规则组合实现大语言模型的数字理解能力

陈钶杰
学号: 22135030

19. März 2024

Inhaltsverzeichnis

1 摘要(Abstract)

在以往的研究中,通常大型语言模型在不借助外部工具的情况下,无法准确进行算术运算。因为现有的语言模型都是在进行记忆数学题,而不是理解数学题,而数学计算是无穷无尽的,而语言模型无法记忆无穷无尽的数学规则,自然无法进行精准的数学计算了.本文旨在挑战这种误解。本文提出的关于规则学习的语言模型,通过对不同规则的组合来实现数字的精准计算,以及其他相关的复杂任务.我们在本文中提出了规则组合学习的语言模型.首先我们设计了一套关于数学计算中用到的基本规则,复合规则和迭代规则数据集,然后将我们所制定的规则进行文本嵌入并进行预训练处理,最后得到的预训练模型便能够进行复杂的数字计算.模型实验中,我们的模型在数字计算上,特别针对一些高位数字计算上,我们的模型在各类数字计算问题上准确度都优于现有的SOTA模型;通过测试在随机生成的数值问题上计算的准确度实验证明了我们模型在解决问题能力的全面性.通过测试同时解决两种不同数值问题的实验证明了我们的模型在解决问题能力上具有一定的多样性.

关键词:Transformer;NLP;Arithmetic Caluculation;Meta Learning;
(英文版文献综述)

2 引言(introduction)

在自然语言处理领域的最新进展中，如GPT-4等大型语言模型已经取得了显著的成功，在许多任务中均展示了惊人的理解能力。然而，在处理数字问题等专业领域时，这些模型仍然面临巨大挑战。所谓的数学问题,涵盖非常广,包括加减乘,求导,积分和解方程等.然而语言模型却在最简单的数字加减问题上就遭遇巨大的困难.
研究人员探索了基于Transformer架构的语言模型解决数学问题的潜力,并在这一领域取得了许多的进展,compute-gpt,mathgpt是相对成功解决复杂数学问题的语言模型中的优秀变体:compute-gpt提出了一种语言模型和外部工具(如python工具)结合方式来解决数值计算问题;MathGPT借鉴了传统语言模型微调的方法,通过提高基本数据集的训练大小,在更大基数的数据集训练之下，相比传统的语言模型能解决更多数字计算问题。尽管上述模型都在一定程度上优化了语言模型解决数学问题的能力，但是这些模型部分是通过借助外部工具，部分是通过训练提高数据量的方式来提升数字计算能力，并没有真正的让语言模型自己掌握一种数字的计算。因此，我们引入了元学习中提到的规则组合能力，我们尝试令语言模型学规则的调用和组合,并通过对问题的多次迭代，来解决具有逻辑性的问题。实验结果显示，直接通过喂养大量数学问题的方式虽然在数据量喂养较大的时候，可以使得语言模型展现出一定的数字计算能力，但通常数字计算问题的数量是无限的或者巨大的，语言模型在数字问题这种具有严格逻辑的问题上所展现出来的泛化能力也非常的有限。相比之下，我们的模型通过规则层面的学习，可以用极少量的数字规则训练数据集，模型通过对问题进行迭代计算，可以解决任意的数字计算问题。此外，我们的简易语言模型也可同时接受多种不同的任务，我们的模型可以同时完成多种任务，比如不仅可以进行任意数值的加减任务，同时也可以完成向量计算中比较复杂的外积任务。并且在进行外积任务时可以利用已经学得的部分数字规则来进行外积的计算。后续的实验结果显示，我们提出的模型，在数字计算任务以及处理复杂逻辑问题上除了与chatgpt3.5,chatgpt4.0持平以外，超越了现有的SOTA(State-of-the-art model)模型。这些贡献总结如下:

1. 设计了一套适用于语言模型的规则数据集
2. 通过对规则的编码嵌入，来实现语言模型对规则的调用和组合.
3. 设计了一种全新的模型迭代方法，更加接近人类对问题逐步求解的思考过程。
4. 可以通过少量的规则学习，来解决大量的问题，远小于其他语言模型的训练数据集的数量，同时保证解决问题的准确度。

3 相关工作(relate work)

3.1 背景介绍(back ground)

大语言模型通过大规模数据学习语言规则，但在数学计算中，数字之间的运算是无穷无尽的，但是数字计算是有明确规则的,数字运算对于模型而言是一项具有挑战性的任务。当前的语言模型在这方面仍然存在一些限制，因此有必要进一步研究数字规则学习的方法，以增强其在数字运算中的适用性。

3.2 Large Language Models

大型语言模型(LLM)在自然领域展示了强大的功能,即语言处理任务，在大量语料库上进行训练多样化且未标记的数据，在不同任务中均展示出了良好的通用能力,显著改变了该领域的研究范式。语言模型在经过对广泛语料库的预训练，这些模型获得了强大的语言理解能力和生成能力，使其在各种基准测试中具有卓越的性能，例如情感分析,机器翻译等任务,均有良好的表现.
尽管如此，目前最优秀的大型语言模型ChatGPT和GPT-4在语言理解和生成中，对文本问题具有精确的理解且给出相当精确的回答,但是在解决数学问题时仍然遇到挑战,数学问题都是只有唯一答案的,这对于模式识别的语言模型而言无疑是困难的,语言模型本质上是一种概率模型,是将可能性最大的结果告诉我们,然而数字计算,数学推理是非常严谨的,每一步都需要严格遵循,这就导致语言模型以概率计算的方式去进行数字计算会出现巨大的偏差。这项工作致力于解决和提高语言模型在解决数学问题领域的表现，包括算术任务和以及严格的逻辑推理问题。

3.3 Meta Learning

元学习，是机器学习领域的一个重要分支。元学习的核心在于让机器学习算法能够利用过去的经验来提高其未来学习新任务的效率和效果。与传统的机器学习相比，元学习更加关注于如何在多个任务上学习通用的学习策略，从而在面对新任务时能够快速适应。人类语言和思维的力量源于系统的组合性，即从已知成分中理解和产生新组合的代数能力。Fodor 和Pylyshyn 曾提出了一个著名的论点:即人工神经网络缺乏这种能力，因此不是可行的思维模型,虽然从那以后的几年里,神经网络取得了巨大的进步，但此挑战仍然存在。Meta Learning [**?**] 成功地解决了Fodor 和Pylyshyn 的挑战，并提供了证据，证明神经网络在优化其组合技能时可以实现类似人类的系统性。为此，他们引入了组合性元学习(MLC)方法，用于通过动态的组合任务流来指导训练。MLC 实现了类人泛化所需的系统性和灵活性，还在几个系统的泛化基准中提高了机器学习系统的组合技能。

3.4 Arithmetic Calculation

早期已经有不少研究集中在使用大型语言模型解决数字计算问题。比如MetaMath[5],该模型通过喂养大量的算术和符号推理任务数据集,在经过大量数据训练后,来实现语言模型对数学问题的理解以及求解,该方法能够实现对简单数字问题的求解,一旦问题变长,比如数字长度变长,模型求解起来将会非常困难,究其本质是因为模型只是记忆了数学问题,但并没有深刻理解其中的数学规律和算法,同时这也是当前主流模型所面临的问题.因此,ComputeGPT[3],开始探索了语言模型和算术任务的外部工具的集成,这种方法的好处是计算能力强大,且计算的数值不受任何位数的影响,但在这种情境下语言模型只是起到了一个文本识别器,将数字,运算符号识别出来,真正起到运算的作用的是算术任务的外部工具,而语言模型本身并未进行数字计算的.同理,像程序辅助语言模型[1]也是类似的,这是一种新的自然语言推理方法，使用程序作为中间推理步骤。与现有的基于LLM的推理方法不同，主要思想是将求解和计算使用外部的Python解释器，而不是使用LLM本身来对问题进行理解和求解。这种方法相比ComputeGPT,语言模型的需要处理更多的问题,但是核心的计算依旧是使用外部工具.最新的模型MathGPT[4]旨在不使用计算器工具情况下执行算术运算，其20亿参数的语言模型可以相对准确地进行多位算术运算，此文的核心思路是通过训练较多的低位数的数值计算方法,以此来对高位数运算的准确性.该模型所用的方法与利用外部工具不同，其专注于探索如何在不依赖外部工具的情况下增强LLM的固有的算术能力,但是这个模型依旧没有让模型理解数字运算的基本规律.而是通过记忆大量基本计算规律来进行问题的求解.

4 研究方法(Methodology)

为了提高语言模型在复杂逻辑推理问题中的有效性和泛化性，我们提出了MetaMathRule模型，其旨在提升语言模型在数学推理中的性能来展示其强大的泛化能力。首先，我们的模型MetaMathRule受到了元学习的启发，更加注重学习通用的学习策略，致力于使用规则来准确的执行算术任务。它通过在训练过程中动态的整合数学计算的基本规则以及复合规则，来提高模型组合规则的能力。

MetaMathRule模型在直接面对一个复杂的未见过的算术表达式时，通过将迭代策略整合到其架构之中，对每一次迭代过程中会根据算式表达式选用最合适的规则进行计算，经过有限次迭代后，即可得到相对精确的计算结果。这与直接解决复杂算术表达式不同，MetaMathRule采用的这种策略，使用规则的方式进行计算更接近人类解决数学问题的方式。使用规则学习的方式解决任务，不仅可以使用更少的训练数据来训练模型，在处理复杂计算问题的时候还能媲美当前主流大语言模型,达到更高的计算准确度。

此外,利用这一策略，MetaMathRule不仅可以处理单任务，同时还可学习多种不同的任务，对不同任务中每个规则是会出现交叉的，我们的模型依旧可以通过动态的学习交叉的规则，可同时完成多种不同的任务，且相互不受影响。

4.1 学习算术任务的具体规则学习

我们的模型主要示例任务为三种,分别是加法算术任务,减法算数任务,以及向量外积计算任务.对于上述三种任务,我们需要分别使用不同的规则进行训练.如下表所示,我们训练加法任务的时候需要学习九九加法表,进位规则,数字映射规则,以及计算规则.学习了这些任务以后,我们的模型就可以精准的运算各种高位数的加法. 如果我们的模型在仅仅学习了加减法以后,我们希望模型学会向量的加减计算,那我们可通过仅添加向量格式规则及外积计算规则,就可以进行向量外积的计算了.即对于复杂的向量外积计算而言,整个网络的泛化能力会很强.
关于上述每种任务的计算规则选择主要如下表所示

Tabelle 1: 不同位数下的结果对比

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	√	√	√

4.1.1 算数训练数据集

用于训练的算术数据集经过精心设计，其中包含了最简单的个位数算术任务以及各种算术规则。这个数据集被深思熟虑地设计以包含各种操作，其中包括对齐规则，进位规则，退位规则，简单计算规则，组合规则等。我们创建了一个小型数据集，大约是2万条记录不等。% 在这些数据集中，每个算术表达式由2到10个操作步骤组成，涵盖了一系列数学操作，如加法(+)、减法(-)、向量外积运算(^)。
在这些数据集中，算术表达式仅仅包含了最最简单的个位数计算，例如九九加法表，九九减法表等，其他所涵盖的均是一系列的数学规则操作，比如数字的对齐规则，数字的进位规则，数字的退位规则，外积的规则，组合规则的学习等。为了与人类的计算习惯保持一致，我们采用逐步策略，不是直接计算每个复杂算术表达式的最终答案，而是将复杂表达式分解成一系列更简单的步骤，逐步生成答案。这种策略反映了人们通常在解决复杂算术任务时遵循的过程。通过在这样的数据集上训练，我们的模型在数值计算上具有良好的算术性能，因为它所学习的是底层的计算规则而不是单单依靠记忆得到的结果。图x中提供了一些从算术数据集中抽取的训练示例，展示了数据集中逐步策略的具体实现手段。
(将训练数据集都放上来*)

4.1.2 模型和训练程序

模型和训练程序。表中报告了所有不同模型参数的模型概览。我们对几个不同参数量的模型都使用相同规模的数据集进行训练，比较最终的结果。我们的训练主要包含了以下4种不同参数量的模型，其中最大的模型配备了10亿（1B）参数，使其在容量方面最为强大。在此之后，我们还有用5亿（500M）参数训练的第二个模型，以及用7000万（70M）参数训练最小的模型。值得注意的是，尽管模型参数大小存在差异，数据集大小规格一致，该数据集包含2万多条训练记录。
(准备一个模型规格大小的表格)

4.2 多任务上的学习训练架构

对于多任务的数据集而言，算术表达式依旧只包含了个位数计算，但是对于其他规则的涵盖需要作出一些改变，我们的测试例子是学会做加减乘，以及向量的外积，这是两种完全不相同的任务，一个是数字的加减，一个是向量的外积。但是在规则上有一些共性，比如向量的外积具体计算时，也需要进行个位数的加减运算，也需要进行对齐等，因此我们找出了两种不同任务的特性和共性，分别设计出相应的公共规则和特殊规则，最后合并训练数据，得到相应的结果。同时我们的模型在针对不同的任务时会根据所习得的规则作出相应的回答，具体回答依旧采用逐步策略，逐步生成答案。图x中提供了一些从算术数据集中抽取的训o示例，展示了数据集中逐步策略的具体实现手段。

4.2.1 算数训练数据集

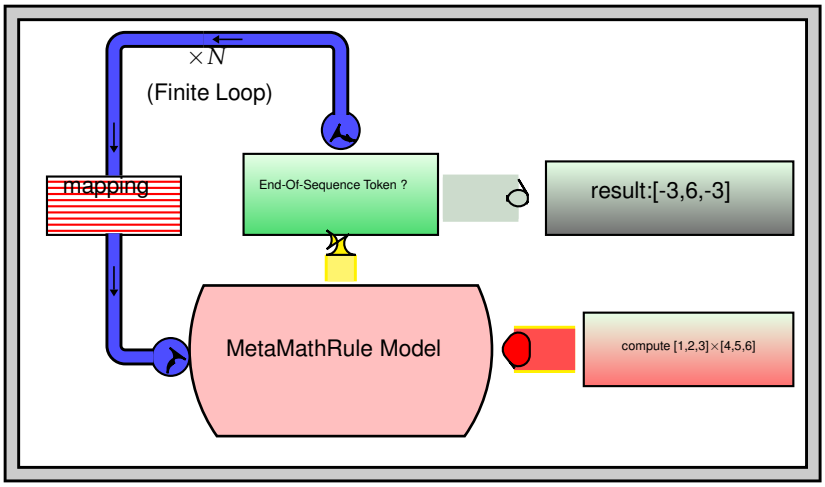
数据包含了最简单的个位数算术任务以及各种算术规则，同时还包括数字，向量的对齐规则，进位规则，退位规则，简单计算规则，组合规则等。由于我们主要测试方法的实用性，我们创建了一个小型数据集，大约是1千条记录左右.

4.2.2 模型和训练程序

由于我们训练的数据集比较小，所以我们只用了3000万(30M)的参数训练模型。已经能对数据集有较好的学习效果。

4.3 模型架构

4.3.1 模型训练结构



在关注算术任务的同时，我们训练了一系列基于Transformer的语言模型，称为通用语言模型，以解决数学问题。我们的模型架构分别如上图所示，在模型架构中分别有以下部件，即训练好的MetaMathRule模型，mapping工具，根据模型输出是否包含结束标识符的判定器。

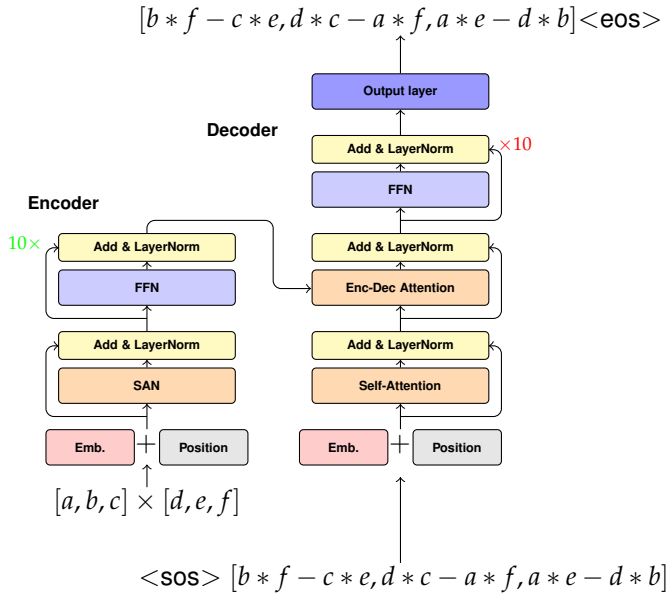


Abbildung 1: Transformer训练架构图

4.3.2 模型训练策略

对于语言模型而言，在面对一串字符'123 + 456'，它并不能直接归纳出两个数值的相加，因为对语言模型而言，它是将每一个字符进行嵌入，每个字符都代表了相同的意义，甚至是+和数字的意义是等价的。对于任何一个问题，我们均通过从识别对齐，位数计算，进位退位，输出结果，这4步法来解决问题从而得到我们的规则数据集合。因此在计算过程中，我们首要解决的问题就是如何让语言模型能够识别对齐数字以及符号。例如我们希望模型能够识别3位数加3位数的加法，或许一种粗暴的办法就是，我们将所有可能的情况都列出来，那样就有10^6个训练集了,但是这样的效率显然是极低的。因此，我们提出了使用规则的方法解决，我们希望我们的模型在面对类似'1 2 3 + 4 5 6'结构的问题时，能够统一识别结果。因此我们需要模型掌握识别位数的能力，我们通过将构造一个特殊的规则，如'a,b,c,+,d,e,f'使模型学会处理的不只是一个具体的数值，而是一种结构。能够主动分离出a,b,c以及d、e、f，之后使得模型学习a,b,c和1,2,3之间的对应规则，即可使得模型对字符串'1 2 3 + 4 5 6'进行数字符号分离。

在数学算术中分离就是为了进行对齐，因此我们可以直接对我们的结果进行对齐，即将'a,b,c,+,d,e,f'经过对齐以后的结果就是'a+d|b+e|c+f'.我们令模型能够学会将一种结构'a,b,c,+,d,e,f'对应成'a+d|b+e|c+f'的对齐规则，来处理数值识别问题。之后再将其中对应的字母使用具体的数值替换规则，将数字替换，这样以来我们的模型就成功的将一个字符串的内容通过我们的数值规则将内容识别了出来。

之后我们需要将解决位数计算的问题，例如'1+4|2+5|3+6'，我们需要如何进行特定位数的加法计算!我们依旧是通过结构化规则的学习，即学习如何对于'a+b|c+d|e+f'进行结构分离，分别进行计算a + b，c + d，e + f等使得模型能够明白，它面对一串字符时，所需处理的就是各个位数的和。同时我们的模型已经学习了大量的子规则，比如1+4,2+5,3+6的值。模型就可以直接将计算出中间结果'5|7|9'。

对于此问题而言，'5|7|9'已经是最终结果了，但是如果中间结果是'5|13|12'的话，我们还需要进行进位操作，即我们希望'5 + 1| 3 + 1| 2' 经过进位以后计算出最终结果'6|4|2'。参考前面的方式，我们使得模型学习进位规则'a|b c | d e '进行相应的进位操作，即'a + b | c + d | e'.将数字和字符再做相应的映射，得到最终的进位结果。

最后，对于我们所得到的结果'5|7|9'是否为我们希望的计算结果，这需要有一个模型结束规则。即判定'5|7|9'所类似的格式是我们所预期的计算结果，我们依旧进行结构化处理为'a|b|c'的形式，如果是我们的预期结果，那么在即可转化成'a|b|c|&',由此得到我们最终的计算结果。

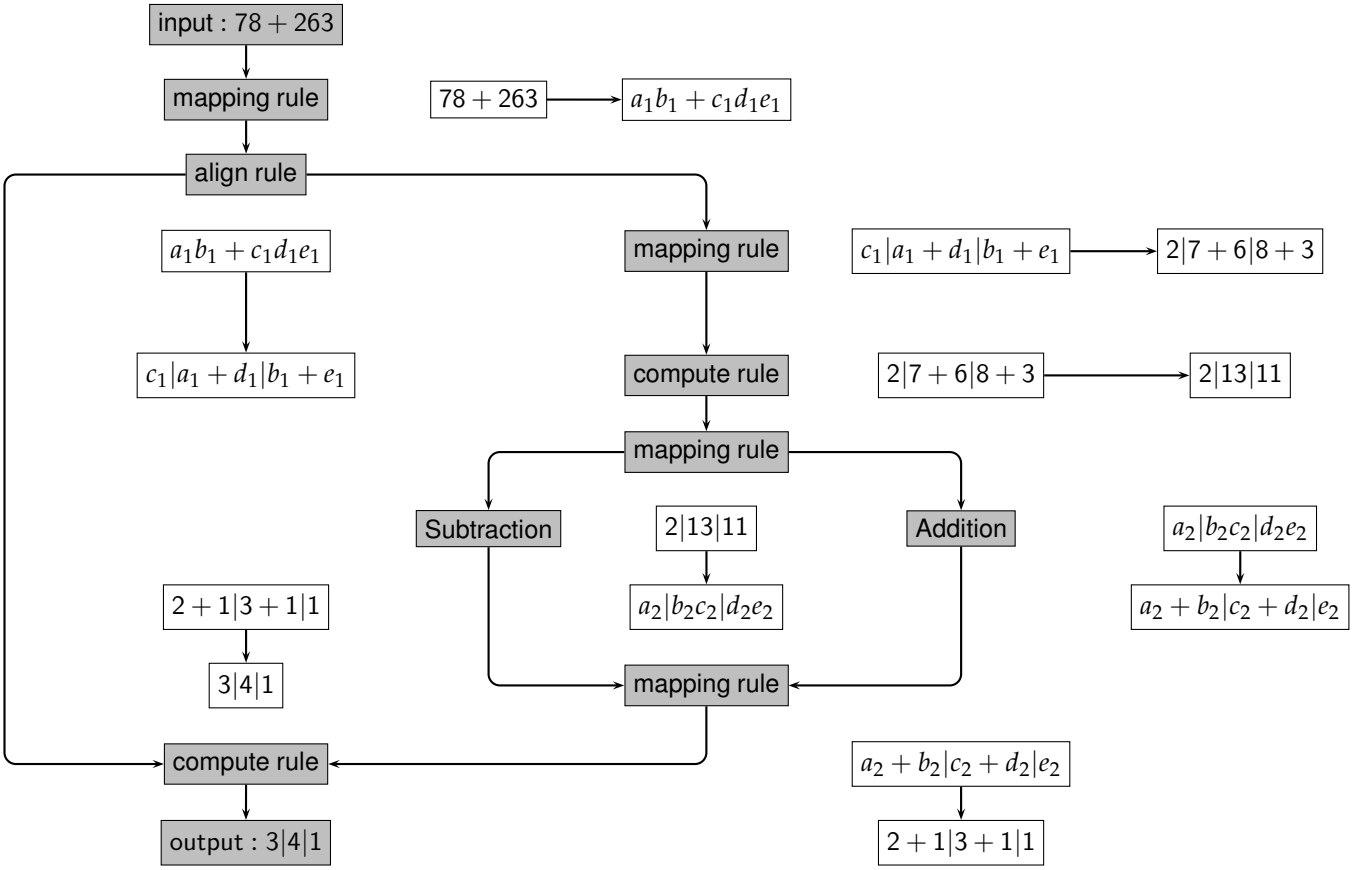


Abbildung 2: 如何调用规则求解算式

4.3.3 模型计算示例一(加法)

我们通过计算一个加法示例来说明我们模型的架构思路。我们现在对我们的模型输入78+263这个问题。

1. 首先模型对我们输入的问题进行结构化处理，即将78+263转化成'78 + 263->$a_1b_1 + c_1d_1e_1$',之后根据已学得的对齐规则，变成'78 + 263->$c_1|a_1 + d_1|b_1 + e_1$'，输出中间过程'2|7 + 6|8 + 3'.
2. '2|7 + 6|8 + 3'，输入进行结构化处理，并逐步计算中间过程，得到最终结果2|13|11
3. 2|13|11作为最新的输入，首先进行结构化处理，得到了'2|13|11->$a_2|b_2c_2|d_2e_2$'然后进行进位规则的调用即可得到'2|13|11->$a_2 + b_2|c_2 + d_2|e_2$',最终输出'2 + 1|3 + 1|1'
4. '2 + 1|3 + 1|1'，输入并结构化处理，同理计算中间过程，得到最终结果'3|4|1'
5. 3|4|1输入，使用结束判定规则，得到最终结果'3|4|1->EOS'

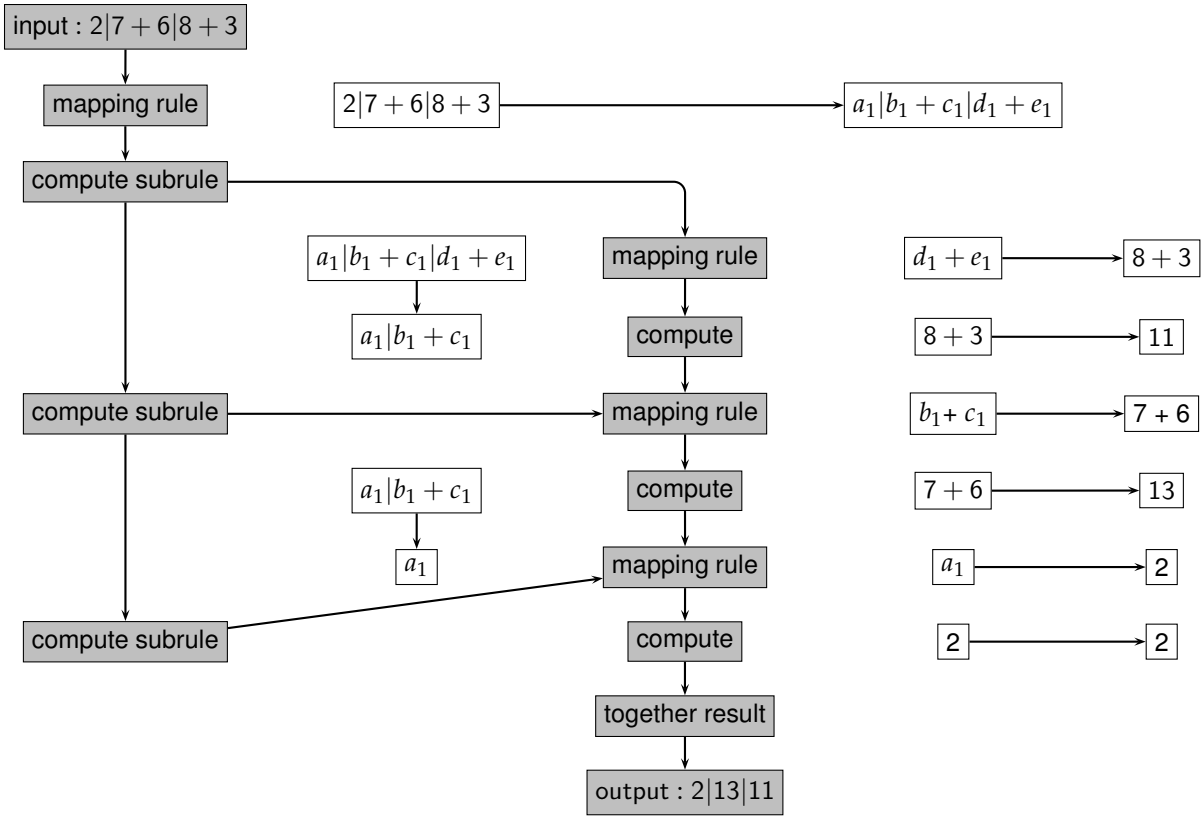


Abbildung 3: compute rule

4.3.4 模型计算中间过程

关于中间过程的计算其实是非常复杂的，比如模型对于一个问题$2|7+6|8+3$如何能够筛选出每一项的内容2、7+6、8+3，并进行相应的计算，这一步是非常困难的。我们通过设定特定的子计算规则来解决这个问题。第一步依旧是使得模型正确分离字符串$2|7+6|8+3$为2、7+6、8+3。这一步我们依旧可以通过结构化规则得到，即通过$a_1|b_1+c_1|d_1+e_1$进行适当的分离，即可得到a_1,b_1+c_1,d_1+e_1,在将输入的2,7,6,8,3分别经过一一映射后，对应到相应的字母，模型会对每一个算式通过所学习的子规则中进行相应的计算，比如8 + 3 = 11,7 + 6 = 13 ， 2 = 2,最后由组合规则将所有内容进行合并，即可得到我们的最终计算结果:'$2|13|11$'.

4.3.5 模型多任务规则训练策略

我们的模型能够同时学习多种任务的规则，且经过学习以后，依旧在上述模型的架构中可以进行问题的解决。比如我们设计了一种数据集其中包含数值运算的所有规则，以及向量外积的规则，这两种规则有一定的共性也有一定的特性，我们将这些规则进行统一，最终进行训练得到了我们所需的结果，最后将我们的模型进行合适的计算就可以得到我们所预期的结果了。

5 实验(Experimental settings)

5.1 实验设定

MetaMathRule的总体目标是展示语言模型在学习规则后在推理领域的泛化能力。为了验证这一点，我们设计了两种不同类型的实验，包括算术任务和交叉计算任务的解决。这两类任务全面涵盖了基础计算能力和高阶问题的解决技巧，为评估模型在推理方面的熟练度提供了强有力的评估。

5.2 数据集

在算术领域内，我们创建了一系列专门针对算术任务的多样化数据集。这套训练数据集包括了多种多样的数值类型，其中包括多进位加法，小数减大数，错位数减法，随机数加减法等。我们的评估数据集由8000个测试用例组成，为了保证我们模型泛化能力的体现，这些测试用例从与训练数据集无任何关联。这套精心生成的数据集作为全面的基准，用于评估和量化MetaMathRule在算术任务上的计算能力。为了更深入地探索生成数据集的具体细节，可以在附录xx中找到。表xx展示了我们主要测试的一些任务平均准确度。

一共有4种特殊的数据集合

Data Type	Randomized Procedure	Perfect Decadic Addition	Reverse Magnitude Subtraction	Interleaved Subtraction
测试	6729132856+1854307391	6659891948+340108052	62103-2386797965	1824453209-482835016
	1392400+4437047	5616608419+383391581	24024-3727865739	5192638457-273344939
数据	⋮	⋮	⋮	⋮
	65584019-1142850735	3511809736+488190264	65223-4484476664	1253479182-944088576
类型	1554887316-817095695	4376628072+623371928	53006-7764286617	8858241744-261714262

5.3 评估指标

关于最终计算结果的生成评价中，我们不仅仅考虑了计算结果与我们真实结果的匹配率，即准确率，同时我们还考虑了是否我们的计算结果与准确答案的差值率，理论上差绝对值越小，说明我们越是能够通过适当的微调来达到更高的模型准确度.
假设正确预测的数量为TP，总预测数量为N，那么准确率可以表示为：

$$准确率 = \frac{TP}{N} \times 100\%$$

假设我们模型计算结果是y,实际计算结果为$\hat{y}$,一共有N个数,那么我们的最终的差值率就是:

$$差值率 = \frac{\sum_{i=0}^{N} |y_i - \hat{y}_i|}{N} \times 100\%$$

5.4 实验过程

为了验证我们模型的数学推理能力，我们考虑与其他流行的语言模型进行比对，例如国内大公司阿里的语言模型QWen，国外的Google-Palm、LLama2，以及当下最强大的语言模型ChatGPT-3.5和ChatGPT-4.0等。这样的比对可以提供对不同模型性能的参考，并帮助我们评估和比较它们在数学推理方面的表现。
对于现有的大型语言模型，其无法精确计算数学问题的原因可能在于它们缺乏对计算底层逻辑的深入理解。然而，我们的模型从数学的底层逻辑出发，并能够更精准的解决数学问题。为了证明这一观点，我们设计一些特殊的数据类型，并提供相关的计算数据集和结果，来验证我们的模型在数学推理方面的能力。通过这种方式，不仅展示了我们模型能够理解并应用数学的底层逻辑来解决计算问题能力，另一方面展现出我们模型在解决问题时具有强大的泛化能力。
以下是我们验证模型的能力所进行详细的实验和评估，以确保其在不同问题上的一致性和准确性。同时，我们与其他大型语言模型进行了比较和评估，这也是评估我们模型性能的重要步骤。这样将有助于更好地展示我们模型在数学推理方面的优势和特点。我们的模型在这些特殊问题上的表现超越了当前大部分流行的模型，并且我们的模型训练过程中，展现出更强的准确性和泛化能力。

1. 完美十进制加法

$$34572843 + 65427159$$

对于上述算式，这是一个非常有规律的算式，结果为1000000002。尽管这个算式在我们人类看起来很简单，但对语言模型而言，是非常复杂的，原因是这个算式需要进行多次进位操作。每一步计算都需要相应的进位，我们发现从最后一位开始计算，每向前一位，刚好凑成一个十，这个时候又需要进行进位了，这个算式一共需要经过8次进位，且内部计算涉及多次计算规则的应用，我们将这种类型的加法称为完美十进制加法。
而对语言模型来说它仅仅是一个文本输入，在处理这种内部逻辑相对复杂的问题时，由于缺乏对数字的深入理解，很难精确执行每一步的进位操作。如果在中间的任何一步出现问题，最终结果就会产生巨大的偏差。
为了验证这一观点，我们使用python随机生成400个满足上述规律的完美十进制加法算式，其中高低位数的数据集各200个。我们测试当前流行的语言模型在处理3-5位数和7-10位数的完美十进制加法，下表是我们测得的语言模型在进位加法问题上的准确性结果。

Tabelle 2: Perfect Decadic Addition(完美十进制加法)

Size	3-5 digits	7-10 digits
GPT-4	100%	98%
GPT-3.5	97.2%	91.5%
llama2-7b	19.4%	5%
llama2-13b	3.6%	3%
llama2-70b	5%	6%
Google-PaLM	52.6%	27.5%
Qwen-72b-Chat	85.8%	67.5%
MetaMathRule	100%	100%

2. 反向幅度减法

$$342 - 32478857$$

关于这个算式，如果从右往左计算是非常容易的，但语言模型本身是一个从左往右识别字符的过程，无法像人类一样进行交换数值，只能从左向右计算，这个问题难点在于符号判断，这个计算结果应该是一个负数，而且减法的进位又比较复杂，在这里每一个位数都需要借位(退位)，对于这样的问题，我们称为反向幅度减法。一个对数字理解能力较弱的语言模型而言处理这个问题是非常困难的，特别当数的位数变长以后。同时，我们通过实验对比了，当前流行的语言模型在这些数据集上的表现。下表展示了所有模型在低,高位数的反向幅度减法数据集上的表现

Tabelle 3: Interleaved Subtraction(交错式位值减法)

Size	3-5 digits	7-10 digits
GPT-4	98.3%	96%
GPT-3.5	95.2%	91%
llama2-7b	2.3%	0%
llama2-13b	21.9%	2%
llama2-70b	76.1%	2%
Google-PaLM	95.9%	19%
Qwen-72b-Chat	93.9%	74.5%
MetaMathRule	100%	100%

3. 错位减法

$$93847638 - 39281729$$

上述算式中两个数的各个位次的数值大小是循环交替的，比如个位后者大，十位前者大，依次循环下去，对于计算此类算式，需要进行适当的退位，进位操作，甚至连正负也是不确定的，语言模型面对此类算式将会很难进行精确计算。我们依旧通过构造400个随机的数据集合，进行测试所有语言模型，进行比对。下表展示了各个语言模型在低，高位数上的错位减法上的表现

Tabelle 4: Reverse Magnitude Subtraction(反向幅度减法)

Size	3-5 digits	7-10 digits
GPT-4	97.8%	96.5%
GPT-3.5	99.4%	88.5%
llama2-7b	20.8%	0.0%
llama2-13b	4%	0.0%
llama2-70b	50.2%	0.5%
Google-PaLM	43.6%	0.0%
Qwen-72b-Chat	86.4%	3.5%
MetaMathRule	100%	100%

4. 多功能模型
   在解决数值加减问题上，我们的模型取得了良好的效果，但我们的模型能力并不仅仅局限于处理这些简单计算问题上。即使问题变得更加复杂，更加多样化，我们依旧可以通过规则学习预训练我们的模型得到所需的结果。
   我们特意选择一个较为复杂的关于向量外积的求解问题作为我们模型拓展学习的一个尝试。我们尝试让模型学会如何求解向量的外积以及计算数值加减运算。并测试我们模型在计算外积过程中的准确度，与当下的模型进行对比。
   下表展示了我们模型在计算向量外积上的能力。

5. 参数规模的重要性
   为了验证我们模型强大的泛化能力，我们通过训练了不同大小参数模型进行比对，来验证不同大小的模型对我们模型能力的影响。如下表分别展示了我们使用3种不同大小的语言模型进行对比

5.5 复杂交叉任务计算

我们的模型将能够同时完成两种不相关的任务，比如计算两个数值的大小与计算两个向量的外积，我们通过精心设计了两种不同的数据集合，来训练我们的模型

5.6 随机加减法(对比实验)

随机加减法
上面这些特定的数据集均需要较多推理，为了验证我们模型的通用能力，我们也生成了800个随机加减算式来展现我们模型在普通数据集上的计算能力，同时我们比对了当前大模型在这些随机加减数值上的能力。
下表展示了我们的模型和当前主流模型的结果比对图。我们的模型不仅仅在这些特殊数据集上具有良好的表现，在更

Tabelle 5: Randomized Subtraction Procedure(随机减法过程)

Size	3-5 digits	7-10 digits
GPT-4	100%	78.5%
GPT-3.5	95.5%	76.5%
llama2-7b	25.5%	3%
llama2-13b	31%	1%
llama2-70b	73.5%	12.5%
Google-PaLM	80.5%	47%
Qwen-72b-Chat	93.5%	81.5%
MetaMathRule	100%	88%

通用的数据集上依旧保证了较高的准确度。

Tabelle 6: Randomized Addition Procedure(随机加法过程)

Size	3-5 digits	7-10 digits
GPT-4	100%	85.5%
GPT-3.5	99%	72%
llama2-7b	49.5%	0.5%
llama2-13b	28.5%	2%
llama2-70b	84%	11%
Google-PaLM	94%	39.5%
Qwen-72b-Chat	97%	75%
MathRuleGLM	100%	100%

Tabelle 7: 3种特定的数据集实验结果

Task	进位加法	负差	逐位差分
GPT-3.5	97.2%	99.4	95.29%
GPT-4	100%	97.8%	98.32%
llama2-70b	19.4%	50.2%	76.09%
llama2-13b	5%	20.8%	21.89%
llama2-7b	3.6%	4%	2.3%
Google-PaLM	52.6%	43.6%	95.96%
Qwen-72b-Chat	85.8%	86.4%	93.94%
MathRuleGLM	100%	100%	100%

6 结果和讨论(Result And Discussion)

6.1 结果

对于算术任务，我们的模型MetaMathRule模型是基于Transformer的模型进行预训练，其模型参数为3000万个，用于预训练和推断。为了准确评估MetaMathRule的有效性，我们将其性能与主流大语言模型（LLMs）如GPT-4和ChatGPT等进行了对比。结果一致表明MetaMathGLM在各种任务上均优于所有其他模型，在处理算术任务方面性能出色。此外我们是更小的模型变体，仅有3000万个参数，结果却显示了一个令人惊讶的现象。尽管其参数规模较小，但MetaMathRule在一系列综合算术任务中均优于GPT-4和ChatGPT。这一令人惊讶的结果表明了MetaMathRule的方法的有效性，该方法涉及将复杂的算术表达式分解为多个步骤，使其能够根据已经学习的各种规则，理解算术任务中的微妙之处。它有效地学习了算术运算的基本规则和原理，使其能够生成准确和精确的解决方案。总而言之，对复杂算术任务的评估结果突显了MetaMathRule的卓越性能。通过分解算术任务，这些模型明显超过了当前主流大语言模型(LLMs)的性能。
我们所希望我们模型所展现出来的不仅仅是精确的计算能力，更重要的是其泛化能力。一个成功的模型必须能够从仅仅几个例子中以系统的方式学习和使用规则，并偏好捕获结构化输入，输出关系的假设。我们的模型，在面对一个未知任务时，依靠的正是这种泛化能力才能克服对先前系统性的限制。这种方法旨在模拟成人的组合技能，而不是成人获取这些技能的过程。我们的模型在仅仅学习了低位数的加法后，通过规则的组合，可以对从未见过的高位数进行相应的计算，并且对高位数计算的准确度高于大部分现有的SOTA模型。MetaMathRule的总体目标围绕着语言模型在逻辑推理问题的泛化能力上的体现，通过在数学推理领域的能力来展现。从上述实验结果中可以清晰得看到我们的模型不论是在各种特殊数据集以及随机数据集上的测试结果均是最优的.特别在某些高位数的计算中,我们的模型的准确度是远高于现有的大部分语言模型的，并且我们可以在模型的各种复杂的算式中表现均为良好.其中我们可以看到如下结果

6.1.1 特殊加减法测试结果

1. 在完美十进制加法计算中,我们可以清晰的看到，当位数提高以后，大部分主流语言模型的准确度都有明显的下跌，chatgpt-3.5和chatgpt-4相对下跌缓慢，说明除了我们的模型以及chatgpt对于进位是有一定的理解能力的。

Tabelle 8: Perfect Decadic Addition(完美十进制加法)

Size	3-5 digits	7-10 digits
GPT-4	100%	98%
GPT-3.5	97.2%	91.5%
llama2-7b	19.4%	5%
llama2-13b	3.6%	3%
llama2-70b	5%	6%
Google-PaLM	52.6%	27.5%
Qwen-72b-Chat	85.8%	67.5%
MetaMathRule	100%	100%

2. 对于反向幅度减法而言的问题，更加佐证了我们的观点，我们可以发现，对于llama2模型而言，当位数变大以后，准确率接近0了，而当位数比较小的时候，却能够进行适当的计算，从中可以看出这就属于我们所谓的背题，通过回忆问题得到答案，这种学习方式下，语言模型的学习能力会受到极大的限制。而我们的模型依旧能够达到百分百的准确度，chatgpt虽然也能够大致的理解，但是其超高的准确度下更多的是巨大规模数据的训练，而我们的模型是真正从逻辑出发，通过迭代的方式，一步一步将问题回答出来，同时表现出强大的泛化能力。

3. 在交错式位值减法任务中,同样可以看出，在5位数的时候大部分普通模型还是具有一定的计算能力,但当位数一旦变高以后，大部分语言模型就无法进行相对精确的计算。如LLama2的各种参数大小的模型,准确率均不超过5%.我们的模型按照所学得的规则，严格推理能够进行严格推导得到我们所预期的结果,甚至超越了当前最强大的chatgpt模型.

同时我们的模型所使用训练数据也是最少的,我们仅训练了极少量的数学算式,并且使用了少量规则来处理这个问题.所以总的来说我们的模型是相对简单但是又更加富含内容的模型.

Tabelle 9: Reverse Magnitude Subtraction(反向幅度减法)

Size	3-5 digits	7-10 digits
GPT-4	97.8%	96.5%
GPT-3.5	99.4%	88.5%
llama2-7b	20.8%	0.0%
llama2-13b	4%	0.0%
llama2-70b	50.2%	0.5%
Google-PaLM	43.6%	0.0%
Qwen-72b-Chat	86.4%	3.5%
MetaMathRule	100%	100%

Tabelle 10: Interleaved Subtraction(交错式位值减法)

Size	3-5 digits	7-10 digits
GPT-4	98.3%	96%
GPT-3.5	95.2%	91%
llama2-7b	2.3%	0%
llama2-13b	21.9%	2%
llama2-70b	76.1%	2%
Google-PaLM	95.9%	19%
Qwen-72b-Chat	93.9%	74.5%
MetaMathRule	100%	100%

6.1.2 随机加减法测试结果

Tabelle 11: Randomized Subtraction Procedure(随机减法过程)

Size	3-5 digits	7-10 digits
GPT-4	100%	78.5%
GPT-3.5	95.5%	76.5%
llama2-7b	25.5%	3%
llama2-13b	31%	1%
llama2-70b	73.5%	12.5%
Google-PaLM	80.5%	47%
Qwen-72b-Chat	93.5%	81.5%
MetaMathRule	100%	88%

从随机的测试结果来看，各类语言模型在测试过程中，做加法的性能是明显是优于减法的，而且我们上面所列出的三种特殊数据集仅仅代表数值计算中三类推理略多的问题类型，实际计算中还会有更会有更复杂的计算问题。而在随机生成的加减算式中，可以看得出对于低位数而言，我们的模型以及当前大部分模型都能有较高的准确率，而一旦计算位数变大以后，大部分语言模型的性能都急剧下降，除了chatgpt以外，其他模型由于并未理解计算的规则，在计算高位时极易出现误差。与此同时，我们的模型在高位的随机加法计算过程中依旧能达到百分百的准确率，但是在对于高位随机减法的过程中也遇到了部分问题，但是总体来看我们的模型的准确率依旧是在所有语言模型中准确率最高的，甚至超越了chatgpt 10%的准确率。

Tabelle 12: Randomized Addition Procedure(随机加法过程)

Size	3-5 digits	7-10 digits
GPT-4	100%	85.5%
GPT-3.5	99%	72%
llama2-7b	49.5%	0.5%
llama2-13b	28.5%	2%
llama2-70b	84%	11%
Google-PaLM	94%	39.5%
Qwen-72b-Chat	97%	75%
MathRuleGLM	100%	100%

6.1.3 向量外积求解的结果

我们通过训练两种任务的规则,对模型进行训练,我们可以发现,我们的模型对于两种不同的任务,可以在一个架构的模型下进行学习,体现出了我们模型强大的泛化能力.而且我们的模型在针对两种任务,能够利用两种任务中公共的规则,极大的提高了我们模型学习的效率.并且对于外积这类复杂的任务,我们的模型也能够进行完美的计算,准确得到我们的预期结果.

6.2 讨论

6.2.1 可控性能

我们知道现有的语言模型虽然功能已经很强大了,但是依旧在某些方面上有很大的挑战,其中一个方面就是可控性能上,大部分语言模型无法对我们的问题在可控范围内进行回答,往往容易出现很大的偏差.这个是我们面临的很大的一个问题,我们的模型相对而言,具有较好的可控性,但是这个可控性能也是在我们模型的一个非常有限的范围之内,比如我们的模型在计算10位数以内的问题效果良好,但是计算更高位数的时候,会出现不可控的现象,会于实际结果误差很大,这些都是我们依旧面临的问题,我们致力于将我们的模型在之后的优化中加入其他规则数据,使得模型整体变得更加可控的.

6.2.2 模型架构的选择

我们的模型虽然在解决数字问题上性能远远优于当下流行的模型,但是我们的模型架构中可以看出我们的模型并不能一步到位完成任务,我们的模型在面对一个复杂的问题时,是通过一步一步前进,把问题逐渐解决,以得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,是否能够明显提高我们模型的效果.

7 总结与展望(Summary and Outlook)

7.1 总结

我们受到元学习的启发,通过让我们的语言模型仅仅在学习了九九加法表以及简单的数值实验后我们的模型通过增加大量的实验增加了我们模型的特殊能力.
在本文中，我们的主要焦点是令语言模型学习组合技能以模仿人类解决问题的泛化能力,我们通过对数学运算任务以及复杂推理任务作为任务示例，通过采用迭代模型的架构，从头开始训练一个基于Transformer的语言模型。通过对复合规则以及子规则的数据集进行全面训练，我们确定一个拥有3000万参数的语言模型在仅学习个位数算术任务以及一些计算规则，通过类人组合技能的方式，在面对未见过的高位数算式以及复杂的推理任务时，获得了出色的表现，超过了当前主流的大型语言模型。这一发现有力地挑战了当前认知，即语言模型也可以具有一定的逻辑理解能力。现有的语言模型因为理解能力有限，在处理多位数计算等严格逻辑问题时，不依赖外部计算辅助的情况下，在执行准确的算术操作方面存在限制。我们训练的模型不仅能够进行的精确计算问题，同时还展现出了语言模型强大的泛化能力和逻辑理解能力。

7.2 展望

本文的研究工作还有诸多值得探讨的问题，例如：

1. 虽然我们的模型在一定规则学习下能够表现出一定的泛化能力和理解能力,但是由于算力有限，我们的模型所处理的问题种类上是非常有限的,我们期待在扩大模型容量的同时并使用更多规则数据集训练，我们模型能够同时在多种不同的逻辑任务上均得到良好的表现。

2. 我们的模型在处理问题的细节上还有待提高，模型对部分问题的细节没能做到绝对的准确，对于人类而言，如果不考虑计算错误，只要方法正确就能百分百回答问题，而我们的模型虽然在大部分问题上能够模仿人类精确解决问题，但模型未能泛化到它未针对优化的归纳偏见中的细微之处，比如对0的理解，我们的模型很容易机械的理解成一种正数或者负数，无法像人类一样，把0进行特殊处理。即我们的模型在处理问题的细腻程度与人类还有较大的差距，需要通过更多的规则加以规范。

3. 我们的模型不能自动处理未经学习的泛化形式或超出元学习分布之外的概念，这减少了它能正确处理完全新颖结构的范围。因此，它是否能够实现类似人类的系统性，在所有方面并从现实的训练经验中，仍然有待确定。

Literatur

[1] 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.

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

[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.