arXiv:2302.03145v1 [cs.CL] 6 Feb 2023

提升神经数学应用题求解器的技术

Youyuan Zhang学生，计算机科学系，麦吉尔大学，
youyuan.zhang@mail.mcgill.ca

摘要

开发自动数学应用题（MWP）求解器是一项艰巨的任务，需要具备理解自然语言并进行数学推理的能力。近期基于神经网络的方法主要使用语言模型对题目文本进行编码，并迭代地解码出由数量和运算符构成的数学表达式。值得注意的是，数学应用题的题目文本由背景部分和问题部分组成。近期一项研究发现，这些神经求解器可能只是在背景文本与标准表达式之间进行浅层模式匹配，而未能充分利用问题文本。与此同时，现有的解码过程未能将数学定律纳入设计，导致数学等价表达式的表示各不相同。为了解决这两个问题，我们提出了一种新型的编码器-解码器架构，能够充分利用问题文本并在逐步计算中保持交换律。除了生成数量嵌入外，我们的编码器还会对问题文本进行编码，并利用它来引导解码过程。在每个步骤中，我们的解码器使用 Deep Sets 计算表达式表示，使这些嵌入在数量的任意排列下保持不变。在四个成熟基准数据集上的实验表明，我们的框架优于最先进的神经数学文字题求解器，证明了所提技术的有效性。我们还对结果进行了详细分析，以展示该方法的局限性，并进一步讨论未来工作的潜在方向。[1]。

1 引言

数学应用题（MWP）求解是回答以自然语言描述的数学问题的任务（Bobrow，1964年）。图 1 展示了一个数学应用题示例。输入由上下文文本和问题文本组成，其中上下文

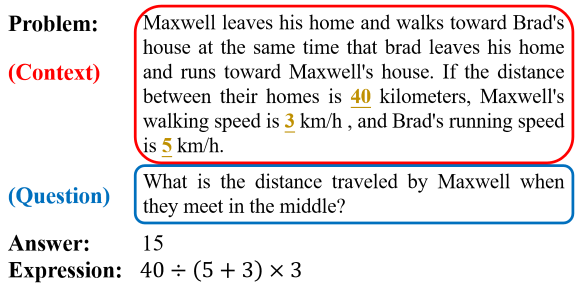

图 1：一个数学应用题示例。

部分描述了包含若干数量的多个叙述，而问题部分则定义了目标。输出是作为答案的数值。要获得这样的答案，需要推断出一个数学表达式，用于规定对问题中各数量执行的运算符。要解决数学应用题，机器不仅需要理解问题的描述和查询，还需要对文本进行数学推理。

近年来，随着深度学习模型在自然语言处理领域的应用（Chung 等人，2014；Vaswani 等人，2017；Devlin 等人，2019），越来越多基于神经网络的方法被提出，用于有效求解数学文字题（Faldu 等人，2021；Lan 等人，2022；Sundaram 等人，2022）。受机器翻译研究的启发（Sutskever 等人，2014），一系列研究（Wang 等人，2017；Amini 等人，2019）将这项任务建模为从自然语言到数学表达式的序列到序列（Seq2Seq）翻译问题。他们使用编码器通过语言模型（例如，门控循环单元（Chung 等人，2014）、BERT（Devlin 等人，2019））对题目文本进行嵌入，然后利用解码器按序生成表达式。后续研究（Xie 和 Sun，2019；Zhang 等人，2020b；Cao 等人，2021；Jie 等，2022）开发了 Seq2Tree 或序列到有向无环图神经网络模型，其中解码器根据特定结构迭代生成运算符和数量。尽管近期取得了进展，

[1]代码可在以下地址获取 https://github.com/sophistz/Question-Aware-Deductive-MWP。

这些神经求解器要构建出像人类一样的鲁棒表示并给出有效解，仍有很长的路要走。一些研究（Patel 等人，2021）发现，即使无法访问题目中所提问的问题部分，现有的基于神经网络的求解器仍能解答很大一部分数学文字题，这表明这些求解器可能只是在进行浅层推理或模式匹配，而非充分理解并推理数学文本。此外，大多数神经求解器在解码表达式时并不遵循数学定律（例如交换律）（Faldu 等人，2021），因此，数学上等价的表达式可能具有不同的表示。例如，表达式 $(1-2)+3$ 的隐藏表示不同于表达式 $(-2+1)+3$的隐藏表示，尽管它们计算的是同一个数量。

为缓解上述两个问题，我们提出了一种新颖的神经框架，该框架充分利用问题中的问题部分，并在每一步强制遵循交换律。具体而言，将上下文和问题输入编码器后，我们得到问题文本的表示，并将其视为查询，以引导解码过程。这模拟了人类的问题求解过程：我们通常会根据目标来寻找问题的答案。从数量和运算符生成新的表达式时，对于加法和乘法等可交换运算符，我们采用 Deep Sets（Zaheer 等人，2017）；对于加法逆元和乘法逆元等不可交换运算符，则采用多层感知机。这样的设计在每一步都保留了交换律，因此预测更加稳健。为评估所提出的方法，我们在四个多语言数学应用题基准数据集（MAWPS（Koncel-Kedziorski 等人，2016）、Math23k（Wang 等人，2017）、SVAMPS（Patel 等人，2021）和 MathQA（Amini 等人，2019）上进行了实验，结果表明，我们的框架优于最先进的基于神经网络的求解器。我们还对结果进行了详细分析，以探讨其局限性和潜在的改进方向。

我们的贡献总结如下：

- 我们提出了一种新颖的编码器-解码器框架，充分利用数学应用题的问题部分，并强制执行逐步交换律，以改进数学应用题求解。

- 我们在两种语言的四个成熟基准数据集上的实验结果表明，我们的模型优于最先进的神经网络数学文字题求解器。我们进一步分析了实验结果，并讨论了这些技术的局限性和未来的潜在研究方向。

2 相关工作

数学应用题求解研究的历史可以追溯到20世纪60年代（Bobrow，1964年）。早期的大多数研究（Fletcher，1985年；Dellarosa，1986年）采用基于规则的方法，将文本输入转换为底层的预定义模式。尽管这种类似槽位填充的方法能够稳健地处理无关信息，但这些规则难以穷尽地捕捉语言的诸多细微差异，因此无法很好地泛化到不同的语言风格。

随着机器学习的发展，一系列研究（Kushman 等人，2014；Zhou 等人，2015年；Mitra 和 Baral，2016；Roy 和 Roth，2018）将统计机器学习技术应用于数学应用题求解。这些技术在基于优化的评分框架中为多个表达式评分，随后为给定文本得出正确的数学模型。然而，与基于规则的方法类似，这些方法从训练数据中发现表达式模板，无法泛化到推理阶段未见过的模板。

近年来，基于学习的模型已成为求解数学应用题的新趋势（Faldu 等人，2021；Lan 等人，2022；Sundaram 等人，2022）。这些神经求解器通常先将题目文本编码到潜在空间中，然后迭代地使用运算符和数量解码出表达式。Wang 等人（2017）是开创性工作，它采用循环神经网络作为编码器和解码器，以 Seq2Seq 的方式生成目标方程。后续工作通过强化学习（Huang 等人，2018）、多头注意力（Li 等人，2019）和大语言模型（Tan 等人，2021）增强了 Seq2Seq 模型。然而，尽管人类按从左到右的顺序写出表达式，数学表达式实际上具有特定的结构。因此，Xie 和 Sun（2019）提出了一种目标驱动的树结构模型，在解码过程中生成表达式树。这种 Seq2Tree 方法的性能显著优于传统的 Seq2Seq 方法，并成为神经数学文字题求解器的主流方法。后续工作利用语义信息和数量关系改进 Seq2Tree 模型（Zhang

等人，2020b）或句法依存关系等外部知识（Lin 等人，2021）以及常识知识（Wu 等人，2020）。

尽管近期基于神经网络的方法在数学应用题求解方面取得了一些令人鼓舞的成果，但最先进的基于神经网络的求解器仍无法与人类匹敌，即使面对初等数学应用题也是如此。此外，神经求解器是否真正理解题目并像人类一样执行推理过程，仍不清楚（Sundaram 等人，2022）。Patel 等人（2021）发现，即使删去题目中的问题部分，基于神经网络的方法仍能生成表达式和解。他们指出，神经网络可能只是学会了生成表达式的浅层启发式，而非进行真正的数学推理。此外，大多数神经求解器在设计中并未纳入数学定律（即交换律、结合律和分配律）（Faldu 等人，2021），因此数学等价表达式的表示形式各不相同。尽管这些等价表达式计算出的数量相同，神经求解器仍会倾向于生成某个特定表达式，导致预测缺乏鲁棒性。为解决这一问题，一些工作致力于在编码过程中生成多样化的数学等价表达式，具体做法包括采用多个解码器生成多个表达式（Zhang 等人，2020a；Shen 和 Jin，2020）或通过增加更多标准表达式来扩充数据集（Yang 等人，2022）。然而，这些方法带来的改进十分有限，问题依然存在。

3 问题形式化

数学应用题可以描述为一种自然语言中的词语集合，其中包含 l 词语$\mathcal{S} = \{w_1, w_2, ..., w_l\}$。给定数量集合 $\mathcal{Q} = \{q_1, q_2, ..., q_n\} \subset \mathcal{S}$、固定的常数集合 $\mathcal{C} = \{c_1, c_2, ..., c_m\}$ 和固定的二元运算符集合 $\mathcal{O} = \{o_1, o_2, ..., o_k\}$，模型旨在生成一个 T 步的数学表达式列表，从而得出最终答案。该表达式列表可表示为 $\mathcal{E} = \{e_1, e_2, ..., e_T\}$，其中 $e_t = (e_i, \ e_j, \ o_t)$，满足 $i < t$ 或 $e_i \in \mathcal{Q}$、$j < t$ 或 $e_j \in \mathcal{Q}$，以及 $o_t \in \mathcal{O}$。e_t 表示对数量或先前生成的中间数学表达式应用二元运算符 o_t 所得的表达式。该题的最终数学表达式为 e_T。

4 方法

我们提出了一种新颖的编码器-解码器框架，将问题部分信息融入解码过程，并确保数学等价表达式具有逐步不变性。我们的模型概览如图所示 2。

我们的编码器首先使用语言模型获取题目文本中每个数量的表示，以及整个问题文本的表示。给定数量嵌入和问题嵌入后，我们的解码器采用自底向上的方式，迭代生成表达式。在每一步中，解码器以问题嵌入为条件生成数学表达式，并利用问题文本引导解码过程。关于我们的编码器和解码器的更多细节将在以下小节中介绍。

4.1 编码器

首先将语言模型应用于原始题目文本，以获取文本中出现的数量的表示。具体而言，文本中的数量会被转换为词元“*<quant>*”，转换后的文本经过分词后输入语言模型。实践中，我们使用 RoBERTa（Liu 等人，2019；Cui 等人，2021）作为编码器，生成表示 h_q 每个数量的表示 q。

除了生成文本中数量的表示外，我们还生成问题文本的表示。请注意，在一些数学文字题中，问题文本可能没有明确界定，因此我们将问题部分 $\mathcal{S}_{qn} \subset \mathcal{S}$ 定义为问题文本中以句号分隔的最后一个完整句子。为了获得问题表示h_{qn}，我们对所有问题词元的嵌入进行均值池化。

4.2 解码器

我们的解码器由三个组件构成：一个构造器、一个评分器和一个合理化器。

4.2.1 构造器

给定数量的嵌入表示后，我们的构造器首先在每一步枚举数量与运算符的所有可能组合。每种组合都是当前步骤的候选表达式，我们从中选择一个作为该步骤的输出表达式。为了计算每个候选项的表示，构造器会使用

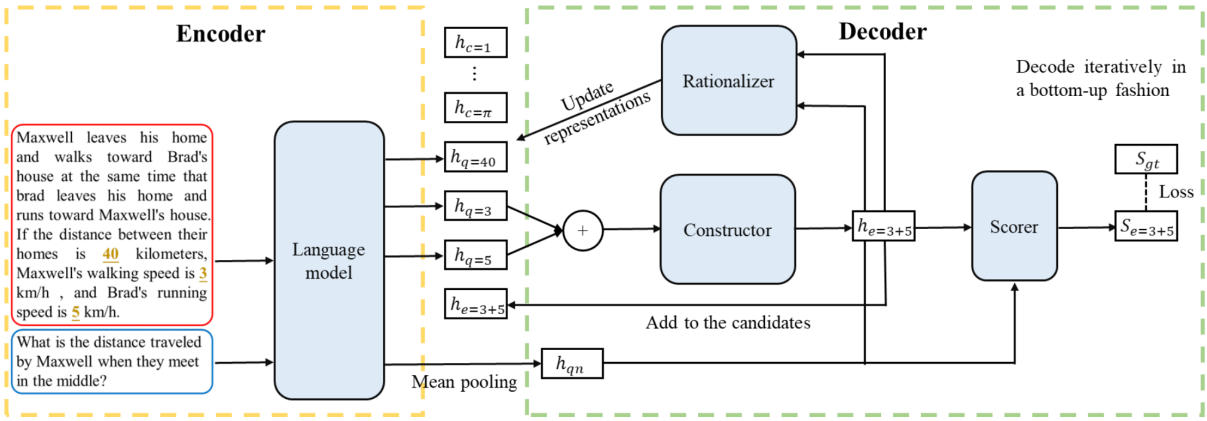

图2：我们框架的概览。给定编码器生成的数量嵌入和问题嵌入后，解码器在第一步生成表达式 $e = 3 + 5$。

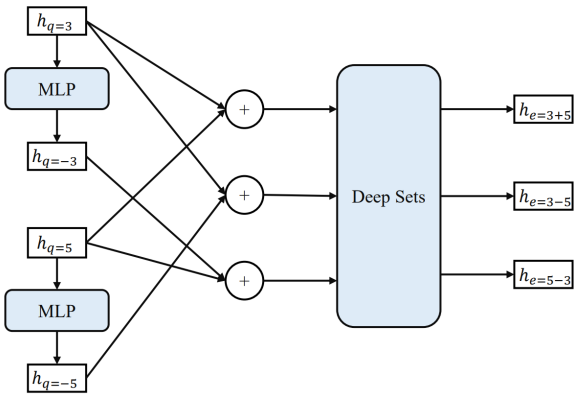

图3：我们构造器的架构。

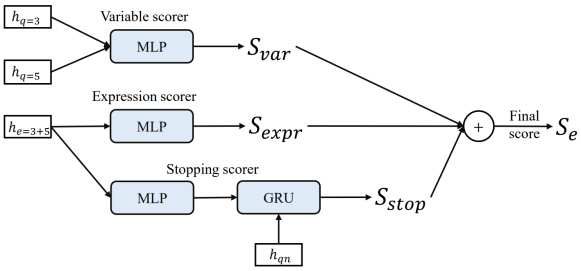

图4：我们评分器的架构。

一种用于加法和乘法等可交换运算符的置换不变函数，以及一种用于加法逆元和乘法逆元等不可交换运算符的非线性变换。在我们的实现中，我们对可交换的二元运算符使用 Deep Sets，对不可交换的一元运算符使用多层感知机。构造器的示意图如图 3所示。为了计算表达式 $e = 3 - 5$的表示，我们首先对数量嵌入表示$h_{q=5}$ 应用一个多层感知机，得到表示 $h_{q=-5}$，并对两个数量的嵌入表示使用 Deep Sets（Zaheer 等人，2017）：

$$h_{q=-5} = \text{MLP}_{\text{add_inv}}(h_{q=5}) \tag{1}$$

$$h_{e=3-5} = \text{MLP}^2_{\text{add}}\left(\sum_{v=3,-5} \text{MLP}^1_{\text{add}}(h_{q=v})\right) \tag{2}$$

需要强调的是，候选表达式的表示在数量的任意置换下保持不变——（例如，$3 - 5$ 和 $-5 + 3$、$3 \div 5$ 和 $1/5 \times 3$），因此我们的构造器能够保证逐步满足交换律，并统一使用 $+$和 $-$运算符或×和 $\div$运算符的表达式表示。

4.2.2 评分器

利用构造器生成的所有候选表达式的表示，我们的评分器会为每个候选表达式打分，以确定当前步骤应生成哪个表达式，以及它是否为最终表达式。图4 展示了我们评分器的架构。

我们的评分器由三部分组成：变量评分器、表达式评分器和停止评分器。变量评分器和表达式评分器会计算继续分数，即将该表达式选作中间表达式的分数。停止评分器会计算终止分数，表示将该表达式选作最终结果表达式的分数。具体而言，变量评分器会接收表达式中两个量的表示，并计算变量分数 s_{var}，表示应使用量

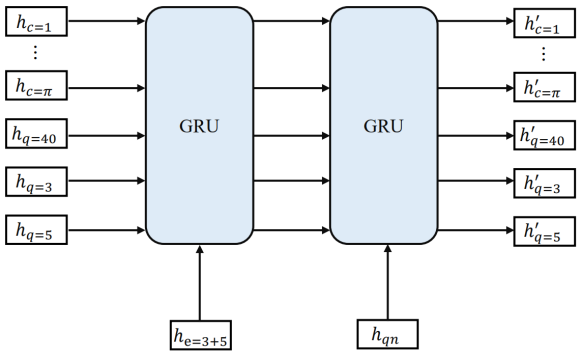

图 5：合理化器的架构。

的概率 q_i，q_j 在当前步骤使用：

$$s_{\text{var}} = \text{MLP}_{\text{var}}(h_{q=q_i}) + \text{MLP}_{\text{var}}(h_{q=q_j}) \quad (3)$$

表达式评分器接收候选表达式 h_e 的表示，并使用多层感知机计算表达式得分s_{expr}：

$$s_{\text{expr}} = \text{MLP}_{\text{epxr}}(h_e) \quad (4)$$

为了确定是否应停止解码过程，停止评分器将候选表达式 h_e 和问题 h_{qn} 的各自表示作为输入，并使用多层感知机和 GRU 单元计算停止得分s_{stop} (Chung 等人，2014)：

$$s_{\text{stop}} = \text{GRU}_{\text{stop}}\left(\text{MLP}_{\text{stop}}(h_e), h_{qn}\right) \quad (5)$$

最后，候选表达式 s_e的得分可以通过将这三部分相加得到：

$$s_e = s_{\text{var}} + s_{\text{expr}} + s_{\text{stop}} \quad (6)$$

4.2.3 合理化器

遵循Jie 等人(2022）的方法，我们也采用合理化器，在每个步骤结束时更新所有数量和中间表达式的表示。该模块至关重要，因为如果不更新这些表示，最初排名靠前的表达式就会一直更受青睐。例如，表达式 $e = 3 + 5$ 在第一步中得分最高；如果其表示保持不变，那么在后续步骤中它的得分很可能仍然很高。

图 5 展示了我们合理化模块的架构。在我们的实现中，我们使用新选出的表达式表示和问题表示，通过两个 GRU 单元对数量表示进行合理化：

$$h_q^{'} = \text{GRU}_{\text{rat}}^{2}\left(h_{qn}, (\text{GRU}_{\text{rat}}^{1}(h_q, h_e)\right) \quad (7)$$

第一个 GRU 单元以数量表示作为输入，以新的表达式表示作为隐藏状态。第二个 GRU 单元以第一个 GRU 单元的表示作为输入，以问题表示作为隐藏状态。最终输出是每个数量更新后的表示。

4.3 训练与推理

我们采用教师强制策略（Williams 和 Zipser，1989）在训练过程中，每一步都用标准表达式引导模型。将框架中所有可学习参数记为 θ，我们的损失定义如下：

$$\mathcal{L}_\theta = \sum_{t=1}^{T}\left(\max_{e\in\text{candidates}^t} s_e - s_{e_t^*}\right) \quad (8)$$

其中，T 是生成真实表达式所需的总步数，候选集^t 表示第 t步的所有候选表达式，$^{s_{e_t^*}}$ 是第 t步的真实表达式。对于每一步，该损失都会最小化黄金表达式与得分最高的表达式之间的分数差距，从而鼓励框架为真实表达式生成最高分数。

在推理过程中，我们迭代地选择每一步得分最高的表达式，直到 t达到预先设定的最大步数 T。对于生成的表达式 e_1，e_2，$\cdots$，e_T，选择终止分数最高的表达式作为输出表达式，并计算其对应的数值答案作为最终答案。

5 实验

5.1 数据集

我们在四个成熟的数学应用题基准数据集上进行实验：MAWPS（Koncel-Kedziorski 等人，2016）、Math23k（Wang 等人，2017）、SVAMPS（Pa- tel 等人，2021）和 MathQA（Amini 等人，2019）。表1 展示了这些数据集的统计信息。MAWPS 和 Math23k 常用于以往的研究，包含小学数学题。MathQA 和 SVAMP 则难得多。MathQA 包含许多领域中更复杂的 GRE题目，包括物理学、几何和概率，因此其中包含大量涉及更多运算的方程。SVAMP 包含人工设计的挑战性题目，这些题目通过对问题

数据集	#训练集	#验证集	#测试集	运算	语言
MAWPS	1,589	199	199	+ − × ÷	英语
Math23k	21,162	1,000	1,000	+ - × ÷ ^	中文
MathQA	16,191	2,411	1,605	+ - × ÷ ^	英语
SVAMP	3,138	-	1,000	+ − × ÷	英语

表1：数据集统计。

从 MAWPS 中改编而来，其中包含许多无关的数量信息和上下文文本。四个数据集中的题目示例见附录 A.1 。

5.2 评估指标

我们使用最终值准确率（验证准确率）作为评估指标，它表示我们根据方程计算出的数值答案与真实答案相等的准确率。请注意，可能有多个数学表达式得出相同的数值答案（例如，$(1+2)\times 3$和 $1\times 3+2\times 3$），因此生成其中任意一个都视为正确。

5.3 基线方法

基线方法分为序列到序列（Seq2Seq）模型和序列到树（Seq2Tree）模型。我们在四个基准数据集上，将提出的模型与四种 Seq2Seq 模型和六种 Seq2Tree 模型进行比较。在这些 Seq2Seq 数学文字题求解器中，GroupAttn（Li 等人，2019）提出在 Seq2Seq 模型中设计多种注意力机制，例如数量相关注意力。mBERT-LSTM（Tan 等人，2021）使用多语言 BERT 作为编码器，LSTM 作为解码器。BERT-BERT（Lan 等，2022）和 RoBERTa-RoBERTa（Lan 等，2022）则分别使用 BERT 和 RoBERTa 同时作为编码器和解码器。在这些 Seq2Tree 模型中，GTS（Xie 和 Sun，2019）是开创性工作，它使用双向门控循环单元对题目文本进行编码，并以自顶向下的方式通过树结构解码数学表达式。Graph2Tree（Zhang 等人，2020b）通过使用 GCN 对数量关系和顺序信息进行建模，改进了 GTS 的编码器。BERT-Tree（Liang 等人，2021）通过 8 个预训练任务对 BERT 进行预训练，以解决数字表示问题，并将其用作编码器。RoBERTa-GTS 和 RoBERTa-Graph2Tree 使用 RoBERTa 替换 GTS 和 Graph2Tree 原有的编码器。与我们的方法最相似的是 RoBERTa-DeductiveReasoner（Jie 等，2022），它使用 RoBERTa 作为编码器，并以数学——表达式的自底向上解码方式进行解码。然而，它将题目文本整体编码，没有对问题文本给予任何特别关注，并且在计算每个候选表达式的表示时，未能保留任何数学规律。

类别	模型	验证准确率
Seq2Seq	GroupAttn	76.1
	Transformer	85.6
	BERT-BERT	86.9
	RoBERTa-RoBERTa	88.4
Seq2Tree	GTS	82.6
	Graph2Tree	85.6
	RoBERTa-GTS	88.5
	RoBERTa-Graph2Tree	88.7
	RoBERTa-DeductReasoner	92.0 ± 0.20
	Ours	$\mathbf{92.9 \pm 0.14}$

表 2：MAWPS 上 5折交叉验证结果对比。

5.4 实现细节

对于英语数据集 MAWPS、MathQA 和 SvAMP，我们使用 RoBERTa（Liu 等人，2019）作为编码器。对于中文数据集 Math23K，我们使用中文 RoBERTa（Cui 等人，2021）作为编码器。预训练的 RoBERTa 模型采用 HuggingFaces Transformers（Wolf 等人，2020）中的参数进行初始化。所有多层感知机均有 2 个隐藏层，每层包含 768 个单元，并使用 ReLU 作为激活函数。在这 4 个数据集上训练时，我们使用 30 的批次大小。在 Math23k 、 MathQA 和 SVAMP 上，我们训练模型 1，000 个轮次。在 MAWPS 上，我们训练模型 100 个轮次。我们使用学习率为 2×10^{-5} 的 Adam 优化器来优化损失函数。所有实验均在配备 4 个 NVIDIA RTX-8000 GPU 的集群上针对每个数据集运行 3 次。遵循先前的工作（Lan 等人，2022；Jie 等人，2022），我们报告 MAWPS 上的 5 折交叉验证准确率，以及 Math23k、MathQA 和 SVAMP 上的测试准确率。基线方法的结果取自其原始论文。

5.5 主要结果

表2 展示了不同模型在 MAWPS 数据集上的 5折交叉验证结果。表3、表4、表5 分别展示了 Math23k、MathQA 和 SVAMP 数据集上的测试准确率比较。结果表明，与 Seq2Seq 模型相比，Seq2Tree 模型通常表现更好，我们推测这是因为树解码器比序列解码器更善于处理数学表达式的结构。同时，GTS 和 Graph2Tree 等模型通过使用 BERT 或 RoBERTa 等预训练大语言模型作为编码器，性能得到了显著提升，这表明大语言模型的表征能力有助于数学应用题求解。

类别	模型	验证准确率
Seq2Seq	GroupAttn	69.5
	mBERT-LSTM	75.1
Seq2Tree	GTS	75.6
	Graph2Tree	77.4
	RoBERTa-DeductReasoner	85.1 ± 0.24
	Ours	**85.3 ± 0.21**

表 3：Math23k 测试准确率对比。

类别	模型	验证准确率
Seq2Seq	BERT-LSTM	77.1
Seq2Tree	Graph2Tree	69.5
	BERT-Tree	73.8
	RoBERTa-DeductReasoner	77.2 ± 0.11
	Ours	**81.1 ± 0.13**

表 4：MathQA 上的测试准确率对比。

是因为树解码器比序列解码器更善于处理数学表达式的结构。同时，GTS 和 Graph2Tree 等模型通过使用 BERT 或 RoBERTa 等预训练大语言模型作为编码器，性能得到了显著提升，这表明大语言模型的表征能力有助于数学应用题求解。

我们的模型在 MAWPS、Math23k、MathQA 和 SVAMP 上的测试准确率分别为 92.9%、85.3%、81.1% 和 44.4%。在所有基准数据集上，我们的模型均取得了最先进的性能，分别比最佳基线模型 RoBERTa-DeductReasoner 高出 0.9%、0.2%、3.9% 和 0.4%。尤其值得注意的是，在四个数据集中规模最大、难度最高的 MathQA数据集上，我们的模型比原始 RoBERTa-DeductReasoner 提升了 3.9%。这些显著的提升表明，在设计中利用问题文本信息并融入数学定律，都有助于求解数学应用题。总而言之，这些比较充分证明了我们提出的改进神经数学文字题求解器技术的有效性。

5.6 消融研究

为详细分析我们的方法，我们进一步进行消融研究，以探究所提出的两种方法的有效性。我们用 RoBERTa-DeductReasoner（Jie 等人，2022）中的模块替换构造器，作为不保留交换律的模型；并在解码器中去掉问题表示，作为不包含问题表示的模型。结果如表 6 所示。与该模型相比

类别	模型	验证准确率
Seq2Seq	GroupAttn	21.5
	BERT-BERT	24.8
	RoBERTa-RoBERTa	30.3
Seq2Tree	GTS	30.8
	Graph2Tree	36.5
	RoBERTa-DeductReasoner	44.0 ± 0.20
	Ours	**44.4 ± 0.11**

表5：SVAMP上的测试准确率对比。

数据集	模型	验证准确率
MAWPS	RoBERTa-DeductReasoner	92.0
	Ours	**92.9**
	- 不保留交换律	92.6
	- 不使用问题表示	92.3
Math23k	RoBERTa-DeductReasoner	85.1
	Ours	85.3
	- 不保留交换律	**85.8**
	- 不使用问题表示	84.6
MathQA	RoBERTa-DeductReasoner	77.2
	Ours	**81.0**
	- 不保留交换律	79.8
	- 不使用问题表示	78.1
SVAMP	RoBERTa-DeductReasoner	44.0
	Ours	**44.4**
	- 不保留交换律	44.2
	- 不使用问题表示	43.8

表6：我们模型的消融研究。将我们的模型与不包含问题部分信息的模型，以及不保留交换律的模型进行比较。

在不保留逐步交换律的情况下，我们的完整模型在 MAWPS、MathQA 和 SVAMP 测试集上的准确率分别提高了 0.3%、1.2% 和 0.2%。值得注意的是，在 Math23k 上，准确率下降了 0.5%。我们推测，这是因为 MathQA 中的数学表达式相对复杂，因此仅为表达式保留逐步交换律还不够。另一方面，与不使用问题嵌入的模型相比，完整模型在 MAWPS、Math23k、MathQA 和 SVAMP 测试集上的准确率分别提高了 0.6%、0.7%、2.9% 和 0.6%。这些结果表明，既施加数学规律，又对问题文本进行显式建模，以问题表示引导解码过程，通常都能提升神经数学文字题求解器的性能。结合这两种技术可以进一步提升模型性能。比较这两种策略可以发现，利用问题部分信息对准确率的提升，大于为数学表达式保留逐步交换律所带来的提升。

6 讨论

6.1 局限性

我们当前的模型提供了一种技术，可在每一步强制遵守交换律，并统一一组运算符（如加法与减法，或乘法与除法）的表示。例如，表示 $h_{e=1+2} = h_{e=2+1}$和 $h_{e=2-1} = h_{e=2+(-1)} = h_{e=(-1)+2}$。但是，在计算包含多个运算符的复杂表达式时，交换律无法得到保留。例如，$h_{e=(1+2)+3} \neq h_{e=(3+2)+1}$，不同的运算顺序会产生不同的表示。此外，分配律和结合律等其他数学规律也无法得到保留。例如，$h_{e=(1+2)\times 3} \neq h_{e=1\times 3+2\times 3}$。

6.2 未来工作

未来，我们计划在神经数学文字题求解器中强制应用更多数学定律，并为所有数学上等价的表达式构建一种置换不变表示。该想法可以通过设计一种统一格式来表示所有数学表达式，并基于这种格式计算其表示来实现。具体而言，我们可以采用一种没有括号的细粒度格式来表示所有表达式。例如，要计算（$1 + 2$）$\times 3$的表示，与其先生成（1 + 2）的嵌入，再生成（$1 + 2$）$\times 3$，不如先将该表达式转换为 $1 \times 3 + 2 \times 3$，然后在这种格式上应用一些置换不变函数来计算其表示。因此，所有数学表达式都将具有相同的嵌入，模型也能够保留分配律和结合律。

6.3 结论

现有的神经数学文字题求解器大多未能充分利用问题文本，也未能保留任何数学定律。本文提出一种新的编码器-解码器框架，通过两种技术解决这些问题：（1）我们的编码器为问题文本生成嵌入，并用其引导解码过程。（2）我们的解码器应用 Deep Sets 计算候选表达式的表示，以确保逐步满足交换律。在四个标准数学文字题基准测试上的实验表明，这两种技术能够提升神经数学文字题求解器的性能，并使我们的模型达到最先进的水平。

6.4 致谢

我谨感谢 Zhaoyu Li 先生、Xujie Si 教授和 Siva Reddy 在本研究中的协助。特别感谢 Zhaoyu Li 先生，他调研了相关出版物并提出了这一想法。

参考文献

Aida Amini、Saadia Gabriel、Shanchuan Lin、Rik Koncel-Kedziorski、Yejin Choi 和 Hannaneh Hajishirzi。2019。MathQA：通过基于运算的形式化方法实现可解释的问题求解。载于*2019*年北美计算语言学协会分会会议论文集：人类语言技术。

Daniel G Bobrow。1964年。面向计算机问题求解系统的自然语言输入。

Yixuan Cao、Feng Hong、Hongwei Li 和 Ping Luo。2021年。用于数学应用题的自底向上DAG结构提取模型。载于*AAAI*人工智能会议论文集。

Junyoung Chung、Caglar Gulcehre、KyungHyun Cho 和 Yoshua Bengio。2014。门控循环神经网络在序列建模中的实证评估。*arXiv*预印本 *arXiv:1412.3555*。

Yiming Cui、Wanxiang Che、Ting Liu、Bing Qin 和 Ziqing Yang。2021。面向中文BERT的全词掩码预训练。*IEEE/ACM*音频、语音与语言处理汇刊。

Denise Dellarosa。1986年。儿童算术应用题求解的计算机模拟。行为研究方法、仪器与计算机。

Jacob Devlin、Ming-Wei Chang、Kenton Lee 和 Kristina Toutanova。2019。BERT：用于语言理解的深度双向Transformer预训练。载于*2019*年计算语言学协会北美分会会议论文集：人类语言技术，*NAACL-HLT*。

Keyur Faldu、Amit Sheth、Prashant Kikani、Manas Gaur 和 Aditi Avasthi。2021年。迈向易处理的数学推理：解决数学文字题的挑战、策略与机遇。*arXiv*预印本 *arXiv:2111.05364*。

Charles R Fletcher。1985年。理解并解决算术应用题：计算机模拟。行为研究方法、仪器与计算机。

Danqing Huang、Jing Liu、Chin-Yew Lin 和 Jian Yin。2018。基于强化学习的神经网络数学应用题求解器。载于第*27*届计算语言学国际会议论文集。

Zhanming Jie、Jierui Li 和 Wei Lu。2022。学习演绎推理：将数学应用题求解视为复杂关系抽取。载于计算语言学协会第*60*届年会论文集。

Rik Koncel-Kedziorski、Subhro Roy、Aida Amini、Nate Kushman 和 Hannaneh Hajishirzi。2016。MAWPS：一个数学应用题语料库。载于 计算语言学协会北美分会第*2016*届会议论文集：人类语言技术。

Nate Kushman、Yoav Artzi、Luke Zettlemoyer 和 Regina Barzilay。2014。学习自动求解代数应用题。载于计算语言学协会第*52*届年会论文集。

Yihuai Lan、Lei Wang、Qiyuan Zhang、Yunshi Lan、Bing Tian Dai、Yan Wang、Dongxiang Zhang 和 Ee-Peng Lim。2022。Mwptoolkit：一个用于基于深度学习的数学应用题求解器的开源框架。载于*AAAI*人工智能会议论文集。

Jierui Li、Lei Wang、Jipeng Zhang、Yan Wang、Bing Tian Dai 和 Dongxiang Zhang。2019。利用不同的函数式多头注意力对数学应用题中的内部关系进行建模。载于 计算语言学协会第*57*届年会论文集。

Zhenwen Liang、Jipeng Zhang、Jie Shao 和 Xian-gliang Zhang。2021 年。Mwp-bert：数学文字题的强基线。*arXiv* 预印本 *arXiv:2107.13435*。

Xin Lin、Zhenya Huang、Hongke Zhao、Enhong Chen、Qi Liu、Hao Wang 和 Shijin Wang。2021 年。Hms：一种利用依赖关系增强的数学应用题层次求解器。发表于*AAAI*人工智能会议论文集。

Yinhan Liu、Myle Ott、Naman Goyal、Jingfei Du、Mandar Joshi、Danqi Chen、Omer Levy、Mike Lewis、Luke Zettlemoyer 和 Veselin Stoyanov。2019 年。RoBERTa：一种经过稳健优化的 BERT 预训练方法。*arXiv* 预印本 *arXiv:1907.11692*。

Arindam Mitra 和 Chitta Baral。2016 年。学习使用公式解决简单算术题。发表于第 *54* 届计算语言学协会年会论文集。

Arkil Patel、Satwik Bhattamishra 和 Navin Goyal。2021 年。自然语言处理模型真的能够解决简单的数学文字题吗？载于*2021* 年计算语言学协会北美分会会议论文集：人类语言技术。

Subhro Roy 和 Dan Roth。2018。利用声明性知识进行文字题求解。计算语言学协会会刊。

Yibin Shen 和 Cheqing Jin。2020。使用多编码器和多解码器求解数学文字题。载于第*28*届国际计算语言学会议论文集。

Sowmya S Sundaram、Sairam Gurajada、Marco Fisichella、Savitha Sam Abraham 等。2022 年。为什么自然语言处理模型在初等数学上频频失误？基于深度学习的文字题求解器综述。*arXiv* 预印本 *arXiv:2205.15683*。

Ilya Sutskever、Oriol Vinyals 和 Quoc V Le。2014 年。使用神经网络进行序列到序列学习。神经信息处理系统进展，27。

Minghuan Tan、Lei Wang、Lingxiao Jiang 和 Jing Jiang。2021 年。使用预训练多语言语言模型研究数学文字题。*arXiv* 预印本 *arXiv:2105.08928*。

Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N Gomez、Łukasz Kaiser 和 Illia Polosukhin。2017 年。Attention is all you need。神经信息处理系统进展，30。

Yan Wang、Xiaojiang Liu 和 Shuming Shi。2017 年。用于数学文字题的深度神经求解器。载于*2017*年自然语言处理实证方法会议论文集。

Ronald J Williams 和 David Zipser。1989 年。连续运行全循环神经网络的学习算法。神经计算。

Thomas Wolf、Lysandre Debut、Victor Sanh、Julien Chaumond、Clement Delangue、Anthony Moi、Pierric Cistac、Tim Rault、Rémi Louf、Morgan Funtowicz 等。2020 年。Transformer：自然语言处理的最先进水平。载于*2020*年自然语言处理实证方法会议：系统演示论文集。

Qinzhuo Wu、Qi Zhang、Jinlan Fu 和 Xuan-Jing Huang。2020。用于数学应用题求解的知识感知序列到树网络。载于*2020*年自然语言处理实证方法会议论文集。

Zhipeng Xie 和 Shichao Sun。2019。用于数学文字题的目标驱动树结构神经模型。载于第*28*届国际人工智能联合会议论文集。

Zhicheng Yang、Jinghui Qin、Jiaqi Chen 和 Xiaodan Liang。2022。用于缓解求解偏差的无偏数学应用题基准。载于计算语言学协会研究成果。

Manzil Zaheer、Satwik Kottur、Siamak Ravanbakhsh、Barnabas Poczos、Russ R Salakhutdinov 和 Alexander J Smola。2017。Deep Sets。神经信息处理系统进展。

Jipeng Zhang、Roy Ka-Wei Lee、Ee-Peng Lim、Wei Qin、Lei Wang、Jie Shao 和 Qianru Sun。2020a。采用多个解码器求解数学应用题的师生网络。载于第*29*届国际人工智能联合会议论文集。

Jipeng Zhang、Lei Wang、Roy Ka-Wei Lee、Yi Bin、Yan Wang、Jie Shao 和 Ee-Peng Lim。2020b。通过图到树学习求解数学文字题。载于第*58*届计算语言学协会年会论文集。

Lipu Zhou、Shuaixiang Dai 和 Liwei Chen。2015年。使用二次规划学习求解代数应用题。载于*2015*年自然语言处理实证方法会议论文集。

A 附录

附录A.1 数据集样本

表7 展示了每个数据集的两个样本。

数据集	MAWP
输入	玛丽正在烤蛋糕。食谱需要8杯面粉。她已经放入了2杯。她还需要加多少 她需要添加多少杯?
答案	6
表达式	8 - 2
输入	抽屉里有139块橡皮擦和118把剪刀。杰森把131块橡皮擦放进了 抽屉里。现在一共有多少块橡皮擦?
答案	270
表达式	139 + 131

数据集	Math23k
输入	一个工程队 dig soil，第一天挖了316立方米，从第二天开始每天都挖230立方米，连续挖 了6天，这个工程队一周共 dig soil 多少立方米?
翻译	工程队第一天挖了 316 立方米，之后每天挖 230 立方米，连续 挖了 6 天。一周内工程队一共挖了多少立方米? 工程队一周内挖了多少立方米?
答案	1466
表达式	316 + 230 × (6 - 1)
输入	从甲地到乙地，如果骑自行车每小时行驶16千米，4小时可以到达；如果乘汽车 只需要2小时，汽车每小时行驶多少千米?
翻译	从甲地到乙地，如果骑自行车每小时行驶16千米，4小时可以到达 。如果乘汽车只需要2小时，汽车每小时行驶多少千米? 小时?
答案	32
表达式	16 × 4 ÷ 2

数据集	MathQA
输入	Sophia读完了一本书的2/3。她计算出自己读完的页数比 尚未读完的页数多90页。她的书有多少页?
答案	270
表达式	90 ÷ (1 - 2 ÷ 3)
输入	6名工人应在8天内完成一项工作。工作了3天后，又有4名工人加入他们。 他们还需要多少天才能完成同一项工作?
答案	2.5
表达式	(6 × 8 - (6 × 3)) ÷ (8 + 4)

数据集	SVAMP
输入	Bobby 吃了一些糖果。然后他又吃了 25 颗。如果他总共吃了 43 颗 糖果。他一开始吃了多少颗糖果?
答案	18
表达式	43 - 25
输入	Bobby 吃了 28 颗糖果。然后他又吃了 42 颗。他还吃了 63 块巧克力。 Bobby 吃了多少颗糖果?
答案	70
表达式	28 + 42

表7：MAWP、Math23k、MathQA 和 SVAMP 中的样本。