用 Transformer 近似 CKY

Ghazal Khalighinejad杜克大学
gk126@duke.edu

Ollie Liu南加州大学
zliu2898@usc.edu

Sam Wiseman杜克大学
swiseman@cs.duke.edu

arXiv:2305.02386v2 [cs.CL] 5 Nov 2023

摘要

我们研究 Transformer 模型近似 CKY 算法的能力，并利用它们直接预测句子的句法分析结果，从而避免 CKY 算法对句子长度的三次方依赖。我们发现，在标准成分句法分析基准测试中，这种方法的表现与使用 CKY 的同类句法分析器不相上下，甚至更好，同时速度更快。我们还评估了这种方法在随机概率上下文无关文法下进行句法分析的可行性。在此，我们发现，随着语法歧义性增加，性能会下降，这表明 Transformer 并未完全捕捉 CKY 的计算过程。不过，我们还发现，加入额外的归纳偏置有所帮助，并提出了一种新方法：借助相对于分析表表示的梯度来预测句法分析结果，这与 CKY 算法是配分函数某个变体相对于分析表的次梯度这一性质相类比。

1 引言

基于 Transformer 的句法分析器（Vaswani 等人，2017年）目前代表着成分句法分析领域的最先进水平（Mrini 等人，2020年；Tian 等人，2020年），近期研究（Tenney 等人，2019年；Jawahar 等人，2019年；Li 等人，2020年；Murty 等人，2022年；Eisape 等人，2022年；Zhao 等人，2023年）发现，Transformer 能够学习文本跨度的类似成分的表示。鉴于这些成果，我们很自然会想知道，Transformer 是否掌握了我们与成分句法分析联系在一起的算法过程，例如 CKY 算法（Kasami，1966年；Younger，1967年；Baker，1979年）。事实上，人们可能会猜想，Transformer 的各层正在逐步构建短语级表示，就像 CKY 算法本身逐步构建其分析表一样。鉴于近期有研究探讨 Transformer（以及更广泛而言，图神经网络）实现或近似经典算法的能力（Xu 等人，2019年；Csordás 等人，2021年；Dudzik 和 Veli˘ckovi´c，2022年；Delétang 等人，2022年，等等）。

如果标准 Transformer 确实是在近似 CKY，那么这将带来若干影响。从实践角度看，这一发现或许能催生更快的神经句法分析器。当前最先进的句法分析器（例如 Kitaev 和 Klein (2018年) 及后续工作（Kitaev 等人，2019年；Tian 等人，2020年））往往在 Transformer 表示之上实现 CKY，因此句法分析的时间复杂度会随句子长度呈立方增长；而我们或许只需对句子运行 Transformer，就能提取出句法分析结果，其时间复杂度仅随句子长度呈平方增长。此外，鉴于学界和工业界都在投入大量精力来加速标准 Transformer（例如 Dao 等人 (2022年) 的工作），这些进展或许也能自动应用于句法分析。

除了更实际的考量之外，使用 Transformer 生成 CKY 解析结果这一任务，为研究赋予 Transformer 额外的归纳偏置是否有助于它们实现经典算法提供了绝佳机会；这是近期由Csordás 等人(2021年)和其他研究者探讨的课题。这类研究的结果也将有助于理解近期有关以标准方式训练的 Transformer 的计算能力的研究成果（Delétang 等人，2022年；Liu 等人，2023年）。

因此，我们首先证明，只需让一个预训练的 Transformer通过独立标注每个跨度来预测完整的句法分析——这与Corro (2020年)在训练时间采用的方法类似，只不过应用于不同的情境——就足以在标准成分句法分析基准上取得有竞争力甚至更好的表现，同时速度显著更快。

虽然这些成分句法分析结果令人鼓舞，

但这并不意味着训练后的 Transformer 正在实现 CKY 算法，因为 Transformer 可能只是在预测一个句法分析，而该分析在某个语法下未必得分最高。因此，我们继续研究 Transformer 在随机生成的概率上下文无关文法下预测 CKY 句法分析的表现。给定一个随机生成的概率上下文无关文法，检查预测的句法分析是否确实得分最高当然很容易。在这种情况下，我们发现 Transformer 的表现与概率上下文无关文法的歧义程度呈负相关，这表明它们实际上并未实现类似 CKY 的机制。同时，我们发现，在标准架构中融入额外的归纳偏置有所帮助，因此提出了一种新方法，利用梯度相对于分析表表示的梯度来预测句法分析。这种归纳偏置的灵感来自这样一个事实：CKY 算法可以被视为计算“最高分”配分函数的梯度（Eisner，2016年；Rush，2020年），我们发现，这显著提升了模型在随机概率上下文无关文法上的表现。

总而言之，我们表明：

- 使用 Transformer 直接预测解析结果，其表现可与显式 CKY 方法相媲美，同时速度更快；
- 这种令人瞩目的表现并非很可能是因为 Transformer 隐式实现了 CKY 算法，因为它们无法准确地对存在歧义的合成概率上下文无关文法进行句法分析；
- 引导模型根据分析表的梯度生成解析结果，能显著提升合成概率上下文无关文法的句法分析性能。

复现所有实验的代码可在此处获取：https://github.com/ghazalkhalighinejad/approximating-cky。

2 背景与符号约定

CKY 算法（Kasami，1966年；Younger，1967年；Baker，1979年）在概率上下文无关文法（PCFG）下计算句子的最高分句法分析。设 $G = (\mathcal{N}, \Sigma, \mathcal{R}, S, \mathcal{W})$ 为一个概率上下文无关文法，其中 $\mathcal{N}$ 是非终结符集合，Σ是终结符集合，$\mathcal{R}$ 是规则集合，S 是起始非终结符，且 $\mathcal{W}$ 是每条规则对应的概率集合（按左部归一化）。给定句子 $x \in \Sigma^T$，CKY 算法使用动态规划计算最高分句法分析在 G下对 x 进行分析，计算时间为 $O(|\mathcal{R}|T^3)$。

遵循Rush (2020年）中的记法，令 $\ell^R \in \mathbb{R}^{|\mathcal{R}|\times|\mathcal{N}|\times|\mathcal{N}|}$ 表示与概率上下文无关文法中的规则相关联的对数势（例如对数概率）G，并令 $\ell^E(x) \in \mathbb{R}^{T\times|\mathcal{N}|}$表示与输入句子中的每个词元相对应的对数势$x$。我们可以使用这些对数势来计算 $\beta \in \mathbb{R}^{T\times T\times|\mathcal{N}|}$对于 x的分析表，其中 $\beta[i,j,a]$ 表示在特定半环下，所有与 a-th 非终结符生成 $x_{i:j}$相关的权重之和。这些对数势还可用于计算得分最高的句法分析，我们将其称为 $\beta^* \in \{0,1\}^{T\times T\times|\mathcal{N}|}$；它是句法分析的独热表示，其中 $\beta^*[i,j,a] = 1$ 当且仅当$j \geq i$ 且非终结符 a 生成$x_{i:j}$ 出现在该句法分析中。

将 CKY 视为次梯度 近期研究（Eisner，2016年；Rush，2020年）强调，像 CKY 这样的推理算法，可以很好地视为一种特殊情形：针对与输入相关的对数势（的某种函数），计算广义对数配分函数的梯度或次梯度。具体而言，CKY 算法可视为针对输入句子 x的分析表，计算“最大得分”版本对数配分函数的次梯度；详情见附录 A。因此，β^* 可以借助自动微分框架相对容易地获得。然而，以这种方式计算它，实际上完全等同于运行传统 CKY 算法，因此仍需要 $O(|\mathcal{R}|T^3)$ 时间。

基于 Transformer 的 CKY 近似 本文其余部分将重点介绍如何训练一个模型（即推理网络（Kingma 和 Welling，2014年；Johnson 等人，2016年；Tu 和 Gimpel，2018年））直接预测 β^*，以期更高效地进行句法分析。我们不再像传统 CKY 算法那样，使用 ℓ^R 和 $\ell^E(x)$ 递归计算 β^*，也不再通过对 β^* 求导来计算 β，而是提出直接训练一个 Transformer，将句子映射为其正确的 β^*。我们希望训练好的 Transformer 能够在其权重中隐式地表示有关语法的信息，进而无需对数势，就能用于分析来自同一语法的未见句子。图 1以示意图的形式展示了这一方法。

我们希望在来自同一分布（即，采样得到的）留出句子上评估我们训练好的推理网络对 β^* 的近似效果。

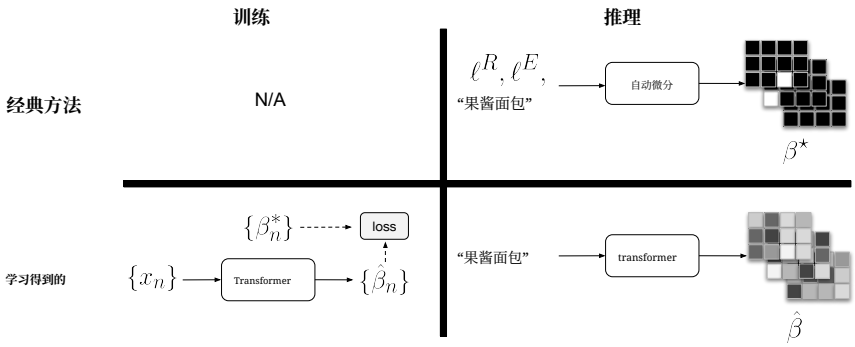


图 1：经典方法（上）和学习式 Transformer 近似（下）的训练与推理示意图。经典解析没有训练阶段，而是使用预定义的对数势对未见句子进行句法分析。我们提出的学习式解析器（即“推理网络”）在句子对及其根据一组对数势计算出的解析结果上进行训练，随后无需访问这些势即可对未见句子进行句法分析。

（来自与训练句子相同的概率上下文无关文法）。这与使用神经网络近似经典算法的绝大多数研究形成对比（Graves 等人，2014年；Kaiser 和 Sutskever，2016年），后者则是在分布外的输入上评估性能，尤其是比模型训练所用输入更长的输入。我们不关注长度泛化，首先是因为我们认为，即使是在域内进行泛化，在解析应用中也很有用；其次是因为我们发现，在随机概率上下文无关文法上训练的 Transformer 即使面对长度相同的输入，也难以实现泛化。

CKY 变体 现代基于跨度的神经成分句法分析器并不假设存在底层的概率上下文无关文法。相反，这些解析器以组合方式为跨度评分（Stern 等人，2017年；Gaddy 等人，2018年；Kitaev 和 Klein，2018年），所用对数势由神经网络（例如 Transformer）$\ell^S \in \mathbb{R}^{T\times T\times|\mathcal{N}|}$生成。这类解析器使用一种 CKY 变体，其计算时间要求为 $O(T^3)$。我们重点研究如何使用 Transformer 来近似经典 CKY 算法及这一更简单的变体。

3 预测解析结果

如第 2 节所述，我们将 $\beta^* \in \{0,1\}^{T\times T\times|\mathcal{N}|}$定义为一个与分析表大小相同的张量，用来表示句子 x 的 CKY 句法分析，其中 $\beta^*[i,j,a]=1$ 当且仅当非终结符 a 生成 $x_{i:j}$ 时取值。我们将使用一种非常简单的方法，通过 Transformer 预测 β^*。

令$\mathbf{h}_i \in \mathbb{R}^d$ 表示仅编码器 Transformer 最后一层对 x 中第 i 个词元的表示；令$\mathbf{h}_{i,\,:d/2}$ 和$\mathbf{h}_{i,\,d/2:}$（均属于 $\mathbb{R}^{d/2}$）分别表示$\mathbf{h}_i$中的前 $d/2$ 个和最后 $d/2$ 个元素。我们然后将 $\hat{}\beta_{ij} \in \Delta^{|\mathcal{N}|}$ 定义为

$$\hat{\beta}_{ij} \overset{\text{def}}{=} \text{softmax}(\text{FFN}([\mathbf{h}_{i,:d/2}; \mathbf{h}_{j,d/2:}])), \quad (1)$$

其中，我们将 $\mathbf{h}_i$ 的前半部分与 $\mathbf{h}_j$的后半部分拼接起来；FFN是一个类似 BERT 的（Devlin 等人，2019年）分类头，由一个隐藏层、GELU 非线性函数（Hendrycks 和 Gimpel，2016年）、LayerNorm（Ba 等人，2016年）以及最终映射至 $|\mathcal{N}|+1$ 的线性投影组成。该分类头会为每个非终结符标签以及一个非成分 标签分别生成一个分数。此表示与 Kitaev 和 Klein（2018年）在使用 CKY 处理$\hat{}$计算出的分数之前得到的表示相似（但并不相同）。我们将 的下三角部分（即沿前两个维度）置零，并将其作为对 β^*的近似。

$\hat{}$解码 请注意，根据$\mathbf{h}_i$ 构造 β 只需句子长度的二次时间，因此，如果我们能以低于三次方的时间从中提取句法分析，就能在渐近意义上优于 CKY。在实践中，我们只需为每个跨度选取得分最高的标签，并忽略最高分标签为非构成成分的跨度。随后，我们按预测跨度的结束词元位置递减排序，再按起始词元位置递增排序（采用稳定排序），并从左向右逐步构建树。因此，解码成本仅为 $O(T^2(|\mathcal{N}|+\log_2 T))$。[1]

训练 给定一个金标准（二叉化）句法分析 β^*，我们只需将预测每个跨度的标签视为 $T(T+1)/2$ 独立的多类别分类问题，

[1]注意，该过程仅适用于成分句法分析，因为其中允许使用一元产生式。对于以乔姆斯基范式表示的概率上下文无关文法进行句法分析时，我们只需取 $T-1$ 得分最高的跨度。

并使用标准交叉熵损失。请注意，Corro (2020年) 在不连续成分句法分析的背景下提出了这种独立跨度分类训练方法（不过对 非成分 标签的权重固定为零）。我们注意到，Yang 和 Tu(2023年) 的同期研究也通过独立地进行跨度预测，探索了训练和解码。

3.1 从 ChartGradients 解码句法分析

如第 2 节所述（另见附录 A），CKY 句法分析是“最大分数”配分函数的一个次梯度。该函数计算在概率上下文无关文法下，句子及其句法树的最大对数（联合）概率。如果我们希望做出类似 CKY 的预测，那么从某个评分函数的梯度中构造预测的句法分析，可能是一种有用的归纳偏置，正如 CKY 所做的那样。我们提出用 Transformer 来定义这个评分函数，并根据该评分函数相对于其输入的梯度来预测句法分析。

更具体地说，同样令$\mathbf{h}_i$ 表示仅编码器 Transformer 最后一层对句子 x 中第i个词元的表示，我们定义 x的以下“内在”分数：

$$\text{score}(x) \stackrel{\text{def}}{=} \text{FFN}(1/T \sum_{i=1}^{T} \mathbf{h}_i),$$

其中，FFN 是一个类似 BERT 的分类头，只输出一个逻辑值。因此，我们只需对 Transformer 最后一层的词元表示取平均池化，并将其输入前馈网络，以得到一个标量分数。

令$\mathbf{h}_i^{(l)}$ 表示仅编码器 Transformer 的l层对 i个词元的表示。由于分数是所有$\mathbf{h}_i^{(l)}$ 的可微函数，我们可以对它们求梯度。具体而言，令$\mathbf{g}_i \stackrel{\text{def}}{=} \frac{1}{L} \sum_{l=1}^{L} \nabla_{\mathbf{h}_i^{(l)}} \text{score}(x)$。也就是说，$\mathbf{g}_i$ 是score 相对于每个 Transformer 层中 i个词元表示的梯度的平均值。接着，我们可以根据$\mathbf{g}_i$构造跨度预测，并在方程（$\mathbf{h}_i$ ）中将其替换为式（1），得到

$$\hat{\beta^+}_{ij} \stackrel{\text{def}}{=} \text{softmax}(\text{FFN}([\mathbf{g}_{i,:d/2}; \mathbf{g}_{j,d/2:}])). \quad (2)$$

在本文余下部分，我们将使用方程（2）的模型称为“梯度解码”。

训练一个梯度解码模型，需要通过反向传播来计算参数梯度，经过一个反向传播相对于$\mathbf{h}_i^{(l)}$ 。幸运的是，借助现代自动微分框架（如 PyTorch），现在可以轻松实现这一点。我们在所有实验中都使用了该框架（Paszke 等人，2019年）。

讨论 值得注意的是，尽管 β^* 本身是最大分数配分函数的一个次梯度ˆ，但我们上述方案只是从内部分数 β^+ 函数的梯度中进行解码。也就是说，梯度分数函数的梯度$\mathbf{g}_i$ 会被拼接后输入一个额外的分类头，用于生成尺寸与分析表相同的结果。造成这一差异的原因在于计算开销：如果我们将尺寸与分析表相同的内容输入分数函数，并且分数函数需要对其输入运行 Transformer，那么我们的方法在 T上将呈四次方增长。相反，我们的方法对 T的依赖仍为二次方。不过，由于需要计算相对于尺寸为 LTd的 Transformer 表示的梯度，实际运行时它比使用方程（1）开销更大；详见附录C 。

4 成分句法分析实验

我们在两个主要设置下开展实验。首先，我们考察标准基准数据集上的现代神经成分句法分析。然后，我们考察随机生成的概率上下文无关文法下的句法分析。我们强调这两个设置之间的几个重要差异。首先，现代成分句法分析不依赖语法。因此，现代成分句法分析器不会预测某个概率上下文无关文法下得分最高的句法分析结果，并且它们使用 CKY 的一种更简单变体，该变体组合跨度，但没有语法规则的概念。尽管直接预测句法分析结果是否能与运行这一更简单的 CKY 变体相竞争仍然很有趣（至少从计算效率的角度来看），但很难区分 Transformer 是在学习这一 CKY 变体，还是仅仅在学习预测标准句法分析树。这一担忧促使我们采用第二个设置，即随机生成的概率上下文无关文法；在该设置下，每个概率上下文无关文法都有定义明确的最高得分句法分析结果，并且我们可以评估 Transformer 是否预测出了这些结果。我们在第 5节考察这一随机概率上下文无关文法设置。

数据集 我们在英语宾州树库（宾州树库；Marcus 等人，1993年）、中文树库（中文树库；Xue 等人，2005），以及 SPMRL 2013 和 2014 共享任务中的树库（Seddah 等人，

	英语		中文		德语		韩语	
	开发集	测试集	开发集	测试集	开发集	测试集	开发集	测试集
Kitaev 等人（2019年）[a]	-	95.59	-	91.75	-	90.20	-	88.80
Kitaev 等人（2019年）[b]	**95.61**	95.48	**94.23**	**92.13**	93.39	**90.32**	**89.74**	88.55
Ours	95.50	**95.64**	93.17	90.54	**93.47**	90.13	**89.74**	**89.05**
我们的 + 梯度解码	95.07	95.16	93.75	91.25	93	89.63	89.26	88.46

表 1：F_1 在 PTB、CTB 以及 SPMRL 2014 共享任务中的德语和韩语树库上的分数。所有模型都使用相同的预训练初始化；详情请见正文。Kitaev 等人(2019年)[a] 的结果摘自论文，其中使用了一个额外的因子化自注意力层；而Kitaev 等人(2019年)[b] 则是运行其代码但未使用该额外层所得的结果。

2013年）。我们全程采用标准的数据集划分。

模型详情 我们改进了由 Kitaev 和 Klein(2018年) 首次提出的分析表句法分析器，随后又根据 Kitaev 等人 (2019年) 的工作进行了完善，使其能够通过微调预训练模型来实现。具体而言，Kitaev 等人(2019年) 使用 CKY 句法分析结果与标准句法分析树之间基于间隔的结构化损失，对预训练的 BERT 模型（Devlin 等人，2019年）进行微调；而我们则按照公式（1）直接微调 BERT，使其预测句法分析结果，并采用第 3 节所述的独立跨度交叉熵损失进行训练。 Kitaev 等人 (2019年) 的部分实验还使用了一个额外的因子化自注意力层，该层以 BERT 的表示为输入；但在根据公式（1）或（2）进行预测时，我们并未使用这一层。

我们的实现是对公开的 Kitaev 等人 (2019年) 实现的修改版，[2] 该实现也是我们的主要基线。虽然这个句法分析器并非始终达到最先进水平，但其性能相当接近，而且最先进的句法分析器通常采用它的架构和方法（Mrini 等人，2020年；Tian 等人，2020年）。

用于初始化 Kitaev 等人 (2019年) 模型以及我们为英语和中文树库训练的模型的预训练模型分别为 BERT-large- uncased (Devlin 等人，2019年) 和 BERT-base- chinese[3],；用于初始化韩语、德语及其余 SPMRL 树库模型的则是 BERT-base-multilingual- cased (Devlin 等人，2019年)（见附录D）。与 Kitaev 等人 (2019年) 的实现相同，我们使用 AdamW (Loshchilov 和 Hutter，2018年；Kingma 和 Ba，2015年) 进行训练，直到验证集上的解析性能不再提升。我们采用 Kitaev 等人 (2019年) 提出的相同学习率调度器，

[2]https://github.com/nikitakit/ self-attentive-parser[3] https://huggingface.co/bert-base-chinese

推理速度（句/秒）	PTB	CTB
Kitaev 等人 (2019年)	485 (1.00x)	704 (1.00x)
Ours	949（1.96 倍）	1466（2.08 倍）

表 2：基于 CKY 的 Kitaev 等人 (2019年) 句法分析器与我们模型的推理速度（句子/秒），取宾州树库和中文树库开发集上的平均值。

先进行 160 步预热，然后当 F_1 分数不再提升时，将学习率乘以 0.5 以降低学习率。我们使用网格搜索来选择超参数，并在附录 F 中提供网格搜索的详细信息以及找到的最优超参数。

结果 在表 1中，我们使用标准的 `evalb` 评估，报告了这些数据集上的句法分析结果。我们发现，我们的方法与 Kitaev 等人 （2019年）的方法具有竞争力：在英语和韩语数据集上略胜于该方法，而在中文和德语数据集上略逊于该方法（其余 SPMRL 树库的结果见附录D ）。在这种设置下，梯度解码并未优于直接使用公式（1）。然而，正如我们在第 5节所讨论的，共享 Transformer 层对于体现梯度解码的优势非常重要，而使用预训练 BERT 模型无法有效做到这一点。

鉴于速度快得多，我们最简单的方法在这些成分句法分析基准测试中具有竞争力，这一事实颇为有趣。在表 2中，我们比较了我们的方法（每秒句法分析的句子数）与基线解析器的速度，发现前者大约快两倍。这些数字反映了使用 Kitaev 等人 (2019年) 的解析器及其改进版本，对宾州树库和中文树库开发集进行句法分析所需的时间。两项实验都在同一台机器上运行，使用 NVIDIA RTX A6000 GPU。我们还要强调，这一比较在某种程度上对基线解析器更有利，因为我们使用了原版

	PTB	CTB
Kitaev 等人（2019年）	89.83	81.72
Ours	**90.07**	80.98
我们的方法 +共享层	89.91	**81.76**
+ 梯度解码		

表 3：我们的方法与基线模型在宾州树库和中文树库开发集上的性能（分别以 F_1 衡量），训练时从头开始。“共享层”表示共享 Transformer 层。

代码的批处理方式，而我们的方法可以极其轻松地创建大型填充批次，从而进一步加快句法分析。此外，我们还与 Supar 的（Zhang 等人，2020b，a）实现进行了比较，但未将其纳入，因为它比 Kitaev 等人 (2019年) 的实现更慢。

虽然上述实验都是对预训练的 BERT 风格模型进行微调以用于句法分析，但考察这些模型从头开始训练时的性能也很有意义。因此，我们使用与表 1 中相同规模的模型，在宾州树库和中文树库训练集上从头开始训练 Transformer，并在表 3中报告开发集上的结果。我们再次发现，不基于 CKY 的模型具有竞争力。此外，由于现在可以轻松共享 Transformer 层，而不牺牲预训练层带来的优势，梯度解码的效果更加显著。当不使用梯度解码时，我们没有发现共享 Transformer 层能带来相应的收益。

5 随机 PCFG 实验

为了生成随机概率上下文无关文法，我们采用 Clark and Fijalkow （2021年）的方法。[4] 他们的方法首先生成一个合成的上下文无关文法（CFG），其中指定终结符、非终结符、二元规则和词法规则的数量。随后，他们使用基于 EM 的估计程序（Lari and Young，1990年；Carroll and Charniak，1992年）更新生成规则，为规则赋予概率，使估计得到的概率上下文无关文法的长度分布与宾州树库语料库的长度分布相似（Marcus 等人，1993年）。

5.1 数据

Clark and FijalkowClarkand Fijalkow（2021年）提出的方法使我们能够构造具有指定数量非终结符和规则的随机文法；我们生成了含有 20 个非终结符的文法，[5] 以及分别含有 100、400，以及分别含有 800 条规则的文法。终结符和词法规则的数量设为 5000。每个非终结符对应的规则越多，通常歧义性越高，因此我们预计，含有 800 条规则的合成文法会比含有 100 条规则的文法难以进行句法分析得多；而含有 20 个非终结符的合成文法会比含有更多非终结符的文法更难进行句法分析。

通常用给定句子时句法树的条件熵来量化歧义性（Clark andFijalkow，2021年），可以通过从概率上下文无关文法中采样句法树，并对给定其对应句子时这些树的条件概率的负对数取平均来估计。概率上下文无关文法的条件熵 G 因此估计为

$$\hat{H}_G(\tau|x) = -\frac{1}{N}\sum_{n=1}^{N} \log \frac{p_G(x^{(n)}, \tau^{(n)})}{p_G(x^{(n)})},$$

其中 $(x^{(n)}, \tau^{(n)})$ 从 G 中采样，而 $p_G(x^{(n)})$ 使用 inside 算法计算。我们使用 $N = 1000$ 个样本。下面我们报告每个随机文法的 $H_G(\tau|x)$ 以及解析性能；我们将看到，$\hat{H}_G(\tau|x)$ 与性能呈负相关。

我们的数据集由从这些概率上下文无关文法中采样的句子及其对应的解析结果组成，这些解析结果是使用 CKY 算法进行句法分析得到的。

数据集大小 由于我们旨在测试 Transformer 捕捉 CKY 算法的能力，因此必须确保训练集足够大，使预测错误可以归因于 Transformer 未能学会该算法，而非训练数据稀疏。为此，我们只需不断添加数据，直到验证性能趋于平稳。具体而言，我们使用 ∼200K 个句子进行训练，并保留 2K 个句子。为了控制数据集的复杂度，我们还将句子长度上限设为 30，这使平均句子长度从每句 20–25 个词降至 18 个词。</final>

5.2 模型与基线

鉴于上一节中的实验主要采用标准的类 BERT 架构（包括经过微调的架构和从头训练的架构），本节还将考虑使用旨在提升“算法”任务性能的 Transformer 变体来进行句法分析预测（如公式（1）和（2）所示）。事实上，近期

[4]参见 https://github.com/alexc17/syntheticpcfg。[5]我们发现，当使用超过 20 个非终结符时，性能模式保持一致，但结果略高

由于语法变得不那么含糊。

$\lvert\mathcal{R}\rvert$	100		400		800	
$\hat{H}_G(\tau\lvert x)$	0.35		3.99		9.74	
	F_1	Δ_{lp}	F_1	Δ_{lp}	F_1	Δ_{lp}
CKY	100.0	0	100.0	0	100.0	0
Transformer	98.31	0.04	87.86	0.29	74.80	0.45
Transformer +共享层	98.61	0.03	89.89	0.19	79.91	0.29
Transformer +共享层 +复制门	98.97	0.02	89.43	0.21	78.64	0.31
Transformer +共享层 +梯度解码	**99.06**	**0.01**	**91.07**	**0.17**	**81.78**	**0.28**

表4：随机生成的概率上下文无关文法中，带标签跨度 F_1 性能（如 evalb 所示），分别包含 $|\mathcal{N}| = 20$ 个非终结符，以及 $|\mathcal{R}| = 100$、400 和 800 条规则。$\hat{}$ $H_G(\tau|x)$ 表示对语法条件熵的估计。 Δ_{lp} 是 CKY 与预测句法树之间的平均对数概率差。“SL”表示共享 Transformer 层，“CG”表示使用复制门，“GD”表示使用梯度解码。

研究表明，Transformer 往往无法在算法任务上实现泛化（Saxton 等人，2019年；Hupkes 等人，2020年；Dubois 等人，2020年；Chaabouni 等人，2021年），这促使研究者对架构进行修改，例如通用 Transformer（UT；Dehghani 等人，2018年）和神经数据路由器（NDR；Csordás 等人，2021年）。

UT 和 NDR 都采用的一项重要架构改动是让所有 Transformer 层共享相同的参数；这一改动旨在体现这样一种直觉：递归计算通常需要多次应用同一个函数。NDR 还使用了“复制门”，允许将第 l 层的 Transformer 表示直接复制为第 $l+1$ 层的表示，而不再进行进一步处理；此外还使用了“几何注意力”，使自注意力更倾向于关注附近的词元。我们在初步实验中发现，几何注意力会显著降低性能，因此报告的结果仅采用共享层和复制门这两项改动。

本节中的所有模型均从头开始训练。这对于 UT 和 NDR 风格的架构来说很方便，因为我们没有大型预训练模型；不过，我们在初步实验中也发现，在随机PCFG上，预训练的原始 Transformer（例如 BERT）并不比从头开始训练的 Transformer 表现更好。我们确实发现，对一个 BERT 风格模型进行微调能带来适度的收益。该模型是在我们随机生成的语法中的句子上预训练的（而非自然语言），这与近期Zhao 等人 (2023年) 的研究结果一致；详情见附录 E 。我们的实现基于 Hugging Face Transformer 库中的 BERT 实现（Wolf 等人，2020年）。

句子/第二			
批大小	32	64	128
CKY	698	1162	1576
Ours	1817	2132	2161

表 5：在包含 $|\mathcal{R}| = 400$ 条规则的随机文法上，基于图形处理器的 torch-struct CKY 句法分析与我们基于 Transformer 的方法的推理速度。我们报告模型和 CKY 在 60K 个样本上分别以 32、64 和 128 为批量大小运行 5 次的中位速度。

评估我们使用以下模型返回的 CKY 句法分析结果，评估预测句法树：`torch-struct` (Rush，2020年)，并按照[1] 中对跨度标签的处理方式进行评估，参见 `evalb`。由于可能存在多个得分最高的解析结果，我们还报告每条产生式上 CKY 句法分析结果与预测句法分析结果之间的对数概率差，并对所有产生式取平均值。在表格中，我们将此指标称为“Δ_{lp}”。由于基于 Transformer 的句法分析器可能会预测无效的产生式，因此我们通过向每个产生式概率加上 10^{-5}并重新归一化，对概率上下文无关文法进行平滑处理，然后再计算 Δ_{lp}。

结果 这些合成句法分析实验的结果见表4，其中展示了三种具有不同条件熵的随机文法的解析性能。我们发现，随着歧义性增加，Transformer 的表现越来越吃力，这表明这些模型实际上并未实现类似 CKY 的处理方式。同时，我们发现，引入归纳偏置几乎在所有情况下都有帮助，而梯度解码与共享 Transformer 层相结合时，在所有文法上表现最佳。

我们在表5中进行了速度评估，并将其与 `torch-struct` （Rush，2020年）中的 CKY 实现进行了比较；该实现经过了性能优化

，可在图形处理器上高效运行。对于固定数量的非终结符，`torch-struct` 的 CKY 实现速度和我们基于 Transformer 的近似方法的速度，都不会显著受到规则数量的影响。因此，我们针对仅包含 400 条规则的合成文法，展示了三种不同批次大小下的句法分析速度结果。所有计时实验均在同一台机器上运行，并使用 NVIDIA RTX A6000 GPU。我们发现，基于 Transformer 的近似方法始终更快，不过其优势会随着批次大小的增加而减小。不过需要注意的是，由于我们的实现使用了 Hugging Face `transformers` 组件（Wolf 等人，2020年），而这些组件目前尚未利用 FlashAttention（Dao 等人，2022年）等优化技术，因此我们的方法很可能还可以进一步提速。

6 相关工作

现代神经成分句法分析器通常分为三类：基于图表的句法分析器（Stern et al., 2017年；Gaddy et al., 2018年；Kitaev and Klein, 2018年；Mrini et al., 2020年；Tian et al., 2020年；Tenney et al., 2019年；Jawahar et al., 2019年；Li et al., 2020年；Murty et al., 2022年）、转移式句法分析器（Zhang and Clark, 2009；Cross and Huang, 2016年；Vaswani and Sagae, 2016年；Vilares andGómez-Rodríguez, 2018年；Fernandez Astudillo et al., 2020年），或序列到序列式句法分析器（Vinyals et al., 2015年；Kamigaito et al., 2017年；Suzuki et al., 2018年；Fernández-González and Gómez-Rodríguez, 2020年；Nguyen et al., 2021年；Yang and Tu, 2022年）。

我们的方法与基于分析表的方法最为相似，因为我们会为所有非终结符的所有跨度计算分数。然而，与基于图表的句法分析器不同，我们的目标只是近似执行 CKY 算法，或将其运行过程摊销化。与基于转移或序列到序列风格的方法不同，我们的近似方法尝试联合且独立地预测句法分析，而不是逐步进行。在基于分析表的神经句法分析器领域，Kitaev 和 Klein (2018年) 以及 Kitaev 等人 (2019年) 的工作影响深远；而 Mrini 等人 (2020年) 和 Tian 等人(2020年) 提出的先进成分句法分析器则在改进这一方法的同时对其加以调整。

采用独立跨度分类训练目标，并在推理时使用贪心解码的方法也已得到探索，Zhang 等人(2019年) 在依存句法分析领域中探讨了这一方法。依存句法分析。值得强调的是，尽管Zhang 等人(2019年) 的研究表明，二次复杂度的评分模型可以近似二次复杂度的 Chu-Liu-Edmonds 算法（Chu 和 Liu，1965），该算法用于依存句法分析中的解码；而我们的工作着眼于使用二次复杂度的评分模型，近似三次复杂度的解码算法。

近期关于神经算法推理的研究也为我们提供了启发（Xu 等人，2019年；Csordás 等人，2021年；Dudzik 和 Veli˘ckovi´c，2022年；Delétang 等人，2022年；Ibarz 等人，2022年；Liu 等人，2023年），其中一些研究致力于使用图神经网络和 Transformer 分别求解经典动态规划程序（Veli˘ckovi´c 等人，2022年）和马尔可夫决策过程（Chen 等人，2021年）。学习计算 CKY 的一个特点是，它与许多其他算法推理任务（包括其他动态规划问题）有所不同：除了离散的句子输入外，CKY 还会接收连续的对数势。

最后，将模型训练为能够求解优化问题的做法称为训练“推理网络”，这种方法曾广泛用于近似概率推理场景（Kingma 和 Welling，2014年），也用于近似梯度下降（Johnson 等人，2016年）。与我们的方法最为相似的是，Tu 和 Gimpel（2018年）训练推理网络来执行 Viterbi 式序列标注，不过他们并未考虑句法分析；由于采用了大间隔方法，他们在训练和测试时预测时都需要使用推理网络。

7 讨论与结论

我们关于 Transformer 能否近似 CKY 的研究结果显然喜忧参半。一方面，使用 Transformer 独立预测成分句法分析中的跨度，其表现可与非常强大的神经图表分析器相媲美，而且采用这种独立的、非基于 CKY 的方式预测解析结果要快得多。另一方面，如果 Transformer 能够实现 CKY 计算，那么它们理应能够在随机概率上下文无关文法下进行句法分析；然而显然，随着语法的歧义性增加，它们在这方面会遇到困难。

我们还发现，让 Transformer 的计算过程更接近经典算法的计算过程会有所帮助；具体做法可以是共享计算层（如 Dehghani 等人(2018年) 和 Csordás 等人(2021年) 所提出的做法），也可以让其利用相对于评分函数的

梯度。这一发现既证实了该领域先前的研究结果，也表明我们试图融入模型的归纳偏置可能需要与问题紧密匹配。

未来研究有许多可探索的方向，而尝试寻找一种能够在随机概率上下文无关文法下实际进行句法分析的最小通用架构，是一项重要挑战。具体而言，值得探究其他形式的预训练（即不同于 BERT 的预训练）是否能更好地助益于这项任务。另一个需要应对的重要未来挑战是，模型是否能够同时接收输入句子和参数（正如 CKY 算法所做的那样），而不只是使用基于这些参数生成的已解析句子进行预训练。

局限性

学习以算法方式计算的一般范式有一个局限性，即它需要训练阶段，这一阶段在计算上可能代价高昂，并且需要带标注的数据。不过，在成分句法分析的情况下，这一点的限制较小，因为无论如何，我们很可能都会训练模型。

我们工作的另一个重要局限性是，我们仅在特定的实验设置下提供了 Transformer 无法实现 CKY 的证据。尽管我们努力寻找表现最佳的模型与损失函数组合（并且这一组合在成分句法分析方面似乎表现良好），但其他基于 Transformer 的架构或其他损失函数可能会显著提升对随机文法进行句法分析的能力。

我们还注意到，我们提出的梯度解码方法存在一个局限：我们发现，与基线方法相比，该方法对优化超参数更加敏感。

最后，我们注意到，我们表现最佳的成分句法分析结果使用了大型预训练模型。这些模型训练成本高昂，而且我们希望进行句法分析的所有语言未必都有相应的模型。

伦理声明

我们认为，除了在图形处理器上训练中等规模机器学习模型所涉及的问题之外，这项研究不存在重大伦理问题。

参考文献

Jimmy Lei Ba、Jamie Ryan Kiros 和 GeoffreyE Hinton。2016年。层归一化。*arXiv* 预印本 *arXiv:1607.06450*。

James K Baker。1979年。用于语音识别的可训练文法。美国声学学会期刊，65(S1)：S132–S132。

Glenn Carroll 和 Eugene Charniak。1992年。关于从语料库中学习概率依存语法的两项实验。计算机科学系，大学。

Rahma Chaabouni、Roberto Dessì 和 Eugene Kharitonov。2021年。Transformer 能否在自然语言中灵活跳转？评估从 SCAN 迁移的性能。载于第四届用于分析和解读自然语言处理神经网络的 *BlackboxNLP* 研讨会论文集，第 136–148 页，多米尼加共和国蓬塔卡纳。计算语言学协会。Lili Chen、Kevin Lu、Aravind Rajeswaran、Kimin Lee、Aditya Grover、Misha Laskin、Pieter Abbeel、Aravind Srinivas 和 Igor Mordatch。2021年。决策 Transformer：通过序列建模实现强化学习。神经信息处理系统进展，34：15084–15097。Y. J. Chu 和 T. H. Liu。1965年。论有向图的最短树形图。《中国科学》，14。Alexander Clark 和 Nathanaël Fijalkow。2021年。锚定概率上下文无关文法的一致无监督估计器。载于*2021*年计算语言学学会论文集，第 471–473 页，线上。计算语言学协会。Caio Corro。2020年。基于跨度的不连续成分句法分析：一系列精确的、基于分析表的算法，时间复杂度从 O(nˆ6) 降至 O(nˆ3)。载于 *2020* 年自然语言处理实证方法会议论文集（*EMNLP*），第 2753–2764 页，线上。计算语言学协会。James Cross 和 Liang Huang。2016年。采用结构-标签系统和可证明最优动态预言机的基于跨度的成分句法分析。载于*2016*年自然语言处理实证方法会议论文集，第 1–11 页，得克萨斯州奥斯汀。计算语言学协会。Róbert Csordás、Kazuki Irie 和 Jürgen Schmidhuber。2021年。神经数据路由器：Transformer 中的自适应控制流提升了系统泛化能力。载于国际学习表征会议。Tri Dao、Daniel Y. Fu、Stefano Ermon、Atri Rudra 和 Christopher Ré。2022年。FlashAttention：具备输入输出感知能力的快速、节省内存且精确的注意力机制。载于神经信息处理系统进展。

Mostafa Dehghani、Stephan Gouws、Oriol Vinyals、Jakob Uszkoreit 和 Lukasz Kaiser。2018年。通用变换器。载于国际学习表征会议。

Grégoire Delétang、Anian Ruoss、Jordi Grau-Moya、TimGenewein、Li Kevin Wenliang、Elliot Catt、Marcus Hutter、Shane Legg 和 Pedro A Ortega。2022年。神经网络与乔姆斯基层级。*arXiv* 预印本 *arXiv:2207.02098*。

Jacob Devlin、Ming-Wei Chang、Kenton Lee 和 Kristina Toutanova。2019年。BERT：用于语言理解的深度双向变换器预训练。收录于计算语言学协会北美分会*2019*年会议论文集：人类语言技术，第*1*卷（长短篇论文），第4171–4186页，明尼阿波利斯，明尼苏达州。计算语言学协会。

Yann Dubois、Gautier Dagan、Dieuwke Hupkes 和 Elia Bruni。2020年。利用位置注意力外推至更长序列。收录于第*58*届计算语言学协会年会论文集，第403–413页，线上。计算语言学协会。

Andrew Dudzik 和 Petar Veli˘ckovi´c。2022年。图神经网络是动态规划器。载于*ICLR 2022*几何与拓扑表征学习研讨会。

Tiwalayo Eisape、Vineet Gangireddy、Roger Levy 和 Yoon Kim。2022年。探测自回归语言模型中的增量解析状态。载于计算语言学协会论文集：*EMNLP 2022* 研究成果，第 2801–2813 页，阿布扎比，阿拉伯联合酋长国。计算语言学协会。

Jason Eisner。2016年。内外算法和前向-反向算法不过就是反向传播（教程论文）。载于自然语言处理结构化预测研讨会论文集，第 1–17 页，奥斯汀，得克萨斯州。计算语言学协会。

Ramón Fernandez Astudillo、Miguel Ballesteros、Tahira Naseem、Austin Blodgett 和 Radu Florian。2020年。使用堆栈变换器进行基于转移的句法分析。载于计算语言学协会论文集：*EMNLP 2020* 研究成果，第 1001–1007 页，线上。计算语言学协会。

Daniel Fernández-González 和 Carlos Gómez- Rodríguez。2020年。用于更快速序列到序列成分句法分析的增强的顺序线性化。载于计算语言学协会第*58*届年会论文集，第4092–4099页，线上。计算语言学协会。

David Gaddy、Mitchell Stern 和 Dan Klein。2018年。神经成分句法分析器中发生了什么？一项分析。载于*2018*计算语言学协会北美分会：人类语言技术，第 *1* 卷（长篇论文），第 999–1010 页，新奥尔良，路易斯安那州。计算语言学协会。Jonas Geiping 和 Tom Goldstein。2022年。填鸭式训练：一天内在单个图形处理器上训练一个语言模型。Alex Graves、Greg Wayne 和 Ivo Danihelka。2014年。神经图灵机。*arXiv* 预印本 *arXiv:1410.5401*。

Dan Hendrycks 和 Kevin Gimpel。2016年。高斯误差线性单元（GELU）。*arXiv* 预印本 *arXiv:1606.08415*。

John Hewitt 和 Christopher D. Manning。2019年。一种用于在词语表征中寻找句法的结构探针。载于《计算语言学协会北美分会*2019*年会议论文集：人类语言技术》，第*1*卷（长短篇论文），第4129–4138页，明尼阿波利斯，明尼苏达州。计算语言学协会。

Dieuwke Hupkes、Verna Dankers、Mathijs Mul 和 Elia Bruni。2020年。组合性分解：神经网络如何实现泛化？（扩展摘要）。载于 第二十九届国际人工智能联合会议论文集，*IJCAI-20*，第5065–5069页。国际人工智能联合会议组织。期刊赛道。

Borja Ibarz、Vitaly Kurin、George Papamakarios、Kyriacos Nikiforou、Mehdi Bennani、Róbert Csordás、Andrew Dudzik、Matko Bošnjak、Alex Vitvitskyi、Yulia Rubanova 等。2022年。一种通用神经算法学习器。*arXiv*预印本 *arXiv:2209.11142*。

Ganesh Jawahar、Benoît Sagot 和 Djamé Seddah。2019年。BERT 学到了哪些关于语言结构的知识？载于计算语言学协会第*57*届年会论文集，第3651–3657页，意大利佛罗伦萨。计算语言学协会。

Justin Johnson、Alexandre Alahi 和李飞飞。2016年。用于实时风格迁移和超分辨率的感知损失。载于欧洲计算机视觉会议，第694–711页。施普林格。

Lukasz Kaiser 和 Ilya Sutskever。2016年。神经图形处理器学习算法。载于第四届国际学习表征会议，*ICLR 2016*，波多黎各圣胡安，*2016*年*5*月*2*日至*4*日，会议论文集。

Hidetaka Kamigaito、Katsuhiko Hayashi、Tsutomu Hirao、Hiroya Takamura、Manabu Okumura 和 Masaaki Nagata。2017年。 用于序列到序列成分句法分析的监督注意力。载于第八届自然语言处理国际联合会议论文集（第 *2* 卷：短篇论文），第 7–12 页，台湾台北。亚洲自然语言处理联合会。

Tadao Kasami。1966年。一种用于上下文无关语言的高效识别与语法分析算法。协调科学实验室报告第 *R-257* 号。

Diederik P. Kingma 和 Jimmy Ba。2015年。Adam：一种随机优化方法。载于 第三届国际学习表征会议，*ICLR 2015*，圣迭戈，加利福尼亚州，美国，*2015*年*5*月*7*日至*9*日，会议专题论文集。

Diederik P. Kingma 和 Max Welling。2014年。自动编码变分贝叶斯。载于*ICLR*论文集。

Nikita Kitaev、Steven Cao 和 Dan Klein。2019年。基于自注意力和预训练的多语言成分句法分析。载于 计算语言学协会第 *57* 届年会论文集，第 3499–3505页，佛罗伦萨，意大利。计算语言学协会。

Nikita Kitaev 和 Dan Klein。2018年。使用自注意力编码器进行成分句法分析。载于计算语言学协会第*56*届年会论文集（第*1*卷：长篇论文），第2676–2686页，墨尔本，澳大利亚。计算语言学协会。

K. Lari 和 S. J. Young。1990年。使用 inside-outside 算法估计随机上下文无关文法。*ComputerSpeech and Language*，4：35–56。

Bowen Li、Taeuk Kim、Reinald Kim Amplayo 和 Frank Keller。2020年。Heads-up！通过自注意力头进行无监督成分句法分析。载于《亚太计算语言学协会第*1*届会议暨第*10*届国际自然语言处理联合会议论文集》，第409–424页，中国苏州。计算语言学协会。

Bingbin Liu、Jordan T. Ash、Surbhi Goel、Akshay Krishnamurthy 和 CyrilZhang。2023年。Transformer 学会了通往自动机的捷径。

Ilya Loshchilov 和 Frank Hutter。2018年。解耦权重衰减正则化。载于《国际学习表征会议》。

Christopher D. Manning、Kevin Clark、John Hewitt、Urvashi Khandelwal 和 Omer Levy。2020年。自监督训练的人工神经网络中涌现的语言结构。《美国国家科学院院刊》，117：30046–30054。

Mitchell P. Marcus、Mary Ann Marcinkiewicz 和 Beatrice Santorini。1993年。构建大型英语标注语料库：宾夕法尼亚树库。《计算语言学》，19(2)：313–330。

Rowan Hall Maudslay 和 Ryan Cotterell。2021年。句法探针是否真的在探测句法？使用 Jabberwocky 探测进行的实验。载于*2021*年北美分会会议论文集 计算语言学协会：人类语言技术，第 124–131 页，线上。计算语言学协会。

Khalil Mrini、Franck Dernoncourt、Quan Hung Tran、Trung Bui、Walter Chang 和 Ndapa Nakashole。2020年。重新思考自注意力：迈向神经句法分析的可解释性。收录于计算语言学协会：*E MNLP 2020* 论文集，第731–742页，线上。计算语言学协会。

Shikhar Murty、Pratyusha Sharma、Jacob Andreas 和 Christopher D Manning。2022年。利用树投影刻画 Transformer 中的内在组合性。*arXiv* 预印本 *arXiv:2211.01288*。

Thanh-Tung Nguyen、Xuan-Phi Nguyen、Shafiq Joty 和 Xiaoli Li。2021年。一种用于高效成分句法分析的条件拆分框架。载于第*59*届计算语言学协会年会暨第*11*届自然语言处理国际联合会议论文集（第*1*卷：长篇论文），第5795–5807页，线上。计算语言学协会。

Adam Paszke、Sam Gross、Francisco Massa、Adam Lerer、James Bradbury、Gregory Chanan、Trevor Killeen、Zeming Lin、Natalia Gimelshein、Luca Antiga 等。2019年。PyTorch：一种命令式风格、高性能的深度学习库。神经信息处理系统进展，32。

Alexander Rush。2020年。Torch-struct：深度结构化预测库。载于计算语言学协会第*58*届年会论文集：系统演示，第335–342页，线上。计算语言学协会。

David Saxton、Edward Grefenstette、FelixHill 和 Pushmeet Kohli。2019年。分析神经模型的数学推理能力。*ArXiv*，abs/1904.01557。

Djamé Seddah、Reut Tsarfaty、Sandra Kübler、Marie Candito、JinhoD.Choi、RichárdFarkas、JenniferFos- ter、Iakes Goenaga、Koldo Gojenola Galletebeitia、Yoav Goldberg、Spence Green、Nizar Habash、Marco Kuhlmann、Wolfgang Maier、Joakim Nivre、Adam Przepiórkowski、Ryan Roth、Wolfgang Seeker、Yan- nick Versley、Veronika Vincze、Marcin Woli´nski、Alina Wróblewska 和 Eric Villemonte de la Clerg- erie。2013年。SPMRL 2013共享任务概述：对形态丰富语言句法分析的跨框架评估。载于第四届形态丰富语言统计句法分析研讨会论文集，第146–182页，美国华盛顿州西雅图。计算语言学协会。

Mitchell Stern、Jacob Andreas 和 Dan Klein。2017年。基于最小跨度的神经成分句法分析器。载于第*55*届计算语言学协会年会论文集（第*1*卷：长篇论文），第818–827页，加拿大温哥华。计算语言学协会。

Jun Suzuki、Sho Takase、Hidetaka Kamigaito、Makoto Morishita 和 Masaaki Nagata。2018年。构建强大成分句法分析基线的实证研究。载于第*56*届计算语言学协会年会论文集（第*2*卷：短篇论文），第612–618页，澳大利亚墨尔本。计算语言学协会。

Ian Tenney、Dipanjan Das 和 Ellie Pavlick。2019年。BERT 重新发现了经典的自然语言处理流程。载于第*57*届计算语言学协会年会论文集，第4593–4601页，意大利佛罗伦萨。计算语言学协会。

Yuanhe Tian、Yan Song、Fei Xia 和 Tong Zhang。2020年。通过跨度注意力改进成分句法分析。载于 计算语言学协会论文集：*EMNLP 2020*，第1691–1703页，线上。计算语言学协会。

Lifu Tu 和 Kevin Gimpel。2018年。学习用于结构化预测的近似推理网络。发表于*ICLR*。

Ashish Vaswani 和 KenjiSagae。2016年。使用神经网络和错误状态进行基于转移的句法分析的高效结构化推理。计算语言学协会会刊，4:183– 196。

Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N Gomez、Łukasz Kaiser 和 Illia Polosukhin。2017年。注意力就是你所需的一切。神经信息处理系统进展，30。

Petar Veli ˇckovi ´c、Adrià Puigdomènech Badia、David Budden、RazvanPascanu、AndreaBanino、Misha Dashevskiy、Raia Hadsell 和 Charles Blundell。2022年。CLRS 算法推理基准。*arXiv* 预印本 *arXiv:2205.15659*。

David Vilares 和 Carlos Gómez-Rodríguez。2018年。采用更轻量前馈网络的基于转移的句法分析。载于第二届通用依存研讨会论文集（*UDW 2018*），第 162–172 页，比利时布鲁塞尔。计算语言学协会。

Oriol Vinyals、Łukasz Kaiser、Terry Koo、Slav Petrov、Ilya Sutskever 和 Geoffrey Hinton。2015年。《语法作为一门外语》。神经信息处理系统进展，28。

Thomas Wolf、Lysandre Debut、Victor Sanh、J ulien Chaumond、Clement Delangue、Anthony Moi、Pier- ric Cistac、Tim Rault、Remi Louf、Morgan Funtow- icz、Joe Davison、Sam Shleifer、Patrick von Platen、Clara Ma、Yacine Jernite、Julien Plu、Canwen Xu、Teven Le Scao、Sylvain Gugger、Mariama Drame、Quentin Lhoest 和 Alexander Rush。2020年。Transformer：自然语言处理的最先进技术。载于*2020*年自然语言处理实证方法会议论文集：系统演示，第 38–45 页，线上。《计算语言学》协会。

Keyulu Xu、Jingling Li、Mozhi Zhang、Simon S Du、Ken-ichi Kawarabayashi 和 Stefanie Jegelka。2019年。神经网络可以进行哪些推理？载于国际学习表征会议。

NaiwenXue、FeiXia、Fu-dongChiou 和 Marta Palmer。2005年。宾夕法尼亚中文树库：大型语料库的短语结构标注。自然语言工程，11(2)：207–238。

Songlin Yang 和 Kewei Tu。2022年。使用指针网络进行自底向上的成分句法分析和嵌套命名实体识别。载于计算语言学协会第*60*届年会论文集（第*1*卷：长篇论文），第2403–2416页，爱尔兰都柏林。计算语言学协会。

Songlin Yang 和 Kewei Tu。2023年。不要进行句法分析，选择跨度！通过自回归跨度选择进行连续和非连续成分句法分析。载于计算语言学协会第*61*届年会论文集（第*1*卷：长篇论文），第8420–8433页，加拿大多伦多。计算语言学协会。

Daniel H Younger. 1967年. 在时间 n^3 内识别和句法分析上下文无关语言. 信息与控制, 10(2):189–208.

Biao Zhang、Ivan Titov 和 Rico Sennrich。2020年a。快速交错双向序列生成。载于第五届机器翻译会议论文集，第 503–515 页，线上。计算语言学协会。

Yu Zhang、Zhenghua Li 和 Min Zhang。2020年b。用于神经依存句法分析的高效二阶 TreeCRF。载于第*58*届计算语言学协会年会论文集，第3295–3305页，线上。计算语言学协会。

Yue Zhang和Stephen Clark。2009年。使用全局判别模型对中文树库进行基于转移的句法分析。载于第*11*届国际句法分析技术会议（*IWPT'09*）论文集，第162–171页，法国巴黎。计算语言学协会。

Zhisong Zhang、Xuezhe Ma和Eduard Hovy。2019年。基于图的神经网络依存句法分析中结构化输出建模的实证研究。载于计算语言学协会第*57*届年会论文集，第5592–5598页，意大利佛罗伦萨。计算语言学协会。

Haoyu Zhao、Abhishek Panigrahi、Rong Ge 和 Sanjeev Arora。2023年。Transformer 会在预测掩码词的同时进行句法分析吗?

A CKY 的补充背景信息

令 $\ell^R \in \mathbb{R}^{|\mathcal{R}|\times|\mathcal{N}|\times|\mathcal{N}|}$ 表示与概率上下文无关文法 G中的规则相关联的对数势（例如对数概率），令 $\ell^E(x) \in \mathbb{R}^{T\times|\mathcal{N}|}$表示输入句子 x中每个词元对应的对数势，我们计算x的分析表 $\beta \in \mathbb{R}^{T\times T\times|\mathcal{N}|}$，其中 $\beta[i,j,a]$表示生成 $x_{i:j}$的第 a 个非终结符所关联的所有权重在某个特定半环下的求和。具体而言，对于半环运算 $\oplus$和 $\otimes$，且当 $1 \le i < j \le T$时，我们有 $\beta[i,j,a] = \bigoplus_{k,\ b,\ c} \ell^R_{a,\ b,\ c} \otimes \beta[i,k,b] \otimes \beta[k+1,j,c]$。假设 ℓ^R 沿第一维的第一个切片（即 $\ell^R_{1,\ :,\ :}$）对应于左侧为 S 的规则，则对数配分函数为 $A(\ell^R,\ \ell^E) = \beta[1,T,1]$。

此外，在最大加半环下，$A(\ell^R, \ell^E)$ 关于 β 的一个次梯度，是在 x 下、针对 G (Eisner, 2016年；Rush，2020年) 的最高分句法分析的独热表示。我们将这种独热次梯度称为 $\beta^* \in \{0,1\}^{T\times T\times|\mathcal{N}|}$。因此，Rush (2020年) 提议使用自动微分计算 β^*，并提供了针对图形处理器使用进行优化的快速实现。

B 模型、训练和数据集详情

数据集详情 我们在表格中提供了实验所用标准成分句法分析数据集的详细信息 6。

	训练	dev	test
PTB	39,831	1,700	2,416
CTB	17,544	352	348
德语	40,472	5,000	5,000
韩语	23,010	2,066	2,287

表 6：英语宾州树库（Marcus 等人，1993年）、中文宾州树库（Xue 等人，2005）和 SPMRL 树库（Seddah 等人，2013年）标准划分中的样例数量。

使用条款 我们按照相应的许可协议使用了标准英语宾州树库、中文宾州树库，以及 SPMRL 2013 和 2014 共享任务提供的树库（阿拉伯语除外）。PTB 和 CTB 均采用语言数据联盟（LDC）许可协议。德语、希伯来语、韩语和瑞典语树库没有特定许可协议。巴斯克语树库采用知识共享许可协议。波兰语树库采用 GPL v3 许可协议。我们还按照其许可协议使用了 Hugging Face 的代码和模型，（Apache 2.0）。对于深度学习框架，我们使用 PyTorch（Paszke 等人， 2019年），其采用 BSD-3 许可协议。我们还使用了 Kitaev 等人 (2019年)[6]提供的代码。其代码采用 MIT 许可协议。

计算预算 我们的所有实验均在 NVIDIA RTX A6000 GPU 上运行。我们提供了成分句法分析数据集和随机概率上下文无关文法实验的计算时间。由于使用 $|\mathcal{R}| = 100$、400 和800 条规则的随机概率上下文无关文法的训练时间相近，因此我们仅提供使用 $|\mathcal{R}| = 400$ 条规则的概率上下文无关文法的训练时间。

模型详情 在表7 和8中，我们分别给出了成分句法分析和随机概率上下文无关文法实验所使用的模型参数数量。

	模型	Size
PTB	BERT-large-uncased	345M
CTB	BERT-base-chinese	102M
SPMRL	BERT-base-multilingual-cased	178M

表7：成分句法分析实验的模型规模。所有模型均从 Hugging FaceTransformer 库（Wolf 等人，2020年）初始化。

模型	Size
Transformer	89.6M
Transformer +共享层	11.1M
Transformer +共享层 + 复制门	1190万
Transformer +共享层 + 梯度解码	1170万

表8：随机 PCFG 实验的模型规模。

	PTB	CTB	德语	韩语
Ours	1	0.5	4.5	1.5
我们的 +梯度解码	8	4.5	8.5	2.5

表9：我们的模型在句法分析数据集上使用和不使用梯度解码时的近似总训练时间（以小时数计）。我们发现梯度解码模型需要更长时间才能收敛。

C 梯度解码详情

速度比较在表11 中，我们比较了我们的模型（使用和不使用梯度解码）与 CKY 的推理时间。

[6]https://github.com/nikitakit/self-attentive-parser

	$\|\mathcal{R}\| = 400$
Transformer	11
Transformer + 共享层	7
Transformer +共享层 +复制门	6
Transformer +共享层 +梯度解码	15

表10：我们模型在随机PCFG上的总训练时间近似值（以小时数计）。“SL”表示Transformer 层共享，“CG”表示使用复制门，“GD”表示使用梯度解码。

	句子/秒		
批大小	32	64	128
CKY	698	1162	1576
Ours	1817	2132	2161
我们 + 梯度解码	1111	1524	1632

表11：在含有 $|\mathcal{R}| = 400$条规则的随机文法上，基于图形处理器的 torch-struct CKY 解析与我们基于 Transformer 的方法的推理速度。我们报告了模型和 CKY 分别在 60K 个样本上以 32、64 和 128 为批大小运行 5 次的中位速度。

内存比较我们的模型使用梯度解码时，最大内存分配为 1。52 GB，相比不使用梯度解码的模型仅高出 10%；后者的最大内存分配为 1。38 GB。

D SPMRL 上的成分句法分析结果

表12 显示，我们的方法在一半的Ki- taev 等人（2019年）SPMRL 树库上优于他们的方法。由于无法获得相应的许可证，我们排除了阿拉伯语树库。

鉴于梯度解码并未显著改善宾州树库、中文树库、德语和韩语的解析结果，如表1所示，考虑到计算资源有限，我们决定不在其余 SPMRL 树库上进行梯度解码实验。

E 在随机PCFG生成的数据上进行MLM预训练

多项研究表明，预训练模型能够捕捉句法信息（Hewitt 和 Manning，2019年；Manning 等人，2020年；Maudslay 和 Cotterell，2021年；Zhao 等人，2023年）。具体而言，Zhao 等人（2023年）认为，MLM 与内外算法之间存在联系。通过探测，他们表明模型在合成PCFG数据上预训练的模型可能是在近似内外算法。由于内外算法与CKY相关，因此自然会让人思考，MLM预训练是否也有助于近似CKY。

因此，我们使用由包含 800 条规则的随机 PCFG生成的 50 万个句子，对一个 Transformer模型进行预训练。我们按照 Cramming训练方案（Geiping 和 Goldstein，2022年）选择了训练设置。训练时，我们采用了 4096 的大批大小，累积梯度，并每 32 步进行一次更新。我们使用线性预热调度，在总训练步数的 10% 内进行预热，峰值学习率为 1×10^{-4}。模型的训练困惑度为 92.01，验证困惑度为 92.20。

表 13显示，对预训练模型进行微调可以提升性能，尤其是在各层不共享参数时。需要强调的是，尽管预训练有所帮助，但与单纯依赖归纳偏置相比，其计算成本更高，而且相较于梯度解码，带来的提升较小。

F 超参数

表14 展示了我们用于搜索成分句法分析和随机 PCFG 实验中所用模型的最优超参数的网格。我们还在表15 和16中提供了最优超参数。对于表Kitaev 和 Klein，2018年）中的基线结果，我们使用了其代码所推荐的超参数。表1和4 中的结果使用了最优超参数，并且所有实验均使用固定随机种子进行一次运行。

	德语	韩语	巴斯克语	法语	希伯来语	匈牙利语	波兰语	瑞典语
Kitaev 等人（2019年）	**90.20**	88.80	90.70	87.35	**92.95**	94.60	**96.26**	**89.94**
Ours	90.13	**89.05**	**90.93**	**87.59**	92.69	**94.64**	95.86	89.29

表12：F_1 分数在 SPMRL 树库测试集上的比较。Kitaev 等人（2019年）的结果为论文中报告的结果。

	$\lvert\mathcal{R}\rvert = 800$
Transformer	78.14
Transformer + 共享层	78.88
Transformer +共享层 +梯度解码	80.64

表 13：具有 $\lvert\mathcal{N}\rvert = 20$ 个非终结符和 $\lvert\mathcal{R}\rvert = 800$ 条规则的随机生成概率上下文无关文法上预训练模型的 F_1 性能。

成分句法分析实验	
超参数	值
调度器	[默认[a]，线性[b]]
学习率	[5e-5, 6e-5, 7e-5, 8e-5]
梯度裁剪	[0, 0.2, 0.3, 0.4]
权重衰减	[0, 1e-3, 1e-2]
注意力 dropout	[0.1, 0.2]
随机 PCFG 实验	
调度器	常数 + 预热
学习率	[1e-4, 1.3e-4, 1.5e-4, 1.7e-4]
梯度裁剪	[0.3, 0.4, 1]
权重衰减	[0, 1e-3, 1e-2]
注意力 dropout	[0.1, 0.2]
预热步数	[2000, 4000]

表 14：我们用于搜索成分句法分析和随机 PCFG 实验中最优超参数的网格。[a]中使用的默认调度器 Ki- taev 等人（2019年），如第4节所述，[b]160 个预热步骤，然后线性降低学习率。

Ours					
	调度器	LR	梯度裁剪	WD	AD
英语	线性	5e-5	0.3	0	0.1
中文	默认	6e-5	0.4	0	0.1
德语	默认	5e-5	0.3	0	0.1
韩语	默认	5e-5	0	0	0.1
巴斯克语	默认	5e-5	0.3	0	0.1
法语	默认	6e-5	0	0	0.1
希伯来语	默认	5e-5	0.5	0	0.1
匈牙利语	默认	5e-5	0.3	0	0.1
波兰语	默认	5e-5	0.3	0	0.1
瑞典语	默认值	6e-5	0.3	0	0.1
我们的 +梯度解码					
英语	默认值	7e-5	0.2	0	0.1
中文	默认值	8e-5	0.3	0	0.1
德语	默认	8e-5	0.3	0	0.1
韩语	默认	7e-5	0.2	0	0.1
Kitaev 等人 (2019年)					
All	默认	5e-5	0	0.001	0.2

表 15：我们用于成分句法分析实验的最优超参数。我们将学习率记为“LR”，权重衰减记为“WD”，注意力 dropout 记为“AD”。基线模型也使用了类似的超参数（Kitaev 等人，2019年），适用于各个数据集。结果见表1。

Transformer					
	LR	梯度裁剪	WD	AD	WS
$\lvert\mathcal{R}\rvert = 100$	1e-4	1	1e-3	0.3	4000
$\lvert\mathcal{R}\rvert = 400$	1e-4	1	1e-3	0.3	4000
$\lvert\mathcal{R}\rvert = 800$	1e-4	1	1e-3	0.3	4000
Transformer +共享层					
$\lvert\mathcal{R}\rvert = 100$	1e-4	1	1e-3	0.3	4000
$\lvert\mathcal{R}\rvert = 400$	1.3e-4	1	1e-3	0.3	4000
$\lvert\mathcal{R}\rvert = 800$	1.3e-4	1	1e-3	0.3	4000
Transformer +共享层 + 复制门					
$\lvert\mathcal{R}\rvert = 100$	1.3e-4	1	1e-3	0.3	4000
$\lvert\mathcal{R}\rvert = 400$	1e-4	1	1e-3	0.3	4000
$\lvert\mathcal{R}\rvert = 800$	1e-4	1	1e-3	0.3	4000
Transformer +共享层 + 梯度解码					
$\lvert\mathcal{R}\rvert = 100$	1e-4	0.4	1e-3	0.3	4000
$\lvert\mathcal{R}\rvert = 400$	1.3e-4	0.4	1e-3	0.3	4000
$\lvert\mathcal{R}\rvert = 800$	1e-4	0.4	1e-3	0.3	4000

表16：随机 PCFG 实验所使用的最优超参数。我们将学习率记为“LR”，权重衰减记为“WD”，注意力 dropout 记为“AD”，预热步数记为“WS”。结果见表4。