分类号：TM888　　单位代码：10335

密　级：　　学　　号：21638888

浙江大学

硕士学位论文

中文论文题目：语言模型的规则遵循

英文论文题目：Rule Compliance in Language Models

申请人姓名：陈钶杰

指导教师：张庆海教授

专业名称：计算数学

研究方向：大语言模型

所在学院：数学科学学院

论文提交日期：二〇二四年四月

语言模型的规则遵循

论文作者签名： _____

指导教师签名： _____

论文评阅人 1：陈某某　教授　XX 大学
评阅人 2：张某某　教授　XX 大学
评阅人 3：李某某　教授　XX 大学
评阅人 4：吴某某　教授　XX 大学
评阅人 5：杨某某　教授　XX 大学

答辩委员会主席：郑某某　教授　浙江大学
委员 1：郑某某　教授　浙江大学
委员 2：刘某某　教授　浙江大学
委员 3：程某某　教授　浙江大学
委员 4：杨某某　教授　浙江大学
委员 5：吴某某　教授　浙江大学

答辩日期：二〇二四年六月

Rule Compliance in Language Models

Author's signature: _____

Supervisor's signature: _____

External Reviewers:

Moumou Chen Prof. XX University

Moumou Zhang Prof. XX University

Moumou Li Prof. XX University

Moumou Wu Prof. XX University

Moumou Yang Prof. XX University

Examining Committe Chairperson:

Moumou Zheng Prof. Zhejiang University

Examining Committe Members:

Moumou Zheng Prof. Zhejiang University

Moumou Liu Prof. Zhejiang University

Moumou Chen Prof. Zhejiang University

Moumou Yang Prof. Zhejiang University

Moumou Wu Prof. Zhejiang University

Date of oral defence: June

浙江大学研究生学位论文独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得浙江大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名：　　签字日期：　　年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解浙江大学有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交本论文的复印件和磁盘，允许论文被查阅和借阅。本人授权浙江大学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和传播，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后适用本授权书）

学位论文作者签名：　　导师签名：

签字日期：　　年　　月　　日　　签字日期：　　年　　月　　日

摘 要

在先前的研究中，我们注意到不依赖外部辅助工具的大语言模型在执行逻辑运算方面存在准确性问题，语言模型通常依赖于存储逻辑问题的表面模式，而不是深入理解其潜在逻辑。这一局限性可能源自于模型对逻辑运算学习方法的不足。以数学计算为例，考虑到数学运算的无限性，仅依靠有限的实例数据集训练，模型难以准确完成计算。受到元学习的启发，我们期望模型不单单学习特定任务的数据和解决方案，更重要的是学习如何学习的策略和方法。

我们在本文中提出了一种新型架构的语言模型。该模型通过学习和组合不同规则，能够精确地完成数值计算及其他复杂逻辑运算任务。该模型基于 Transformer 架构，采用动态的任务组合流程来指导训练。通过设计一套涵盖数学计算的基本规则、复合规则及迭代规则的数据集，这些规则被文本化并嵌入模型中进行预训练，由此得到能够处理复杂数值运算的 MetaRuleGPT 预训练模型。实验结果表明，特别是在高位数计算方面，我们的模型在准确性上超越了现有的知名大语言模型，如 Google 的 Palm，Llama2，阿里的 QWen 等，且能媲美 ChatGPT-3.5 和 ChatGPT-4。同时，对于更加复杂的数学逻辑推理问题，MetaRuleGPT 能够模仿人类的规则遵循能力，化繁为简，逐步推导得到准确的计算结果。

关键词： Transformer; 元学习; 大语言模型; 规则组合学习; 数值计算

Abstract

In our previous research, we observed that large language models, when operating without external tools, encounter accuracy issues in performing logic operations. These models tend to rely on recognizing the superficial patterns of logical problems rather than understanding their underlying logic. This limitation may stem from the models' insufficient approach to learning logical operations. For example, considering the vastness and complexity of mathematical operations, models trained only on a limited set of example datasets struggle to compute accurately. Inspired by the concept of meta-learning, we aspire for models to learn not just the data and solutions for specific tasks but more importantly, the strategies and methods of learning.

In this paper, we introduce a new language model architecture based on the Transformer framework. This model can precisely execute numerical calculations and other complex logic operations by learning and applying various rules. We have developed a dataset encompassing basic, composite, and iterative rules for mathematical calculations. These rules are textually represented and embedded into the model during pre-training, leading to the creation of the MetaRuleGPT pre-trained model capable of handling advanced numerical operations. Experimental results demonstrate that our model, especially in high-digit calculations, surpasses the accuracy of existing renowned large language models like Google's Palm, Llama2, and Alibaba's QWen, and is comparable to ChatGPT-3.5 and ChatGPT-4. Moreover, when faced with more complex mathematical logic reasoning tasks, MetaRuleGPT can mimic human rule-following capabilities, simplifying complex issues and methodically deriving precise outcomes.

Keywords: Meta-learning; Large language models; Numerical computations; Rule-based learning; Transformer

图 目 录

表 目 录

目 次

1 引言

1.1 选题背景与意义

在自然语言处理 (Natural Language Processing，NLP) 的世界里，大语言模型如 GPT-4[10,11] 已经取得了令人瞩目的进展，它们在各式各样的任务中展示了惊人的理解和处理能力。然而，当这些模型遇到数学问题以及其他需要专业知识的领域时，它们仍然面临着相当大的挑战。以数学问题为例，其覆盖的内容极其广泛，包括但不限于基础的加减乘除、求导、积分以及方程求解等。尽管拥有强大的语言理解能力，这些模型在解决基础的数学加减法和数值计算问题上仍显得力不从心。例如，对于简单的高位数加法问题：

$$241257284 + 758742716 = 1000000000$$

目前大多数主流语言模型都难以正确完成这类基本的数学运算。

在探索基于 Transformer[6] 架构的语言模型以挖掘其在解决数学问题方面的潜能时，研究人员取得了重要进展。Compute-GPT[3] 和 MathGPT[4] 是在这一领域表现较好的模型变种。Compute-GPT 采用了一种创新的方法，通过结合语言模型与外部工具 (如 Python)，提出使用 Python 工具来解决数学问题，为数值计算提供了一种独特的解决方案。另一方面，MathGPT 利用了传统语言模型的微调技术，通过增大训练数据集的规模，即对所有五位数以内的数值计算问题进行训练。此外，MathGPT 还采用了分步求解策略，将复杂问题分解成更小、更易管理的部分，从而相较于传统模型，在解决广泛数值问题上展现出了显著的能力。

现如今，部分语言模型变体在处理数学问题方面已经取得了一定的成就，但此类模型主要是通过融合外部计算工具或扩大训练数据集实现的，这些努力并未根本上深化语言模型对数学计算深层次的理解。研究表明，虽然通过大量数学问题的训练，语言模型能够展现出一定的计算能力，但鉴于数学问题的无限多样性，这种训练方法训练出的语言模型在问题解决的泛化方面存在一定局限性。

受到元学习中规则组合能力的启发，我们尝试让语言模型学习如何调用及组合规则，并通过迭代策略，逐步简化算式，从而更精确的求解数值问题。不同于之前的方法，我们的模型通过在规则层次上的学习，只需学习少量数学规则，便能处理大量的数值问题。此外，我们的模型还具备处理多样化任务的能力，利用掌握的数学规则，不仅能够执行基础的运算任务，也能够解决更复杂的问题，比如向量的叉积运算。更深入的实验结果表明，我们的模型在数学计算和处理复杂逻辑问题上不仅超越了现

有的大部分先进水平 SOTA(State-of-the-art model) 模型，甚至能与 ChatGPT-3.5[10,11] 和 ChatGPT-4.0 媲美，

我们的主要贡献总结如下:

1. 受到元学习的启发，我们采用了规则组合的方法，促使语言模型学会数值计算的本质规律，从而展现出良好的泛化能力。
2. 为了满足语言模型的规则学习需求，我们精心设计了一套专门的规则数据集。
3. 通过对规则进行编码和嵌入，我们赋予了语言模型高效调用和组合规则的能力。
4. 解决问题时，开发了一种新颖的迭代思路，更接近于人类在解决问题时的逐步分析和逻辑思考过程。
5. 相较于现有的大部分语言模型，我们优化了训练数据量，在解决数值问题上拥有较高的准确性。

1.2 文章结构

本文旨在探索使语言模型具备人类在解决问题时所展现的泛化与组合规则的能力，并开发了一个遵循规则的 MetaRuleGPT 模型，该模型在处理复杂算术任务时展现出良好的准确性。本文的结构安排如下：

第一章，引言。首先阐述了本文的选题背景以及意义。

第二章：相关工作。本章概述了大语言模型的基本概念，详细介绍了 Transformer 模型的核心架构，包含编码器、解码器及注意力机制等关键部分。此外，还探讨了语言模型的预训练方法，元学习神经网络的原理及其在实际应用中的角色，以及大语言模型在数学问题解决和逻辑推理方面的当前进展。

第三章：研究方法。本章首先指出了现有语言模型在数学问题求解方面的泛化能力不足的问题，随后介绍了引入规则学习策略的背景和一些规则数据集的示例。提出了 MetaRuleGPT 模型的架构与预训练方案，并详细描述了模型解决问题的逻辑机制，最后讨论了在模型开发过程中遇到的主要挑战和难点。

第四章：实验设计。本章详细说明了 MetaRuleGPT 模型性能的测试，评估方法。在数学计算和向量叉积运算任务上，我们构建了大量的测试数据集，将 MetaRuleGPT 与现有主流语言模型进行比较。

第五章：结果与讨论。基于实验结果，本章对 MetaRuleGPT 模型的性能进行了详细分析，并就模型在解决问题时的表现、可控性和运作策略等方面进行了深入讨论。

第六章：总结与未来方向。最后，总结了本研究的主要发现和贡献，并对模型的规模扩展、问题细节捕捉能力以及系统性泛化能力的未来发展提出了展望。

2 相关工作

2.1 Transformer 模型

Transformer[6] 模型自 2017 年由 Vaswani 等人在论文《Attention Is All You Need》中提出以来，已经彻底改变了自然语言处理 (NLP) 领域的面貌。它基于自注意力机制 (Self-Attention)，能够在处理序列数据时捕获长距离依赖关系，相较于之前的循环神经网络 (Recurrent Neural Network，RNN)[18,19] 和长短期记忆网络 (Long-Short Term Memory，LSTM)[17]，Transformer 显示出更高的效率和更强的性能。

2.1.1 Transformer 模型的输入和输出

Transformer 模型接受一个序列作为输入，并输出另一个序列，这使得我们能以任何形式的输入输出序列对作为其训练目标。例如，以字母大小写转换为例，序列组可以是：

Input	Output
a, b, c	A, B, C
b, c, d	B, C, D
c, d, e	C, D, E

经过训练，当模型接收到输入序列“a,b,c”时，它将输出对应的“A,B,C”。同样地，若输入“b,c,d”，模型则会产生“B,C,D”。

2.1.2 Transformer 架构

Transformer 完全放弃了传统的循环层，完全依赖于自注意力机制来处理序列数据。它由编码器 (Encoder) 和解码器 (Decoder) 组成，每个部分都包含多个相同的层，每层都有两个子层：一个是多头自注意力机制 (Multi-Head Self-Attention)，另一个是简单的位置全连接前馈网络。此外，Transformer 引入了位置编码 (Positional Encoding)，以保持序列中单词的位置信息。

2.1.3 自注意力机制

自注意力机制是 Transformer 的核心，它允许模型在处理每个序列元素时，考虑到序列中的所有元素，这样模型就可以捕获长距离依赖信息。多头自注意力进一步扩展了这一概念，允许模型在不同的表示子空间中并行学习信息。下面是自注意力机制的详细说明，包括相关的公式。

2.1.3.1 注意力机制是通过以下步骤实现的

1. 输入表示：首先，将输入序列的每个元素 (例如，每个单词或字符) 转换为一个固定大小的向量，通常通过嵌入层 (Embedding Layer) 实现。

2. 计算 **Query**、**Key** 和 **Value**：对于输入序列中的每个元素的向量，通过乘以三个不同的权重矩阵 (分别对应于 Query、Key 和 Value)，生成三组新的向量：Query 向量、Key 向量和 Value 向量。这些权重矩阵是通过模型训练学习得到的。

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V. \tag{2.1}$$

其中 X 是输入向量，W^Q、W^K 和 W^V 是对应于 Query、Key 和 Value 的权重矩阵。

3. 计算注意力得分：接下来，对于每个 Query 向量，计算它与所有 Key 向量的点积，得到一个注意力得分。

$$\text{Attention Score} = QK^T. \tag{2.2}$$

4. 标准化注意力得分：将注意力得分除以 $\sqrt{d_k}$(其中 d_k 是 Key 向量的维度)，然后通过 softmax 函数进行标准化，这样每一列的得分和为 1。

$$\text{Softmax Score} = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right). \tag{2.3}$$

5. 计算加权的 **Value** 向量：将 softmax 标准化后的得分与 Value 向量相乘，使得对于每个 Query，其对应的输出是所有 Value 向量的加权和，权重由注意力得分确定。

$$\text{Output} = \text{Softmax Score} \cdot V. \tag{2.4}$$

6. 多头注意力：为了让模型能够同时从不同的表示子空间学习信息，Transformer 使用了多头注意力机制。它包括将 Query、Key 和 Value 向量分割成多个“头”，然后对每个头分别进行上述步骤，最后将所有头的输出拼接起来，并通过一个线性层。

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O. \tag{2.5}$$

其中每个 head 是：

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V). \tag{2.6}$$

并且 W^O 是另一个权重矩阵，用于将不同头的输出合并成最终的输出向量。

这个过程允许模型在计算每个元素的输出时，同时考虑输入序列中的所有元素及其相互关系。自注意力机制的这种全局视野使得 Transformer 模型特别适用于处理序列数据，如文本和时间序列分析。

2.1.4 应用与进展

Transformer 模型的提出，开启了 NLP 领域的新时代，促成了 BERT[16]、GPT[4] 系列、T5[15] 等一系列基于 Transformer 的模型的出现。这些模型在多种 NLP 任务上取得了突破性的成果，包括文本分类、机器翻译、文本生成和问答系统等。

除了 NLP，Transformer 也被成功应用于其他领域，如计算机视觉中的图像识别、生成图像任务，甚至音频处理领域。例如，Vision Transformer(ViT)[23] 将 Transformer 应用于图像分类任务，展示了与传统卷积神经网络 (Convolutional Neural Network，CNN)[22] 相媲美甚至更好的性能。

Transformer 的成功证明了自注意力机制的强大潜力，也激发了对于模型架构和注意力机制进一步研究的兴趣。目前，研究者们正致力于优化 Transformer 结构，以提高其效率、减少计算资源消耗，并探索其在 NLP 之外领域的新应用。随着研究的深入，Transformer 及其衍生模型有望在人工智能领域继续发挥核心作用。

2.2 大语言模型

大语言模型 (Large Language Models，LLMs) 如 GPT 系列、BERT 和 T5 已成为自然语言处理 (NLP) 领域的革命性力量。这些模型通过在大量文本数据上进行预训练，学习了语言的深层结构和语义，从而能够在多种语言处理任务上实现良好的性能。

2.2.1 GPT 系列

GPT 系列由 OpenAI 开发，从 GPT-1 开始，到最新的 GPT-4，每一代模型都在规模、复杂性和性能上有了显著的提升。GPT 模型使用了 Transformer 架构，特别是自注意力 (Self-Attention) 机制，允许模型捕获文本中长距离的依赖关系。GPT-3，作为其中的佼佼者，通过在数十亿参数上训练，展示了令人惊叹的语言理解和生成能力，能够进行写作、翻译、摘要以及问答等多种任务。

2.2.2 BERT

BERT 由谷歌团队开发，采用了双向训练的方法来更全面地理解语言上下文。与 GPT 主要用于文本生成不同，BERT 优化了文本的理解任务，如文本分类、命名实体识别和问题回答。BERT 的创新之处在于其预训练包括两个任务：掩码语言模型和下一个句子预测，这使得它能够更好地理解文本的双向性质。

2.2.3 T5

T5 将所有 NLP 任务统一为文本到文本的问题，无论是翻译、文本分类还是摘要，都可以被框定为将输入文本转换为输出文本的问题。这种设计简化了不同任务之间的迁移学习，使得模型训练更加高效。T5 也通过大规模预训练，在许多 NLP 任务上取得了先进的结果。

2.2.4 相关工作的进展

随着计算能力的提升和算法的优化，大语言模型的发展速度迅猛。除了上述模型外，还有许多其他重要的模型，如 ERNIE[20]、RoBERTa[21] 等，它们在特定领域或任务上做了进一步的优化和创新。此外，大语言模型的应用范围不断扩大，从基本的文本处理任务到更复杂的情感分析、自动摘要、对话系统和甚至编程助手等领域都有广泛的应用。

总之，大语言模型正成为 NLP 领域的基石，它们的发展不仅推动了人工智能技术的进步，也为人机交互、信息检索和自动化决策等应用打开了新的可能性。目前大语言模型 (LLM) 在自然语言处理领域展现出了优秀的理解能力，特别是在处理语言任务上。通过在广泛且未标记的数据集上进行训练，这些模型在多种任务中展示了出色的泛用性，从而根本性地改变了该领域的研究范式。经过对大量语料库的预训练，语言模型不仅获得了强大的语言理解能力，还拥有了先进的生成能力，使它们在各种基准测试中表现出色，包括但不限于情感分析和机器翻译等任务，都展现了令人印象深刻的性能。

尽管如 ChatGPT 和 GPT-4 这样的先进大语言模型在语言理解和生成领域表现良好，能够精准地解读文本问题并给出准确的答案，它们在可控性能和逻辑问题理解上却遇到了挑战。特别是在处理数学问题时，这些问题往往只有一个确切的答案，数学运算和逻辑推理往往需要极高的精确性，且每个步骤都必须严格遵循特定的规则，这对于依赖模式识别的语言模型来说构成了困难。由于语言模型本质上是基于概率的，旨在提供最可能的结果，这可能导致语言模型在进行数学计算时出现较大的误差。此外，由于语言模型是通过训练大量数据获得的，这也使得它们在可控性方面存在限制，进而在处理某些问题时面临挑战，例如，在处理敏感或禁忌话题时难以精确控制输出内容。

2.3 预训练和微调策略

在自然语言处理 (NLP) 领域，大型语言模型 (LLM) 通过预训练和微调策略展现了显著的性能提升。这一进展不仅极大地推动了 NLP 技术的发展，也为解决实际问题提供了强大的工具。

预训练是指在大规模语料库上训练语言模型以学习通用语言表示的过程。此过程不侧重于特定的下游任务，而是旨在使模型掌握广泛的语言知识和语义理解能力。著名的预训练模型如 BERT 和 GPT 通过这种方式学习了丰富的语言特征，从而为处理更专业化的任务奠定了基础。

微调策略则是在预训练的基础上，将模型应用于特定的下游任务，并对其进行进一步训练的过程。通过在任务相关的数据集上进行微调，模型能够调整其预训练获得的知识，以更好地适应特定任务的需求。这一策略在各种 NLP 任务中表现出色，如文本分类、情感分析、问答系统等。

近年来，研究人员探索了多种预训练和微调技术，以进一步提升模型的性能和泛化能力。例如，XLNet[24] 引入了置换语言建模，通过最大化输入序列所有排列的预测似然来优化模型。RoBERTa 通过移除 BERT 中的 Next Sentence Prediction(NSP) 任务，并在更大的数据集上训练更长时间，显示了更好的结果。

此外，研究还表明，通过适当的微调策略，如渐进式学习率调整和适当的正则化，可以有效减少过拟合的风险，从而提高模型在特定任务上的表现。

预训练和微调策略的成功展示了迁移学习在 NLP 中的巨大潜力。通过从大规模文本数据中提取通用知识，然后适应具体任务，LLM 正推动着自然语言处理技术的边界，解锁了前所未有的应用场景和可能性。随着未来更多创新的预训练和微调方法的出现，我们有理由期待 LLM 将在 NLP 乃至更广泛的人工智能领域中发挥更加重要的作用。

2.4 元学习神经网络实现类人的系统性泛化

元学习，作为机器学习领域的一项关键技术，旨在通过让算法借鉴过往的经验，提高其在学习新任务时的效率与性能。这一方法与传统的机器学习相比，更注重于跨多个任务学习通用的学习策略，以便能够迅速应对新的挑战。人类的语言和思维能力之强大，很大一部分原因是因为其系统的组合性–即能够利用现有的元素进行理解和创新组合。Fodor 和 Pylyshyn[25] 曾指出，人工神经网络缺乏这种组合性能力，因此不适合作为一个有效的思维模型。

尽管此后神经网络技术取得了巨大进步，这一挑战仍旧存在。近期，元学习神经网络实现类人的系统性泛化的研究[1] 成功地解决了 Fodor 和 Pylyshyn 提出的问题，证明了通过优化组合技巧，神经网络能够实现与人类相似的系统性泛化。该研究引入了组合性元学习 (MLC) 方法，该方法通过动态的任务组合流程来指导训练。展示了如何使用 MLC 标准神经网络通过优化其组合技能来模仿或超越人类的系统性概括。MLC 表现出比标准训练方式的神经网络更强的系统性，并且表现出比原始符号模型更微妙的行为。MLC 还允许神经网络解决其他现有的挑战，包括系统性地使用

孤立的原语和使用互斥性来推断含义。此外，MLC 通过元学习获得其能力，其中系统性概括和人类偏见都不是神经网络架构的固有属性，而是从数据中诱导出来的。

MLC 架构采用标准的 Transformer 编码器来处理查询输入及一系列学习示例，并由标准解码器接收编码器的信息以生成相应的输出序列。经过针对不同语法生成的数据集的优化，Transformer 能够在使用冻结权重的情况下执行新任务。

研究通过特定的实验设计，评估人类是如何根据给定指令或指导来学习新任务或概念，并记录他们的行为反应及学习效率和成果。通过与人类行为实验的对比分析，研究显示，与完全系统化但僵化的概率符号模型和完全灵活但缺乏系统性的神经网络模型相比，MLC 模型成功地实现了类人的泛化所需的系统性与灵活性，并在多个系统性泛化的基准测试中提高了机器学习系统的组合能力。

2.5 数学计算与逻辑推理能力

早期研究已经展示了大语言模型 (LLM) 在解决数学计算问题方面的潜力。例如，MetaMath[5] 通过训练模型处理大量算术和符号推理任务数据集，让语言模型能够理解并求解数学问题。这种方法对于简单的数字问题相对有效，但面对更复杂的情形——如数字长度增加时，模型的性能会显著下降。这主要是因为模型更多地是记忆了数学问题的解决方案，而不是真正理解数学的基本规则和算法，这也是当前大部分语言模型普遍存在的问题。

为了应对这一挑战，ComputeGPT 开始尝试将语言模型与外部算术工具进行集成。这种集成方法的优点是，可以借助具有强大计算能力的外部工具来处理任何长度的数字，从而大大扩展了处理能力的范围。在这一过程中，语言模型的主要角色是识别数字和运算符，而真正的计算任务则是由外部工具承担的。同样地，程序辅助的语言模型[2](Program-aided) 引入了一种新颖的自然语言推理方法，通过使用程序作为中间步骤来进行推理，主要依赖外部 Python 解释器来执行中间的求解和计算任务，而不是直接通过大语言模型 (LLM) 进行。尽管这种方法使得语言模型能够处理更为复杂的问题，但核心的计算过程仍然需要依靠外部工具，这意味着它并没有从根本上增强语言模型自身的计算能力。

近期推出的经过优化的语言模型，MathGPT，无需借助计算器工具即可进行算术运算。这一模型拥有 20 亿参数，能够相对准确地完成多位数字的算术运算。它的主要策略是通过大规模训练简单的低位数字计算，从而提升模型在高位数字运算上的准确度。与依赖外部工具的方法不同，MathGPT 致力于增强大语言模型 (LLM) 自身的计算能力。然而，这种方法并没有使模型真正深入理解数字运算的根本原则，而是仍旧依靠喂养大量的计算实例来解决问题。虽然这种方式在一定程度上提升了语言模型的计算能力，但它更多的是一种优化而非根本的改进。我们相信，要想从

本质上增强模型的计算能力，关键在于改变语言模型的基础学习策略。

3 研究方法

为了增强语言模型在解决复杂逻辑推理和数值计算任务等方面的准确性与泛化性，我们引入了 MetaRuleGPT 模型。该模型以增强语言模型的推理能力和泛化潜力为目标，灵感源自于元学习理念。MetaRuleGPT 专注于掌握通用学习策略，旨在通过应用已学习的规则，精确地完成复杂的逻辑推导任务。模型通过动态整合数学计算的基础规则与高阶运算规则，增强了对规则组合的处理能力。这样的设计使得 MetaRuleGPT 模型在面对复杂逻辑推理挑战时，如数学推理问题，能够拥有良好的准确性和泛化能力。

MetaRuleGPT 模型采用了一种全新的方法来处理复杂的算术表达式。通过将迭代策略融入模型架构中，模型在每次迭代中会自动将算术表达式匹配至最适用的规则以进行计算。整个计算过程不是直接一步到位完成的，而是对表达式通过逐步解析并应用组合规则，逐渐计算出最终的结果，模拟了人类解决数学问题的思维过程。

此外，采用这种策略，MetaRuleGPT 模型不限于处理单一任务，而是具备了学习和执行多种不同任务的能力。在处理多任务时，不同任务间的规则可能会存在交叉，我们的模型能够灵活地学习这些交叉规则，并动态应用它们来完成多项任务，且各任务间相互独立，互不干扰，例如，数值的加法运算和向量的叉积运算均需要学习基本的数字规则，我们模型只需学习一次通用的数字规则，即可同时完成两种相互独立的任务，不仅提高了规则的使用效率，同时保证结构的统一性。

3.1 语言模型的泛化能力

大语言模型对于文本具有强大的理解能力，但是其在数值计算任务上却有着很大的局限性。其中一个明显的问题就是语言模型的泛化能力受到很大的限制，例如，在模型学习了一些基本的数值计算问题之后，它仍然不能像人类那样，通过掌握个位数的运算便能灵活应对任意位数的计算任务。因此，我们正致力于增强语言模型的泛化能力，期望模型能够以接近人类的方式对各种问题进行泛化和计算。

我们尝试用当前主流的一些模型进行计算一个简单的高位数加法:

$$6659891948 + 340108052$$

大部分语言模型均无法正确计算这个结果。我们认为想要从本质上改变模型的泛化能力，需要从模型的学习内容以及模型架构上进行调整。

对比人类学习数值计算的方法，当前大语言模型学习的方式无法准确回答原因在于模型并未学会计算的规则，仅仅是记住了一些计算问题的实例，而人类在学习

加法以后，还学习了进位规则，后续将加法规则和进位规则组合起来，从而实现对任意位数的数值问题进行计算。基于上述，我们开始尝试令语言模型学会规则遵循的能力，即类似人类能够学会具体进位的规则。在后续的实验中，我们证明适当的规则遵循确实可以有效的提高模型的泛化能力，从而来提高模型的数值计算能力。

3.2 算术任务的具体规则学习

在我们的研究中，旨在通过两种算术任务来评估模型的逻辑推理与计算能力，其中这两种任务分别是高位数的数值计算和向量的叉积计算。针对这些任务，我们设计了特定的规则训练数据集。例如，在进行加法计算的训练时，模型被引导学习包括个位数加法规则、进位规则、数字映射规则以及基本计算规则等在内的关键知识。通过掌握这些基础规则，经过精心的预训练，我们的模型能够灵活运用并组合这些规则，准确地完成包括高位数加减法在内的复杂数学运算。

在我们的模型成功掌握了高位数值运算之后，我们计划将其能力进一步拓展至向量的叉积运算。在处理向量叉积运算时，模型需完成一系列复杂的推导过程。通过逐步的推导，融合叉积的计算规则以及基础数值运算规则，以此来赋予模型执行向量叉积计算的新能力。为了达成这个目标，我们尝试将向量表示以及叉积运算等规则重新引入到我们已经掌握数值运算能力的预训练模型 MetaRuleGPT 中。通过学习这些新规则，模型能够将其与既有的数值运算规则相结合，进而实现向量叉积的精确计算。此方法体现了模型通过学习并融合多样规则，解决更复杂的数学运算挑战，展现出其逻辑推理和泛化的能力。

我们以表格的形式详细展示了完成各项数学计算任务所需学习具体规则，表3.1展示了在执行相应数学任务时所需的规则种类数。

表 3.1 各任务学习规则汇总表

Train Rule Type	Numerical Addition	Numerical Subtraction	Vector Cross Product
Vector Table	-	-	√
Nine Addition Table	√	-	√
Nine Subtraction Table	-	√	√
Nine Multiplication Table	-	-	√
Mapping Rule	√	√	√
Carrying Rule	√	-	√
Borrowing Rule	-	√	√
Vector Product Rule	-	-	√
Compute Rule	√	√	√

3.2.1 算数训练数据集

我们精心设计了用于训练的计算规则数据集，它覆盖了从最基本的个位数算术任务到各种复杂算术规则的广泛范围。该数据集经过周密的规划，涵盖了多种算术

操作，包括对齐规则、进位规则、借位规则、基础计算规则以及组合规则等。我们构建的这个数据集大约包含 2 万条记录。

在这个数据集中，每个算术表达式包含了 2 到 10 个操作步骤，涉及了一系列的数学计算操作，如加法 (+)、减法 (−) 以及向量叉积运算 (×)。这样的设计旨在为模型提供一个全面且多样化的数学操作学习环境。

在这些数据集中，我们训练的算术表达式中仅包含了最基础的个位数运算，如九九加法表和九九减法表等简单计算。此外，数据集还包含了一系列细致的数学计算规则，这包括数字对齐规则、进位规则、退位规则、叉积规则，以及数字映射规则等。这样的设计旨在为模型提供一个坚实的基础，以便模型能够掌握从基本到复杂的数学运算所需的关键规则的组合能力。

以 k 进制下 n 位数的加法为例，我们可以初步估计所需的训练数据量如下：

1. 基本加法表数据: $O(k^2)$
   k 进制的个位数加法计算一共需要 $k \times k$ 个基本计算规则。

2. 对齐规则数: $O(n^2)$
   最高位数为 n 的加法计算中，最多有 $n \times n$ 个对齐规则。

3. 进位规则数: $O(2^n)$
   加法计算过程中，每一位只有两种可能，即需要进位和不需要进位，因此对于 n 位数加法计算，最多需要 2^n 个规则。

4. 计算规则数: $O(n)$
   我们在对每一位进行计算时，最多需要同时计算 n 位，因此最多需要 n 个规则。

5. 映射规则数: $O(1)$
   对于每一个数值问题的输入，映射规则是一致的，将数字映射成英文字符，对于固定长度的输入，只需有限个规则。

因此总计规则数大约为

$$O(2^n + n^2 + k^2), n \geq 10, k \geq 2. \tag{3.1}$$

而当前主流语言模型想要准确实现 k 进制下的 n 位数的加法所需要的训练数据量是:

$$O(k^{n+n}), n \geq 10, k \geq 2. \tag{3.2}$$

相比之下，我们的数据训练数据量是远小于当前主流模型训练的数据量的。

图 3.1呈现了部分规则训练数据集的示例内容：

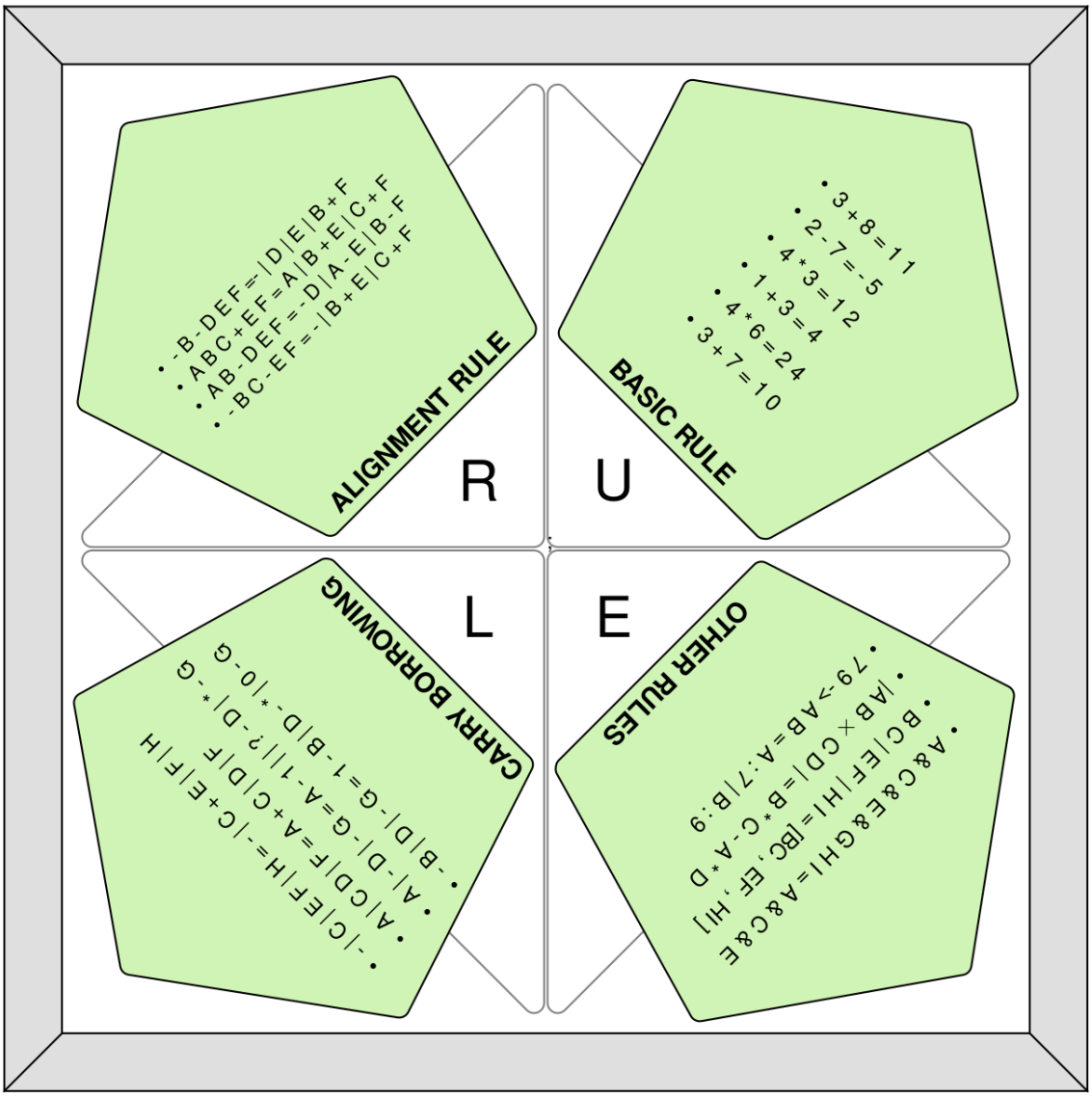

图 3.1 部分训练数据集展示

3.2.2 模型参数和训练

为了验证我们的研究假设，我们构建了一个基于 Transformer 架构的模型。考虑到我们在数值实验中所处理的词汇类别相对较少，我们选择了 Word2Vec 词嵌入技术。Word2Vec 能够将单词转化为向量，这些向量在多维空间中捕捉并表示单词的语义特性。使用 Word2Vec 的优势包括：

1. 语义信息捕捉：这一特性使得模型能够识别并理解单词之间的相似性和关系。
2. 密集向量表示：与传统的 one-hot 编码相比，Word2Vec 提供了更为紧凑的向量表示，显著降低了数据维度的同时保留了丰富的信息。
3. 增强泛化能力：通过学习到的词向量，模型能够对新的、未知的单词进行有效推断，提升了泛化能力。
4. 简化模型复杂度：低维的词向量有助于降低后续模型处理的复杂度，从而提高训练效率。

基于这些考虑，我们利用 Word2Vec 技术将数字和规则表达式映射至 256 维的实数向量空间，这样的词嵌入设计有助于深入探究数字与规则之间的深层联系。

为了评估不同参数量的 Transformer 模型性能，我们采用了统一规模的训练数据集对一系列模型进行训练。具体而言，我们测试了三种不同参数规模的模型：最大的模型配置了 100M 参数; 紧随其后的是 30M 参数的模型; 最小的模型则拥有 10M 参数。尽管这些模型的参数量各不相同，它们都在相同条件下，使用包含逾 2 万条训练数据的同一数据集进行训练，以保证实验的公平性和一致性。

考虑到我们的训练数据集规模相对较小，我们发现经过大约 3000 次训练迭代，我们的预训练模型就能在原始数据集上展现出优良的性能。因此，不论模型大小，我们均对所有模型进行了 3000 次的训练周期。具体模型参数如下表所示:

表 3.2 模型参数规模对比表

MetaRuleGPT Model	Dimension	Batch size	Heads	Layers	Parameters	Trainning Steps
MetaRuleGPT-10M	256	10	16	5	10M	3000
MetaRuleGPT-30M	256	30	16	15	30M	3000
MetaRuleGPT-100M	256	100	32	20	100M	3000

3.3 模型架构

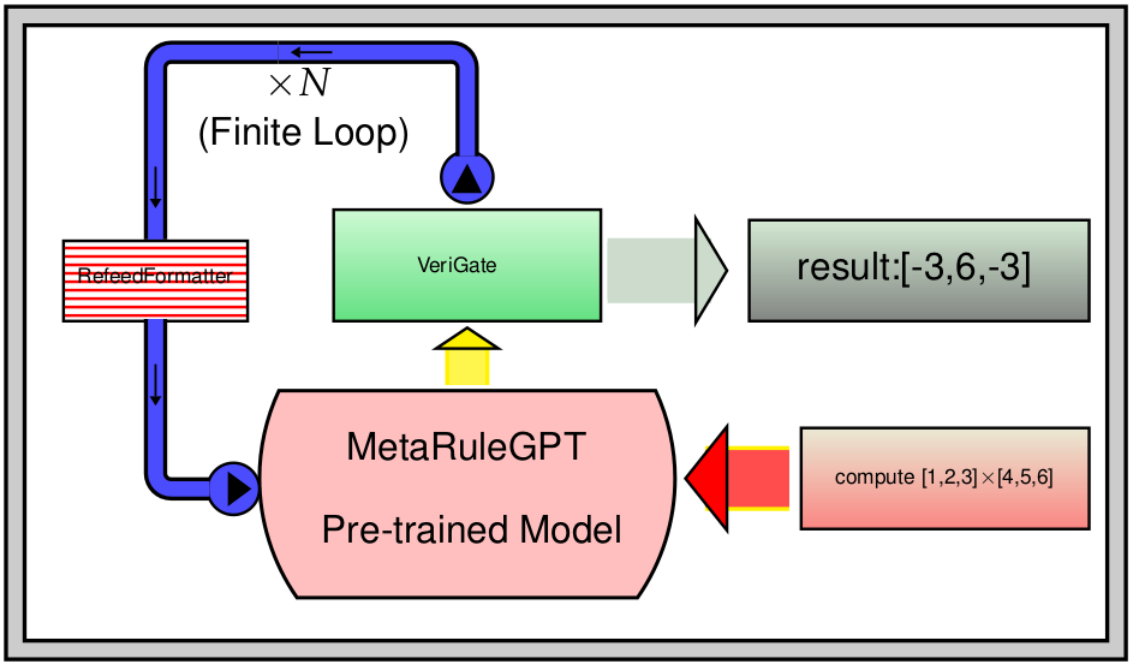

图 3.2 MetaRuleGPT 模型架构图

3.3.1 MetaRuleGPT 模型结构

为了更贴近人类解决数学问题的自然过程，我们不是直接求解每一个复杂的算术表达式，而是采取了一种迭代且分步的策略。通过这种方式，我们的模型将复杂的表达式分解为一系列更简单和基本的计算步骤，逐步推理出最终答案。这种方法使得语言模型在学习过程中对特定规则有了更深入的理解和更有效的应用，从而在解决问题时能够灵活地组合和应用这些规则。我们模型在数学计算任务上表现良好，这主要归功于它掌握了计算的核心规则，而非简单地依赖于对特定案例的记忆。

在专注于算术任务的同时，我们开发了基于 Transformer 架构的语言模型，其旨在解决数学问题，我们将其称为 MetaRuleGPT 语言模型。如图 3.2所示的模型架构包括几个关键组件：MetaRuleGPT 预训练模型、格式化工具 (RefeedFormatter)，验证门 (VeriGate)。我们设计了一种自我迭代的方法，使模型能够通过对复杂问题的不断迭代，化繁为简，最终在有限的步数内获得正确的答案。

各个组件的具体功能如下所述：

1. MetaRuleGPT 预训练模型：该模型是一个经过数值和规则深度学习优化的预训练模型，它能够利用组合规则对输入的问题进行精确的单步逻辑分析，并生成相应的处理结果。

2. RefeedFormatter：该工具负责将中间输出进行格式转换，并将转换后的结果作为下一轮 MetaRuleGPT 预训练模型的输入。它的作用主要在于实现输入和输出格式之间的转换。

3. VeriGate：此组件的任务是辨识我们预训练模型的输出是中间结果还是最终结果。为了达成此目的，我们在最终输出中引入了一些不常用的 Unicode 字符作为结束标识符。通过检测输出中是否含有这些 Unicode 字符标识，我们可以判断该输出是否应被视为最终结果，或需要被重新送回预训练模型进行更深层次的处理。

3.3.2 MetaRuleGPT 预训练模型

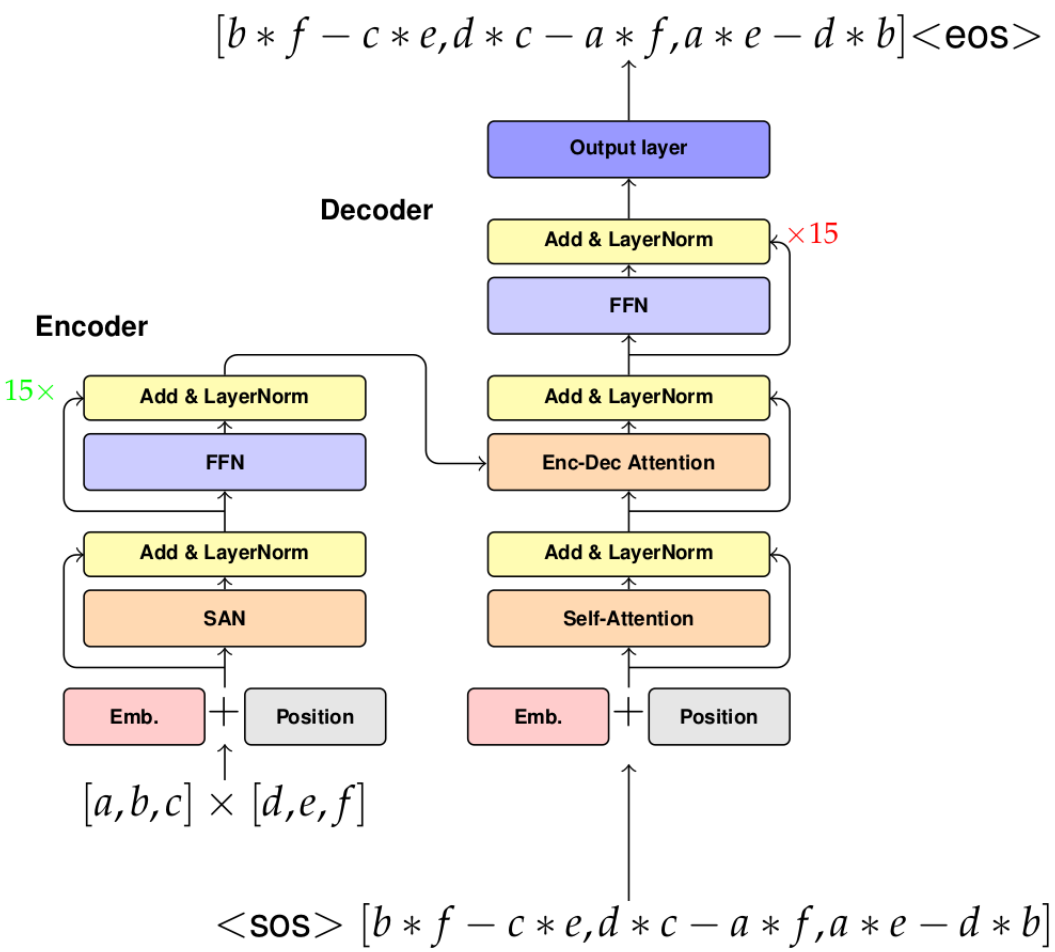

图 3.3 MetaRuleGPT 预训练模型

如图 3.3所示，我们采用了基于 Transformer 架构的语言模型对数据集进行了训练。为了更灵活地调整模型参数规模和内部结构，我们设计并实现了一个自定义的

Transformer 模型。在模型设计阶段，我们选择了配置有 16 个注意力头和 15 层编码器和解码器的模型，并且特别选用了精心挑选的位置编码和词嵌入策略。鉴于我们面临的问题并不涉及庞杂的词汇量，我们采取了基于单字节的训练方法。这种训练策略与传统的基于词汇或字符的方法相比，具有明显的优势和重要性。

基于字节的语言模型为处理多语言文本和未知字符提供了一种灵活而有效的手段。正如图 3.3所示，这是我们利用 Transformer 模型训练向量叉积计算规则的示例。通过逐个字符的处理方式，可以确保模型对规则的学习更为准确，为解决复杂的逻辑任务打下了坚实的基础。

3.3.3 MetaRuleGPT 模型运行策略

对于语言模型而言，面对如 “123 + 456” 这类字符串序列时，它并不能直接进行两个数值相加的操作。这是因为语言模型对每个字符进行单独嵌入处理，赋予每个字符平等的意义，没有区分加号和数字的不同。为了解决数值计算问题，我们认为语言模型需要首先识别问题，然后理解问题，最后解决问题。因此，我们设计的模型主要按照以下几个步骤操作：

1. 数字识别与对齐：借助对大规模数据集的学习，我们的语言模型已经发展出了基本的识别能力。结合模型掌握的对齐规则，这两项能力的结合使得模型能够有效地执行所需的数字对齐操作。

2. 进行位数计算：在识别每个字符的含义之后，接下来的步骤是进行位数的具体计算。

3. 处理进位或借位：根据语言模型的识别结果，如果计算过程中需要进位或借位，相应的操作会被执行。

4. 输出结果：完成上述步骤后，模型将输出最终计算结果。

通过对文本的识别和采用相应的规则进行计算，经过多次迭代后，我们的模型能够得出准确的计算结果。

在计算过程中，我们遇到的主要挑战是确保语言模型能够精确地识别并对齐数字以及它们的运算符。例如，当模型遇到字符串 “123 + 456” 时，我们希望它能正确识别出这是两个三位数进行加法运算的情况。目前大部分模型所采取的解决方案是

枚举所有可能的场景，即模型需要训练如下 10^6 个数据集：

$$\begin{gathered}\text{“}000+001=001\text{”},\\ \text{“}000+002=001\text{”},\\ \vdots\\ \text{“}999+998=1997\text{”},\\ \text{“}999+999=1998\text{”}\end{gathered}$$

遵循传统方法，在只处理正数和不考虑更高位数的情况下，仅为了使模型学会三位数与三位数之间的加法运算，就需要大约一百万个训练样本。此外，经过这种方式训练，语言模型还面临着难以将学习成果泛化到更高维度数值计算的问题。尽管大多数当前主流模型仍采用这种训练方法，其性能优化主要依靠监督学习的中间步骤，并没有从根本上解决这些问题。为了克服这些限制，我们采用了一种不同的策略，从根本上改变训练内容和模型架构，从依赖大量数据的学习模式转变为基于规则的学习策略进行细微调整。通过学习几种关键规则，我们的模型能够更加高效地处理像 “$123+456$” 这样的算术问题。

我们应用映射规则，将每个数字定位到其特定位置，从而让模型能够识别并统一处理此类结构，增强了模型对位数的识别和处理能力。如

$$\begin{gathered}123+456\\ \downarrow\\ a:1,b:2,c:3,e:4,f:5,g:6\end{gathered}$$

在模型清晰识别出每个字符所承载的意义后，语言模型就能够启动对齐规则，以便进行后续的处理步骤。

$$abc+efg=a+e|b+f|c+g. \tag{3.3}$$

当模型成功识别出 a,b,c 与 d,e,f 之间的对应关系后，它就能够对字符串 “$123+456$” 进行数字与符号的分离处理。即

$$\begin{gathered}a:1|b:2|c:3|+|e:4|f:5|g:6->a+e|b+f|c+g\\ \downarrow\\ 1+4|2+5|3+6\end{gathered}$$

至此，我们成功地对字符串进行了识别和对齐，即

$$123+456\rightarrow 1+4|2+5|3+6. \tag{3.4}$$

当然，我们的模型的能力不止于识别和对齐上述示例中的字符串; 它可以对任何具有类似结构的字符串进行相似的识别和对齐操作。

接下来，我们需要着手解决位数计算的问题，即使语言模型在遇到以下类型的字符串时，也能进行有效的识别和计算。

$$1+4|2+5|3+6. \tag{3.5}$$

在我们的模型识别并标记出字符串中的数字后，它将运用掌握的个位数加法规则对问题进行解算，对于带有特定字母标记位置的字符串进行求和，通过组合数字识别与加法计算规则的能力，以获得期望的结果，如下所示:

$$\begin{gathered} 1+4|2+5|3+6->a+c|e+g|i+k \\ \downarrow \\ a:1+c:4|e:2+g:5|i:3+k:6 \\ \downarrow \\ 5|7|9 \end{gathered}$$

至此，我们通过两个步骤，利用规则组合的方法完成了字符串中数字表达式的计算:

$$123+456=579. \tag{3.6}$$

经过计算之后，模型会判断输出是否符合我们的要求，若满足，则对结果进行标记，并将其作为最终输出; 反之，则继续进行计算，直到通过迭代得到最终的结果。

对于上述问题，仅需两步便可得出最终结果。然而，对于需要进位操作的大多数问题，我们的处理逻辑仍然遵循先前的对齐和计算方法。具体的处理流程和示例可参见图 3.4。

3.3.4 MetaRuleGPT 模型计算示例

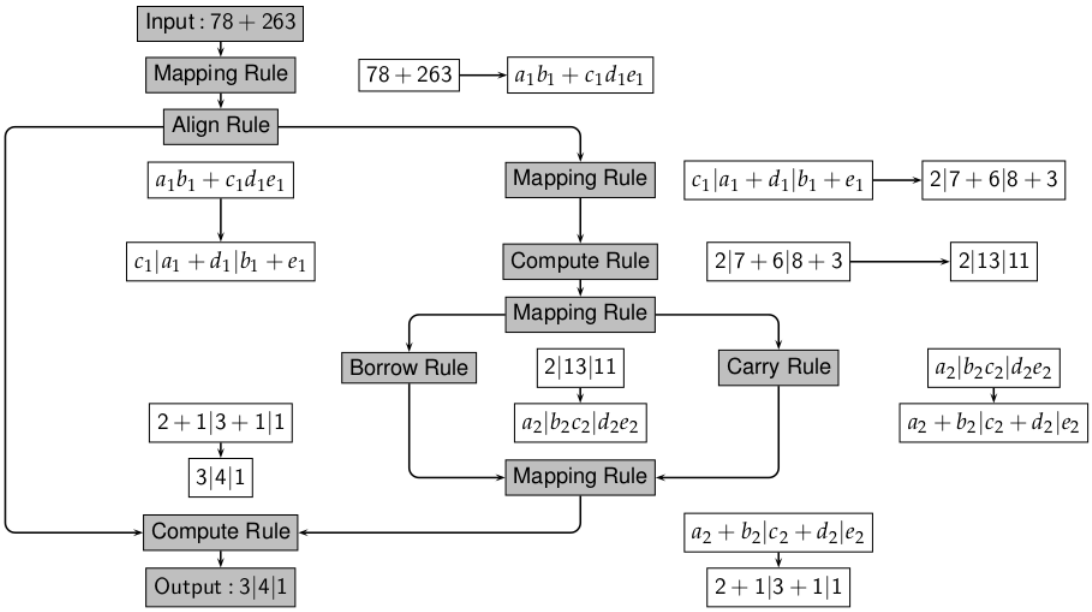

图 3.4 MetaRuleGPT 语言模型计算示例图

图 3.4中展示了我们模型的内部运行原理。为清晰展示模型的工作方式，我们借助一个简单的加法示例进行详细阐述。当向模型输入“Input: $78 + 263$”时，该输入将依次经过映射规则 (Mapping Rule)、计算规则 (Compute Rule)、对齐规则 (Align Rule) 和进位规则 (Carry/Borrow Rule) 等，逐步推导出计算结果。图 3.4阐释了如何将初始输入转化为最终结果的过程。

1. 首先模型对我们输入的问题进行结构化处理，其中 “$78 + 263$” 经过映射规则 (Mapping Rule) 会得到：

$$a_1 : 7, b_1 : 8, c_1 : 2, d_1 : 6, e_1 : 3. \tag{3.7}$$

“$a_1b_1 + c_1d_1e_1$” 经过对齐规则 (Align Rule) 得到：

$$c_1|a_1 + d_1|b_1 + e_1. \tag{3.8}$$

通过对齐，映射规则的组合调用得到了我们的中间输出:“$2|7 + 6|8 + 3$”。

2. 同理对于 “$2|7 + 6|8 + 3$”，组合调用映射规则 (Mapping Rule) 和个位数加法计算规则 (Add Sub-rule)，即可得到中间输出结果 “2|13|11”

3. “2|13|11” 作为最新的输入时，模型将会调用进位规则 (Carry Rule)，以及映射规则 (Mapping Rule)，执行数字进位操作，进而产生中间输出 “$2 + 1|3 + 1|1$”

4. “$2+1|3+1|1$” 将作为新的输入，再次应用映射规则 (Mapping Rule) 和计算规则 (Compute Rule)，便可获得最终的计算结果 “$3|4|1$”

5. “$3|4|1$” 作为最后的输入阶段，我们的模型将调用格式化规则并采用特殊符号进行标记。最终通过使用 VeriGate 进行格式化处理，从而输出计算结果:“Output:341”

简言之，在计算过程中，MetaRuleGPT 模型通过规则的组合运用，进行对齐，进位等操作并输出最终结果，即有:

$$\begin{gathered}\text{Input:}78+263\\ \downarrow\\ 2|7+6|8+3\\ \downarrow\\ 2|13|11\\ \downarrow\\ 2+1|3+1|1\\ \downarrow\\ 3|4|1\\ \downarrow\\ \text{Output:}341\end{gathered}$$

3.3.5 计算规则 (Compute Rule)

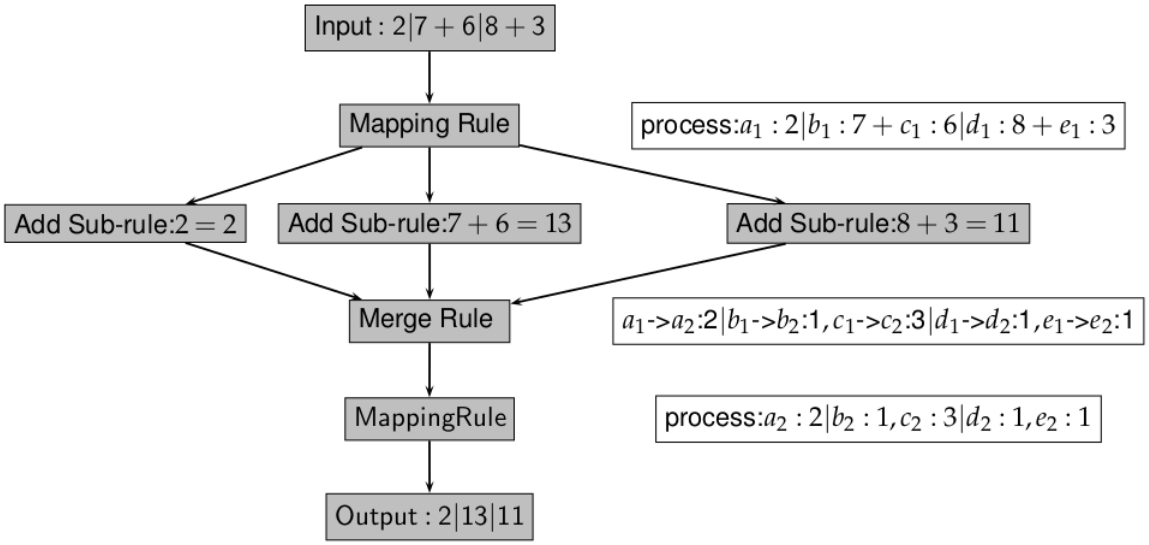

图 3.5 Compute Rule 机制图

计算规则 (Compute Rule) 是一种复合规则，其机制如图 3.5所展示，对于模型输入 “2|7 + 6|8 + 3”，我们首先需要识别并分离出各个部分:“2”，“7 + 6”，“8 + 3”，并对它们分别进行计算。具体来说，我们模型可以识别出每个位置，并按照顺序进行标记，即有

$$a_1 : 2, b_1 : 7, c_1 : 6, d_1 : 8, e_1 : 3. \tag{3.9}$$

通过对 “$a_1|b_1 + c_1|d_1 + e_1$” 进行分离并组合调用个位计算规则 (Add Sub-rule) 以及映射规则 (Mapping Rule)，即有

$$a_1 : 2 = a_2 : 2, \tag{3.10}$$

$$b_1 : 8 + c_1 : 3 = b_2 : 1, c_2 : 1, \tag{3.11}$$

$$d_1 : 7 + e_1 : 6 = d_2 : 1, e_2 : 3. \tag{3.12}$$

对中间输出调用合并规则 (Merge Rule) 处理，得到

$$a_2 : 2|b_2 : 1, c_2 : 3|d_2 : 1, e_2 : 1. \tag{3.13}$$

经过映射规则 (Mapping Rule) 处理，模型完成了对字符串的计算，并输出

Output:2|13|11

简言之，模型采用标记技术来识别并分离出各个部分，运用已习得的基本加法子规则，对各部分进行数值运算，进而合并得到中间计算结果，并输出。

3.4 模型构造的重点和难点

3.4.1 重点:

在模型开发过程中，我们遇到了许多挑战。语言模型其内部的工作机制往往被视为一个黑盒。为了增强模型对数字的理解能力，我们必须深入探索这个黑盒的内部。我们依据模型的已知架构进行推理，尝试使模型能够恰当地运用它所学习到的规则并进行组合。通过对大规模规则数据集的学习，模型掌握了各种规则，能够在面对逻辑推理问题时，将问题进行归类以后，严格遵循规则推理，逐步剖析问题，将复杂问题简化，从而找到解决方案。通过这种方法，我们不仅严格监控了解决问题的每一步骤，也将原本的黑盒模型转化为更透明的“白盒”，从而更精确地控制模型的能力。

3.4.2 难点:

我们的语言模型在处理数值问题时，遇到的主要困难包括以下几点: 首先，模型对字符串的解析能力有限，我们选用多字符时，无法掌控模型对单个字符的识别，因此我们通过选择单字符词嵌入的方式，促进模型对各个字符的理解。

其次，目前尚不清楚语言模型是否能够真正理解数字。因此，我们通过让语言模型对已有字符串进行标记和识别的方式来解析字符串以此达到对统一格式的字符串进行结构化理解。

最后对于不同的问题所需的中间步骤数是不确定的，比如有些问题需要进位，有些不需要，为了克服这些障碍，我们通过迭代输入，并在过程中让模型对特定输出进行特殊标记，来为后续判定重新作为输入，亦或作为结果输出。

我们采用了规则学习方法。通过仅学习基础的个位数加法规则，并逐步引入其他规则，MetaRuleGPT 模型经过学习掌握本质规则后，拥有了一定的逻辑推理能力，从而更好地理解和解决了数字相关的问题。

4 实验

4.1 实验设定

为了展现我们的模型 MetaRuleGPT 在推理任务中的准确性和泛化能力, 我们精心设计了两项实验：数值算术任务和向量叉积计算任务。这些实验不仅测试了模型的基本计算能力, 也考察了其解决复杂问题的能力, 为全面评估模型在逻辑推理方面的性能提供了坚实的基础。同时, 为了进一步证明 MetaRuleGPT 的优势, 我们将其与当下几款知名的大型语言模型, 其中包括阿里巴巴的 QWen、谷歌的 Palm、Llama2, 以及最新且极具实力的 ChatGPT-3.5 和 ChatGPT-4.0 等, 进行了比较, 来佐证 MetaRuleGPT 模型的计算能力。

4.2 测试数据集

当前的大型语言模型在处理逻辑严密的数学问题时存在一定局限性, 这部分原因可能归咎于对数学逻辑深层次理解的不足。与此相反, 我们开发的模型从数学的基本原理出发构建, 因此在解决数学算术任务方面展现出更高的精确度。为验证这一优势, 我们精心设计了多种不同的测试数据类型, 并准备了大量的测试数据集。这样做的目的是为了突显我们模型在数学推理能力方面的优势。通过这一系列的验证, 不仅能证明我们的模型能够掌握并运用数学的基础逻辑运算, 还展示了其在问题解决过程中展现出一定的泛化能力。

在算术任务领域, 我们构建了一个包含广泛算术操作的多样化训练数据集。为了全面评价我们模型的计算准确性和泛化能力, 我们设计了一个包含 8000 个测试案例的评估数据集, 这些案例与训练集完全不重叠。该数据集覆盖了多种数值运算类型, 包括但不限于完美十进制加法、反向幅度减法、错位减法, 以及基于随机生成数的加减运算等。下面是每类数据集的具体说明:

1. 完美十进制加法 (Perfect Decadic Addition)

   示例:$6659891948 + 340108052$

   这类加法算式特点在于需要连续多次进位操作：从个位开始, 每向前进一位, 数值恰好累加至十, 因而触发连续的进位。这样的计算过程涉及到多次隐式的进位计算, 我们将这种特殊的加法过程称为“完美十进制加法”。

2. 反向幅度减法 (Reverse Magnitude Subtraction)

   示例:$62103 - 2386797965$

这类减法算式的特征在于它要求多次连续借位, 同时还需要判断最终结果的正负性。在进行这类减法时, 从左至右计算, 每一位数字都需借位以求差, 且在这些算式中, 减数明显小于被减数。我们将这类需要连续借位和具有结果正负判断需求的减法称作“反向幅度减法”。

3. 错位减法 (Interleaved Subtraction)

$$\text{示例:} 1824453209 - 482835016$$

在这类减法算式中, 减数与被减数的每个位数上的数值大小呈现循环交替的形式。具体来说, 如个位上减数较大, 而在十位上被减数则更大, 这种模式在更高位次上会继续交替出现。计算这种算式时, 不仅需要进行恰当的借位或进位操作, 还必须对最终结果的正负性进行准确判定。我们将这种特征为各数位上数值大小循环交替, 且结果正负不定的减法过程称为“错位减法”。

4. 随机生成数 (Random Number Generation)

$$\text{示例:} 465584019 - 1142850735$$

为了测试我们模型在数值计算方面的广泛适用性, 我们利用 Python 生成了大量随机数, 以此作为评估模型在数值计算领域性能的平均基准。

5. 向量组叉积测试数据集 (Vector Cross Product)

$$\text{示例:}(6,5,7) \times (9,3,7)$$

为了测试模型解决复杂数学计算问题的能力, 我们选择在三维空间中随机生成 400 组向量, 对每组向量进行叉积运算.

我们精心准备的这个数据集, 作为一个全面的评价标准, 旨在量化地衡量 MetaRuleGPT 模型在解决算术任务上的性能表现。以下表格展示了在各类测试集中的部分案例。

表 4.1 部分测试数据展示表

Data Type	Test Dataset Examples
Randomized Procedure	$6729132856 + 1854307391, \ldots, 1554887316 - 817095695$
Perfect Decadic Addition	$6659891948 + 340108052, \ldots, 4376628072 + 623371928$
Reverse Magnitude Subtraction	$62103 - 2386797965, \ldots, 53006 - 7764286617$
Interleaved Subtraction	$1824453209 - 482835016, \ldots, 8858241744 - 261714262$
Vector Cross Product	$(6,5,7) \times (9,3,1), \ldots, (8,2,0) \times (6,4,9)$

4.3 评估指标

在评估最终计算结果时，我们不仅考虑了模型的计算结果与真实答案之间的匹配度，即准确率 (Accuary)，还关注了计算结果与正确答案之间的误差，为此，我们定义了标准化最大误差。从理论上讲，标准化最大误差越小，表明模型的计算结果越接近真实值，这意味着通过适当调整模型参数，有更大的可能性提升模型的准确性。相反，当标准化最大误差大于 1 时，这表明该模型在处理此类问题时缺乏可控性或未能展现出解决这些问题的能力。

假定正确计算的问题数为 TP，问题总数为 N，那么准确率可定义为：

$$\text{Accuracy} = \frac{TP}{N} \times 100\% \tag{4.1}$$

假设模型计算结果是 y，实际计算结果为 $\hat{y}$，一共有 N 组结果，那么标准化最大误差可定义为:

$$\text{Normalized Maximum Error} = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{y_i - \hat{y}_i}{\max(y_i, \hat{y}_i)} \right| \tag{4.2}$$

4.4 语言模型深度数值优化实验

为了检验 MetaRuleGPT 模型在数学上推理能力以及泛化能力，我们使用了当前知名的语言模型进行对比，如阿里巴巴的 QWen、谷歌的 Palm、Llama2，以及目前非常强大的 ChatGPT-3.5 和 ChatGPT-4.0 等。通过这样的比较，我们可以全面了解不同模型之间性能差异，并评估我们的模型在数学推理任务上的表现。

我们选择了参数大小为 30M 的 MetaRuleGPT 模型进行对比实验，将先前整理的各类测试数据集合分别使用上述各类大语言模型进行测试，并将各个模型的计算结果进行保留统计，并比对结果。我们进行了一系列详细的实验和评估，得到了各类语言模型在各项任务测试数据集上的计算结果。

表 4.2展示了各模型对完美十进制加法的测试结果：

表 4.2 完美十进制加法表

Compute Digits	5-digit		10-digit	
Rate	Error	Accuracy	Error	Accuracy
GPT-4	**0.0**	100%	2.9e − 09	98%
GPT-3.5	4.2e − 05	97.2%	2.0e − 4	91.5%
Llama2-7b	15	13.4%	30	5%
Llama2-13b	0.16	3.6%	10	3%
Llama2-70b	0.14	5%	1.8	6%
Google-PaLM	0.89	52.6%	24	27.5%
Qwen-72b-Chat	0.27	85.8%	0.21	67.5%
MetaRuleGPT	**0.0**	**100%**	**0.0**	**100%**

表 4.3展示了各模型对反向幅度减法的测试结果：

表 4.3 交错式位值减法表

Compute Digits	5-digit		10-digit	
Rate	Error	Accuracy	Error	Accuracy
GPT-4	0.016	98.3%	5.368e − 07	96%
GPT-3.5	0.0033	95.2%	0.037	91%
Llama2-7b	0.64	2.3%	0.92	0%
Llama2-13b	0.52	21.9%	0.69	2%
Llama2-70b	0.061	76.1%	0.79	2%
Google-PaLM	0.0076	95.9%	0.54	19%
Qwen-72b-Chat	0.0092	93.9%	0.0027	74.5%
MetaRuleGPT	**0.0**	**100%**	**0.0**	**100%**

表 4.4展示了各模型对错位减法的测试结果：

表 4.4 反向幅度减法表

Compute Digits	5-digit		10-digit	
Rate	Error	Accuracy	Error	Accuracy
GPT-4	0.027	97.8%	1.3e − 08	96.5%
GPT-3.5	0.0033	99.4%	8.3e − 04	88.5%
Llama2-7b	0.64	20.8%	2.1	0.0%
Llama2-13b	0.52	4%	1.5	0.0%
Llama2-70b	0.061	50.2%	$4.6e-06$	0.5%
Google-PaLM	0.0076	43.6%	1.0	0.0%
Qwen-72b-Chat	0.0092	86.4%	0.065	3.5%
MetaRuleGPT	**0.0**	**100%**	**0.0**	**100%**

表 4.6，4.5分别展示了各模型对随机生成数的测试结果：

表 4.5 对比实验-随机减法表

Compute Digits	5-digit		10-digit	
Rate	Error	Accuracy	Error	Accuracy
GPT-4	**0.0**	100%	0.019	78.5%
GPT-3.5	4.9e − 05	95.5%	0.052	76.5%
Llama2-7b	1.4	25.5%	2.1	3%
Llama2-13b	0.49	31%	0.67	1%
Llama2-70b	0.085	73.5%	1.5	12.5%
Google-PaLM	0.15	80.5%	0.68	47%
Qwen-72b-Chat	2.0e − 4	93.5%	**0.0017**	81.5%
MetaRuleGPT	**0.0**	**100%**	0.063	**88%**

表 4.6 对比实验-随机加法表

Compute Digits	5-digit		10-digit	
Rate	Error	Accuracy	Error	Accuracy
GPT-4	**0.0**	100%	0.092	85.5%
GPT-3.5	1.6e − 05	99%	0.046	72%
Llama2-7b	0.7434	49.5%	6.6	0.5%
Llama2-13b	0.5075	28.5%	0.47	2%
Llama2-70b	0.0165	84%	0.64	11%
Google-PaLM	0.0017	94%	0.34	39.5%
Qwen-72b-Chat	0.0005	97%	0.0082	75%
MetaRuleGPT	**0.0**	**100%**	**0.0**	**100%**

4.5 语言模型驱动的向量叉积计算实验

为了展示我们模型在处理复杂逻辑问题上的能力，我们精心挑选了数学中较为复杂的向量叉积计算作为测试案例。通过这项测试，我们不仅验证了模型的精确计算能力，还将其与当前领域最领先大语言模型进行了准确度对比。

表 4.7详细展示了不同模型在执行向量叉积计算的数据集上的准确性比较结果：

表 4.7 向量外积计算表

Vector Compute	Cross Product
GPT-4	17%
GPT-3.5	5.5%
llama2-7b	-
llama2-13b	-
llama2-70b	0%
Google-PaLM	0%
Qwen-72b-Chat	23%
MetaRuleGPT	**98.5%**

5 结果和讨论

5.1 结果

表 5.1 语言模型在数值计算任务上的平均测试结果

Model	Model Parameter	5-digit	10-digit
GPT-4	100000B	99.22%	90.9%
GPT-3.5	175B+	97.26%	83.9%
Llama2-7b	7B	22.3%	1.7%
Llama2-13b	13B	17.8%	1.6%
Llama2-70b	70B	57.76%	6.4%
Google-PaLM	110B	73.32%	26.6%
Qwen-72b-Chat	72B	91.32%	60.4%
MetaRuleGPT	30M	**100%**	**97.6%**

我们开发的基于 Transformer 架构的语言模型 MetaRuleGPT, 拥有 300 万参数, 专为算术任务的优化和推理而设计。我们将 MetaRuleGPT 与当前领先的大型语言模型（LLMs）, 如 GPT-4 和 ChatGPT 等进行了性能对比。

尽管 MetaRuleGPT 的规模相对较小, 参数量仅为 300 万, 但其在算术任务上的表现出乎意料地超越了参数规模远大于它的 GPT-4 和 ChatGPT 等模型。这一结果明确展现了 MetaRuleGPT 在处理算术问题上的能力领先于所有其他测试模型。

这一成果突出了 MetaRuleGPT 采用的组合规则方法的高效性。该模型通过分解复杂的算术表达式为一系列中间步骤, 并通过组合学习到的多种规则来深入理解算术任务, 从而掌握了算术运算的核心规则与逻辑, 提供了更准确的答案。

从表 5.1中可知,MetaRuleGPT 在特定的算术任务评估中展现了良好的性能, 优于当前 NLP 领域上的主流大型语言模型（LLMs）。这一结果不仅表明了 MetaRuleGPT 在解决特定数值计算挑战及常规计算问题上相较于现有主流模型有更高的准确度, 还证明了采用组合规则学习方法的有效性。

此外, 我们期望我们的模型不仅展现出精确的计算能力, 更希望它具备一定的泛化能力。一个理想的模型应该能够从少量的示例中, 通过系统化的方式学习和应用规则, 并偏向于理解结构化输入与输出之间的关系。我们的模型正是依托这样的泛化能力, 来克服先前遇到的系统性限制, 以此来应对未知的任务。这种策略旨在模拟人类的组合技能。经过初步学习个位数加法的基础上, 我们的模型能够通过规则的组合处理, 计算从未遇见的高位数问题, 并且在处理高位数计算任务时，其准确性超越了大多数现有的语言模型。

我们开发的 MetaRuleGPT 模型旨在深入探索语言模型在逻辑和数学推理问题

上的泛化能力。通过我们进行的一系列实验, 不论是针对特定数据集还是随机数据集,MetaRuleGPT 都展现了良好的性能。虽然在低位数数值计算任务上, 我们的模型与当前主流模型都达到了较高的准确率, 但在处理高位数计算任务时,MetaRuleGPT 的表现明显优于大多数现有语言模型。此外,MetaRuleGPT 在向量叉积的计算上也展示了优异的能力。不仅如此, 我们还对每一项的测试结果进行了分析。

5.1.1 测试数据结果分析

5.1.1.1 完美十进制加法

表 4.2明确展示了我们的模型无论在低位数还是高位数的计算任务上均优于其他模型, 特别是在处理高位数计算时, 其优势尤为突出。我们注意到, 随着位数的增加, 大多数主流语言模型的准确度相比于处理低位数计算时显著下降, 这表明尽管这些模型在处理低位数计算时具备一定的进位处理能力, 但一旦位数增加, 它们就难以有效应对进位问题。这种现象揭示了一个重要事实：大多数模型可能只是在“记忆”计算过程, 而并没有真正理解进位的本质。与之形成鲜明对比的是, 我们的模型确实掌握了进位的核心能力, 因此无论计算的位数如何, 都能维持较高的准确率。

5.1.1.2 交错式位值减法

如表 4.3所示, 在处理五位数计算任务时, 除了 Llama2-7b 和 Llama2-13b 模型外, 大多数模型都超过 75% 的计算准确率。然而, 当面临更高位数的计算挑战时, 我们观察到 Llama2 系列模型和 Google-Palm 模型的性能显著下降, 准确率均未超过 5%。这一现象揭示了这些模型在泛化能力上受到了显著的限制。相比之下, 我们的模型能够根据学习到的规则进行严格的推理, 以较高的准确率计算出预期结果。这不仅超越了目前最先进的 ChatGPT 模型, 还证明了我们模型在泛化方面的潜力。

5.1.1.3 反向大幅度减法

如表 4.4所示, 对于这类高位数的减法问题, 最优的方法是通过简单地逆转计算顺序并在最终结果前添加负号, 计算过程可以大为简化, 且这种方法应用于反向大幅度减法, 也能轻松得出结果。我们确实可以教导模型掌握加减法的交换律, 但这样做无法充分展示我们模型处理高位数任务的能力, 因此我们依旧选择常规的计算逻辑来测试模型的能力。从表 4.4可以看出, 对于 Llama2 和 Google-palm 模型, 随着位数的增加, 它们的准确率几乎降至零, 而在处理较小位数的计算时, 则能够完成一定的计算任务。这暗示这些模型可能主要依赖记忆来解决数学问题, 而不是真正的数学逻辑推理。依靠记忆来寻找答案的学习方式极大限制了语言模型的学习潜力。

尽管 ChatGPT 在许多场景下也能达到很高的准确率, 我们发现在面对上述的减法问题时, 它通过将小数减大数的问题转换为大数减小数, 从而显著降低了问题的复

杂性。与之相对的是,我们的模型从逻辑角度出发,通过迭代过程逐步解析问题,无论是在低位数还是高位数的计算上,都展现了出色的准确性。这种方法不仅证明了我们模型在解决复杂数学问题上的强大能力,也展示了它在逻辑推理方面的深度和广度。

5.1.1.4 随机加减法测试结果

如表 4.5和表 4.6所展示,为了评估我们的模型在解决一般数值问题上的性能,我们采用 Python 生成了大量随机数值的实验数据以进行测试。初步结果显示,在执行低位数的加减法运算上,我们的模型与其他语言模型测试的准确率均超过了 75%,展现了较高的计算精确度。然而,随着计算位数的增加,大部分语言模型的性能明显下降。除了 ChatGPT 之外,其他模型由于未能深入掌握计算规则,在处理高位数计算时常常出现错误,几乎失去了计算能力。

特别值得一提的是,即便在面对高位数的随机加法计算任务时,我们的模型依然维持了 100% 的准确率。尽管在高位数随机减法计算的任务上遇到了一定挑战,我们的模型在所有参与测试的语言模型中,仍然展现了最高的准确率,比 ChatGPT 高出约 10%。这一成就不仅凸显了我们模型在解决复杂数值问题上的良好性能,也证明了其具有一定的泛化能力。

5.1.1.5 标准化最大误差分析

标准化最大误差这一指标用于衡量模型计算偏差的相对大小，这个指标大于 1 时，说明模型计算结果是完全偏离真实值的。从表 4.2，4.3，4.4，4.5，4.6中可以看出，在进行五位数数值计算的测试中，各类模型均能够确保标准化误差的平均值低于 1，即便计算不完全准确，误差也处于一个可控的范围之内。然而，当我们将计算的复杂度提升至十位数后，发现 Llama2-7b 模型的平均标准化误差均超过了 1，该结果表明模型在处理高位数计算时偏离真实值较远，结果不稳定。Llama2-13b 模型在准确率上与 Llama2-7b 相当，但其标准化误差更小，计算结果更为接近真实值。Qwen-72b-Chat 模型在各类任务中数值计算的标准化误差均是较小的，说明其计算结果相对接近真实值，表现更加稳定。而 ChatGPT-3。5 在某些测试任务中虽然准确率较高，其标准化误差较大。相比之下，ChatGPT-4。0 无论是在准确率还是标准化误差上都展现出良好的性能。MetaRuleGPT 模型在多项测试任务中有较高的准确率，在表 4.5的计算中，即使在规则执行出错的情况下，对计算结果的标准化误差相对较小，计算结果稳定。

5.1.1.6 向量叉积求解的结果

从表 4.7的数据中可以明显看到,Llama2 的 7b,13b 参数大小的模型甚至无法进行向量叉积的计算, 而 Llama2-70b 作为 llama2 系列最大的参数模型, 虽然可以进行叉积计算, 但是准确率为 0%。甚至是当前最强大的语言模型 ChatGPT, 在不借助外部工具的情况下, 最终的准确率也低于 50%。相比之下, 我们的模型能够在三维空间中准确计算向量的叉积, 且准确率高达 98□5%, 这进一步证实了通过组合不同规则来提升模型能力的方法是有效的。我们通过综合学习加法、减法、乘法以及叉积等基础运算规则, 实现了对这些规则的精确调用, 并成功输出了精确的计算结果。更为重要的是, 在同一个预训练模型中针对两种不同类型的任务进行规则训练, 我们的模型展示了多任务的泛化能力。这表明我们的模型不仅能够适应多种不同的任务场景, 而且能够识别并应用这些任务之间的共通规则, 从而显著提升了学习的效率。这进一步展示了我们模型的良好性能和灵活性。

5.2 讨论

5.2.1 可控性能

虽然现有的语言模型已经展现出了强大的功能, 但它们在可控性方面仍面临一些挑战。特别是, 大多数模型很难在可控范围内精确回答问题, 常常会出现较大的偏差。这是当前语言模型所需解决的一个重要问题。与此相比, 我们的模型严格按照规则执行任务, 因此在可控性上表现得相对较好, 尽管这种可控性可能会随着所需执行任务增加而产生变化, 但按照规则执行的方式总体来看可控性是相对有保障的。我们也将在后续模型训练中添加更多的任务规则, 对模型可控性进一步评估。

此外, 我们希望我们的模型在处理能力范围之外的任务上也拥有较强的可控性, 例如, 我们的模型在处理数值计算的计算任务时表现良好, 但在尝试解决函数积分问题时, 会遇到不可预测的结果, 导致很大的误差。这是我们目前面临的挑战之一。我们计划在未来的优化工作中, 引入更多规则数据, 以提高模型的整体可控性, 使其在更广泛的应用场景中表现的更加稳定和准确。

5.2.2 模型架构的选择

我们的模型虽然在解决数字问题上的性能优于当下流行的模型, 但是我们模型架构中可以看出 MetaRuleGPT 模型并不能一步到位完成任务, 在面对一个复杂的问题时, 通过一步一步前进, 把问题逐渐简化, 添加中间计算过程, 以达到我们最终的预期结果。语言模型并没有直接一步到位解决问题, 因此我们将会在之后考虑多步并行的方式来解决问题, 以此来提高我们模型的性能。

6 总结与展望

6.1 总结

受到元学习的启发，本研究旨在探索语言模型遵循规则的能力，即掌握人类在解决问题时所表现出的组合技能和泛化的能力。以数值计算为例，我们采用了基于Transformer 架构的方法，构建了一个采用迭代策略的语言模型 MetaRuleGPT。通过对一系列复合规则及其子规则的深入训练，我们成功得到了一个具有 300 万参数的预训练语言模型 MetaRuleGPT。在学习了基础的个位数算术运算和一些核心计算规则后，我们的模型能够通过组合和迭代已掌握的技能，有效处理未遇到过的高位数算式和更为复杂的向量叉积运算，展现出较高的准确性，甚至在准确度方面超越了当前的主流大型语言模型。实验证明，在不借助外部计算工具的情况下，现有的主流语言模型，在处理高位数计算和其他复杂运算问题时，常因理解能力的限制而受挫。我们的模型不仅准确完成了这些具有挑战性的计算任务，还展现出了语言模型在数值计算上的泛化潜力。这一发现有力地支持了我们的假设：通过规则遵循，语言模型能够展现出一定的泛化能力。

6.2 展望

本文的研究提出了一些值得深入探讨的问题，包括:

1. 尽管我们的模型在规则学习后显示出了一定的泛化和理解能力，但受限于计算资源，我们能够处理的问题种类相对有限。我们期待在扩大模型参数规模并采用更多规则数据集进行训练后，使模型能够处理更广泛的逻辑任务。

2. 我们的模型在捕捉问题细节上还有进步空间。MetaRuleGPT，大体上能够模仿人类解决数值问题，但在一些细节处理上还不能达到人类的精确度。例如，模型对 0 的处理往往是机械的，无法像人类一样，将 0 当做一种既不是负数也不是偶数的特殊存在。这表明我们的模型在处理问题的精细度上与人类还存在着很大的差距。因此我们后续会优化学习策略，细化问题，以此提升模型的性能。

3. 我们的模型目前无法自动处理未经训练的泛化形式或超出元学习分布的新概念，这大大限制了其处理全新问题的能力。因此，模型是否能够在各个方面利用现实世界的训练经验，从而实现人类般的系统性泛化，仍是一个开放的问题。

上述讨论为未来的研究指出了方向。我们期望通过持续的优化与扩展，进一步增强模型的性能和实用性。目前，我们已经使语言模型在特定数学任务上具备了遵循规则的能力，但这对于一个能够处理广泛任务的模型来说仅仅是个开端。它的全面潜力还有待于我们去挖掘。因此，对我们来说，促进语言模型掌握规则遵循的能力更是一个有价值的尝试。

参考文献

[1] Brenden M Lake and Marco Baroni. Human-like systematic generalization through a meta-learning neural network. Nature, 623(7985):115–121, 2023.

[2] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. arXiv preprint arXiv:2211.10435, 2022.

[3] Ryan Hardesty Lewis and Junfeng Jiao. Computegpt: A computational chat model for numerical problems. arXiv preprint arXiv:2305.06223, 2023.

[4] Zhen Yang, Ming Ding, Qingsong Lv, Zhihuan Jiang, Zehai He, Yuyi Guo, Jinfeng Bai, and Jie Tang. Gpt can solve mathematical problems without a calculator. arXiv preprint arXiv:2309.03241, 2023.

[5] Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. arXiv preprint arXiv:2309.12284, 2023.

[6] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.

[7] Bai J, Bai S, Chu Y, et al. Qwen technical report[J]. arXiv preprint arXiv:2309.16609, 2023.

[8] Anil R, Dai A M, Firat O, et al. Palm 2 technical report[J]. arXiv preprint arXiv:2305.10403, 2023.

[9] Touvron H, Martin L, Stone K, et al. Llama 2: Open foundation and fine-tuned chat models[J]. arXiv preprint arXiv:2307.09288, 2023.

[10] Wu T, He S, Liu J, et al. A brief overview of ChatGPT: The history, status quo and potential future development[J]. IEEE/CAA Journal of Automatica Sinica, 2023, 10(5): 1122-1136.

[11] Lim Z W, Pushpanathan K, Yew S M E, et al. Benchmarking large language models' performances for myopia care: a comparative analysis of ChatGPT-3.5, ChatGPT-4.0, and Google Bard[J]. EBioMedicine, 2023, 95.

[12] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. arXiv preprint arXiv:2204.02311, 2022.

[13] Brown T, Mann B, Ryder N, et al. Language models are few-shot learners[J]. Advances in neural information processing systems, 2020, 33: 1877-1901.

[14] Thoppilan R, De Freitas D, Hall J, et al. Lamda: Language models for dialog applications[J]. arXiv preprint arXiv:2201.08239, 2022.

[15] Bujard H, Gentz R, Lanzer M, et al. [26] A T5 promoter-based transcription-translation system for the analysis of proteins in vitro and in vivo[M]//Methods in enzymology. Academic Press, 1987, 155: 416-433.

[16] Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.

[17] Yu Y, Si X, Hu C, et al. A review of recurrent neural networks: LSTM cells and network architectures[J]. Neural computation, 2019, 31(7): 1235-1270.

[18] Sherstinsky A. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network[J]. Physica D: Nonlinear Phenomena, 2020, 404: 132306.

[19] Yin W, Kann K, Yu M, et al. Comparative study of CNN and RNN for natural language processing[J]. arXiv preprint arXiv:1702.01923, 2017.

[20] Zhang Z, Han X, Liu Z, et al. ERNIE: Enhanced language representation with informative entities[J]. arXiv preprint arXiv:1905.07129, 2019.

[21] Liu Y, Ott M, Goyal N, et al. Roberta: A robustly optimized bert pretraining approach[J]. arXiv preprint arXiv:1907.11692, 2019.

[22] He K, Gkioxari G, Dollár P, et al. Mask r-cnn[C]//Proceedings of the IEEE international conference on computer vision. 2017: 2961-2969.

[23] Arnab A, Dehghani M, Heigold G, et al. Vivit: A video vision transformer[C]//Proceedings of the IEEE/CVF international conference on computer vision. 2021: 6836-6846.

[24] Yang Z, Dai Z, Yang Y, et al. Xlnet: Generalized autoregressive pretraining for language understanding[J]. Advances in neural information processing systems, 2019, 32.

[25] Fodor J A, Pylyshyn Z W. Connectionism and cognitive architecture: A critical analysis[J]. Cognition, 1988, 28(1-2): 3-71.