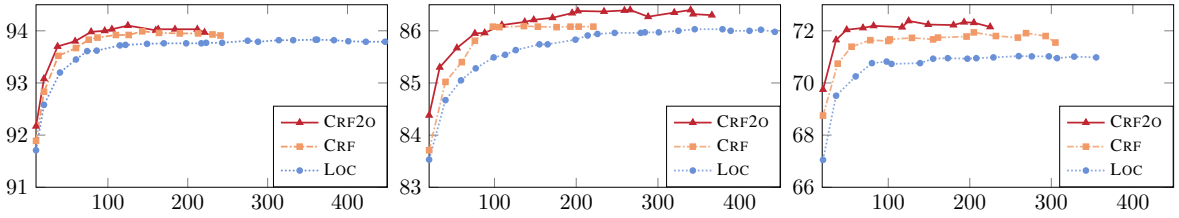


图 2-5　PTB、CoNLL09 和 NLPCC19 的 dev 数据的训练收敛曲线（LAS 相对于训练迭代次数).

表 2-2　test 数据上子树和完全树的结果.

		SIB			UCM	LCM
		P	R	F		
PTB	LOC	91.16	90.80	90.98	61.59	50.66
	CRF	91.24	90.92	91.08	61.92	50.33
	CRF2O	**91.56**	**91.11**	**91.33**	**63.08**	**50.99**
CoNLL09	LOC	79.20	79.02	79.11	40.10	28.91
	CRF	79.17	79.55	79.36	40.61	29.38
	CRF2O	**81.00**	**80.63**	**80.82**	**42.53**	**30.09**

2.4.3　分析

MBR 解码的影响. 对于 CRF 和 CRF2O，我们默认进行 MBR 解码，也就是直接在边缘概率上应用 Eisner 算法 (Smith 等[28]) 来找到最优的句法树.

$$\boldsymbol{y}^* = \arg\max_{\boldsymbol{y}} \sum_{i \to j \in \boldsymbol{y}} p(i \to j \mid \boldsymbol{x}) \tag{2.13}$$

表 2-1报告了 MAP 解码（即根据依存分值直接寻找最优句法树，关于与 MBR 解码的联系详见附录 A）. 除了 PTB 数据集的结果已经比较高了之外，MBR 解码对于 CRF 和 CRF2O 都带来了不多但是一致的提升.

收敛行为. 图 2-5比较了不同模型的收敛曲线. 清晰起见，我们每 20 次迭代选择一个结果最高的数据点放入到图里面. 可以明显看到，结构化学习和高阶建模都带来了一致的提升. CRF2O 稳定的达到了最好的结果，并且收敛要远远快于基本的 LOC.

子树和完全树的结果. 除了弧级别的准确率（UAS/LAS），我们还希望能够评估模型在子树和完全树上的性能. 表 2-2展示了相关的结果. 这里我们忽略了局部标注的 NLPCC19 数据. UCM（Unlabeled Complete Match）代表无标签完全匹配率，即所有

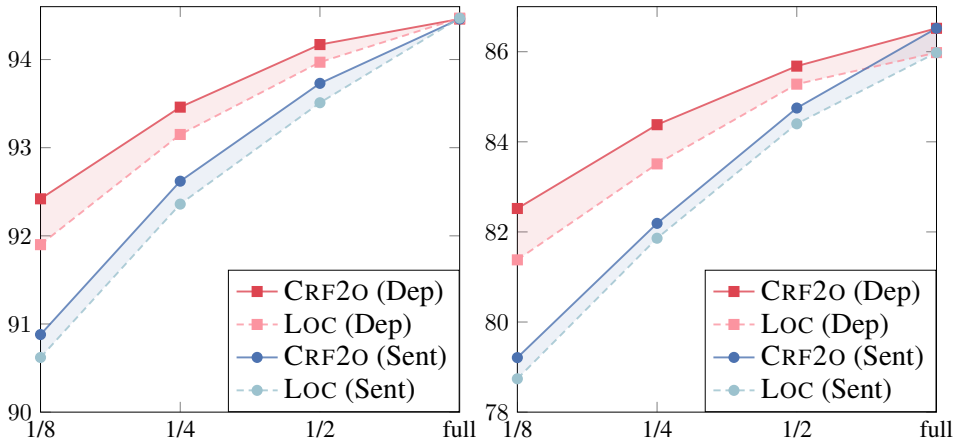


图 2-6 PTB（左）和 CoNLL09（右）的测试集有关训练数据量（弧数以及句子数）变化之后的 LAS 比较.

对应无标签树完全正确的句子的比率，而 LCM 进一步要求树上的所有标签也是正确的.

对于 SIB，我们评估了模型在邻接兄弟子树上的结果（系统输出和对应的正确子树）. 根据公式 2.6，当且仅当词 w_k 和词 w_j 都是词 w_i 的同侧孩子，并且没有其他词 w_i 的孩子在这两者之间，才称 $i \rightarrow \{k, j\}$ 是一个邻接兄弟子树. 给定两棵树，我们可以收集所有的邻接兄弟子树，然后组成一个三元组集合. 之后我们评估相应的 P/R/F_1 值. 请注意在局部标注下评估 SIB 是做不到的.

可以明显看到，通过建模邻接兄弟子树的分值，SIB 的性能相比于 CRF 和 LOC 得到了更大的提升，并且这进一步带来了在完全树匹配率（UCM/LCM）上的进步.

学习局部标注树的能力. 为了更好的理解为什么 CRF2O 在局部标注的 NLPCC19 上表现的比较好，我们通过随机丢弃一定比例的训练数据来进行了更多的比较实验. 我们要么随机丢弃了一部分的句子（完全树），要么每个句子随机丢弃一部分弧（局部树）. 图 2-6列出了结果.

可以看到当我们渐渐地丢弃一定数量的训练句子的时候，性能差距相当稳定. 与此相反的是，当每个训练句子有更少的标注弧时，性能差距明显变得更大了. 这表明 CRF2O 在利用局部标注数据训练模型上要优于 LOC.

2.4.4 多语言 UD 数据集上的结果

表 2-3比较了 UD 数据上不同模型的结果. 在 UD 数据中包含了大量的非投影树. 我们采用了 Nivre 等[81] 的伪投影方法来将许多语言中大量存在的非投影树转换为投

表 2-3　UD2.2 和 UD2.3 的 test 数据的 LAS 结果. 同样地，† 和 ‡ 各自表示基于 Loc 分析器，$p < 0.05$ 以及 $p < 0.005$ 的显著性级别.

	BG	CA	CS	DE	EN	ES	FR	IT	NL	NO	RO	RU	AVG.
						UD2.2							
LOC	90.45	91.14	90.97	80.02	86.83	90.56	87.76	91.14	87.72	90.74	86.20	93.01	88.88
LOC^{MST}	90.44	91.11	91.04	80.21	86.86	90.67	87.99	91.19	88.24	90.35	86.24	93.01	88.95
CRF	90.73	91.25	91.01	**80.56**†	86.92	90.81†	**88.16**	91.64†	88.10	90.85	86.50	93.17†	89.14‡
CRF2O	**90.77**	**91.29**	**91.54**†	80.46	**87.32**†	**90.86**†	87.96	**91.91**‡	**88.62**‡	**91.02**†	**86.90**‡	**93.33**‡	**89.33**‡
						using raw text							
Ji 等[15]	88.28	89.90	89.85	77.09	81.16	88.93	83.73	88.91	84.82	86.33	84.44	86.62	85.83
CRF2O	**89.72**	**91.27**	**90.94**	**78.26**	**82.88**	**90.79**	**86.33**	**91.02**	**87.92**	**90.17**	**85.71**	**92.49**	**88.13**
						UD2.3							
LOC	90.57	91.10	91.85	81.68	86.54	90.47	88.40	91.53	88.18	90.65	86.31	92.91	89.19
LOC^{MST}	90.56	91.03	91.98	81.59	86.83	90.64	88.23	91.67	88.20	90.63	86.51	93.03	89.23
CRF	90.52	**91.19**	92.02	81.43	86.88†	90.76†	88.75	91.76	88.08	**90.79**	86.54	93.16‡	89.32‡
CRF2O	**90.76**	91.12	**92.15**‡	**81.94**	**86.93**†	**90.81**‡	**88.83**†	**92.34**‡	**88.21**†	90.78	**86.62**	**93.22**‡	**89.48**‡
						using gold POS tags							
Zhang 等[21]	90.15	91.39	91.10	83.39	88.52	90.84	88.59	92.49	88.37	92.82	84.89	93.11	89.85
CRF2O	**91.32**	**92.57**	**92.66**	**84.56**	**88.98**	**91.88**	**89.83**	**92.94**	**89.85**	**93.26**	**87.39**	**93.86**	**90.76**

影树. 基本的想法是将非投影树转换为投影树的时候利用更多更复杂的标签来标记，以方便后处理的时候能够恢复.

可以看到在基础的局部模型上，直接的基于 MST 算法的非投影解析方法 LOC^{MST} 和伪投影方法 LOC 达到了非常相似的性能.

更重要的是，CRF 和 CRF2O 在大部分语言上相比于基线模型都出现了一致的提升. 在 UD2.2 和 UD2.3 上，我们提出的 CRF2O 模型在 12 个语言中的 10 个上达到了最高的解析准确率，并且在超过 7 个语言上获得的提升是显著的. 总体上，平均分别在 UD2.2 和 UD2.3 各自达到了 0.45 和 0.29 的提升，这同样是显著的（$p < 0.005$）.

在 UD2.2 上，我们和 CoNLL18 任务一样采用的是毛文本，并且我们的 CRF2O 分析器上比 Ji 等[15] 平均要高 2.30. 在 UD2.3 上，Zhang 等[21] 使用了正确词性作为输入，我们平均比他们高 0.91. 需要注意的是 Ji 等[15] 里德语（DE）结果错误地使用了正确的句子分割，符号化以及词性，而表中的结果是用了预测的词性，并重新运行他

们的分析器得到的. 我们的 Crf2o 分析器在使用他们的词性之后平均的 LAS 达到了 87.64.

2.5 本章小结

本章首次在神经依存句法分析模型上提出了一个二阶 TreeCRF 的扩展，并提出使用 Triaffine 结构来为二阶子树打分. 我们提出了批次化的 Inside 算法，以适应在 GPU 上的并行运算. 我们同样从经验上验证了复杂的 Outside 算法可以隐式地通过高效的自动求导机制来完成，相应可以自然地产生梯度以及边缘概率. 我们在 13 个语言的 27 个数据集上进行了大量的实验和详尽的分析，发现结构化学习和高阶建模可以进一步地从不同的方面提升当前最佳的 Biaffine Parser：1）更好的收敛行为；2）在子树和完全树上更优越的性能；3）对局部标注数据更好的利用.

第三章　基于 TreeCRF 的高阶成分句法分析

本章节提出了一个基于 TreeCRF 的高阶成分句法分析器. 估计概率分布一直是自然语言处理领域的一个核心问题. 但是，在深度学习时代和前深度学习时代，不同于线性链条件随机场（linear chain CRF）在序列标注任务中的大量应用，由于 Inside-Outside 算法的高复杂度，还很少有工作将 TreeCRF 应用到成分句法分析任务当中. 这里我们提出应用 TreeCRF 到成分句法分析，核心的想法是批次化计算损失函数用到的 Inside 算法，使其能够支持在 GPU 上的大规模张量并行计算，与此同时结合基于高效自动求导机制的反向传播，避免了复杂的 Outside 算法的计算. 考虑到高阶建模为依存句法模型带来的显著提升，我们还进一步设计了一个二阶 TreeCRF 拓展到成分句法模型. 我们还提出一个简单的两阶段解析方法，bracketing-then-labeling，来进一步提升分析器的效率. 为了提升解析的性能，受依存句法分析器的启发，我们引入了一个基于边界表示和仿射注意力的新打分方法，以及一个有效的 Dropout 策略. 在 PTB、CTB5.1 和 CTB7 上的实验表明，二阶 TreeCRF 模型和一阶模型的结果相当. 在一阶模型上，我们的两阶段条件随机场分析器在使用和不使用 BERT 的两种设置上，达到了新的最佳性能，并且解析速度达到了 1,092 句每秒.

3.1　引言

成分句法分析是自然语言处理领域一个基础但是富有挑战性的任务. 由于诸如宾州树库（Penn Treebank，PTB）、中文宾州树库（Penn Chinese Treebank，CTB）等大规模树库的标注，成分句法分析吸引了一大批研究者的关注. 同样的，句法分析输出的句法树也被证明对于大量的下游任务 (Akoury 等[82], Wang 等[83]) 都有用.

作为最有影响力的工作之一, Collins[37] 首次将概率上下文无关文法 (Probabilistic Context-Free Grammars, PCFGs) 扩展到了词汇化上下文无关文法（Lexicalized PCFGs）. 由此开始，成分句法分析方法一直是这样的生成式模型（generative models）占据主导地位，并且其中广泛使用的 Berkeley Parser 采用了带隐式非终端结点标注的非词汇化概率上下文无关文法（Unlexicalized PCFGs）(Matsuzaki 等[38], Petrov 等[84]). 而在判别式方法（discriminative models）上存在着两种主要方向. 第一种采取了以动态规划解码为基础的基于图的方法，训练时使用局部 max-entropy 估计 (Kaplan 等[85])

或者全局 Max Margin 方法 (Taskar 等[51]). 第二类则通过基于贪婪解码或者集束搜索（beam search）产生 shift-reduce 这样的转移序列来构建一棵树，这种方法被称为基于转移的方法 (Sagae 等[86], Zhu 等[87]).

最近，得益于深度神经网络在上下文表示方面令人印象深刻的发展，成分句法分析取得了显著的进展. 其中，Cross 等[43] 的基于转移的分析器，以及 Stern 等[10] 的基于图的分析器是两个具有代表性的工作. 作为判别式模型，两个分析器有很多的共同点，他们都使用了 1）多层双向 LSTM 作为编码器；2）从双向 LSTM 的输出得到的 minus feature 作为区块的表示；3）利用 MLP 层来为区块打分；4）Max Margin 的训练损失函数. 后续的大多数工作 (Gaddy 等[12], Kitaev 等[45]) 的主要设置都和这两个分析器一样，并且都相比传统的非神经网络模型达到了更好的准确率，这特别是得益于在大规模无标记文本上训练的语言模型输出的上下文词表示的使用 (Peters 等[79], Devlin 等[80]).

然而尽管有这些显著的进展，现有的成分句法分析的研究仍然受两个相互关联的缺点困扰. 首先，解析速度（训练速度同理）很慢，并且很难满足现实系统的需要. 其次，显式的树/子树概率建模的缺失一定程度上影响了分析器输出的利用. 一方面，估计概率分布一直是自然语言处理领域的核心问题 (Le 等[70]). 另一方面，与没有上下界的树的分值相比，树的概率可以作为一种软特征，更好的被更高层级的任务所利用 (Jin 等[88])，且子树的边缘概率可以支持更加复杂的最小贝叶斯风险解码 (Smith 等[28]).

事实上，Finkel 等[49], Durrett 等[50] 都通过建模树的条件概率 $p(\boldsymbol{t} \mid \boldsymbol{x})$，提出了基于 TreeCRF (Lafferty 等[89]) 的成分句法分析器. 但是，由于损失函数和梯度计算需要的 Inside-Outside 算法的高复杂度（尤其是 Outside 算法），这些模型都极端低效. 而在深度学习时代，由于以前所有的工作都直接在 CPU 上运行 Inside-Outside 算法，而让模型频繁在 CPU 和 GPU 切换所需要的时间代价是昂贵的，因此效率问题变得更加严重.

本章节通过极大拓展 Stern 等[10] 的基于图的分析器，提出了一个 TreeCRF 成分句法分析器. 类似于章节二，我们应用了 Inside 算法的批次化方法，并且将 Outside 算法用高效的反向传播代替 (Eisner[56]). 我们同样批次化了 CKY（Cocke–Kasami–Younger）算法，以支持高效的解码. 考虑到章节二发现引入高阶特征有显著的提升，因此我们进一步尝试在一阶 TreeCRF 的基础上提出了一个二阶 TreeCRF 的拓展.

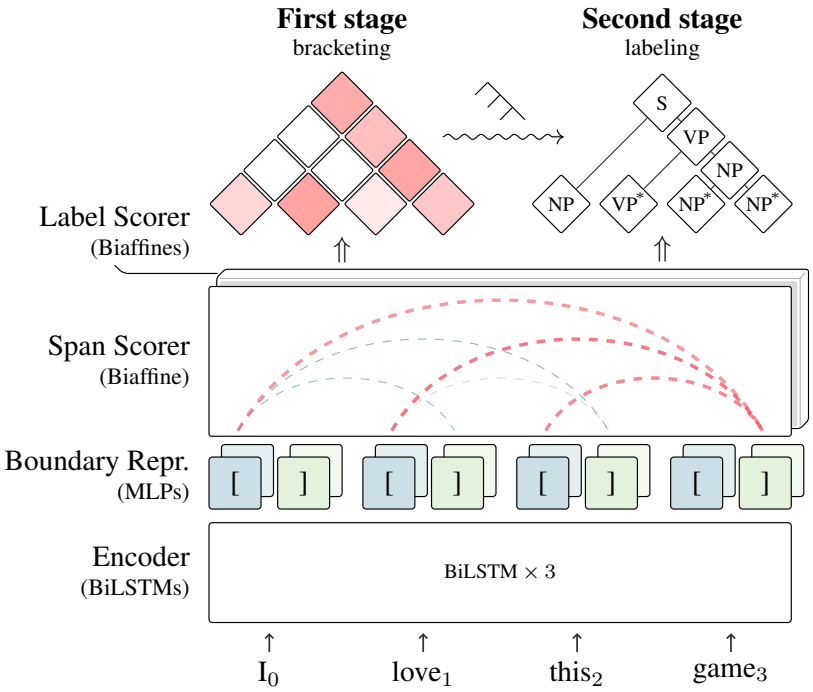


图 3-1　模型架构.

总体而言，我们在本章节做了如下的贡献.

- 为了直接建模树和子树的概率，我们首次提出了一个基于 TreeCRF 的高阶成分句法分析器. 通过批次化技术支持 Inside 算法和 CKY 算法在 GPU 上的直接计算，长期以来一直困扰句法分析社区的效率问题在这里被很好的解决了. 我们还进一步引入了二阶拓展，发现二阶模型和一阶 TreeCRF 的结果相当.
- 我们提出了一个两阶段的解析方法 bracketing-then-labeling：先产生无标记树骨干树（bracketing）再标注标签（labeling）的解析方法，这不仅更加高效，并且达到了比一阶段解析方法稍好的性能.
- 我们提出了基于区块表示的一个新的打分方法，以及基于仿射注意力机制的打分方法，比 minus feature 方法表现的更好. 我们同样表明，通过更好的模型及参数设置，比如 Dropout，解析的性能可以被极大提升.
- 在中英文的三个基准数据集上的实验表明，我们提出的两阶段解析方法在使用 BERT 和不使用 BERT (Devlin 等[80]) 的两种设置下，在性能上都分别达到了新的最佳水准. 在解析速度方面，我们的一阶解析器可以达到大约 1,000 句每秒的解析速度.

3.2　两阶段 TreeCRF 解析

如图 1-2a所示，对于一个由 n 个词组成的句子 $\boldsymbol{x} = w_1, \dots w_n$，一棵成分句法树 $\boldsymbol{t} = \{((i, j), l) \mid 1 \le i \le n, i \le j \le n, l \in \mathcal{L}\}$ 可以被分解为两个部分，即，$\boldsymbol{t} = (\boldsymbol{y}, \boldsymbol{l})$，其中 $\boldsymbol{y}$ 是一棵无标签树（又称为 bracketed tree），而 $\boldsymbol{l}$ 是树中所有区块的标签按一定顺序产生的标签序列. 区块 $(3, 5\text{VP})$ 可以等价表示为 $\text{VP}_{3,5}$.

为了能够适应 Inside 算法和 CKY 算法，我们用 NLTK 工具包[1]将原始的树转换为了遵循乔姆斯基范式的二叉树形式，如图 1-2b所示. 特别地，连续的单链产生式被压缩为了一个标签，比如 $\text{X}_{i,j} \to \text{Y}_{i,j}$ 会被压缩为一个 $\text{X+Y}_{i,j}$. 根据前置实验，我们在二叉化原始树的时候，采用了左二叉. 在用 CKY 解码获得一棵最佳的树之后，CNF 树会被恢复为原来的 n-ary 形式.

3.2.1　模型定义

在本章中，我们对成分句法分析采用了一个两阶段的解析框架 bracketing-then-labeling. 这与传统的一阶段解析方法 (Stern 等[10], Gaddy 等[12]) 相比，不仅简化了模型架构，同样也提升了效率.

第一阶段：bracketing. 给定句子 $\boldsymbol{x}$，第一阶段的目标是找到一个最优的无标签树 $\boldsymbol{y}$. 一棵树的分值被分解为所有包含的区块（子树）的分值.

$$\text{s}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{(i,j) \in \boldsymbol{y}} \text{s}(i, j) \tag{3.1}$$

对于 TreeCRF，条件概率为

$$p(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{\exp(\text{s}(\boldsymbol{x}, \boldsymbol{y}))}{Z(\boldsymbol{x}) \equiv \sum_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x})} \exp(\text{s}(\boldsymbol{x}, \boldsymbol{y}'))} \tag{3.2}$$

其中 $Z(\boldsymbol{x})$ 被称为配分项（partition term），$\mathcal{Y}(\boldsymbol{x})$ 是输入句子 $\boldsymbol{x}$ 对应的所有合法句法树的集合.

给定所有的区块分值 $\text{s}(i, j)$，我们可以用 CKY 算法来找到一棵最优的无标签句

[1] https://www.nltk.org

法树 $\boldsymbol{y}$.

$$\boldsymbol{y} = \arg\max_{\boldsymbol{y}'} \mathrm{s}(\boldsymbol{x}, \boldsymbol{y}') = \arg\max_{\boldsymbol{y}'} p(\boldsymbol{y}' \mid \boldsymbol{x}) \tag{3.3}$$

第二阶段：labeling. 给定一个句子 $\boldsymbol{x}$ 和一棵树 $\boldsymbol{y}$，第二阶段独立地给每个区块 $(i, j) \in \boldsymbol{y}$ 预测一个标签.

$$l = \arg\max_{l' \in \mathcal{L}} \mathrm{s}((i, j), l') \tag{3.4}$$

请注意训练时我们使用正确的无标签树来进行损失函数的计算. 对于一个长度为 n 的句子，所有的 CNF 树都包含同样 $2n - 1$ 多个的区块. 因此，这一阶段的时间复杂度为 $O(n \cdot |\mathcal{L}|)$.

时间复杂度分析. CKY 的时间复杂度为 $O(n^3)$. 因此，我们两阶段解析方法的总时间复杂度为 $O(n^3 + n \cdot |\mathcal{L}|)$. 相对应的，对于一阶段解析的 CKY 而言，算法需要为所有 n^2 个区块决定最优的标签，因此需要 $O(n^3 + n^2 \cdot |\mathcal{L}|)$，其中 $|\mathcal{L}|$ 通常来说都很大（比如对于表 1-2中的英语来说为 138）.

3.2.2　打分方法

本章节引入了给区块和标签打分的模型架构，如图 3-1，大部分设置都遵循 Stern 等[10]，除了两个重要的修改：1）针对分值计算的边界表示和仿射注意力；2）和 Dozat 等[9] 一样的更好的参数设置.

模型输入. 对于第 i 个词，其对应的输入向量 $\mathbf{e}_i$ 是词向量和字级别表示的拼接：

$$\mathbf{e}_i = \mathbf{e}_i^{word} \oplus \mathrm{CharLSTM}(w_i) \tag{3.5}$$

其中 CharLSTM(w_i) 是将字序列输入到双向 LSTM 一样的输出向量 (Lample 等[74]). 以前的工作表明将词性表示用 CharLSTM(w_i) 代替会带来稳定的提升 (Kitaev 等[45]). 这同样简化了模型，因为不再需要额外预测词性.

双向 LSTM 编码器. 我们在输入向量上应用了三层双向 LSTM 以得到上下文表示. 我们分别用 $\mathbf{f}_i$ 和 $\mathbf{b}_i$ 来表示词 w_i 在顶层 LSTM 的前向和后向输出向量.

在这里，我们借用了大部分 Dozat 等[9] 的依存句法分析器的参数设置（参考章节2.4）. 我们发现其中的 Dropout 设置对于解析性能特别关键，这和原始的实现 (Stern 等[10]) 有两方面的不同.

首先，对于每个词 w_i，$\mathbf{e}_i^{word}$ 和 CharLSTM(w_i) 都作为一个整体被 dropout，要么保持版本，要么称为 **0** 向量. 如果其中一个向量被设置为了 **0**，则另一个会被乘以 2 倍作为补偿. 其次，相同 LSTM 层在不同的时间步（词）共享相同的 dropout 掩码 (Gal 等[90]).

边界表示. 对于每个词 w_i，我们参考 Stern 等[10] 的做法来组成上下文词向量[2]

$$\mathbf{h}_i = \mathbf{f}_i \oplus \mathbf{b}_{i+1} \tag{3.6}$$

$\mathbf{h}_i$ 的维度为 800.

不同于直接应用一个单一的 MLP 层到 $\mathbf{h}_i$ 上，我们发现一个词在一棵给定树的所有的区块中都必须作为其左边界或右边界. 因此，我们应用来两个 MLP 层来做这样的区分，并且分别获取左边界和右边界的表示向量.

$$\mathbf{r}_i^l; \mathbf{r}_i^r = \mathrm{MLP}^l(\mathbf{h}_i); \mathrm{MLP}^r(\mathbf{h}_i) \tag{3.7}$$

$\mathbf{r}_i^{l/r}$ 的维度 d 为 500. 正如 Dozat 等[9] 所指出的，MLP 层缩减了 $\mathbf{h}_i$ 的维度，并且更重要的是只保留了句法相关的信息，因此能够减轻过拟合的风险.

Biaffine 打分器. 给定边界表示，对于候选区块 (i, j)，我们在左边界词 w_i 的表示和右边界词 w_j 的表示上利用仿射注意力为该区块打分.

$$\mathrm{s}(i, j) = \begin{bmatrix} \mathbf{r}_i^l \\ 1 \end{bmatrix}^{\mathrm{T}} \mathbf{W} \mathbf{r}_j^r \tag{3.8}$$

其中 $\mathbf{W} \in \mathbb{R}^{(d+1)\times(d)}$.

计算区块标签的分值 $\mathrm{s}((i, j), l)$ 时，需要在 $\mathbf{h}_i$ 上应用额外两层 MLP 来获取相应的边界表示 $\bar{\mathbf{r}}_i^{l/r}$（维度为 $\bar{d}$）. 接着我们使用 $|\mathcal{L}|$ 个 Biaffine（$\mathbb{R}^{(\bar{d}+1)\times(\bar{d}+1)}$）来获取所有标签的分值. 由于 $|\mathcal{L}|$ 非常大，我们对于 $\bar{\mathbf{r}}_i^{l/r}$ 使用了稍小的维度 $\bar{d} = 100$（对于 $\mathbf{r}_i^{l/r}$ 则是 500）以减轻内存和计算负担.

前人打分方法. Stern 等[10] 对双向 LSTM 的输出使用了 minus feature 方法来得到区块

[2]我们的前置实验表明 $\mathbf{f}_i \oplus \mathbf{b}_{i+1}$ 相比于 $\mathbf{f}_i \oplus \mathbf{b}_i$ 有稳定的提升. 可能的原因是 $\mathbf{f}_i$ 和 $\mathbf{b}_i$ 都使用 $\mathbf{e}_i$ 作为输入，因此提供的信息冗余.

的表示 (Wang 等[32], Cross 等[43])，然后用 MLP 层得到区块的分值.

$$\mathrm{s}(i, j) = \mathrm{MLP}(\mathbf{h}_i - \mathbf{h}_j) \tag{3.9}$$

注意到 Stern 等[10] 等人的打分方法由于得到区块表示带来的高复杂度，难以应用到二阶区块结构上. 在实验部分我们表明我们的打分方法要明显更优越.

3.2.3 训练损失函数

对于一个训练的例子 $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{l})$，训练损失函数由两部分组成.

$$L(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{l}) = L^{tree}(\boldsymbol{x}, \boldsymbol{y}) + L^{label}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{l}) \tag{3.10}$$

第一项是句子级的全局 TreeCRF 损失，目的是最大化无标签树的条件概率:

$$L^{tree}(\boldsymbol{x}, \boldsymbol{y}) = -\mathrm{s}(\boldsymbol{x}, \boldsymbol{y}) + \log Z(\boldsymbol{x}) \tag{3.11}$$

其中 $Z(\boldsymbol{x})$ 可以利用 Inside 算法在 $O(n^3)$ 的时间复杂度内被计算.

第二项是在 labeling 阶段，区块级别的标准交叉熵损失函数.

3.3 二阶 TreeCRF

与章节 二类似，我们还进一步尝试了更加复杂的二阶子结构约束，在这个场景下，包含二阶子树的句法树分值被定义为

$$\mathrm{s}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{(i,j)\in\boldsymbol{y}} \mathrm{s}(i, j) + \sum_{(i,k,j)\in\boldsymbol{y}} \mathrm{s}(i, k, j) \tag{3.12}$$

其中 (i, k, j) 这样的三元组是指 $i \le k < j$ 是区块 (i, j) 的分割点，并满足子区块 (i, k) 和 $(k + 1, j)$ 都在树中.

算法 2 批次化的 Inside 算法.

define: $S \in \mathbb{R}^{n \times n \times B}$ ▷ B is #sents in a batch
initialize: $S_{i,i} = 0, 0 \le i \le n$
for $w = 1$ **to** n **do** ▷ span width
    Parallel computation on $0 \le i, j = i + w < n$, $r, 0 \le b < B$
    $S_{i,j} = \log \sum_{i \le r < j} \exp\left(S_{i,r} + S_{r+1,j}\right) + \mathrm{s}(i, j)$
end for
return $S_{0,n-1} \equiv \log Z(\boldsymbol{x})$

3.3.1 基于 Triaffine 的二阶区块打分

对于考虑分割点 k 的二阶区块 (i, k, j) 的打分，我们参考章节 二，使用了一个 Triaffine 结构. 我们首先用三个额外的 MLP 层来进行和 Biaffine 类似的操作

$$\mathbf{r}_i^{l'}; \mathbf{r}_i^{r'}; \mathbf{r}_i^{s} = \mathrm{MLP}^{l'/r'/s}(\mathbf{h}_i) \tag{3.13}$$

$\mathbf{r}_i^{l'}; \mathbf{r}_i^{r'}; \mathbf{r}_i^{s}$ 是词 w_i 分别作为左边界、右边界和分割点对应的表示. 然后，我们对上述表示应用 Triaffine 结构

$$\mathrm{s}(i, k, j) = \begin{bmatrix} \mathbf{r}_k^{s} \\ 1 \end{bmatrix}^{\mathrm{T}} \mathbf{r}_i^{l'\mathrm{T}} \mathbf{W}^{triaffine} \begin{bmatrix} \mathbf{r}_j^{r'} \\ 1 \end{bmatrix} \tag{3.14}$$

其中 $\mathbf{W}^{triaffine} \in \mathbb{R}^{(d'+1)\times(d'+1)\times(d'+1)}$ 是一个三维的张量. 我们设置 d' 的维度为 100.

3.3.2 高效的 TreeCRF 计算方法

对于 TreeCRF 的损失函数（公式 3.11）里的配分项 $Z(\boldsymbol{x})$ 和特征梯度，所有以前在 TreeCRF 解析上的工作 (Finkel 等[49], Durrett 等[50]) 都是通过显式地在 CPU 上进行了 Inside-Outside 算法计算得到的. 不同于线性链 CRF，树状结构的 CRF 的批次化看起来要复杂很多.

在这里，我们发现实现一个批次化的 Inside 算法是可行的，如算法 2所示. 关键的想法是将一个批次的所有实例中宽度相同的区块分值打包到一个大的张量中. 这使得我们可以通过高效的大规模张量操作进行计算和合并. 由于对所有的 $0 \le i, j < n, r, 0 \le b < B$ 而言，在 GPU 上的计算都是并行的，因此算法仅需要 $O(n)$ 步. 我们的代码会给出更多的技术细节.

3.3.3　通过反向传播完成的 Outside 算法

传统上，Outside 算法被认为对于子树边缘概率和特征梯度的计算是不可或缺的. 事实上，Outside 算法通常至少要两倍慢于 Inside 算法. 而 Outside 算法的批次化也要复杂的多. 幸运的是，这个问题在深度学习时代被很好的解决了，基于自动求导机制支持的反向传播可以方便的获取梯度. Eisner[56] 提出了理论上的等价性证明. 由于我们在前向过程使用来批次化的 Inside 算法，因此反向传播过程也是通过大规模张量并行计算的，可以视为同样高效.

值得注意的是通过用区块分值 $\mathrm{s}(i, j)$ 对 $\log Z(\boldsymbol{x})$ 求偏导（同样由自动的反向传播完成），我们可以自然的得到区块 (i, j) 的边缘概率，也就是梯度.

$$p((i, j) \mid \boldsymbol{x}) = \frac{\partial \log Z(\boldsymbol{x})}{\partial \mathrm{s}(i, j)} \tag{3.15}$$

边缘概率在很多下游任务都可以作为一种很有用的软特征. 更多细节可以参考 Eisner[56].

3.3.4　解码

正如上面提到的，解析过程中，我们应用 CKY 算法来获取一棵最佳的句法树，如公式 3.3所示. 和 Eisner 算法一样，CKY 算法和成分句法分析的 Inside 算法几乎一样，除了其中的 sum product 被替换为了 max product（参考算法 2的行 5），因此可以被同样高效的批次化. 为了进行 MBR 解码，我们直接将区块分值 $\mathrm{s}(i, j)$ 换成式 3.1和式 3.3的边缘概率 $p((i, j) \mid \boldsymbol{x})$. 然而，我们发现这在成分句法上带来了很微弱的提升.

3.4　实验

参数设置. 我们直接采用 Dozat 等[9] 的依存句法分析器里面的大部分参数设置，没有进一步的改动. 唯一的区别是我们用了 CharLSTM 词表示，而非词性 embedding. CharLSTM 里自向量、词向量以及 CharLSTM 输出向量的维度分别为 50、100 和 100. 所有 Dropout 的比率为 0.33. 一个批次数据的大小约为 5,000 个词. 训练过程持续至多不超过 1,000 次迭代，并且如果 Dev 数据上的最高结果连续 100 次不提升，那么训练就会提前停止.

表 3-1　Dev 数据上的结果. 所有模型都使用了随机初始化的词向量.

	PTB			CTB5.1			CTB7		
	P	R	F_1	P	R	F_1	P	R	F_1
Max Margin (one-stage)	93.70	93.73	93.72	90.60	90.48	90.54	86.85	86.08	86.47
CRF (one-stage)	93.44	93.75	93.60	91.08	90.98	91.03	87.10	86.75	86.93
CRF (two-stage)	93.77	93.96	93.86	90.91	91.09	91.00	87.27	87.00	87.13
w/o MBR	93.75	93.85	93.80	**90.93**	91.10	91.02	87.21	86.89	87.05
minus feature	93.40	93.35	93.37	90.60	90.51	90.56	86.96	86.24	86.60
vanilla dropout	92.80	93.00	92.90	89.68	89.68	89.68	85.55	85.54	85.54
CRF2O (two-stage)	**93.87**	**93.98**	**93.93**	90.86	**91.24**	**91.05**	**87.34**	**87.16**	**87.25**

3.4.1　模型比较

我们在 Dev 数据上从三个方面对模型进行深入探究：1）TreeCRF 与 Max Margin 训练损失函数的对比；2）两阶段解析和一阶段解析的对比；2）一阶和二阶模型的对比. 表 3-1的前三行以及最后一行显示了相应的结果. 这四个模型使用了相同的打分方法和参数设置. 参考前人的经验 (Stern 等[10])，一阶段解析的模型只用带标签的区块 $s((i, j), l)$ 的分值. 为了验证两阶段解析的有效性，我们同样列出了 “CRF (one-stage)” 的结果，“CRF (one-stage)” 是指直接在有标签区块上打分.

$$s(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{l}) = \sum_{(i,j),l \in \boldsymbol{t}} s((i, j), l) \tag{3.16}$$

如章节3.2.1的最后一段所讨论的，一阶段解析的 Inside 算法和 CKY 算法相比于两阶段解析要复杂一点.

从表格的前两行我们可以看到，在一阶段解析的框架下，TreeCRF 损失函数在英语上带来了相似的结果，但是在中文数据集上稳定超过 Max Margin 方法 0.5 的 F_1 值. Max Margin 方法由额外的超参数，即 margin 值，该值根据在英语上调整的结果为 1，并且为了方便没有在中文上进一步调整. 我们猜测 Max Margin 方法在中文数据上的结果如果进一步调参的话可能还会有提升. 总体上，我们可以认为两种训练损失函数达到了非常相近的结果，但是 TreeCRF 方法有额外的概率建模的优势.

如果对表中第二行和第三行的结果进行比较，两种 TreeCRF 解析器在 CTB5.1 上

的结果十分相近，而两阶段解析的解析器在 PTB 和 CTB7 上相比于一阶段解析的方法超过了大约 0.2 的 F_1 值. 因此我们可以认为我们提出的两阶段解析在简单性和效率上要好于一阶段解析（参考表 3-3），并且没有损失性能.

最后，我们比较了同样的两阶段解析设置下的一阶 Crf 模型和二阶 Crf2o 的模型的结果. 除了引入了考虑分割点的二阶分值，Crf2o 和一阶模型 Crf 完全一样，都采用了两阶段解析，一致的编码器和打分方法. 可以看到在三个数据集上，Crf2o 取得了最高的结果，但是提升并不显著. 在三个数据集上的提升分别是 0.07、0.05 和 0.12. 这显示引入二阶特征的作用十分微弱.

3.4.2　消融实验

为了了解我们提出的框架中每个单独的模块所发挥的作用，我们在一阶模型上，通过每次去除一个模块，在 Dev 数据上进行了消融实验. 结果见表 3-1的第三到六行.

MBR 解码的影响. 我们默认在边缘概率上使用 CKY 解码. 行 “w/o MBR” 给出了在区块分值上解码的结果. 由于通常认为 MBR 解码在理论上要优于原始的解码方式，这样的比较给出了一些有趣的视角. 但是，从结果中可以明显的看到两种解码方式的结果近乎一样.

打分方法的影响. 为了衡量我们提出的新的打分方法带来的影响，我们将仿射打分器恢复到了原来 Stern 等[10]（参考公式 3.9）采用的 “minus feature” 方法. 可以清晰的看到我们提出的打分方法要优于广泛使用的 minus feature 方法，并且在所有三个数据上有大约 $0.5F_1$ 值的显著一致的提升.

Dropout 策略的影响. 我们保持其他模型设置都不变，然后将我们参考自 Dozat 等[9]的 Dropout 策略换成 Stern 等[10] 采用的原始 Dropout 方式. 这导致了三个数据集上结果的显著下降，分别是 0.96、1.39 和 $1.59F_1$ 值的下降. Kitaev 等[45] 用一个自注意力编码器替换了 Stern 等[10] 的双向 LSTM 编码器，通过分离上下文和位置注意力，达到了 1.0 F_1 值的巨大提升. 类似地，这里我们表明如果基于双向 LSTM 编码器的解析器有合适的参数设置，那么和自注意力编码器相比仍然很有竞争力.

3.4.3　主要结果

表 3-2显示了有和没有 ELMo/BERT 这两种设置下，Test 数据集上的主要最终结果.

表 3-2　Test 数据的结果.

	PTB			CTB5.1			CTB7		
	P	R	F_1	P	R	F_1	P	R	F_1
Stern 等[10]	92.98	90.63	91.79	-	-	-	-	-	-
Gaddy 等[12]	92.41	91.76	92.08	-	-	-	-	-	-
Kitaev 等[45]	93.90	93.20	93.55	88.09	86.78	87.43	-	-	-
Gómez-Rodríguez 等[46]	-	-	90.0	-	-	84.4	-	-	-
Shen 等[48]	92.0	91.7	91.8	86.6	86.4	86.5	-	-	-
Teng 等[91]	92.5	92.2	92.4	87.5	87.1	87.3	-	-	-
Vilares 等[47]	-	-	90.60	-	-	85.61	-	-	-
Zhou 等[52] + pretrained	93.92	93.64	93.78	89.70	89.09	89.40	-	-	-
CRF	93.84	93.58	93.71	89.18	89.03	89.10	87.66	87.21	87.43
CRF2O	94.06	93.84	93.95	89.04	88.68	88.86	87.86	87.40	87.63
CRF + pretrained	94.23	94.02	94.12	89.71	**89.89**	**89.80**	88.84	88.36	88.60
CRF2O + pretrained	**94.29**	**94.15**	**94.22**	**89.97**	89.47	89.72	**88.95**	**88.56**	**88.76**
Kitaev 等[45] + ELMo	95.40	94.85	95.13	-	-	-	-	-	-
Kitaev 等[92] + BERT	95.73	95.46	95.59	91.96	91.55	91.75	-	-	-
CRF + BERT	95.85	**95.53**	**95.69**	92.51	92.04	92.27	91.73	**91.38**	91.55
CRF2O + BERT	**95.86**	95.47	95.67	**92.75**	**92.18**	**92.47**	**91.93**	91.31	**91.62**

除了 Zhou 等[52]，大部分前人的结果都没有使用预训练的词向量，而是用了随机初始化的版本. 他们在英文上使用了 Glove 词向量，中文上用了 structured skip-gram 词向量. 对于预训练词向量，我们在英文 PTB 上用了 100 维的 Glove[3]，中文上采用了 Li 等[33] 的设置，使用基于 Gigaword 3rd Edition 训练的词向量，我们同样尝试了 Zhou 等[52] 分享的 structured skip-gram 词向量，最终的结果和前者差不多. 可以明显看到我们的解析器极大受益于预训练词向量的使用.

我们同样与最近的一些成分句法分析相关工作做了对比. 可以看到我们的基于双向 LSTM 编码器的解析器极大地超越了基本的 Stern 等[10]，这大部分要归功于新的打分方法，以及 Dropout 之类的参数设置. 和在 Dev 数据上的趋势一致，我们的二阶模型 CRF2O 和一阶模型 CRF 的结果近乎一致，其中 CRF2O 在 PTB 和 CTB7 上的结果超越了 CRF，在 CTB51 上要次于 CRF，但是差异都不大. 与之前最佳的自注意力解析

[3]https://nlp.stanford.edu/projects/glove

表 3-3　PTB 的 Test 数据上的解析速度比较.

	F_1	Sents/sec
Petrov 等[84] (Berkeley Parser)	90.1	6
Zhu 等[87] (ZPar)	90.4	90
Stern 等[10]	91.79	76
Shen 等[48]	91.8	111
Kitaev 等[45]	93.55	332
Gómez-Rodríguez 等[46]	90.0	780
CRF (one-stage)	93.71	990
CRF2O (two-stage)	**94.22**	598
CRF (two-stage) w/ MBR	94.12	743
CRF (two-stage) w/o MBR	94.08	**1092**

器相比 (Kitaev 等[45])，我们的一阶解析器分别在英文 PTB 和中文 CTB5.1 有 0.16 以及 1.67 的绝对提升，并且不包含任何语言特定的实验设置.

Zhou 等[52] 的 CTB5.1 结果得自重新运行他们开源的代码，他们的模型使用了预测的词性. 我们参考了他们描述[4]来产生预测词性. 值得注意的是他们的汇报的 CTB5.1 结果意外使用了正确的词性. 我们用正确词性重新跑了他们的代码，在 CTB5 的 Test 数据的结果是 92.14，与他们论文汇报的结果十分接近. 我们的解析器使用正确词性之后的结果达到了 92.66 的 F_1 值. 关于他们论文的另一个细节需要澄清：对于中文上的依存句法分析，他们使用了两种不同的数据分割设置，并且都用了 Stanford Dependencies 3.3.0 以及正确词性.

表格的最后三行列出了在使用 ELMo/BERT 设置下的结果. 对于 PTB，我们参考 Kitaev 等[92]，使用了 bert-large-cased[5]（24 layers, 1024 dimensions, 16 heads），对于 CTB 则使用了 bert-base-chinese（12 layers, 768 dimensions, 12 heads）. 可以明显看到无论是在一阶 CRF 模型和二阶 CRF2O 模型上，使用 BERT 表示都可以极大帮助性能的提升，最终两种解析器的结果趋于一致. 我们的解析器同样超过了 Kitaev 等[92] 的多语言解析器，这里他们用了额外了多语言资源. 总体而言，我们的解析器在两个语言和不同的设置下都达到了当前最佳的结果.

[4]https://github.com/DoodleJZ/HPSG-Neural-Parser
[5]https://github.com/huggingface/transformers

3.4.4 速度比较

表格 3-3列出了不同的解析器关于解析速度方面的比较以及解析器相应的 F_1 值. 为了公平比较的需要，我们的模型都运行在 Intel Xeon E5-2650 v4 CPU 和 Nvidia GeForce GTX 1080 Ti GPU. Berkeley Parser 和 ZPar 是两个最具代表性的非神经网络的解析器，并且没法利用 GPU. Stern 等[10] 应用了 Max Margin 方法训练并且在 CPU 上进行了类 CKY 的解码. Kitaev 等[45] 使用了自注意力编码器，并用 Cython 做了加速.

可以看到我们的一阶段 TreeCRF 解析器由于直接在 GPU 上解码，因此比前人的解析器要快得多. 在一阶模型上，我们的两阶段解析器每秒可以解析 1,092 句，是 Kitaev 等[45] 的三倍快. 当然，值得注意的是如果采用了我们提出的批次化技术，Stern 等[10], Kitaev 等[45] 的方法可能和我们的一样快.

Gómez-Rodríguez 等[46] 的解析器将句法分析任务视为一个序列标注问题，因此同样十分高效. 但是如表中所示，他们的结果还很差.

两阶段解析器相比于一阶段解析器要快大约 10%. 相比于我们之前讨论的时间复杂度差异（参考章节 3.2.1），这个差距显得似乎不是特别明显. 一个理由是这两种解析器都共享了相同的编码和打分组块，这可能占用了解析的大部分时间.

使用 MBR 解码额外需要运行一次 Inside 算法和反向传播来计算边缘概率，因此相对来说效率要差一些. 如表中所示，在有和没有 MBR 解码的两个设置下，结果的差异很小.

在使用 MBR 解码和不使用 MBR 解码两种场景下，一阶模型都分别比二阶模型快了大约 200 句每秒，而二阶模型没有显示出性能上的优势.

3.5 本章小结

本章中，我们提出了一个基于 TreeCRF 的高阶成分句法分析器. 我们表明 Inside 算法和 CKY 算法可以被高效的批次化，以支持在 GPU 上的大规模张量并行计算，进而带来显著的效率提升. 基于自动求导机制的反向传播和 Inside 算法一样高效，并且取代了 Outside 算法来进行梯度的计算. 在中英文三个基准数据集上的实验给出了很多有意义的结论. 第一，我们发现相比较一阶模型，二阶扩展的结果相当. 第二，简单的两阶段 bracketing-then-labeling 解析方法比一阶段解析更加高效，并且没有损害解析器的性能. 第三，我们新提出的打分方法相比于前人的基于 minus feature 的方法

达到了更高的结果. 第四，我们引入的一些诸如 Dropout 的参数设置可以极大提升解析的性能. 最后，我们提出的解析器达到了新的最佳结果，并且一阶模型和二阶模型的解析速度分别达到了 1,000 和 600 句每秒.

第四章 基于变分推断的高阶方法

本章节在章节 二和 三提出的高阶 TreeCRF 句法分析器的基础上，提出了一个近似的 MFVI 算法来代替精确推断的高阶方法. 因为建模简单性能优异，基于图的句法分析方法一直以来是目前最流行的句法分析方法之一. 而其中的高阶 TreeCRF 方法由于融入了包含更多变量交互的子树特征，并且具备估计概率分布的优势，一直以来都是一个有吸引力的研究方向. 然而，由于 Inside-Outside 推断算法高复杂问题，限制了高阶 TreeCRF 的应用. 在本章中，我们提出利用基于 MFVI 的近似算法来代替精确推断的高阶 TreeCRF，从而显著降低了推断算法的复杂度. 与其他近似方法相比，MFVI 方法具备了方便地获取树/子树概率的优势. 我们在依存句法和成分句法的中英文常见的 5 个基准数据集上做了实验，并对比了加入 BERT 之后的效果. 结果表明，MFVI 方法不仅从准确率上相匹敌，并且训练和测试的速度要远远快于章节 二和 三里的高阶 TreeCRF 方法.

4.1 引言

近年来句法分析领域有了长足的进步. 研究者针对句法分析任务提出了一系列的方法 (Dozat 等[9], Ji 等[15], Gómez-Rodríguez 等[46], Zhang 等[93], Wei 等[94])，将英语基准数据集宾州树库（Penn Treebank, PTB）的准确率刷新到了很高的水平.

作为代表性的方法之一，基于图的方法选择将一棵句法树分解为多个部分，然后分别打分，组成树的分值，并训练模型使得能够找到分值最大的句法树. 相比基于转移的方法，基于图的方法建模简单，并且通常不会陷入局部最优，因此依存句法和成分句法中很多工作都遵循了这个范式 (McDonald 等[23,24], Taskar 等[51]). 最简单的一阶基于图的方法让树中的每个变量都互相独立，例如要求依存树的每条弧均互相独立（称为弧分解假设）或者成分树的每个区块互相独立. 后来，有研究者提出放宽一阶假设，考虑包含更多子结构的交互的高阶特征，发现显著提升了句法分析器的性能 (McDonald 等[13], Koo 等[27], Ma 等[58]). 最近，有很多文献在神经网络模型中引入了二阶子树特征 (Chen 等[14], Pei 等[68], Ma 等[95]). 其中，Ji 等[15] 在当前最流行的基于一阶局部训练目标的 Biaffine Parser 的基础上，考虑了祖孙、兄弟等二阶子树，并利用多层 GAT 来编码. 这些工作证明了高阶方法在基于神经网络的模型中是有益的.

尽管有很好的性能，现有的基于图的高阶方法中存在着一些固有问题限制了其应用. 一方面，为了找到最优的句法树，高阶方法需要设计高复杂度的动态规划算法来求解，当引入更多复杂的子树特征时，我们往往无法得到一个可精确推断的算法，尽管我们在章节 二和章节 三展示了如何通过批次化等方法加速，但是推断算法的复杂度仍然带来了不可忽视的时间开销. 另一方面，由于效率原因，现有的工作通常采用 Max Margin 方法 (McDonald 等[13]) 或者局部目标 (Ji 等[15]) 来训练，而非需要全局归一化的 TreeCRF. 而对于句法分析社区而言，一个重要的目的是概率分布推断[1]. 无论是为了获得性能更高的分析器，还是作为特征应用到下游任务中 (Zhang 等[4,6])，树和子树的后验/边缘概率都是重要的.

针对这些问题，在本章中，我们提出基于平均场变分推断的近似方法到高阶依存句法和成分句法分析模型中（Mean Field Variational Inference, MFVI）. 对于不可精确推断问题采用近似方法一直以来都在图像分割、句法分析等领域有广泛的应用 (Wang 等[18], Martins 等[60], Krähenbühl 等[96]). 其中作为与我们最相关的工作，Smith 等[16], Gormley 等[61] 提出应用循环置信传播算法融入大量的全局因子，如约束全局树的 Tree、双变量子树的 Sib 和 Grd 等来帮助依存句法模型，Fonseca 等[97] 则应用了基于对偶分解的 AD^3 算法处理 Siblings/Grandparents 等特征. 我们选择 MFVI，理由有两个：1）为了与前面章节保持一致，我们采用了不超过两个变量的二阶子树特征，在这个场景下，相比于循环置信传播，MFVI 的收敛更快，复杂度更低；2）与对偶分解相比，MFVI 可以方便地得到边缘概率，方便我们进行 MBR 解码.

本章中，针对依存解析和成分解析两种任务的特性，我们分别为他们设计了不同的因子图和 MFVI 更新算法. 依存句法中，由于每个位置都只有一个头，因此参考 Wang 等[67]，我们的 MFVI 算法设计中引入了头选择的结构约束，让每个位置取值在所有可能的头上归一化. 成分句法中，我们参考 Smith 等[16], Wang 等[18], Naradowsky 等[62]，让句法树的每个组块对应变量的取值为 0-1. 我们发现近似算法在获取了和精确推断的高阶模型相接近的准确率的同时，解析速度大大提高.

总体而言，我们在本章节的工作如下：

- 我们提出在句法分析任务上应用引入二阶特征的 MFVI 近似方法来代替精确推断的高阶方法，显著降低了推断算法的复杂度.
- 在依存句法和成分句法两种句法分析任务上，我们分别探索了基于头选择和基

[1]我们提到的句法分析中的概率分布推断通常指的是两个方面：1）MAP Inference，即获取概率最大的句法树；2）Marginal Inference，即得到每个变量的边缘概率.

于二分类目标的两种变分推断方法.

- 我们在中文和英文的 4 个基准数据集上做了实验，发现 MFVI 方法的性能显著超越了一阶方法，达到了和精确推断的高阶方法可比较的水平. 解析速度上，MFVI 在依存和成分句法分析上分别可以达到 1,126 句和 905 句每秒. 在加入 BERT 之后，我们的变分推断方法在一些数据集上达到了新的最佳结果.

4.2 基于 MFVI 的二阶模型

本章在章节 二以及章节 三的基础上，分别设计了一个基于变分推断的二阶依存句法和成分句法模型. 本章保持了前面章节提出的二阶模型架构，区别是用 MFVI 近似算法代替了精确推断的 TreeCRF 方法，从而大大降低了复杂度.

4.2.1 模型定义

我们保持了和精确推断模型一致的两阶段解析策略.

第一阶段：给定输入句子 $\boldsymbol{x}$，目标是找到一棵最优（概率最高）的无标签树 $\boldsymbol{y} = \arg\max_{\boldsymbol{y}'} p(\boldsymbol{x}, \boldsymbol{y}')$. 在精确推断方法中，为了得到树概率，我们需要通过 $O(n^3)$ 复杂度的 Inside 算法计算配分项 $Z(\boldsymbol{x}) \equiv \sum_{\boldsymbol{y}' \in \mathcal{Y}} \mathrm{s}(\boldsymbol{x}, \boldsymbol{y}')$，来对分值 $\mathrm{s}(\boldsymbol{x}, \boldsymbol{y})$ 全局归一化，因此十分低效.

在本节我们提出利用高效的基于平均场理论的变分推断法（Mean Field Variational Inference, MFVI）来近似得到后验概率的方法. MFVI 假设句法树 $\boldsymbol{y}$ 每个位置的变量相互独立，因此可以在线性时间内通过不断迭代得到后验概率的近似分布 $Q(\cdot)$. 关于 MFVI 的通用更新公式以及相关推导见附录 B. 根据任务目标的不同，我们为依存句法和成分句法分别设计了两种不同的因子图，以及相应的更新方法.

第二阶段：我们的目标是为预测的无标签树 $\boldsymbol{x}$ 的每个部分预测标签，得到标签序列 $\boldsymbol{l}$. 依存树上 $\boldsymbol{l}$ 被分解为每条弧的标签，成分树上 $\boldsymbol{l}$ 由每个组块的标签组成. 我们采取和前述一样的方法，以贪婪解码的方式，为无标签依存句法树的每条弧 $i \rightarrow j$ 找到一个分值最大的标签，成分句法树的每个组块 (i, j) 找到一个最优的标签.

4.2.2　二阶子树打分

具体而言，给定一个句子 $\boldsymbol{x}$，我们的模型将对应的词向量输入到 3 层双向 LSTM 来计算每个位置的上下文表示，其中第 i 个词的输出为 $\mathbf{h}_i$. 接着，我们利用 $\mathbf{h}_i$ 对句法树的每个子结构进行打分. 对于依存句法，我们利用公式 2.3得到每条弧 $i \rightarrow j$ 的分值 $\mathrm{s}(i \rightarrow j)$，对于成分句法，我们用公式 3.8得到每个组块 (i, j) 的分值 $\mathrm{s}(i, j)$.

我们在变分推断方法中还引入了二阶子树特征. 其中，依存句法中，我们采用了和章节 二一样的兄弟子树结构. 我们利用公式 2.10得到兄弟子树 $i \rightarrow \{k, j\}$ 的分值 $\mathrm{s}(i \rightarrow \{k, j\})$，其中 k 是 j 的兄弟并且两者的父亲均为 i. 成分句法中，参考章节 三，我们利用公式 3.14得到 $\mathrm{s}(i, k, j)$，代表由 (i, k) 和 (i, j) 两个组块构成的子树 (i, k, j) 的分值，k 和 j 分别是两个组块的右边界位置，并且左边界均为 i.

对于标签的分值，我们利用 Biaffine 结构，分别得到依存树每条弧上的每个标签的分值 $\mathrm{s}(i \rightarrow j, l)$，以及成分句法树每个组块上的每个标签的分值 $\mathrm{s}((i, j), l)$.

4.2.3　依存句法的 MFVI 方法

目前广泛使用的一阶局部模型（对应于章节 二的 Loc 方法，这里我们称为 Loc_{dep}）在训练的时候采用了头选择（head selection）的训练损失函数，要求句子中除了根结点之外的每个词有且仅有一个头. 和 Wang 等[67] 一样，我们选择在依存句法对应变分推断方法中引入头选择约束，相应的因子图设计见图 4-1a. 对于每个位置 j，我们定义变量取值 $y_j \in \{0, 1, \cdots, i \neq j, \cdots, n\}$，代表词 w_j 的可能的头索引.

具体地，对于每个变量 y_j，一阶的势函数定义为

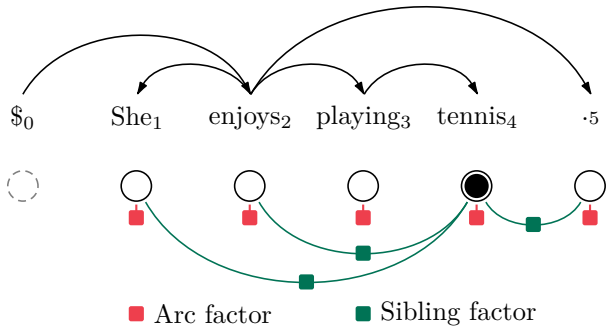
$$\psi_j(y_j) = \exp(\mathrm{s}(y_j \rightarrow j)) \tag{4.1}$$

$\mathrm{s}(y_j \rightarrow j)$ 是弧 $y_j \rightarrow j$ 对应的分值，由公式 2.3计算得到.

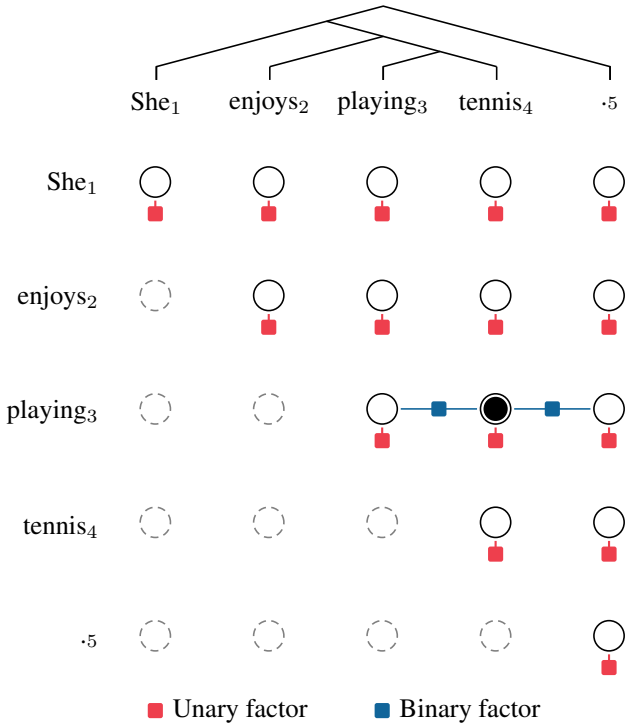
对于包含两个变量 y_j 和 y_k 的二阶因子，对应的势函数定义为

$$\psi_{j,k}(y_j, y_k) = \begin{cases} \exp \mathrm{s}(y_j \rightarrow \{k, j\}) & y_j = y_k \\ 1 & otherwise \end{cases} \tag{4.2}$$

$\mathrm{s}(y_j \rightarrow \{k, j\})$ 是兄弟子树 $y_j \rightarrow \{k, j\}$ 的分值，由公式 2.10计算得到. 需要注意的是这



(a) 依存句法模型的因子图



(b) 成分句法模型的因子图

图 4-1　一个例句和其依存/成分句法模型对应的因子图，我们在例句上方标出了对应的正确无标签依存句法树和成分句法树，其中成分句法树进行了左二叉化. 灰色虚线圆圈代表被屏蔽的变量. 图中标出了所有的一阶（红色）因子，为了简洁起见，对于二阶因子，依存句法图中我们只标出了涉及弧 $3 \rightarrow 4$ 的兄弟（绿色）因子，成分句法图中只标出了和组块 $(3,4)$ 连接的二阶（蓝色）因子.

里的兄弟 k 和二阶 TreeCRF 不同，并不需要一定和 j 邻近 (Smith 等[16]). 我们要求子树 $y_j \rightarrow \{k, j\}$ 中没有环存在，即有 $k \neq 0, j, y_j$.

MFVI 迭代式地更新近似分布 $Q(\cdot)$，来最小化其与真实分布 $P(\cdot)$ 的 KL 散度. 依

存句法模型的迭代更新公式如下 (Wang 等[67]):

$$Q_j^{(t)}(i) \propto \exp\left(\mathrm{s}(i \rightarrow j) + \sum_{k \neq i,j} Q_k^{(t-1)}(i) \cdot \mathrm{s}(i \rightarrow k, j)\right) \tag{4.3}$$

后验分布 $Q_j^{(0)}(i)$ 初始化为一阶势函数的值 $\psi_j(i)$. 每次迭代时，我们都会将 $Q_j^{(t)}(\cdot)$ 的值在所有可能的头取值上进行归一化. 经过 T 次迭代后，我们得到最终的近似后验分布 $Q_j^{(T)}(\cdot)$.

4.2.4　成分句法的 MFVI 方法

我们的成分句法分析模型采用了 Gaddy 等[12] 的方法作为基线方法，称为 Loc_{con}. 具体来说，模型对短语树所有可能的位置进行一个简单的二分类，预测该位置是否是一个区块. Wang 等[18], Dozat 等[98] 在语义依存分析中应用了这样的训练目标. Naradowsky 等[62], Gormley 等[99] 在成分句法分析中应用了这种方法. 我们将其引入到了基于 MFVI 的成分句法分析中，并采用了一个一阶因子和一个二阶因子，对应的因子图见图 4-1b.

具体地，每个位置 ij 的可能的变量取值 $y_{ij} \in \{0, 1\}$. 其中单个变量 y_{ij} 的一阶的势函数定义为

$$\psi_{ij}(y_{ij}) = \begin{cases} \exp\left(\mathrm{s}(i, j)\right) & y_{ij} = 1 \\ 1 & otherwise \end{cases} \tag{4.4}$$

$\mathrm{s}(i, j)$ 是由公式 3.8计算得到的区块 (i, j) 的分值，这里我们要求 $i < j$.

给定两个变量 y_{ij} 和 y_{lk}，我们使用章节 三的二阶兄弟特征，因此二阶的势函数定义为

$$\psi_{ij,lk}(y_{ij}, y_{lk}) = \begin{cases} \exp\left(\mathrm{s}(i, k, j)\right) & i = l \\ 1 & otherwise \end{cases} \tag{4.5}$$

$\mathrm{s}(i, k, j)$ 可以视为 (i, j) 和 (i, k) 都作为组块时，组成的子树的分值，由公式 3.14计算得到. 需要注意的是这里 k 的位置不受动态规划算法的约束，并不要求一定位于 (i, j) 之间，即有 $i < k, k \neq j$.

成分句法模型的 MFVI 迭代更新公式如下：

$$\begin{aligned} Q_{ij}^{(t)}(0) &\propto 1 \\ Q_{ij}^{(t)}(1) &\propto \exp\left(\mathrm{s}(i,j) + \textstyle\sum_{k\neq i,j} Q_{ik}^{(t-1)}(1) \cdot \mathrm{s}(i,k,j)\right) \end{aligned} \tag{4.6}$$

后验概率 $Q_{ij}^{(0)}(y_{ij})$ 初始化为一阶势函数 $\psi_{ij}(y_{ij})$. 每次迭代时，我们将 $Q_{ij}^{(t)}(\cdot)$ 在取值 $\{0,1\}$ 上进行归一化. 经过 T 次迭代后，我们得到成分句法最终的近似后验分布 $Q_{ij}^{(T)}(\cdot)$.

4.2.5 训练损失函数

我们的训练损失函数分为两部分，分别是无标签树的损失和对应标签的损失. 给定所有正确标签，我们的目标是最大化树上每个标签的概率，和前面的章节一致，我们采用了弧/区块级别的标准交叉熵损失函数. 给定输入句子 $\boldsymbol{x}$ 和对应正确的无标签句法树 $\boldsymbol{y}^*$，我们的目标是最大化句法树的概率 $p(\boldsymbol{y}^* \mid \boldsymbol{x})$. 由于 MFVI 将真实分布在每个变量上分解得到近似概率分布 $Q(\cdot)$，因此训练目标等价于最大化每个变量的后验概率.

其中，对于每个句子，依存句法分析树采用了弧分解策略，每个位置的变量取值为所有可能的头，因此相应无标签句法树的损失函数为

$$L_{dep}^{tree} = -\sum_{j\neq 0} \log Q_j(y_j^*) \tag{4.7}$$

成分句法的无标签句法树按照组块分解，每个组块可能的取值为 $\{0,1\}$，因此目标函数为

$$L_{con}^{tree} = -\sum_{i<j} \log Q_{ij}(y_{ij}^*) \tag{4.8}$$

参考 Dozat 等[98]，我们还新引入了一个参数 λ 用于平衡无标签句法树以及标签的损失，相应的成分句法的最终训练目标为

$$L_{con} = \lambda L_{con}^{label} + (1-\lambda) L_{con}^{tree} \tag{4.9}$$

4.2.6 解码

我们在 MFVI 模型中解码时直接应用了 MBR 解码的策略. 我们采用 MFVI 近似得到的后验概率 $Q(\cdot)$ 代替边缘概率作为解码算法的输入. 依存句法中，我们将由公式 4.3得到的概率 $Q_i(\cdot)$ 作为输入，并利用了 Eisner 算法来解码，加速时采用了算法 1的批次化技术. 成分句法中，我们将由公式 4.6得到的概率 $Q_{ij}(\cdot)$ 作为输入，并利用了和章节 三一样的类 CKY 算法来解码，采用了算法 2的批次化技术来加速.

4.3 实验

参数设置. 我们保持两种句法分析模型的编码器和训练方法与前面的章节基本一致. 对于二阶模型，我们设置依存句法分析中使用的兄弟特征以及成分句法分析使用的二阶特征的 MLP 层输出维度为 100. 我们设置变分推断的迭代次数统一为 3 次. 对于在成分句法分析中的用于平衡标签和无标签树的训练损失的参数 λ，我们设置为 0.1.

模型定义. 我们在每种句法分析上都比较了四种模型：Loc、Crf、Crf2o 和 Mfvi，这里有必要对于他们的记号做一下说明. Loc 表示采用了局部损失函数的模型，Crf 和 Crf2o 指的是使用了一阶和二阶精确推断的 TreeCRF 模型，Mfvi 表示采用平均场变分推断的模型. 相同记号的在各自任务的指导不同，例如依存模型中的 Loc 采用了头选择损失函数，而成分模型的 Loc 采用了二分类损失函数. 有时候我们会附加下标 *dep* 和 *con* 以示区分.

4.3.1 主要结果

表格 4-1和表格 4-2分别给出了我们尝试的多种推断算法在依存句法和成分句法分析上的比较性实验结果. 和章节 二以及章节 三一样，我们在所有模型中都使用了预训练词向量作为输入. 英文数据我们统一使用了 100 维的 Glove 词向量，中文数据则使用了 word2vec 在 Giga 数据上训练的 100 维词向量. 我们还汇报了 BERT 的结果，对于英文数据，我们使用了 24 层 1024 维的 bert-large-cased 作为双向 LSTM 的输入，并在训练时固定参数. 中文数据我们则使用了 12 层 768 维的 bert-base-chinese.

可以看到在依存句法的结果上（表格 4-1），由于英文 PTB 的结果非常高，因此这四种推断方法的结果都十分接近，Mfvi 方法超越了 Loc，达到了最好的性能. 中文 CoNLL09 上，精确推断的 Crf2o 仍然是最好的，但是 Mfvi 超越了 Crf 达到了 86.25

表 4-1 依存句法分析模型在 PTB 和 CoNLL09 的 Test 数据上不同推断算法的结果.

	PTB		CoNLL09	
	UAS	LAS	UAS	LAS
Loc	96.08	94.47	89.15	85.98
Crf	96.02	94.33	89.28	86.18
Mfvi	**96.11**	**94.49**	89.35	86.25
Crf2o	**96.11**	94.46	**89.63**	**86.52**
		+BERT		
Loc	96.86	95.32	92.28	89.50
Crf	96.75	95.30	92.29	89.52
Mfvi	**96.90**	**95.37**	92.32	89.55
Crf2o	96.80	95.33	**92.43**	**89.68**

的 LAS. 在两种语言上，Mfvi 都一致超越了没有全局结构约束的 Loc 方法. 这验证了在近似推断的 Mfvi 方法上应用二阶结构约束的有效性.

在成分句法分析上（表格 4-2），中英文三个数据的结果中，无论是一阶 Crf 还是二阶 Crf2o，精确推断算法都仍然是表现最好的. 然而 Mfvi 都达到了和一阶 Crf 十分相近的性能. 并且，Mfvi 一致超越了基于二分类学习的 Loc 模型，在 PTB、CTB51 和 CTB7 上分别提升了 0.11、0.52 和 0.35.

当使用 BERT 之后，上述的几种推断算法在结果上没有显著差异. 依存句法上，PTB 中 Mfvi 表现最好，LAS 为 95.37，达到或超越了当前的最佳性能 (Zhou 等[52], Wang 等[67])，CoNLL09 上仍然 Crf2o 最佳，但是 Mfvi 的差距十分微弱. 成分句法中在 PTB 和 CTB51 这两个数据上，Mfvi 的表现都是最好的，分别达到 95.71 和 92.56 的 F_1 值，高于当前最佳的模型 (Kitaev 等[92]). 在 CTB7 上表现最好的仍是 Crf2o，F_1 值为 91.62，而 Mfvi 的差距不到 0.1. 然而，基于 Mfvi 方法的近似推断在解析效率上有很大的优越性（见章节 4.3.3的复杂度分析）.

4.3.2 样例分析

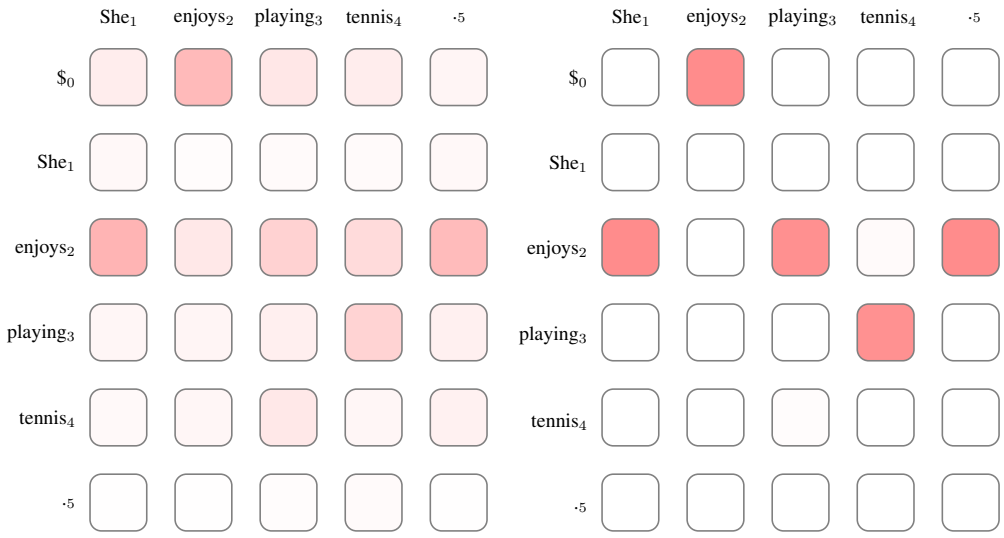
图 4-2给出了一个例句分别在依存和成分句法模型上的分值（log potentials）与经过 MFVI 迭代近似得到的后验概率的热力图对比. 图中颜色的深浅反应了分值/概率的大小.

表 4-2　成分句法分析模型在英文 PTB、中文 CTB5.1 和 CTB7 的 Test 数据上不同推断算法的结果.

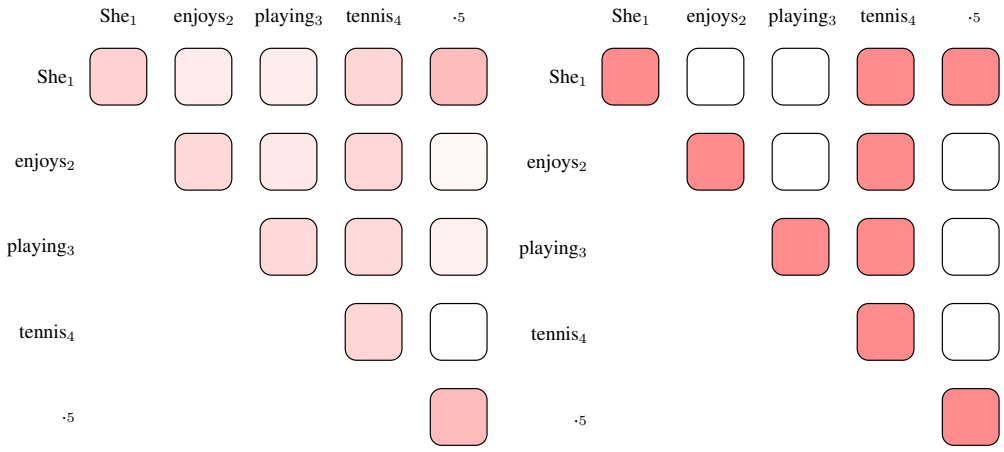
	PTB			CTB51			CTB7		
	P	R	F_1	P	R	F_1	P	R	F_1
Loc	94.16	93.85	94.01	89.39	89.16	89.28	88.54	87.96	88.25
Crf	94.23	94.02	94.12	89.71	**89.89**	89.80	88.84	88.36	88.60
Mfvi	94.11	93.96	94.04	89.82	89.79	**89.81**	88.71	88.16	88.43
Crf2o	**94.29**	**94.15**	**94.22**	**89.97**	89.47	89.72	**88.95**	**88.56**	**88.76**
+BERT									
Loc	95.70	95.43	95.57	92.47	92.09	92.28	91.90	91.24	91.57
Crf	95.85	95.53	95.69	92.51	92.04	92.27	91.73	**91.38**	91.55
Mfvi	95.85	**95.57**	**95.71**	**92.78**	**92.35**	**92.56**	91.89	91.31	91.60
Crf2o	**95.86**	95.47	95.67	92.75	92.18	92.47	**91.93**	91.31	**91.62**

左边的图反应的是模型直接输出的分值之间的对比，这对应于一阶模型的输出. 可以看到模型的输出的分值数值相对来说都比较均匀. 由于依存句法模型基于的是头选择的训练目标，可以看到图中的每一列都有一个极值存在，比如词 She_i 对应的列中，颜色最深（分值最高）的位置为词 $enjoys_2$，这对应了句法树中的弧 $enjoys_2 \rightarrow She_1$. 成分句法的热力图中，由于采用了二分类的目标，因此颜色较深的位置表示更可能成为一个区块. 然而由于没有类似于依存句法的头约束，直接通过对每个位置 arg max 得到的所有区块可能无法组成一棵合法的句法树，因此我们仍需要 CKY 算法来获得全局最优的合法句法树.

当我们利用 MFVI 算法，得到的近似后验概率的热力图如右边图所示. 可以看到，在进一步考虑了二阶特征的分值并聚合到每个变量位置之后，后验概率的置信度远远高于分值. 对于依存句法而言，每个词都有一个最可能的头，并且概率几乎为 1，对应于热力图中每列仅有一个红色结点. 对于成分句法而言，如果对图中每个位置的后验概率取 arg max 得到可能的区块，每个区块的概率同样近似为 1，并且对于图中的例句而言，这种贪婪解码得到的句法树可以直接组成一棵合法的短语结构树，例如，红色结点 $(She_1, tennis_4)$ 和 $(playing_3, tennis_4)$ 对应了例句正确的无标签树左二叉化之后的两个区块. 这显示了 MFVI 在引入二阶结构打分之后，对于模型结构预测更强大的约束作用.



(a) 依存句法树每个位置对应的分值和后验概率



(b) 成分句法树每个位置对应的分值和后验概率

图 4-2 一个例句对应的依存句法模型和成分句法模型的输出. 左边的图是每个位置的分值（log potential），为了方便显示，我们首先对分值进行了归一化. 右边的图是变分推断得到的后验概率. 灰色虚线圆圈代表被屏蔽的不合法位置.

4.3.3 速度和时间复杂度比较

表 4-3给出了 MFVI 和前面章节提及的精确推断模型的速度和复杂度的比较. 为了统一比较，所有的设置下都应用了 MBR 解码，并且每个任务都用了相同的解码算法. 可以看到，无论是依存句法还是成分句法模型，他们相应的 MFVI 在 CPU 上每一次迭代都需要 $O(n^3)$ 的复杂度，这和精确推断的 2 阶 Inside 算法的复杂度相当. 而当利用 GPU 进行并行计算时，MFVI 每个位置的变量仅需要一次遍历来收集其他位置的

表 4-3　依存句法和成分句法使用不同推断算法的时间复杂度，以及相应的在 PTB 的 Test 数据上的解析时间的比较. 依存句法统一使用了 Eisner 算法解码，成分句法统一使用了 CKY 算法解码

	Inference	Complexity		Decoding Alg.	Sents/sec
		CPU	GPU		
Dependency	CRF	$O(n^3)$	$O(n^2)$	Eisner	653
	CRF2O	$O(n^3)$	$O(n^2)$	Eisner	431
	MFVI	$O(n^3)$	$O(n)$	Eisner	**1126**
Constituency	CRF	$O(n^3)$	$O(n^2)$	CKY	743
	CRF2O	$O(n^3)$	$O(n^2)$	CKY	598
	MFVI	$O(n^3)$	$O(n)$	CKY	**905**

信息[2]，因此 GPU 上的算法复杂度为 $O(n)$，大大快于精确推断所需要的 $O(n^2)$ 复杂度.

从表中可以看到，利用 MFVI 的依存句法在 PTB 的 Test 数据的解析速度为 1126 句每秒，是精确推断的二阶 CRF2O（431）的近三倍快，同样也大大快于一阶 CRF 的 557 句每秒. 成分句法的 MFVI 模型的解析速度大约为 905 句每秒，同样显著快于一阶 CRF 的 743 句每秒和二阶 CRF2O 的 598 句每秒.

4.4　本章小结

在本章我们将包含高阶特征的 MFVI 方法方法引入到了依存句法和成分句法分析中，以解决精确推断的高阶方法的高复杂度问题. 根据两种句法分析的特点，在依存句法分析上，我们实现了一个包含头选择约束的 MFVI 方法，在成分句法分析上，我们引入了一个基于二分类学习目标的 MFVI 方法. 在这两个句法任务的中英文个五个数据集上的结果表明，MFVI 方法均显著超越了基于局部学习的一阶方法，并达到和精确推断的二阶 TreeCRF 方法可比较的性能. 在速度上，我们的 MFVI 方法分别在依存和成分句法达到了 1126 句每秒和 905 句每秒的解析速度. 加入 BERT 之后，基于 MFVI 方法的近似推断模型在所有的数据集上都达到或者接近了当前最佳模型的性能.

[2]尽管现代的 CUDA 技术可以通过二叉树等数据结构让并行化的归约操作，例如 sum、min 和 max 等，缩减到 $O(\log n)$ 复杂度的时间 (Wang 等[66])，但是这里我们统一假设归约操作的复杂度为 $O(n)$.

第五章 总结与展望

5.1 总结

得益于深度学习技术的迅速发展，以及神经网络模型的强大建模能力，近年来句法分析领域得到了长足的进步. Dozat 等[9] 提出的 Biaffine Parser 采用了一个强大的编码器结合一个简单的训练目标，是当前最流行的句法分析器. 与此对应的，传统的基于 TreeCRF 的全局训练目标尽管具有概率建模的优势，然而推断的低效问题限制了 TreeCRF 的广泛流行. 本文提出将 TreeCRF 应用到神经句法分析器当中，并借鉴前人工作提出了一个二阶 TreeCRF 拓展来进一步提升句法分析的准确率. 为了解决效率问题，本文提出了多项加速技术：1）批次化 Inside 算法；2）反向传播机制代替 Outside 算法. 二阶 TreeCRF 进一步增加了推断算法的复杂度，因此本文还提出利用基于平均场变分推断的近似算法代替二阶 TreeCRF. 变分法的模型性能超越了一阶方法，并且在速度上显著超越了二阶 TreeCRF.

总的来说，本文的主要内容如下:

(1) 基于 TreeCRF 的高阶依存句法分析

本文以当前最佳的神经依存句法分析模型 Biaffine Parser 为基础，提出了一个二阶 TreeCRF 的扩展. 对于二阶子树，本文提出了一个新颖的 Triaffine 结构来打分. 为了解决 TreeCRF 低效的问题，我们提出对于 $O(n^3)$ 复杂的 Inside 算法进行批次化，利用 GPU 并行计算的能力将算法复杂度降低到了 $O(n^2)$. 此外我们将复杂的 Outside 过程用基于自动求导机制的高效反向传播. 我们的加速方法让二阶模型的解析速度达到 400 句每秒，相比于传统在 CPU 上运算的方式有数十倍的提升，并且没有明显慢于一阶模型. 我们在 13 个语言的 27 个数据集上进行了大量的分析和实验，发现二阶模型带来了显著的准确率提升，并且尤其在全局指标，以及局部标注数据上表现良好.

(2) 基于 TreeCRF 的高阶成分句法分析

本文提出在已有神经网络模型的基础上应用高阶 TreeCRF 到成分句法分析中. 为了解决高复杂度问题，我们采用了和依存句法中一致的加速策略，将训练和解码算法进行了高度批次化，并且用支持自动求导机制的反向传播传播算法代替了复杂的 Outside 算法，从而显著提升了解析速度. 此外，我们提出了简单的两阶段解析方法，

比前人的一阶段解析更加高效，并且没有损害性能. 为了提升解析效果，我们参考依存句法分析对模型架构进行了修改. 我们提出用双仿射打分机制代替前人的 minus feature 方法，并且发现在双向 LSTM 编码器中引入的一些诸如 Dropout 的参数修改可以极大提升解析的性能，达到不输于 Transformer 的效果. 在中英文的三个基准数据集上的实验结果表明，我们提出的一阶和二阶成分句法模型显著超越了前人的方法. 速度方面，一阶和二阶模型分别可以解析 1,092 和 598 句每秒. 在使用 BERT 之后，我们的模型在大部分数据集上都达到了新的最佳结果.

(3) 基于变分推断的高效句法分析方法

考虑到精确推断的 TreeCRF 方法的高复杂度问题，本文中我们将包含二阶特征的平均场变分推断引入到了依存句法和成分句法分析中，使得算法在 GPU 上的复杂度从 $O(n^2)$ 降低到了 $O(n)$. 根据两种句法分析任务的不同学习特性，我们采取了不同的变分推断更新策略，在依存句法分析上我们采取了一个包含头选择约束的更新方法，在成分句法分析上我们则引入了一个基于二分类学习目标的更新方法. 我们在中英文共五个数据集上做了实验，结果表明我们的方法显著超越了采用局部学习目标的方法，并达到了和精确推断的二阶 TreeCRF 方法可比较的性能. 在使用 BERT 之后，变分推断方法在所有数据上都达到或接近了现有模型的最佳水平.

5.2 未来展望

本文在依存句法和成分句法分析两个任务上分别尝试了基于树形随机场的融入高阶特征的结构化学习方法，让句法分析模型达到了新的最佳水平. 本文还尝试了应用基于变分推断的近似推断算法，显著降低了高阶 TreeCRF 的复杂度，大大提升了句法分析速度. 未来，本文打算基于已有成果从一下几个方面继续探索:

(1) 本文的句法分析模型主要采用了 3 层双向 LSTM 作为编码器. 考虑到 Transformer 的迅速发展，以及 BERT 预训练语言模型的强大作用，未来我们将尝试利用 Transformer 替换已有编码器，探讨自注意力机制的效果，以及语言模型不同的利用方式，例如特征集成和微调对模型性能的影响.

(2) 本文中为了方便起见我们只采用了一种高阶特征，例如依存模型中我们只采用了邻接兄弟特征，成分模型中采用了区块分割点特征. 在未来我们将尝试更多的特征设计，探讨他们对模型效果的影响.

(3) 在本文中我们只尝试了基于平均场变分推断的近似推断算法. 然而，仍然有

其他的近似算法在 NLP 社区有广泛的应用，例如循环置信传播、对偶分解、整数线性规划等等. 在未来我们打算尝试更多近似算法，并对他们做一些经验性比较.

参考文献

[1] 张梅山. 中文词法句法语义联合模型研究 [D]. 哈尔滨工业大学, 2014.

[2] 李正华. 汉语依存句法分析关键技术研究 [D]. 哈尔滨工业大学, 2013.

[3] Nivre J, Zeman D, Ginter F, Tyers F. Universal Dependencies [C]//Proceedings of EACL. Valencia, Spain: Association for Computational Linguistics, 2017.

[4] Zhang M, Li Z, Fu G, Zhang M. Syntax-enhanced neural machine translation with syntax-aware word representations [C]//Proceedings of NAACL. Minneapolis, Minnesota: Association for Computational Linguistics, 2019: 1151-1161.

[5] Song L, Zhang Y, Gildea D, Yu M, Wang Z, Su J. Leveraging dependency forest for neural medical relation extraction [C]//Proceedings of EMNLP-IJCNLP. Hong Kong, China: Association for Computational Linguistics, 2019: 208-218.

[6] Zhang B, Zhang Y, Wang R, Li Z, Zhang M. Syntax-aware opinion role labeling with dependency graph convolutional networks [C]//Proceedings of ACL. Online: Association for Computational Linguistics, 2020: 3249-3258.

[7] Jiang W, Li Z, Zhang Y, Zhang M. HLT@SUDA at SemEval-2019 task 1: UCCA graph parsing as constituent tree parsing [C]//Proceedings of SemEval. Minneapolis, Minnesota, USA: Association for Computational Linguistics, 2019: 11-15.

[8] Fu Y, Tan C, Chen M, Huang S, Huang F. Nested named entity recognition with partially-observed treecrfs [C]//Proceedings of AAAI. Online: AAAI Press, 2021.

[9] Dozat T, Manning C D. Deep biaffine attention for neural dependency parsing [C]// Proceedings of ICLR. Toulon, France: OpenReview.net, 2017.

[10] Stern M, Andreas J, Klein D. A minimal span-based neural constituency parser [C]// Proceedings of ACL. Vancouver, Canada: Association for Computational Linguistics, 2017: 818-827.

[11] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A N, Kaiser Ł, Polosukhin I. Attention is all you need [C]//Proceedings of NIPS. Long Beach, California, USA: Neural Information Processing Systems Foundation, Inc., 2017: 5998-6008.

[12] Gaddy D, Stern M, Klein D. What's going on in neural constituency parsers? an analysis [C]//Proceedings of NAACL. New Orleans, Louisiana: Association for Compu-

tational Linguistics, 2018: 999-1010.

[13] McDonald R, Pereira F. Online learning of approximate dependency parsing algorithms [C]//Proceedings of EACL. Trento, Italy: Association for Computational Linguistics, 2006: 81-88.

[14] Chen D, Manning C. A fast and accurate dependency parser using neural networks [C]//Proceedings of EMNLP. Doha, Qatar: Association for Computational Linguistics, 2014: 740-750.

[15] Ji T, Wu Y, Lan M. Graph-based dependency parsing with graph neural networks [C]//Proceedings of ACL. Florence, Italy: Association for Computational Linguistics, 2019: 2475-2485.

[16] Smith D, Eisner J. Dependency parsing by belief propagation [C]//Proceedings of EMNLP. Honolulu, Hawaii: Association for Computational Linguistics, 2008: 145-156.

[17] Martins A, Smith N, Xing E. Concise integer linear programming formulations for dependency parsing [C]//Proceedings of ACL. Suntec, Singapore: Association for Computational Linguistics, 2009: 342-350.

[18] Wang X, Huang J, Tu K. Second-order semantic dependency parsing with end-to-end neural networks [C]//Proceedings of ACL. Florence, Italy: Association for Computational Linguistics, 2019: 4609-4618.

[19] Hajič J, Ciaramita M, Johansson R, Kawahara D, Martí M A, Màrquez L, Meyers A, Nivre J, Padó S, Štěpánek J, Straňák P, Surdeanu M, Xue N, Zhang Y. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages [C]// Proceedings of CoNLL. Boulder, Colorado: Association for Computational Linguistics, 2009: 1-18.

[20] Peng X, Li Z, Zhang M, Wang R, Zhang Y, Si L. Overview of the nlpcc 2019 shared task: cross-domain dependency parsing [C]//Proceedings of NLPCC. Dunhuang, China: Springer, 2019: 760-771.

[21] Zhang Z, Ma X, Hovy E. An empirical investigation of structured output modeling for graph-based neural dependency parsing [C]//Proceedings of ACL. Florence, Italy: Association for Computational Linguistics, 2019: 5592-5598.

[22] Zeman D, Hajič J, Popel M, Potthast M, Straka M, Ginter F, Nivre J, Petrov S. CoNLL

2018 shared task: Multilingual parsing from raw text to universal dependencies [C]// Proceedings of CoNLL. Brussels, Belgium: Association for Computational Linguistics, 2018: 1-21.

[23] McDonald R, Crammer K, Pereira F. Online large-margin training of dependency parsers [C]//Proceedings of ACL. Ann Arbor, Michigan: Association for Computational Linguistics, 2005: 91-98.

[24] McDonald R, Pereira F, Ribarov K, Hajič J. Non-projective dependency parsing using spanning tree algorithms [C]//Proceedings of EMNLP. Vancouver, British Columbia, Canada: Association for Computational Linguistics, 2005: 523-530.

[25] Chu Y J, Liu T H. On the shortest arborescence of a directed graph [J]. Scientia Sinica, 1965.

[26] Carreras X. Experiments with a higher-order projective dependency parser [C]// Proceedings of EMNLP. Prague, Czech Republic: Association for Computational Linguistics, 2007: 957-961.

[27] Koo T, Collins M. Efficient third-order dependency parsers [C]//Proceedings of ACL. Uppsala, Sweden: Association for Computational Linguistics, 2010: 1-11.

[28] Smith D A, Smith N A. Probabilistic models of nonprojective dependency trees [C]// Proceedings of EMNLP. Prague, Czech Republic: Association for Computational Linguistics, 2007: 132-140.

[29] Zhang Y, Clark S. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing [C]//Proceedings of EMNLP. Honolulu, Hawaii: Association for Computational Linguistics, 2008: 562-571.

[30] Huang L, Jiang W, Liu Q. Bilingually-constrained (monolingual) shift-reduce parsing [C]//Proceedings of EMNLP. Singapore: Association for Computational Linguistics, 2009: 1222-1231.

[31] Kiperwasser E, Goldberg Y. Simple and accurate dependency parsing using bidirectional LSTM feature representations [J]. Transactions of ACL, 2016: 313-327.

[32] Wang W, Chang B. Graph-based dependency parsing with bidirectional LSTM [C]// Proceedings of ACL. Berlin, Germany: Association for Computational Linguistics, 2016: 2475-2485.

[33] Li Y, Li Z, Zhang M, Wang R, Li S, Si L. Self-attentive biaffine dependency pars-

ing [C]//Proceedings of IJCAI. Macao, China: International Joint Conferences on Artificial Intelligence Organization, 2019: 5067-5073.

[34] Ma X, Hu Z, Liu J, Peng N, Neubig G, Hovy E. Stack-pointer networks for dependency parsing [C]//Proceedings of ACL. Melbourne, Australia: Association for Computational Linguistics, 2018: 1403-1414.

[35] Falenska A, Kuhn J. The (non-)utility of structural features in BiLSTM-based dependency parsers [C]//Proceedings of ACL. Florence, Italy: Association for Computational Linguistics, 2019: 117-128.

[36] Veličković P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y. Graph attention networks [C]//Proceedings of ICLR. Vancouver Canada: OpenReview.net, 2018.

[37] Collins M. Three generative, lexicalised models for statistical parsing [C]// Proceedings of ACL. Madrid, Spain: Association for Computational Linguistics, 1997: 16-23.

[38] Matsuzaki T, Miyao Y, Tsujii J. Probabilistic CFG with latent annotations [C]// Proceedings of ACL. Ann Arbor, Michigan: Association for Computational Linguistics, 2005: 75-82.

[39] Petrov S, Barrett L, Thibaux R, Klein D. Learning accurate, compact, and interpretable tree annotation [C]//Proceedings of ACL-COLING. Sydney, Australia: Association for Computational Linguistics, 2006: 433-440.

[40] Dyer C, Kuncoro A, Ballesteros M, Smith N A. Recurrent neural network grammars [C]//Proceedings of NAACL. San Diego, California: Association for Computational Linguistics, 2016: 199-209.

[41] Kuncoro A, Ballesteros M, Kong L, Dyer C, Neubig G, Smith N A. What do recurrent neural network grammars learn about syntax? [C]//Proceedings of EACL. Valencia, Spain: Association for Computational Linguistics, 2017: 1249-1258.

[42] Liu J, Zhang Y. In-order transition-based constituent parsing [J]. TACL, 2017: 413-424.

[43] Cross J, Huang L. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles [C]//Proceedings of EMNLP. Austin, Texas: Association for Computational Linguistics, 2016: 1-11.

[44] Fried D, Klein D. Policy gradient as a proxy for dynamic oracles in constituency pars-

ing [C]//Proceedings of ACL. Melbourne, Australia: Association for Computational Linguistics, 2018: 469-476.

[45] Kitaev N, Klein D. Constituency parsing with a self-attentive encoder [C]// Proceedings of ACL. Melbourne, Australia: Association for Computational Linguistics, 2018: 2676-2686.

[46] Gómez-Rodríguez C, Vilares D. Constituent parsing as sequence labeling [C]// Proceedings of EMNLP. Brussels, Belgium: Association for Computational Linguistics, 2018: 1314-1324.

[47] Vilares D, Abdou M, Søgaard A. Better, faster, stronger sequence tagging constituent parsers [C]//Proceedings of NAACL. Minneapolis, Minnesota: Association for Computational Linguistics, 2019: 3372-3383.

[48] Shen Y, Lin Z, Jacob A P, Sordoni A, Courville A, Bengio Y. Straight to the tree: Constituency parsing with neural syntactic distance [C]//Proceedings of ACL. Melbourne, Australia: Association for Computational Linguistics, 2018: 1171-1180.

[49] Finkel J R, Kleeman A, Manning C D. Efficient, feature-based, conditional random field parsing [C]//Proceedings of ACL. Columbus, Ohio: Association for Computational Linguistics, 2008: 959-967.

[50] Durrett G, Klein D. Neural CRF parsing [C]//Proceedings of ACL-IJCNLP. Beijing, China: Association for Computational Linguistics, 2015: 302-312.

[51] Taskar B, Klein D, Collins M, Koller D, Manning C. Max-margin parsing [C]// Proceedings of EMNLP. Barcelona, Spain: Association for Computational Linguistics, 2004: 1-8.

[52] Zhou J, Zhao H. Head-driven phrase structure grammar parsing on Penn treebank [C]//Proceedings of ACL. Florence, Italy: Association for Computational Linguistics, 2019: 2396-2408.

[53] Zhang Y, Li Z, Lang J, Xia Q, Zhang M. Dependency parsing with partial annotations: An empirical comparison [C]//Proceedings of IJCNLP. Taipei, Taiwan: Association for Computational Linguistics, 2017: 49-58.

[54] Jiang X, Li Z, Zhang B, Zhang M, Li S, Si L. Supervised treebank conversion: Data and approaches [C]//Proceedings of ACL. Melbourne, Australia: Association for Computational Linguistics, 2018: 2706-2716.

[55] Li Z, Peng X, Zhang M, Wang R, Si L. Semi-supervised domain adaptation for dependency parsing [C]//Proceedings of ACL. Florence, Italy: Association for Computational Linguistics, 2019: 2386-2395.

[56] Eisner J. Inside-outside and forward-backward algorithms are just backprop (tutorial paper) [C]//Proceedings of WS. Austin, TX: Association for Computational Linguistics, 2016: 1-17.

[57] Rush A. Torch-struct: Deep structured prediction library [C]//Proceedings of ACL. Online: Association for Computational Linguistics, 2020: 335-342.

[58] Ma X, Zhao H. Fourth-order dependency parsing [C]//Proceedings of COLING. Mumbai, India: The COLING 2012 Organizing Committee, 2012: 785-796.

[59] Koo T, Rush A M, Collins M, Jaakkola T, Sontag D. Dual decomposition for parsing with non-projective head automata [C]//Proceedings of EMNLP. Cambridge, MA: Association for Computational Linguistics, 2010: 1288-1298.

[60] Martins A, Smith N, Figueiredo M, Aguiar P. Dual decomposition with many overlapping components [C]//Proceedings of EMNLP. Edinburgh, Scotland, UK.: Association for Computational Linguistics, 2011: 238-249.

[61] Gormley M R, Dredze M, Eisner J. Approximation-aware dependency parsing by belief propagation [J]. TACL, 2015: 489-501.

[62] Naradowsky J, Vieira T, Smith D. Grammarless parsing for joint inference [C]// Proceedings of COLING. Mumbai, India: The COLING 2012 Organizing Committee, 2012: 1995-2010.

[63] Auli M, Lopez A. A comparison of loopy belief propagation and dual decomposition for integrated CCG supertagging and parsing [C]//Proceedings of ACL. Portland, Oregon, USA: Association for Computational Linguistics, 2011: 470-480.

[64] Martins A, Smith N, Xing E, Aguiar P, Figueiredo M. Turbo parsers: Dependency parsing by approximate variational inference [C]//Proceedings of EMNLP. Cambridge, MA: Association for Computational Linguistics, 2010: 34-44.

[65] Li Z, Zhao H, Wang R, Parnow K. High-order semantic role labeling [C]//Findings of EMNLP. Online: Association for Computational Linguistics, 2020: 1134-1151.

[66] Wang X, Jiang Y, Bach N, Wang T, Huang Z, Huang F, Tu K. AIN: Fast and accurate sequence labeling with approximate inference network [C]//Proceedings of EMNLP.

Online: Association for Computational Linguistics, 2020: 6019-6026.

[67] Wang X, Tu K. Second-order neural dependency parsing with message passing and end-to-end training [C]//Proceedings of AACL. Suzhou, China: Association for Computational Linguistics, 2020: 93-99.

[68] Pei W, Ge T, Chang B. An effective neural network model for graph-based dependency parsing [C]//Proceedings of ACL-IJCNLP. Beijing, China: Association for Computational Linguistics, 2015: 313-322.

[69] Zhang X, Cheng J, Lapata M. Dependency parsing as head selection [C]//Proceedings of EACL. Valencia, Spain: Association for Computational Linguistics, 2017: 665-676.

[70] Le P, Zuidema W. The inside-outside recursive neural network model for dependency parsing [C]//Proceedings of EMNLP. Doha, Qatar: Association for Computational Linguistics, 2014: 729-739.

[71] Li Z, Zhang M, Zhang Y, Liu Z, Chen W, Wu H, Wang H. Active learning for dependency parsing with partial annotation [C]//Proceedings of ACL. Berlin, Germany: Association for Computational Linguistics, 2016: 344-354.

[72] Cai J, Jiang Y, Tu K. CRF autoencoder for unsupervised dependency parsing [C]// Proceedings of EMNLP. Copenhagen, Denmark: Association for Computational Linguistics, 2017: 1638-1643.

[73] Eisner J. Bilexical grammars and their cubic-time parsing algorithms [M]//Bunt H, Nijholt A. Advances in Probabilistic and Other Parsing Technologies. Kluwer Academic Publishers, 2000.

[74] Lample G, Ballesteros M, Subramanian S, Kawakami K, Dyer C. Neural architectures for named entity recognition [C]//Proceedings of NAACL. San Diego, California: Association for Computational Linguistics, 2016: 2475-2485.

[75] Drozdov A, Verga P, Yadav M, Iyyer M, McCallum A. Unsupervised latent tree induction with deep inside-outside recursive auto-encoders [C]//Proceedings of NAACL. Hong Kong, China: Association for Computational Linguistics, 2019: 1129-1141.

[76] Nivre J, Goldberg Y, McDonald R. Squibs: Constrained arc-eager dependency parsing [J]. CL, 2014: 249-257.

[77] Hwa R. Supervised grammar induction using training data with limited constituent

information [C]//Proceedings of ACL. College Park, Maryland, USA: Association for Computational Linguistics, 1999: 73-79.

[78] Pereira F, Schabes Y. Inside-outside reestimation from partially bracketed corpora [C]//Proceedings of ACL. Newark, Delaware, USA: Association for Computational Linguistics, 1992: 128-135.

[79] Peters M, Neumann M, Iyyer M, Gardner M, Clark C, Lee K, Zettlemoyer L. Deep contextualized word representations [C]//Proceedings of NAACL. New Orleans, Louisiana: Association for Computational Linguistics, 2018: 2227-2237.

[80] Devlin J, Chang M W, Lee K, Toutanova K. BERT: Pre-training of deep bidirectional transformers for language understanding [C]//Proceedings of NAACL. Minneapolis, Minnesota: Association for Computational Linguistics, 2019: 4171-4186.

[81] Nivre J, Nilsson J. Pseudo-projective dependency parsing [C]//Proceedings of ACL. Ann Arbor, Michigan: Association for Computational Linguistics, 2005: 99-106.

[82] Akoury N, Krishna K, Iyyer M. Syntactically supervised transformers for faster neural machine translation [C]//Proceedings of ACL. Florence, Italy: Association for Computational Linguistics, 2019: 1269-1281.

[83] Wang X, Pham H, Yin P, Neubig G. A tree-based decoder for neural machine translation [C]//Proceedings of EMNLP. Brussels, Belgium: Association for Computational Linguistics, 2018: 4772-4777.

[84] Petrov S, Klein D. Improved inference for unlexicalized parsing [C]//Proceedings of NAACL. Rochester, New York: Association for Computational Linguistics, 2007: 404-411.

[85] Kaplan R, Riezler S, King T H, Maxwell III J T, Vasserman A, Crouch R. Speed and accuracy in shallow and deep stochastic parsing [C]//Proceedings of HLT-NAACL. Boston, Massachusetts, USA: Association for Computational Linguistics, 2004: 97-104.

[86] Sagae K, Lavie A. A classifier-based parser with linear run-time complexity [C]// Proceedings of IWPT. Vancouver, British Columbia: Association for Computational Linguistics, 2005: 125-132.

[87] Zhu M, Zhang Y, Chen W, Zhang M, Zhu J. Fast and accurate shift-reduce constituent parsing [C]//Proceedings of ACL. Sofia, Bulgaria: Association for Computational

Linguistics, 2013: 434-443.

[88] Jin L, Song L, Zhang Y, Xu K, yun Ma W, Yu D. Relation extraction exploiting full dependency forests [C]//Proceedings of AAAI. New York City, NY, USA: AAAI Press, 2020.

[89] Lafferty J D, McCallum A, Pereira F C N. Conditional random fields: Probabilistic models for segmenting and labeling sequence data [C]//Proceedings of ICML. Williams College, Williamstown, MA, USA: Morgan Kaufmann, 2001: 282–289.

[90] Gal Y, Ghahramani Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning [C]//Proceedings of ICML. New York City, NY, USA: JMLR.org, 2016: 1050-1059.

[91] Teng Z, Zhang Y. Two local models for neural constituent parsing [C]//Proceedings of COLING. Santa Fe, New Mexico, USA: Association for Computational Linguistics, 2018: 119-132.

[92] Kitaev N, Cao S, Klein D. Multilingual constituency parsing with self-attention and pre-training [C]//Proceedings of ACL. Florence, Italy: Association for Computational Linguistics, 2019: 3499-3505.

[93] Zhang Y, Zhou h, Li Z. Fast and accurate neural crf constituency parsing [C]// Proceedings of IJCAI. Online: International Joint Conferences on Artificial Intelligence Organization, 2020: 4046-4053.

[94] Wei Y, Wu Y, Lan M. A span-based linearization for constituent trees [C]//Proceedings of ACL. Online: Association for Computational Linguistics, 2020: 3267-3277.

[95] Ma X, Hovy E. Neural probabilistic model for non-projective MST parsing [C]// Proceedings of IJCNLP. Taipei, Taiwan: Asian Federation of Natural Language Processing, 2017: 59-69.

[96] Krähenbühl P, Koltun V. Efficient inference in fully connected crfs with gaussian edge potentials [C]//Shawe-Taylor J, Zemel R, Bartlett P, Pereira F, Weinberger K Q. Advances in NIPS. Granada, Spain: Curran Associates, Inc., 2011: 109-117.

[97] Fonseca E, Martins A F T. Revisiting higher-order dependency parsers [C]// Proceedings of ACL. Online: Association for Computational Linguistics, 2020: 8795-8800.

[98] Dozat T, Manning C D. Simpler but more accurate semantic dependency parsing [C]//

Proceedings of ACL. Melbourne, Australia: Association for Computational Linguistics, 2018: 484-490.

[99] Gormley M R, Eisner J. Structured belief propagation for NLP [C]//Proceedings of COLING. Beijing, China: Association for Computer Linguistics, 2015: 5-6.

[100] Stoyanov V, Eisner J. Minimum-risk training of approximate CRF-based NLP systems [C]//Proceedings of NAACL. Montréal, Canada: Association for Computational Linguistics, 2012: 120-130.

[101] 蒋勇. 结构化隐变量模型的表示和学习 [D]. 中国科学院大学, 2019.

[102] Smith N A. Synthesis lectures on human language technologies: Linguistic structure prediction [M]. Morgan and Claypool, 2011.

[103] Sutton C, McCallum A. An introduction to conditional random fields [J]. Found. Trends Mach. Learn., 2012: 267–373.

附录 A　关于 MBR 解码的推导

大体上讲，解码过程是将一个模型输出的概率分布转化为对应的系统输出. 给定输入 $\boldsymbol{x}$，理想情况下一个解码器将选择一个 $\boldsymbol{y}$，使得 $\ell(\boldsymbol{y}, \boldsymbol{y}^*)$ 最小化，其中 $\ell(\boldsymbol{y}, \boldsymbol{y}^*)$ 称为代价函数，衡量 $\boldsymbol{y}$ 与真实分配 $\boldsymbol{y}^*$ 的差异性. 由于通常情况下解码时不存在正确答案，因此我们转而用正确答案的所有可能取值 $\boldsymbol{y}'$ 的平均来代替

$$\boldsymbol{y} = \arg\min_{\boldsymbol{y}} \sum_{\boldsymbol{y}'} p(\boldsymbol{y}' \mid \boldsymbol{x}) \cdot \ell(\boldsymbol{y}, \boldsymbol{y}') \tag{A.1}$$

上述正是**最小贝叶斯风险**（Minimum Bayesian Risk, MBR）的内涵：给定输入 $\boldsymbol{x}$ 及对应的后验分布，选择 $\boldsymbol{y}$，使得期望代价（即风险，*Risk*）最小化 (Stoyanov 等[100]).

对句法分析,或其他结构化预测任务而言,通常我们采用的是最大后验概率（Maximum *A Posteriori*, MAP）解码,即寻求一个使得后验概率最大化的答案[101]. 这是 MBR 解码的一个特例，而对应的代价函数 $\ell^{MAP}(\cdot)$ 为一个简单的 0-1 指示函数，判断 $\boldsymbol{y}$ 与 $\boldsymbol{y}'$ 是否一致

$$\ell^{MAP}(\boldsymbol{y}, \boldsymbol{y}') = \mathbb{1}(\boldsymbol{y} \neq \boldsymbol{y}') \tag{A.2}$$

代入到公式 A.1得到

$$\begin{aligned}
\boldsymbol{y}^{MAP} &= \arg\min_{\boldsymbol{y}} \sum_{\boldsymbol{y}'} p(\boldsymbol{y}' \mid \boldsymbol{x}) \cdot \ell^{MAP}(\boldsymbol{y}, \boldsymbol{y}') \\
&= \arg\min_{\boldsymbol{y}} \sum_{\boldsymbol{y}'} p(\boldsymbol{y}' \mid \boldsymbol{x}) \cdot \mathbb{1}(\boldsymbol{y} \neq \boldsymbol{y}') \\
&= \arg\min_{\boldsymbol{y}} 1 - p(\boldsymbol{y} \mid \boldsymbol{x}) \\
&= \arg\max_{\boldsymbol{y}} p(\boldsymbol{y} \mid \boldsymbol{x})
\end{aligned} \tag{A.3}$$

上式与 MAP 解码的选择概率最大的 $\boldsymbol{y}$ 的目标等价.

以基于弧分解假设的一阶句法分析模型为例，式 A.3的 MAP 解码进一步写为

$$\begin{aligned}
\boldsymbol{y}^{MAP} &= \arg\max_{\boldsymbol{y}} p(\boldsymbol{y} \mid \boldsymbol{x}) \\
&= \arg\max_{\boldsymbol{y}} \frac{\exp\left(\sum_{i \rightarrow j \in \boldsymbol{y}} \mathrm{s}(i \rightarrow j)\right)}{Z(\boldsymbol{x}) \equiv \sum_{\boldsymbol{y}'} \exp\left(\sum_{i' \rightarrow j \in \boldsymbol{y}'} \mathrm{s}i' \rightarrow j\right)} \\
&= \arg\max_{\boldsymbol{y}} \sum_{i \rightarrow j \in \boldsymbol{y}} \mathrm{s}(i \rightarrow j)
\end{aligned} \tag{A.4}$$

去掉共同的分母，分子为对应句法树的分值，上式的含义是获取分值最大的句法树，这个目标可以通过常见的一些解码算法（如 Eisner，MST 等）达到.

而对于 MBR 解码，我们定义代价函数为在每条弧上的指示函数，

$$\ell^{MBR}(\boldsymbol{y}, \boldsymbol{y}') = \sum_{i \rightarrow j \in \boldsymbol{y}, i' \rightarrow j \in \boldsymbol{y}'} \mathbb{1}(i \neq i') \tag{A.5}$$

代入到公式 A.1得到

$$\begin{aligned}
\boldsymbol{y}^{MBR} &= \arg\min_{\boldsymbol{y}} \sum_{\boldsymbol{y}'} p(\boldsymbol{y}' \mid \boldsymbol{x}) \cdot \ell^{MBR}(\boldsymbol{y}, \boldsymbol{y}') \\
&= \arg\min_{\boldsymbol{y}} \sum_{\boldsymbol{y}'} p(\boldsymbol{y}' \mid \boldsymbol{x}) \cdot \sum_{i \rightarrow j \in \boldsymbol{y}, i' \rightarrow j \in \boldsymbol{y}'} \mathbb{1}(i \neq i') \\
&= \arg\min_{\boldsymbol{y}} \sum_{i \rightarrow j \in \boldsymbol{y}, i' \rightarrow j \in \boldsymbol{y}'} \sum_{\boldsymbol{y}'} p(\boldsymbol{y}' \mid \boldsymbol{x}) \cdot \mathbb{1}(i \neq i') \\
&= \arg\min_{\boldsymbol{y}} \sum_{i \rightarrow j \in \boldsymbol{y}} \sum_{\boldsymbol{y}' : i' \rightarrow j \notin \boldsymbol{y}'} p(\boldsymbol{y}' \mid \boldsymbol{x}) \\
&= \arg\min_{\boldsymbol{y}} \sum_{i \rightarrow j \in \boldsymbol{y}} \left(1 - \sum_{\boldsymbol{y}' : i' \rightarrow j \in \boldsymbol{y}'} p(\boldsymbol{y}' \mid \boldsymbol{x})\right) \\
&= \arg\max_{\boldsymbol{y}} \sum_{i \rightarrow j \in \boldsymbol{y}} \sum_{\boldsymbol{y}' : i' \rightarrow j \in \boldsymbol{y}'} p(\boldsymbol{y}' \mid \boldsymbol{x}) \\
&= \arg\max_{\boldsymbol{y}} \sum_{i \rightarrow j \in \boldsymbol{y}} p(i \rightarrow j \mid \boldsymbol{x})
\end{aligned} \tag{A.6}$$

式 A.6和式 A.4的唯一区别在于要求和的项是边缘概率而非分值. 因此，在一阶依存句法分析的场景下，MBR 解码只需要给定弧的边缘概率，然后直接应用和 MAP 解码一样的解码算法即可 (Smith 等[28], Smith[102]).

■

附录 B　用于结构化预测的平均场变分推断的推导

对于一般的结构化任务而言，我们需要得到后验分布 $P(\boldsymbol{y} \mid \boldsymbol{x})$. 我们可以将输出 $\boldsymbol{y}$ 按照对应的因子图分解为若干个因子的集合 (Sutton 等[103])，相应的 $P(\boldsymbol{y} \mid \boldsymbol{x})$ 定义为[1]

$$P(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{\prod_{\alpha} \psi_{\alpha}(y_{\alpha})}{Z(\boldsymbol{x}) \equiv \sum_{\boldsymbol{y}'} \prod_{\alpha} \psi_{\alpha}(y'_{\alpha})} \tag{B.1}$$

其中 Z 为配分项，$\psi_{\alpha}(\cdot)$ 为因子 α 的势函数，我们定义未归一化的分布 $\tilde{P}(\boldsymbol{y}) = \prod_{\alpha} \psi_{\alpha}(y_{\alpha})$.

变分推断建模一个近似分布 $Q(\boldsymbol{y})$，并最小化其与真实分布的 KL 散度（Kullback–Leibler Divergence）

$$\begin{aligned} Q^*(\boldsymbol{y}) &= \arg\min_{Q} KL(Q \parallel P) \\ &= \arg\min_{Q} E_Q\left[\log \frac{Q(\boldsymbol{y})}{P(\boldsymbol{y})}\right] \\ &= \arg\min_{Q} E_Q\left[\log Q(\boldsymbol{y})\right] - E_Q\left[\log P(\boldsymbol{y})\right] \\ &= \arg\min_{Q} \log Z - \underbrace{\left(E_Q\left[\log \tilde{P}(\boldsymbol{y})\right] - E_Q\left[\log Q(\boldsymbol{y})\right]\right)}_{ELBo} \end{aligned} \tag{B.2}$$

去掉配分项，定义

$$\mathcal{L}(Q) = ELBo = E_Q\left[\log \tilde{P}(\boldsymbol{y})\right] - E_Q\left[\log Q(\boldsymbol{y})\right] \tag{B.3}$$

我们将最小化 KL 散度的目标函数代替为最大化 ELBo

$$\begin{aligned} Q^*(\boldsymbol{y}) &= \arg\max_{Q} \mathcal{L}(Q) \\ &= \arg\max_{Q} E_Q\left[\log \tilde{P}(\boldsymbol{y})\right] - E_Q\left[\log Q(\boldsymbol{y})\right] \end{aligned} \tag{B.4}$$

Mean Field 假设每个变量相互独立，即有 $Q(\boldsymbol{y}) = \prod_i Q_i(y_i)$. 因此上式可应用坐标上升法（Coordinate Ascent），每次迭代优化一个变量 $Q_j(y_j)$，保持其余变量 $Q_{-j}(y_{-j})$

[1]后面方便起见我们省略输入 $\boldsymbol{x}$ 的标记

不变. 对于目标函数

$$\mathcal{L}(Q) = \underbrace{E_Q\left[\log \tilde{P}(\boldsymbol{y})\right]}_{①} - \underbrace{E_Q\left[\log Q(\boldsymbol{y})\right]}_{②} \tag{B.5}$$

分别有

$$\begin{aligned}
① &= E_Q\left[\log \tilde{P}(\boldsymbol{y})\right] \\
&= \int_{\boldsymbol{y}} Q(\boldsymbol{y}) \log \tilde{P}(\boldsymbol{y}) d\boldsymbol{y} \\
&= \int_{\boldsymbol{y}} \prod_i Q_i(y_i) \log \tilde{P}(\boldsymbol{y}) d\boldsymbol{y} \\
&= \int_{y_j} Q_j(y_j) \left(\int_{y_{-j}} Q_{-j}(y_{-j}) \log \tilde{P}(\boldsymbol{y}) dy_{-j} \right) dy_j \\
&= \int_{y_j} Q_j(y_j) E_{Q_{-j}}\left[\log \tilde{P}(\boldsymbol{y})\right] dy_j \\
② &= E_Q\left[\log Q(\boldsymbol{y}))\right] \\
&= \int_{\boldsymbol{y}} Q(\boldsymbol{y}) \sum_i \log Q_i(y_i) d\boldsymbol{y} \\
&= \int_{\boldsymbol{y}} \prod_i Q_i(y_i) \sum_i \log Q_i(y_i) d\boldsymbol{y} \\
&= \sum_i \int_{\boldsymbol{y}} Q_i(y_i) Q_{-i}(y_{-i}) \log Q_i(y_i) d\boldsymbol{y} \\
&= \sum_i \int_{y_i} Q_i(y_i) \log Q_i(y_i) dy_i \int_{y_{-i}} Q_{-i}(y_{-i}) dy_{-i} \\
&= \sum_i \int_{y_i} Q_i(y_i) \log Q_i(y_i) dy_i \\
&= \int_{y_j} Q_j(y_j) \log Q_j(y_j) dy_j + \sum_{i \neq j} \int_{y_i} Q_i(y_i) \log Q_i(y_i) dy_i \\
&= \int_{y_j} Q_j(y_j) \log Q_j(y_j) dy_j + C
\end{aligned} \tag{B.6}$$

C 代表常数项，因此有

$$\begin{aligned}\mathcal{L}(Q) &= ① - ② \\ &= \int_{y_j} Q_j(y_j) \underbrace{E_{Q_{-j}}\left[\log \tilde{P}(\boldsymbol{y})\right]}_{\log \dot{P}_j(y_j)} dy_j - \int_{y_j} Q_j(y_j) \log Q_j(y_j) dy_j + C \\ &= \int_{y_j} Q_j(y_j) \log \frac{\dot{P}_j(y_j)}{Q_j(y_j)} dy_j + C \\ &= -KL\left(Q_j(y_j) \parallel \dot{P}_j(y_j)\right) + C\end{aligned} \tag{B.7}$$

因此最大化 $\mathcal{L}(Q)$ 即最小化 $KL\left(Q_j(y_j) \parallel \dot{P}_j(y_j)\right)$. 当两个分布相等时 KL 散度最小，因此相应的更新公式为

$$\begin{aligned}Q_j^*(y_j) &= \exp\left(E_{Q_{-j}}\left[\log \tilde{P}(\boldsymbol{y})\right]\right) \\ &= \exp\left(E_{Q_{-j}}\left[\log \prod_{\alpha} \psi_\alpha(y_\alpha)\right]\right) \\ &= \exp\left(\int_{y_{-j}} Q_{-j}(y_{-j}) \sum_{\alpha} \log \psi_\alpha(y_\alpha) dy_{-j}\right) \\ &= \exp\left(\sum_{\alpha} \int_{y_{-j}} Q_{-j}(y_{-j}) \log \psi_\alpha(y_\alpha) dy_{-j}\right) \\ &\propto \exp\left(\sum_{\alpha: j \in \mathcal{N}(\alpha)} \int_{y_{-j}} Q_{-j}(y_{-j}) \log \psi_\alpha(y_\alpha) dy_{-j}\right)\end{aligned} \tag{B.8}$$

$\mathcal{N}(\alpha)$ 代表与因子 α 连接的变量，上式中，所有与变量 j 无关的因子对应的积分之和为一个常数项. 继续推导得

$$\begin{aligned}Q_j^*(y_j) &\propto \exp\left(\sum_{\alpha: j \in \mathcal{N}(\alpha)} \int_{y_{-j}} Q_{-j}(y_{-j}) \log \psi_\alpha(y_\alpha) dy_{-j}\right) \\ &\propto \exp\left(\sum_{\alpha: j \in \mathcal{N}(\alpha)} \int_{y_{\mathcal{N}(\alpha)-j}} Q_{\mathcal{N}(\alpha)-j}(y_{\mathcal{N}(\alpha)-j}) \log \psi_\alpha(y_\alpha) dy_{\mathcal{N}(\alpha)-j} \int_{y_{-\mathcal{N}(\alpha)}} Q_{-\mathcal{N}(\alpha)}(y_{-\mathcal{N}(\alpha)}) dy_{-\mathcal{N}(\alpha)}\right) \\ &\propto \exp\left(\sum_{\alpha: j \in \mathcal{N}(\alpha)} \int_{y_{\mathcal{N}(\alpha)-j}} Q_{\mathcal{N}(\alpha)-j}(y_{\mathcal{N}(\alpha)-j}) \log \psi_\alpha(y_\alpha) dy_{\mathcal{N}(\alpha)-j}\right) \\ &\propto \exp\left(\sum_{\alpha: j \in \mathcal{N}(\alpha)} E_{Q_{\mathcal{N}(\alpha)-j}} \log \psi_\alpha(y_\alpha)\right)\end{aligned} \tag{B.9}$$

因此，更新公式为

$$Q_j^*(y_j) \propto \exp\left(\sum_{\alpha: j \in \mathcal{N}(\alpha)} E_{Q_{\mathcal{N}(\alpha)-j}} \log \psi_\alpha(y_\alpha) \right) \tag{B.10}$$

对于包含高阶因子的模型，我们引入记号 α 和 β 分别代表一阶因子和高阶因子. 容易得，相应的更新公式为

$$Q_j^*(y_j) \propto \psi_\alpha(y_\alpha) \cdot \exp\left(\sum_{\beta: j \in \mathcal{N}(\beta)} E_{Q_{\mathcal{N}(\beta)-j}} \log \psi_\beta(y_\beta) \right) \tag{B.11}$$

■

攻读学位期间的成果

· 论文

(1) **Yu Zhang**, Zhenghua Li, Min Zhang. 2020. *Efficient Second-Order TreeCRF for Neural CRF Dependency Parsing*. In Proceedings of ACL, pages 3295–3305, Online. (CCF-A 类会议)

(2) **Yu Zhang***, Houquan Zhou*, Min Zhang. 2020. *Fast and Accurate Neural CRF Constituency Parsing*. In Proceedings of IJCAI, pages 4046-4053, Online. (CCF-A 类会议)

(3) Houquan Zhou*, **Yu Zhang***, Zhenghua Li, Min Zhang. 2020. *Is POS Tagging Necessary or Even Helpful for Neural Dependency Parsing?*. In Proceedings of NLPCC. pages 179-191, Zhengzhou, China (CCF-C 类会议, ***Best Paper Award***)

(4) Wei Jiang, Zhenghua Li, **Yu Zhang**, Min Zhang. 2019. *HLT@SUDA at SemEval 2019 Task 1: UCCA Graph Parsing as Constituent Tree Parsing*. In Proceedings of SemEval, pages 11–15, Minneapolis, Minnesota, USA.

· 比赛

(1) 2020 语言与智能技术竞赛比赛，第六名.

(2) 2019 语义分析国际评测比赛，第一名.

· 实习

(1) 2020/8--2021/2. 杭州-阿里巴巴-达摩院.

致谢

养天地正气，法古今完人. 从本科到硕士，转眼间在美丽的苏州大学校园内度过了七年的求学时光. 在这段不算短的人生旅途中，无论是学识上还是生活阅历上我都成长良多.

首先，我要感谢我的导师李正华老师. 李老师永远以饱满的热情和专注的态度面对工作和生活，永远是我以后求学和工作的一个榜样.

感谢尊敬的张民老师，张老师以高标准要求每一个学生，营造了组内浓厚专一的科研氛围. 此外，张老师敏锐的思维、渊博的知识、平易近人的风格、深深的影响了我，平时的相处让我获益良多. 感谢陈文亮老师，陈老师开朗热情，在学业上给予了我很多指导. 同样感谢周国栋、朱巧明、李寿山、洪宇、段湘煜和李军辉等苏州大学自然语言处理实验室的所有老师，各位老师严谨的治学态度和进取的专业精神是我的榜样.

感谢周厚全师弟，厚全师弟涉猎广博，热爱阅读，富有好奇心，在平时的讨论中总是能给我很多启发. 在课题研究上我们有很多合作，也取得了很多成果，希望以后继续合作，互相促进.

感谢同组的夏庆荣师兄、龚晨师姐、李英师姐和张月师姐，各位师兄师姐总是十分热心的解决我生活和研究上遇到的困难. 感谢章波、黄德朋、江心舟师兄，彭雪师姐，在我还是萌新的时候对我的帮助，以及平时对我的关照. 感谢同届的蒋炜、陆凯华、吴锟、刘亚慧同学，大家在一起互相帮助，互相进步. 感谢沈嘉钰、李嘉诚、侯洋、李帅克、周仕林、刘泽洋、李扬师弟，还有周明月、杨浩萍师妹，十分珍惜与大家相处的美好时光.

此外，还要感谢实习期间相处的王涛师兄、蒋勇师兄，以及王新宇、胡泽川、蔡炯和马欣尹同学. 特别是感谢蒋勇师兄在我实习期间对我的关照，以及在课题研究上的悉心帮助和指导.

感谢我的父母还有家人们，你们总是我心灵上的港湾和寄托，无论何时都能给我最无私的帮助.

最后，我还要感谢各位评审老师，感谢各位老师们在百忙之中抽取时间对本文进行评审，并提出宝贵的修改意见.

学位论文答辩委员会决议

包括：1、对论文的评价，包括选题的理论价值和实践意义，论文理论、方法上的开拓与创新，论据的可靠充分与结论的正确性；论文所反映的作者学术视野（对本学科及相关领域研究动态的把握）、基础理论、专业知识、写作能力等；

2、对答辩的评价；

3、是否同意通过论文答辩，是否建议授予学位或是否建议在规定时间内修改论文后重新答辩一次的结论。

论文在依存句法分析和成分句法分析这两种句法分析任务上，探讨了基于树形条件随机场的高阶方法对句法分析器性能和效率的影响。选题具有很好的理论价值和实践意义，以及一定的创新性。 目前主流的句法分析方法大多基于神经网络方法，并采用了一个简化的学习目标，相对应地，传统方法中大多采用了结构化学习以及高阶建模。论文针对这一对比，提出了将当前的句法分析器与传统方法做一个联结。论文提出在神经网络模型中采用树形条件随机场来最大化树概率，并进一步提出采用高阶建模。为了改善带来的效率问题，论文分别尝试了批次化计算和变分推断近似方法来加速。结果表明结构化建模和高阶方法对于目前的句法分析器仍然是有益的。

作者比较全面地论述了相关研究领域国内外研究情况，所采用的研究方法和技术手段体现了作者良好的学术研究基础和能力。论文成果在研究方法等方面有所创新，其中的新方法、新思路具备了很好的应用价值。论文写作层次清晰，逻辑结构合理，文字流畅，符合学术规范。

论文质量优秀。答辩过程中陈述清楚，回答问题准确。

答辩委员会经讨论，认为该论文已达到硕士学位论文水平，一致同意其通过论文答辩，建议授予硕士学位。

答辩委员会主席：孔芳　　秘书：张雅静

委员：陈文亮、李培峰、钱龙华、朱晓旭

_____、_____、_____、_____

2021 年 5 月 22 日

注：本表内容（包括答辩名单）可手签或打印