SPINDLE：借助图注意力将原始文本转化为 Lambda 项

Konstantinos Kogkalidis♢ 和**Michael Moortgat**♢ 和**Richard Moot** □
♢乌得勒支大学语言科学研究所□ 蒙彼利埃大学 LIRMM、法国国家科学研究中心 {k.kogkalidis,m.j.moortgat}@uu.nl, richard.moot@lirmm.fr

arXiv:2302.12050v1 [cs.CL] 23 Feb 2023

摘要

本文介绍了 SPINDLE[1] ——一个开源 Python 模块，用于实现高效、准确的荷兰语书面文本解析器，可将原始文本输入转换为用于意义组合的程序，表达为 λ项。该解析器整合了近年来取得的多项突破性进展。其输出包含多模态类型逻辑语法的高分辨率推导，刻画句法的两个正交维度，即深层论元结构和依存关系。这些推导由三个相互依赖的系统生成：用于断言语法分析适定性的静态类型检查器、基于异构图卷积且具备结构感知能力的先进超标签器，以及基于 Sinkhorn 迭代的大规模并行证明搜索组件。软件还附带了实用工具和扩展功能，用于证明可视化和推理，旨在方便终端用户使用。

1 引言

λ 演算的透明性和良好的形式性质，使其成为表达组合结构的理想形式；这一点已得到那些主要关注语义学的解析器和工具充分强调。λ 演算是类型逻辑语法的关键组成部分，在其中，它们被用作所谓的语法逻辑的计算对应物。这是一种直觉主义线性逻辑的子结构逻辑，旨在捕捉自然语言句法和语义学的（某些方面）（Moortgat，1997）。对于类型逻辑语法，Curry-Howard 同构保证了逻辑规则、类型构造子和项构造算子之间可以直接对应；简言之，解析 ≡ 证明 ≡ 程序，范畴 ≡ 命题 ≡ 类型。其运作方式很直接：词典将词语与逻辑公式关联起来，而逻辑的推理规则则决定公式如何彼此作用。

进一步说，词语只能以严格且类型正确的方式组合，并在此过程中形成更大的短语。句法分析成为一种逻辑推导过程，最终得出的结果（一个证明）会生成一份构造意义的配方（一个程序）。只要为词汇常量（词语）和项操作（组合指令）指定适当的解释，该程序便会转化为可执行代码。此框架具有通用性，能够轻松适配不同的解释选择；例如，有效的目标可以是（真值条件）形式语义学、分布式组合模型（Sadrzadeh 和 Muskens，2018年）或基于表列法的定理证明器（Abzian-idze，2017年）。

在这项工作中，我们关注语义执行之前发生的事情；也就是说，我们抽象掉词汇语义学，旨在揭示自然语言话语背后的组合规则。为此，我们采用一种类型语法，旨在捕捉两种截然不同的句法轴向特征，而这两者在实际语言中很少同时出现：函数—论元结构和依存关系。为了从输入短语中获得推导，我们设计并实现了一个由三个彼此独立但相互通信的组件组成的系统。第一个组件是语法类型系统的实现——它包含若干实用功能，其中最重要的是静态类型检查器，用于验证所构建分析的句法良构性。第二个组件是超标签器，负责为每个输入词分配一个类型——该标注器基于一种超高效的异构图卷积核构建，在范畴语法数据集中取得了最先进的准确率。第三个也是最后一个组件是神经置换模块，它

[1]风格化的纺锤[2]λ，代表 **spindle** 对短语进行**p**句法分析**并**转化**为**依存关系**修饰**的 λ **表达式**。源代码和用户说明可在以下地址找到：https://github.com/konstantinosKokos/spindle。

利用目标逻辑的线性约束，将证明搜索简化为最优传输学习（Peyré 等人，2019）——这种重新表述实现了大规模并行且易于优化的实现，不受传统解析器中常见的结构操作中断所阻碍。三个组件在处理流水线中交替发挥作用，在低层线性代数例程和高层逻辑推理阶段之间切换（分别主要依赖图形处理器和中央处理器）。它们的集成打造出一个速度极快且准确率很高的神经符号解析器，并以精良的封装公开发布。

2 系统分解

2.1 类型语法

该系统的理论支柱是其类型逻辑——一种独具特色的语义优先的类型逻辑文法，在两个主要方面偏离了范畴语法规范。首先，它关注深层句法结构（即深层语法，用Curry的术语来说），而非表层形式；因此，其函数类型不受方向或位置约束的影响，只遵循线性条件：每个原子命题都必须使用一次，且恰好一次。其次，它对函数类型进行修饰，使其能够编码语法功能，并将其分为补语、中心语和附加语三类。

完整阐述该语法超出了本文的范围，但简要而简化的概述应有助于阐明下文内容。其第一个方面——**函数-论元结构**——使用线性逻辑的蕴含箭头 $\multimap$ 建模，使我们能够处理注重资源的函数应用和变量抽象版本（Girard，1987；Abramsky，1993）。在其语言学用途中，形如 $A \multimap B$ 的函数类型表示这样的谓词：它们消耗一个 A 类型对象的单次出现，其结果是一个 B 类型的复合短语。对空位、省略等现象的推理借助高阶类型完成，即前述模式的实例，其中 A 本身是一个函数——这些高阶类型启动了一个**假设推理**过程：我们可以暂时假设某个资源存在，以便在局部产生一个推导，随后撤回该假设，并在此过程中创建一个新函数。第二个方面——**依存关系**——使用一组带标注的余剩对（一元算子）建模

$$\frac{}{x : A \vdash x : A}\ \mathrm{id} \qquad \frac{(c \mapsto A) \in \mathcal{L}}{c : A \vdash c : A}\ \mathrm{lex}$$

$$\frac{\Gamma \vdash s : A \multimap B \quad \Delta \vdash t : A}{\Gamma, \Delta \vdash s\ t : B} \multimap E \qquad \frac{\Gamma, x : A \vdash s : B}{\Gamma \vdash \lambda x.s : A \multimap B} \multimap I$$

$$\frac{\Gamma \vdash s : \square_\delta A}{\langle \Gamma \rangle^\delta \vdash \blacktriangledown_\delta s : A} \square_\delta E \qquad \frac{\Gamma \vdash s : A}{\langle \Gamma \rangle^\delta \vdash \triangle_\delta s : \Diamond_\delta A} \Diamond_\delta I$$

图 1：类型语法所使用的逻辑推理规则（子集）。恒等规则将一个任意类型 A 的新变量实例化。词汇规则根据从词典 $\mathcal{L}$ （或者在后神经时代，从超标签器）中提取的类型赋值 $c \mapsto A$，将常量 c 声明为 A 类型。引入规则是复合类型构造器，消去规则是析构器。$\multimap E$ 规则说明：从前提 Γ 推导出的 $A \multimap B$ 类型的项 s，可以应用于从前提 Δ 推导出的 A 类型的项 t，从而产生一个 B 类型的复合项 s t；该复合项由 Γ 与 Δ 的合并推导得到。 $\multimap I$ 规则说明：如果某个 B 类型的项 s 的前提中包含一个 A 类型的变量 x，我们可以对后者进行抽象，产生一个 $A \multimap B$ 类型的项 $\lambda x.s$。$\square_\delta E$规则移除一个 $\square_\delta A$ 类型项 s 外部的方框，产生 A 类型的项$\blacktriangledown_\delta s$，并将前提置于括号 $\langle _ \rangle^\delta$ 内。相应地，$\Diamond_\delta I$ 规则将一个 A 类型的项 s 置于菱形算子的作用域之下，产生一个 $\Diamond_\delta A$ 类型的项 $\triangle_\delta s$，并再次将前提置于括号 $\langle _ \rangle^\delta$ 内。

源自时序逻辑。没有任何依存标记的原子类型被赋予语言学上独立的单位和短语，例如 NP 表示名词短语。表示中心语的函数类型会在其所选择的补语上施加菱形算子 $\Diamond_c$，并标记为 c，表示补语所占据的依存槽位，例如 $\Diamond_{su}NP \multimap S_{main}$ 表示一个不及物动词，它需要一个主语标记名词短语来构成主句。与此相对，表示附加语的函数类型本身带有一个方框 $\square_a$，标记为 a，此时 a 表示附加语在应用之前所投射的依存角色，例如 $\square_{mod}(NP \multimap NP)$ 表示形容词；如果去掉它的方框，它便承诺提供一个作用于名词短语的函数。引入菱形算子或消去方框都会留下一个结构印记，以括号包围完整短语；还会留下一个计算印记，要求对被包裹的项进行特殊处理——两者均以所引入的菱形算子（或所消去的方框）的语法功能标记。类型语法的关键逻辑规则及其同构的项操作见图1。

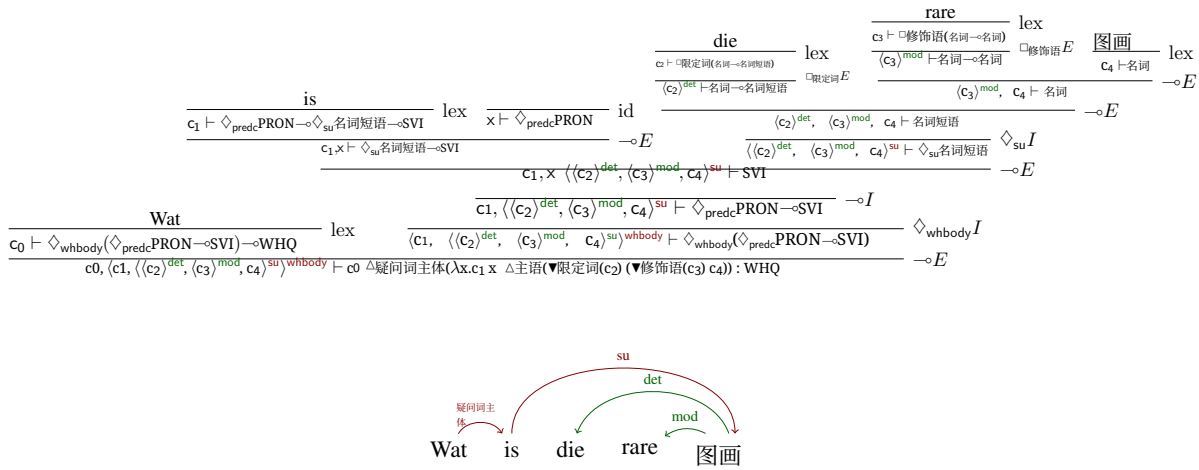

图2：句子*Wat is die rare tekening?* “那幅奇怪的图画是什么？”的自然演绎证明。为节省空间，组合项仅在终结序列（证明底部）中明确写出。根据终结序列的前件结构，我们也可以恢复出一棵依存树。颜色编码用于以非正式方式区分补语（红色）与附加语（绿色）结构括号/依存弧。

2.1.1 证明表示

类型逻辑中的证明传统上采用树状的**自然演绎**格式。自然演绎中的证明易于转换为 (i) λ 表达式，只需遵循图 1中的规则；也易于转换为 (ii) 依存树，只需将结构括号转换为依存弧，从每个短语的中心语指向其依存项（的中心语）。图 2 展示了一个直观示例。另一种表示方式是简洁得多的**证明网**格式。这是一种几何构造，抽象掉了假设推理和树状规则排序中繁琐的簿记工作。图 3展示了当前示例对应的证明网。证明网更适用于神经网络设置中的推理，因为它允许我们将句法分析视为一个大幅简化的问题：将每个出现在负位置（即条件蕴涵的前提）中的原子命题，与出现在正位置（即条件性地已证的陈述）中的原子命题进行匹配。证明网的并行特性使匹配能够在整个证明中同时进行；也就是说，所有决策都在同一时刻完成，无需等待条件以自底向上的方式得到满足。另一方面，与自然演绎证明相比，证明网的规定略显不足，因为它只明确表示函数-论元结构——从一种格式转换到另一种格式，需要先就何为规范证明确立一些约定。

2.1.2 实现

类型系统的句法以 Python 编写的微型领域专用语言实现。[2] 它被用作 Æthel（Kogkalidis 等人，2020a）的表示格式。Æthel 是一个包含约 70000 条书面荷兰语分析的语料库，同时也构成了该系统的训练数据。该实现最初旨在确保语料库的类型安全性，促进自然演绎树、 λ 项和证明网之间的转换，并通过提供搜索、美化打印工具、用于可视化的 LaTeX 交叉编译、证明变换接口等便利功能，简化第三方语料库分析。所有这些功能均原样导入。转换例程使我们能够在证明网这一有利范式下进行神经证明搜索，并仅在最后一步将结果转换为自然演绎格式，以便展示和/或进行健全性测试。重要的是，类型检查器被重新用作验证所构造分析正确性的工具——不构成有效证明的分析将无法通过检查器，并会抛出类型错误，提醒我们这一事实。换言之，我们可以盲目信任解析器给出的任何内容是正确的，至少就（证明论意义上的）句法有效性而言。

[2]源代码可在以下网址找到：https://github.com/konstantinosKokos/aethel。

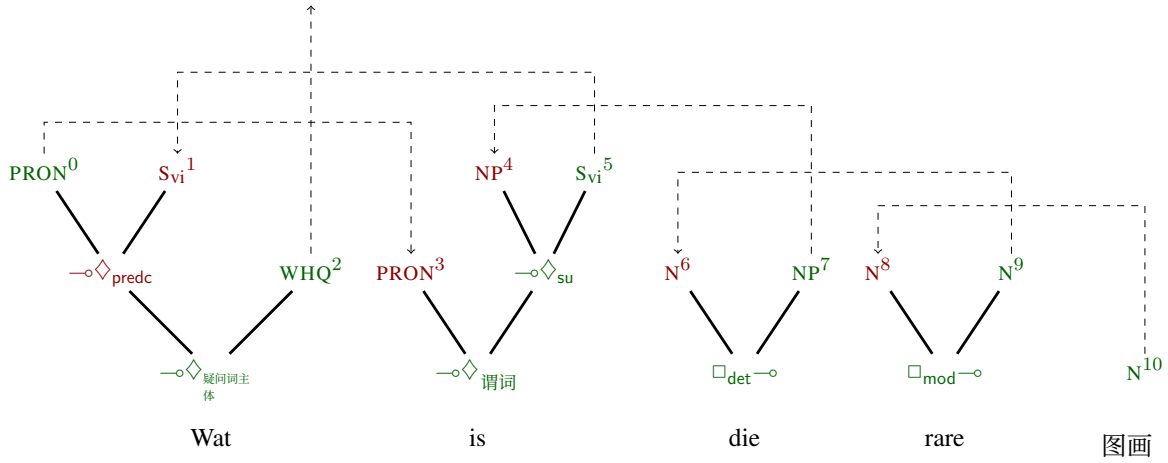

图 3：图 2 中证明的等价证明网。为压缩深度，将一元菱形算子（或方框）与支配它（或受它支配）的蕴含融合。原子命题以枚举编号，便于识别。此处使用颜色区分已有资源（绿色）和所需资源（红色）——规则是从底部开始标为绿色，蕴含的左侧子节点改变颜色，右侧子节点则保持颜色。粗边表示类型赋值所依据的树结构。虚线表示极性相反的资源之间的正确匹配。

2.2 超标签标注模块

词汇类型歧义和词汇类型稀疏性是任何范畴语法中普遍存在的常见问题。事实上的解决方案依赖于超标签器这一神经模块，它取代了固定词典，传统上被构造为序列分类器，并经过训练，为输入句子上下文中的每个词生成最合理的类型分配。在这里，高度精细的类型系统进一步加剧了这些问题。Æthel 约 6 000 个类型中，有约 80% 是奇怪的（即在语料库中的出现次数少于 10 次），而全部句子中约有 10% 至少包含一个此类奇怪的类型。因此，相较于标准的“设置后弃置”方法——将奇怪的类型分配完全视为无关紧要并丢弃——这里需要采取更为积极的处理方式。解决方案是一种**构造式超标签标注器**，即一种自回归神经解码器。它经过训练，能够根据类型的代数分解动态构造类型，而不是将类型视为单一且不透明的整体（Kogkalidis 等人，2019）。这种配置使系统无论此前是否见过某个类型，都能够构造有效类型，从而将覆盖范围扩展到训练数据之外。这里使用的超标签器遵循一种具有几何学依据且面向特定任务的解码顺序，其中类型被表示为二叉树的结构展开。继Prange 等人（2021）之后，树在整个输入序列批次中并行解码，从而建立一个时间上的上限

解码的上限随最大树深度变化——实际应用中为常量。为克服标准树解码器的局部性问题，目标输出不是一批树，而是一批**树序列**（见图 3），因此超标签器被构建为图神经网络，利用消息传递连接，将反馈从树节点传递到其词汇根节点，再从词汇根节点传递到其邻居，从而确保每个解码步骤所做的决策都会受到整个输出中先前决策的影响（Kogka- lidis 和 Moortgat，2022）。因此，它在树形架构的速度和内存效率（允许进行更多训练迭代并加快推理）与序列到序列模型更强的自回归特性（提升性能）之间取得了完美平衡。此外，由于其输出天然受限于树结构，因此在结构上必然正确——在任何情况下，生成的类型都不可能格式错误。单独使用时，该架构目前在涵盖不同形式体系和语言的范畴语法超标签标注准确率排行榜上名列前茅；在此采用的多任务训练设置中，其性能略逊一筹。

2.3 置换模块

通常不宜对证明网进行搜索。传统上，这个问题需要检查正、负原子命题之间所有可能的双射。双射的数量

会随着原子命题数量的增加而按阶乘级增长，很快变得难以处理。为了应对这种组合爆炸的局面，**神经证明网**将证明搜索放宽为连续、可微的问题，在这个问题中，寻找正确的双射被转换为一个传输问题（Kogkalidis 等人，2020b），并由另一个图神经网络学习。具体而言，从解码器中提取所有原子命题出现位置的表示，并根据其句子索引、符号和极性进行分组（例如，一个分组可以包含输入批次中第 #13 个句子里所有正向名词短语的出现位置）。每个分组都与其极性相反的对应分组进行比较，使用某种相似度度量（此处为加权内积）。由此得到一组方阵，每个矩阵都包含注意力权重（或相似度分数），对应于相同符号和句子中正、负项目的笛卡尔积。这些矩阵按基数分组，并使用 Sinkhorn 算子（Sinkhorn 和 Knopp，1967年）使其趋近于二值化和双随机性，从而得到置换矩阵的近似值，以表示目标双射（Mena 等人，2018年）。

以图 3 的贯穿示例为例，具体来说，原子类型 SVI、NP 和 PRON 各有一个负向出现和一个正向出现，因此它们的双射是平凡的（这证明了超标签标注几乎等同于句法分析）。WHQ 的唯一正向出现代表短语的目标类型，并保持未匹配状态。只有 N 需要做出选择，因为它有两种可能的双射。正确的候选项由下方的置换表编码，其中行列出正向项，列列出负向项：

$$\Pi_{\mathrm{N}} := \begin{array}{c|cc} & 6 & 8 \\ 9 & \checkmark & \\ 10 & & \checkmark \end{array}$$

这种重新表述带来了巨大的提速：原本极其缓慢、繁琐的符号证明搜索问题，被转化为简单、经过充分优化且可批量处理的矩阵运算。当前解析器基于这样一个洞见：置换模块不受原子符号表示来源的影响；事实上，它还能从速度更快、准确度更高的任务适配的超级标注器中显著获益。

2.4 集成

神经符号融合构成了一个端到端流程，包含以下几个阶段。首先，用户输入一系列待解析的句子。经过微调的 $\mathrm{BERT}_{\mathrm{BASE}}$ 模型会生成上下文化的词元表示，随后根据输入文本的词边界对这些表示进行聚合。由此得到的词表示作为初始种子，在空白画布上启动解码。在解码过程中，类型逐步构建，同时种子通过彼此交换消息进行更新，消息交换依据的是它们在序列中的距离。经过少量解码步骤后，处理过程终止，为每个输入句子生成一系列类型分配。随后执行一项初步的不变性检查，确认每个序列中两种极性的原子命题数量是否相等。未通过不变性检查的句子不具备进行证明搜索的资格，其分析会提前终止。通过检查的句子将经过符号处理，生成一组稀疏索引张量，用于将解码器的表示收集到前文所述的分箱中。各分箱被批量处理并进行对比，随后执行少量 Sinkhorn 迭代，作为二维 softmax 的类比操作。使用匈牙利算法将 Sinkhorn 软距离离散化，以确保双射性（Jonker 和 Volgenant，1987）。双射通过前一步索引操作的逆操作，重新关联到其来源符号和句子。随后，控制权交给符号组件，由其尝试遍历候选证明网，并在此过程中验证无环性和连通性这两项正确性标准（Danos 和 Regnier，1989）。遍历过程与转换为自然演绎格式相对应，而这种格式的构建相当于对输出进行静态类型检查（Lamarche，2008）。假如没有发现类型不匹配，输出便是一个真正的证明；根据 Curry-Howard 同构，它会被改写为带有依存关系标注的 λ 项。最后，用户会看到每个输入句子的分析结果——理想情况下是一个 λ 项，但有时也可能是一个被拒绝的中间结果及其错误说明。

3 评估

3.1 性能

该系统已在 Æthel 的测试集上进行了评估。在没有任何预过滤或后处理的情况下

训练轮（即不对句子长度、类型稀有度/深度或双射的基数施加任何约束）下，解析器在 5 770 个测试集样本中的 3191 个样本上生成了与真实值满足严格句法相等的证明。这意味着，在所生成的类型分配、短语分块、函数-论元结构和依存标注方面，有 55.30% 的测试句子实现了零错误解析[3]总计有 4 901 个句子被赋予了通过分析，使覆盖率达到较为适中的 84.94%。高准确率与低覆盖率之间的差异源于类型系统的刚性：只有 5 010 个句子满足不变性检查，因此能够接受任意证明。这表明性能瓶颈在于超标签器，而非置换模块；97.82 % 的可解析句子被赋予了解析结果，其中 75.30% 的情况下该解析还是完美解析。这些发现总结于表 1。

可解析性 （可获得某个证明）	**覆盖率** （已获得某个证明）
86.83	84.94
类型正确 （可获得正确的证明）	**准确率** （已获得正确的证明）
56.88	55.30

表1：解析器的句子级评估。

为了更深入地了解性能，我们采用了解析领域青睐的 F_1-分数的一种变体。具体而言，我们收集所有成功生成证明的样本，并将预测结果和真实值分别分解为各自的子证明集合。我们将tp 衡量为两个集合的交集，将 fp 衡量为预测子证明与正确子证明之差，将fn 衡量为正确子证明与预测子证明之差，据此可得精确率$p = {tp}/{(tp + fp)}$、召回率$r = {tp}/{(tp + fn)}$ 及其调和平均数$F_1 = {2pr}/{(p + r)}$。除了这些指标的基本版本，我们还可以结合两个模因素来考察松弛形式。第一种松弛针对逻辑的函数核心，采用遗忘变换去除证明中的模态，以便单独考察带类型的函数-论元结构。第二种松弛针对逻辑的模态增强，将原子类型集合压缩为一个点（因此将形状相同的所有函数类型视为相等），以便单独考察依存结构。沿两个维度同时进行松弛，本质上是将证明转换为无类型的 λ 演算；在这种情况下，我们只关注不依赖类型和依存关系的函数-论元结构——这是与外部理论最具可比性的指标。[4] 请注意，松弛仅在推理之后进行——其目的是确保必须先生成严格证明，才会将其纳入松弛形式的考量（即宽松准确率仍受严格覆盖率的限制）。结果按已覆盖样本取平均值[5]，并列于表2中。

	局部指标		
模	p	r	F_1
–	89.52	89.68	89.39
模态	90.93	91.13	90.85
函数类型	91.09	91.26	90.97
both	92.31	92.52	92.24

表2：分解指标与松弛。

3.2 效率

关于效率，该架构包含总计 1.17 亿个不可忽视的参数，其中 94% 继承自底层 BERT 模型。在测试集上以 64 为批量大小时，网络前向传播的内存占用不超过 3.5GB，因此在家庭环境中使用时相当轻量。使用中端笔记本电脑 GPU，网络为整个测试集添加标注约需 15 秒（即每秒 370 个句子或 6,000 个词元）；添加标注并解析则需 80 秒（即每秒 70 个句子），其中包括分词和后处理。不过请注意，推理模式下的证明搜索尚未实现句子级批处理，即句子会并行添加标注，但证明过程是顺序进行的。目前正在开发跨句填充以及 Sinkhorn 输入的批处理——基准测试表明，对 64 个矩阵组成的批次执行前向传播时，只有当矩阵数量超过

[3]这与类似（事实上更简单）的 CCG解析问题的当前最先进水平相当；Clark (2021) 提供了最新的综述。

[4]证明均为 β 和 η 范式，因此抽象不会带来额外得分。只有名称和类型都匹配时，变量才相同，因此变量实例化也不会带来额外得分。[5]对整个测试集取平均会人为抬高 p 并压低 r 得分，因为失败样本不会返回部分证明。

```
>>>frominference import InferenceWrapper as IW>>>fromaethel.utils.tex import compile_tex, sample_to_tex >>> 解析器 =IW(weight_path='./data/model_weights.pt') >>> 分析 =解析器.analyze(['什么是那奇怪的图画？ ']）[0]>>> 分析
Analysis( 词汇_短语=( LexicalPhrase(string=什么, type=(◇疑问词主体(◇predc(VNW)→SV1))→WHQ, len=1),
LexicalPhrase(string=是, type=◇predc(VNW)→◇主语(名词短语)→SV1, len=1), LexicalPhrase(string=那, type=□限定词(名词
→名词短语), len=1), LexicalPhrase(string=奇怪的, type=□修饰语(名词→名词),len=1), LexicalPhrase(string=图画, type=名词,
len=1), LexicalPhrase(string=?, type=PUNCT, len=1)), 证明=c0, ⟨c1, ⟨⟨c2⟩限定词, ⟨c3⟩修饰语, c4⟩主语⟩疑问词主体 ⊢ c0 △疑
问词主体((λx0.c1 x0 △主语(▼限定词(c2) (▼修饰语(c3) c4)))) : WHQ)>>> tex_证明 =compile_tex(sample_to_tex(分析)) # 请参
见图2查看输出 :)
```

图4：Python 控制台中的用户交互示例。

顺序 2^7，被锁定在此前仅 1 毫秒处，这个时间差微不足道。[6]

4 用户界面

用户界面虽然功能精简，但简单易用。解析器目前以代码仓库的形式提供，下载后可作为 Python 模块使用。前端类封装了解析器内部复杂的运作机制，并提供了便捷的推理程序调用方式。结构检查由内部处理，错误处理也十分妥善：即使发生部分故障，也能保证用户获得输出。输出遵循与 Æthel 语料库样本相同的协议，因此兼容后者的所有附加功能。证明可以美观地排版、交互式处理和转换（例如用于语义应用），也可以借助 LaTeX 进行可视化。对于更有抱负的用户，我们还提供训练和评估工具。

5 结论与未来工作

至此，spind$^2\lambda$e 演示之旅告一段落：它是一款独特的神经符号解析器，能够准确且高效地将原始文本转换为 λ 表达式。与更廉价的替代方案不同，这些 λ 表达式并非通过任意解码生成的毫无结构、临时拼凑的仿制品，而是可执行、类型安全且百分之百**保证正确的**程序。该软件专注于荷兰语，但直觉线性核心具有普适性，因此很容易进行跨语言适配；适配工作基本上只需使用新的类型词典重新训练。目前，法语版本正在开发中（De Pourtales 等人，2023年）。

[6]要理解 2^7 有多么不切实际，不妨想想，这相当于要从 $2^7! = 3.9 \times 10^{215}$ 种可能性中，找出正确的双射，且要同时处理 64 对集合。

展望未来，计划是让该模块与 Æthel 保持同步并及时更新：后者今后每次发布重大版本，都会通过更新前者来体现（无论是采用软补丁还是重新训练）。除兼容性之外，计划中的功能还包括将该模块部署为网络服务、编译为独立软件包，以及编写注释文档（以更好地照顾不熟悉类型系统者）。相关研究取得的任何性能、稳定性或效率改进，以及工程灵感迸发的成果，也很可能会应用到面向用户的版本中。欢迎随时贡献和反馈。

局限性

本文所述的实现利用神经运算与符号运算之间的解耦来提高效率。但这样做的代价是采用缺乏反馈的单向数据流。符号组件唯一的作用是测试和验证神经网络输出，却不会发出任何自身的消息。错误或许能够被发现，但仍然无法挽回——不符合某项结构约束的部分输出，意味着处理流水线会突然且无可协商地终止，从而大幅降低覆盖率。更好的实现方式是利用符号核心持续请求神经网络输出，直到满足结构约束（或用户对所提供的解析结果感到满意）为止。然而，只有在为神经组件加入某种回溯机制后，这种方式才可行。从这个意义上说，超标签器和解析器的并行特性如今成了一把双刃剑，妨碍了束搜索等标准启发式算法的潜在应用。

更广泛地说，该软件存在任何

依赖机器学习的自然语言处理架构都会面临的标准风险，即继承自所集成语言模型无监督预训练的语言偏差，以及源自基于人工标注数据进行监督训练的标注偏差。

致谢

本文所述软件由荷兰研究理事会（NWO，资助编号 360-89-070）资助开发。

参考文献

Samson Abramsky。1993。线性逻辑的计算解释。理论计算机科学，111(1-2)：3–57。Lasha Abzianidze。2017。LangPro：自然语言定理证明器。载于*2017*年自然语言处理实证方法会议论文集：系统演示，第115–120页，丹麦哥本哈根。计算语言学协会。Stephen Clark。2021。新旧兼具：使用Transformer模型进行基于语法的CCG解析。*CoRR*，abs/2109.10044。Haskell B Curry。1961。语法结构的若干逻辑方面。语言结构及其数学方面，12：56–68。Vincent Danos和Laurent Regnier。1989。乘法联结词的结构。数理逻辑档案，28(3)：181–203。Caroline De Pourtales、Julien Rabault、Konstantinos Kogkalidis和Richard Moot。2023。DeepGrail：面向法语的神经证明网。技术报告，LIRMM。即将发表。Matthias Fey和Jan Eric Lenssen。2019。使用pytorch geometric进行快速图表示学习。*arXiv*预印本 *arXiv:1903.02428*。Jean-Yves Girard。1987。线性逻辑。理论计算机科学，50(1)：1–101。Roy Jonker和Anton Volgenant。1987。用于稠密和稀疏线性指派问题的最短增广路径算法。计算，38(4)：325–340。Konstantinos Kogkalidis和Michael Moortgat。2022。使用异构动态卷积的几何感知超标签标注。Konstantinos Kogkalidis、Michael Moortgat和Tejaswini Deoskar。2019。使用自注意力网络进行构造性类型逻辑超标签标注。载于第*4*届自然语言处理表示学习研讨会论文集（*RepL4NLP-2019*），第113–123页，意大利佛罗伦萨。计算语言学协会。Konstantinos Kogkalidis、Michael Moortgat和Richard Moot。2020a。ÆTHEL：自动提取的荷兰语类型逻辑推导。载于第*12*届语言资源与评测会议论文集，第5257–5266页，法国马赛。欧洲语言资源协会。Konstantinos Kogkalidis、Michael Moortgat和Richard Moot。2020b。神经证明网。载于第*24*届计算自然语言学习会议论文集，第26–40页，线上。计算语言学协会。François Lamarche。2008。直觉主义线性逻辑的证明网：本质网。Gonzalo Mena、David Belanger、Scott Linderman和Jasper Snoek。2018。使用Gumbel-Sinkhorn网络学习潜在排列。载于国际学习表征会议。Michael Moortgat。1997。范畴类型逻辑。载于逻辑与语言手册，第93–177页。Elsevier。Gabriel Peyré、Marco Cuturi等。2019。计算最优传输：及其在数据科学中的应用。机器学习基础与趋势®，11(5-6)：355–607。Jakob Prange、Nathan Schneider和Vivek Srikumar。2021。通过复杂范畴的树结构解码为长尾进行超标签标注。计算语言学协会会刊，9：243–260。Mehrnoosh Sadrzadeh和Reinhard Muskens。2018。自然语言λ演算模型的静态与动态向量语义。语言模型期刊，6(2)：319–351。Richard Sinkhorn和Paul Knopp。1967。关于非负矩阵和双随机矩阵。太平洋数学杂志，21(2)：343–348。