大语言模型合成数据方法简述

李培基*
上海人工智能实验室
复旦大学计算机科学技术学院
lipeiji@pjlab.org.cn

马逸川*
上海人工智能实验室
上海交通大学电子信息与电气工程学院
mayichuan@pjlab.org.cn

颜航
上海人工智能实验室
yanhang@pjlab.org.cn

摘要

大语言模型在过去两年受到了极大的关注，并引起了对通用人工智能的广泛讨论。为了实现通用人工智能，合成数据被认为是其中非常关键的一环。本文将当前常见的数据合成方法归为三类，基于蒸馏的合成数据、基于模型自我进化、基于工具的合成数据。针对每一类合成数据方法，我们简要介绍了几种主流的做法，以期概览各类方法的基本思路以及异同。当前大部分合成数据方法都基于蒸馏，尽管这些方法取得了良好的效果，但其实质是将更强的大模型蒸馏到更小的大模型。这样的方法从降低大模型推理成本的角度具有实际意义，但对于进一步提升大模型能力上限作用有限。基于模型自我进化和基于工具的合成数据研究相对偏少，对于持续提升模型能力，这两个方向需要有更多探索。

关键词： 合成数据 ； 大语言模型 ； 知识蒸馏

A Brief Introduction to Synthetic Data for Large Language Model

Abstract

Large language models have gained significant attention over the past two years, sparking extensive discussions on artificial general intelligence (AGI). Synthetic data is considered a crucial component in achieving AGI. This paper categorizes the current common synthetic data methods into three types: distillation-based, model self-evolution-based, and tool-based. For each type, we briefly introduce several mainstream approaches to provide an overview of the basic ideas and differences among these methods. Most current synthetic data methods are based on distillation. Although these methods have achieved good results, their essence lies in distilling a more powerful large model into a smaller one. This approach is practically significant for reducing the inference cost of large models but has limited effectiveness in further enhancing the upper limit of large model capabilities. Research on model self-evolution-based and tool-based synthetic data is relatively scarce, and more exploration is needed in these two directions for continuously improving model capabilities.

Keywords: Synthetic Data , Large Language Model , Knowledge Distillation

* 共同一作

第二十三届中国计算语言学大会论文集，第86页-第97页，太原，中国，2024年7月25日至28日。
卷2：前沿综述

1 引言

大语言模型（Large Language Model，简称LLM）自2022年11月底ChatGPT问世以来受到了各个行业的持续关注，由于其出色的文字掌握能力和逻辑推理，LLM被广泛应用于代码生成[0]、文案写作、AI for Science (Zhang et al., 2024; Jablonka et al., 2024)等领域。尽管各类最先进大模型的具体训练方法并未披露 (OpenAI, 2023; Anil et al., 2023; Anthropic, 2024)，但数据被普遍认为是决定大模型能力的最重要因素 (Touvron et al., 2023a; Bai et al., 2023; Cai et al., 2024b)。

规模定律（Scaling Law）表明大模型的性能会随着使用训练数据量的增加而持续攀升 (Kaplan et al., 2020; Hoffmann et al., 2022)，是否能够持续产生训练数据将很大程度上决定大模型未来的能力上限。然而Epoch AI的预测表明人类将在2026年耗尽高质量数据[1]，尽管近期的一些研究表明可以通过重复使用数据的方式继续提高模型的性能 (Muennighoff et al., 2023)，但这种性能收益会在多次重复之后持续衰减。因此通过合成数据的方式来持续获得高质量训练语料成了一个值得研究的科学问题。除了数据短缺带来的对合成数据的需求，数据的长尾效应也使得在某些场景下必须要借助合成来进行补充，例如一些复杂数学推理 (Trinh et al., 2024a)和代码数据 (Li et al., 2022)，这些场景下需要比较专业的人才来标注相关数据，这类数据在天然文本中数量较少，同时标注成本较高。一些常识推理的数据，尽管对人类来说难度不高，但这类推理路径数据在自然文本中分布也很少。此外，在某些场景下可以直接利用工具进行大量数据合成，从而让模型掌握某些特定能力，例如四则运算 (Yang et al., 2023b; Yuan et al., 2023)。

在本文中，我们将合成数据定义为：借助大模型或者工具生产的数据。在此定义下，本文根据合成数据使用到的方法将过去的工作分成了三类：基于蒸馏的合成数据、基于模型自我进化的合成数据以及外部工具合成数据。其中基于蒸馏的相关工作主要集中在从性能更优的私有模型中获取训练数据，在开源大模型上进行继续训练，缩小开源模型与闭源模型间的性能差异 (Xu et al., 2023; Yu et al., 2023; Wang et al., 2024)；基于模型自我进化的合成数据从前景上，更有可能解决数据的短缺问题，目前这类方法效果提升相较前一种方案不明显，但这类方法未来有很好的发展前景，特别是对于如何持续提升大语言模型性能上限上有重要意义 (Wang et al., 2023b; Zelikman et al., 2022)；除了前两种需要借助语言模型的方法，还可以通过配合使用工具 (Yue et al., 2023; Singh et al., 2024)或者完全依赖工具来构造训练数据 (Yuan et al., 2023; Trinh et al., 2024a)，借助工具的方法可以利用工具的可靠性，提升合成数据的准确性，例如通过规则构造的四则运算数据就不会存在错误，但如何在合适的领域利用对应的工具需要领域知识。尽管自然语言处理领域早在ChatGPT诞生之前就在使用合成数据训练模型，例如ZeroGen (2022)利用生成式模型为判别式模型生成训练数据，Ding等人 (2023)使用GPT-3来进行数据标注训练模型，但本文的讨论的方法主要集中在ChatGPT之后，并且主要以呈现近期合成数据主流方法为主，每种方法主要介绍几篇相关工作，如需更详细和全面的了解，请参阅近期的相关综述论文 (Xu et al., 2024; Liu et al., 2024)。

接下来本文将首先介绍以上三类合成数据方法的典型做法，然后针对这些方法的优缺点以及当前合成数据的不足进行讨论，最后提出对合成数据发展的展望。

2 基于蒸馏的合成数据

知识蒸馏（Knowledge Distillation）通过一定的方法将大模型的知识迁移到小模型上，这样可以在推理的时候使用更小的模型获得相近的性能 (Lin et al., 2021)。过去的知识蒸馏方法一般假设可以获取到大模型的输出概率分布 (Gou et al., 2021)，甚至可以获取大模型的中间层输出 (Jiao et al., 2020)，但随着模型能力的增强，一些商业模型选择了闭源其模型，用户只能拿到其预测的结果。在这种情况下，只能通过模仿大模型的输出来实现蒸馏 (Wallace et al., 2020; Wang et al., 2023b)。自OpenAI于2022年11月底发布ChatGPT以来，各家商业公司的大模型能力不断攀升 (OpenAI, 2023; Anil et al., 2023; Anthropic, 2024)，这些模型都选择了闭源，开源社区模型如果想要复刻这些模型的性能，最简单的方法便是蒸馏这些模型的能力。

[0] https://github.com/features/copilot

[1] https://epochai.org/blog/will-we-run-out-of-ml-data-evidence-from-projecting-dataset

在这一节中，我们将讨论利用一些更强模型，例如ChatGPT、GPT-4 (OpenAI, 2023)，来提升开源模型的方法。由于这些方法都利用了更强的模型，所以它们的性能增益很大一部分来源于对更强模型的知识蒸馏。在过去两年涌现的相关方法，由于没有办法直接获取到模型的输出概率分布，基本均通过让语言大模型生成文本回答，模仿该回答来提升模型能力 (Shridhar et al., 2022; Magister et al., 2022; Fu et al., 2023)。这些方法都等同于扩大了训练数据量（通过扩展了回答的推理路径或者直接扩充了更多的数据条数），这类方法从本质上也类似于传统的数据增强（Data Augmentation） (Feng et al., 2021; Xu et al., 2024)。根据这些方法是否需要利用标注数据标签（例如数学题目的答案），我们将这类方法进一步细分为：借助监督数据标签进行合成和无需监督数据标签的数据合成。

2.1 借助监督数据标签进行合成数据

在OpenAI公司的GPT-3 (Brown et al., 2020)问世之后，人们发现其在经过思维链（Chain-of-Thought）的提示之后可以显著提高推理任务的回答准确率 (Wei et al., 2022)，这种特性同时也在谷歌公司推出的大模型PaLM上得到了印证 (Chowdhery et al., 2023)。经验性的结论发现，模型的参数量需要超过一定的量级才能涌现出一些能力 (Schaeffer et al., 2023)。但这些模型都太大了，推理成本高昂，研究者们希望能够找到方法缩小模型的尺寸同时保持其推理能力，因此Magister (2022)、Shridhar (2022)和Ho (2022)等人尝试了通过借助语境学习（In-Context Learning）的方法来生成思维链数据。

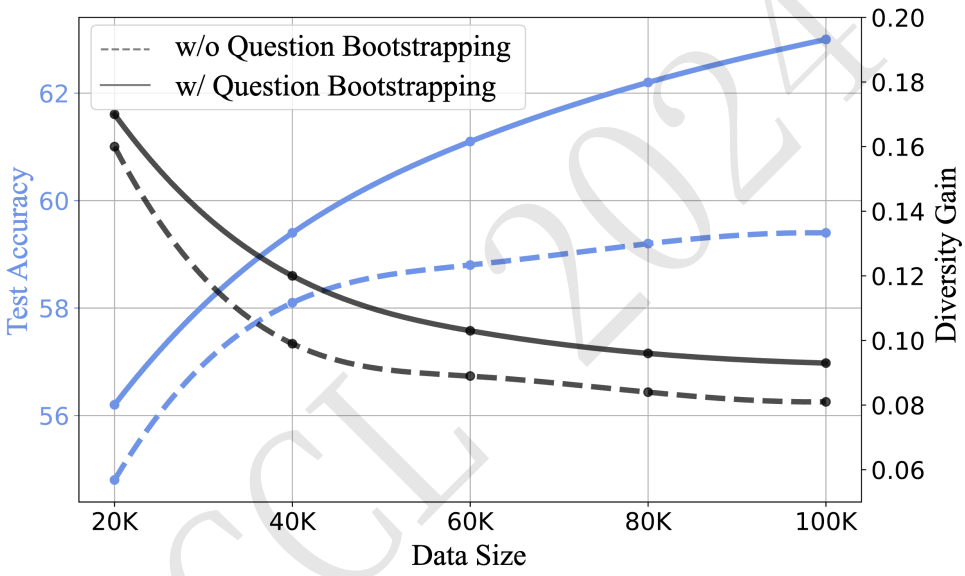


Figure 1: 模型性能随着微调数据量增加的曲线，如果不对数据中的问题进行增广（蓝色虚线），随着训练数据量增加，模型性能增长非常缓慢(Yu et al., 2023)

利用上述的方法可以获得大量的思维链训练数据，然而问题的数量被固定为了原数据集中问题的数量，这会导致训练数据的问题多样性不足，影响模型性能持续提升，如图 1所示。为了规避这个问题，研究者们提出了多种方法来可靠地增加问题的数量。MetaMath (Yu et al., 2023)提出可通过同义改写题目、逆向推理的题目增强以及同义改写答案来增加数据多样性。其中逆向推理的推理题目增强方法为任意将原文中的某个数字更换为未知数，让模型求解其未知数，例如原题为“农场的母鸡每日可以产蛋10颗，3天可产蛋多少颗？”，可修改为“农场的母鸡每日可以产蛋x颗，3天可产蛋30颗。问x的值为多少？”。不论是同义改写还是增强题目方法，都需要借助一个聪明的模型，通过输出最终的正确答案来筛选生成的数据，在该论文中使用了OpenAI的GPT-3.5-Turbo来进行数据生产。

通过借助程序来计算数学解答过程中的一些数值计算可以避免出现因为计算带来的错误，MAmmoTH (Yue et al., 2023)提出解题过程中可以生成类似于思维链的程序链（Program of Thought）来进行解题，即在问题求解过程中直接将解答过程写成可执行的代码，通过代码解释器执行完这些代码得到的结果作为预测结果，这样可以让针对相同的问题提出不同的解

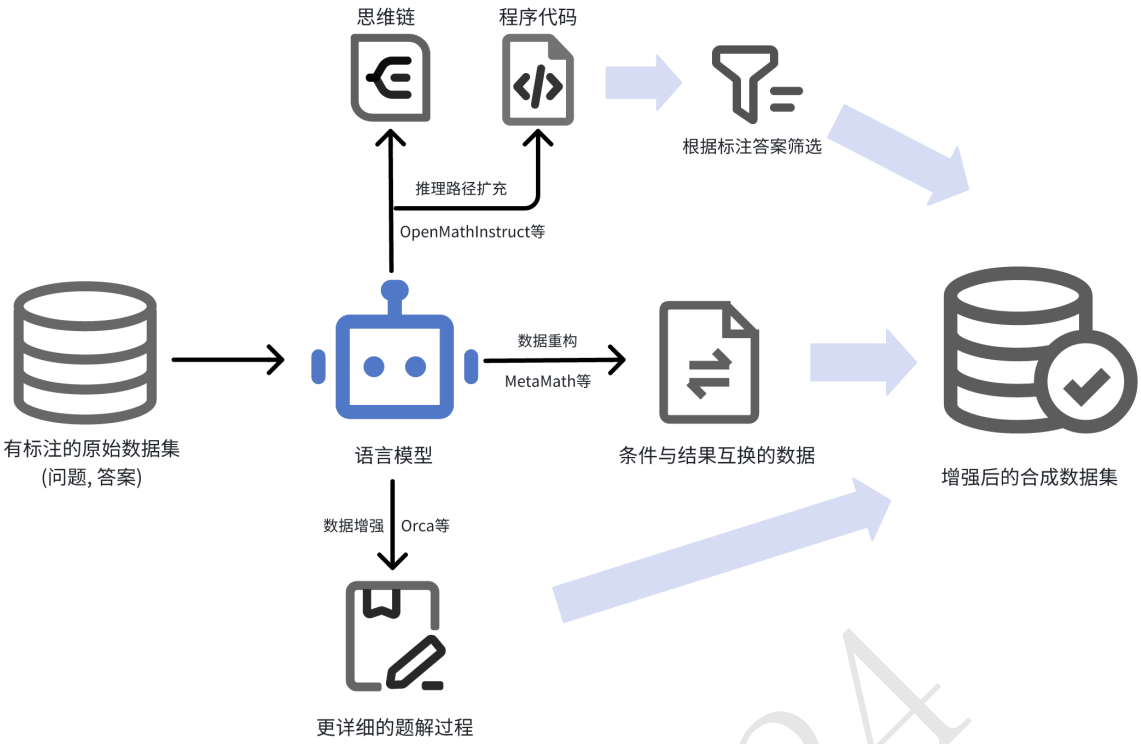


Figure 2: 借助监督数据标签可通过各种方法扩充训练数据

法。同样类似于直接生成答案的方法，如果代码执行完毕之后，预测结果与答案不符合则认为生成代码有误，不予采用。

如果有监督数据标签，除了通过上述的丢掉不符合答案的推理路径方案外，还可以尝试直接将答案用于解答过程的生成。Orca (Mukherjee et al., 2023)尝试将FLAN (Longpre et al., 2023)中的数据集通过大语言模型推理构建回答，并在模型回答前提前将答案拼接到输入中，然后输入诸如下面的提示词“假设我只有5岁，请一步步思考，并将为什么答案是这样解释给我听”，这样可以让小模型学习并模仿大模型的思考过程。在Orca中，作者总共通过GPT-3.5生成了500万条数据，其中的100万条通过GPT-4进行了再次生成。尽管这份数据没有直接开源，社区在这个思想的基础上进行了复现 (Lian et al., 2023)，并将复现结果进行了开源 [2]。

在本小节中我们介绍的几种方法被汇总到了图 2，这类方法由于利用了监督信号来辅助数据生产，所以可以在一定程度上保证生成数据的质量，使得这种方法生成的训练数据在下游任务提升上效果都比较显著。此外，类似的思想也可以用于代码生成之后，在代码中，可以通过生成的代码能否通过单元测试来判断生成的代码是否有误。但正如Orca (Mukherjee et al., 2023)指出的那样，能够利用的数据不够多是这类方法一个比较明显的缺点，尽管Orca里面利用了FLAN来扩大可利用的监督数据集，但这个数据量离预训练数据量还有两到三个数量级的差异。

2.2 无需监督数据标签的合成数据

在过去的蒸馏方法中，可以借助大量的无标签数据集来将大模型的能力迁移到小模型上。同样地，在大语言模型时代仍然可以利用相同的思想。在这一节中我们主要讨论三个相关的方法：（1）借助进化的思想，让合成数据的难度不断增加；（2）通过大模型生成更具教育意义的数据；（3）从预训练语料中挖掘高质量数据。

WizardLM (Xu et al., 2023)设计了Evol-Instruct算法来避免依赖人类生产高质量的指令。Evol-Instruct算法的思想如图 3所示，通过给大模型施加指令，使其在一个初始指令的基础上逐步增加指令的深度和宽度，从而不断提高指令难度。由于不确定大模型会从哪个方向

[2] https://huggingface.co/datasets/Open-Orca/OpenOrca

演化指令，因此生成的指令也不一定有标准答案，此刻就需要利用大模型来根据生成的指令进行回答，收集这些复杂指令和对应的回答可以构成训练数据集用于微调开源模型。通过迭代不断生成更复杂指令的思想也被广泛使用在了后续的工作中 (Luo et al., 2023a; Luo et al., 2023b)。

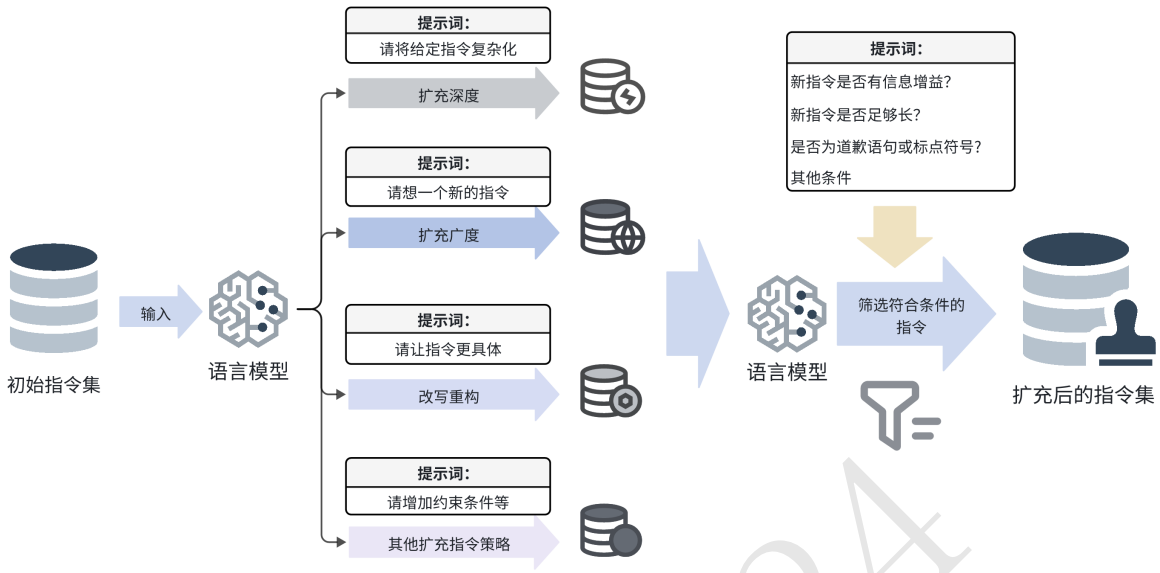


Figure 3: 通过多种策略丰富指令的数量和类型

受到人类学习启发，Gunasekar等人 (2023)提出利用大模型生成更具教育意义的训练数据能够使小模型更快地进行训练。基于此原则，作者发现许多代码数据是不具备良好的教育意义的，因为：（1）很多数据不完备，需要来自其它代码文件的信息；（2）很多代码块都只是类似变量定义等无意义的操作，不具备学习的价值；（3）一些具备复杂逻辑的代码没有很好地注释；（4）存在严重的长尾效应。为了避免这些问题，作者提出了借助大模型来生产更具教育意义的数据，具体来说：（1）使模型仅保留预训练文本中具有教育意义的内容；（2）让大模型直接生成“代码教科书”；（3）让大模型生产“课后习题”。通过以上方案，作者发现一个仅1.3B的代码模型可以在代码相关评测集上达到10倍其规模模型的性能。随后作者在此基础上，将相关思想推广到了推理类文本数据上 (Li et al., 2023b)，并取得了优异的效果。这类大量数据生成的方法容易遭遇模型生成过程中与已生成内容重复的问题，为了避免这个问题，作者首先收集了两万个主题，在模型生成时通过给定主题让模型尽量不重复。

除了通过限制主题来避免得到重复的内容，还可以通过借助大量的预训练语料来获取高质量合成数据。Yue等人 (2024)提出首先从互联网数据中筛选出相关文档，然后抽取出其中的问答对，最后使用开源大语言模型对问答对进行润色的方式来获取大量的训练数据。在筛选阶段，作者训练了一个基于Fasttext (Joulin et al., 2016)的文本分类器来从CommonCrawl (Com,)数据中分类出可能包含高质量问题-答案对的文档，在这个过程中大模型可以用于标注正样本数据，负样本数据则通过随机采样产生；然后通过去掉网页数据中的HTML标签以及广告，在这一步基础之上通过借助开源大模型，如Qwen (Bai et al., 2023)，判断文档中是否存在自然的问题-答案对，如果存在则让模型提取出；在抽取出的问题-答案对中，部分数据只含有问题和对应的答案，缺乏对相关过程的展示，则需要使用大模型来补足这部分内容。通过以上步骤可获取超过千万条相关数据，在上面训练的各种7B大小的模型取得了良好的性能。但由于生产这批数据使用的是Qwen-72B，一定程度上可以将整个过程看做是在蒸馏Qwen-72B模型。

相较于上一小节中借助标注数据标签来确定合成数据质量的方案，不利用标签的合成方法容易导致生成数据的质量不可控，因此不利用标签的方案尽管理论上可以生产近乎无限的数据量，但如何在质量和数据量间取得一个很好的平衡仍然是这类方法需要解决的问题。在代码领域，Song等人 (2024)通过让大语言模型补充代码数据注释的方式产生了超过100B词元的训练语料，由于注释即使发生错误也不影响代码执行的正确性，因此这样可以避免因为引入错误数据

而导致模型训练变坏的问题。此外，让大模型在生成数据过程中保持多样性和生成数据质量的平衡是未来这类方法另一个需要解决的难题，在Evol-Instruct方法中多样性提高的时，也容易出现合成指令不合理或无法被现有大模型很好解决的问题。

利用上述的方法一些工作被总结在表 1中，可以看出这些方法有以下两个特点，第一个是都利用了更强的模型，第二个是都主要集中在代码和数学等领域。将这些方法应用到其他领域仍然需要回答两个问题，其一是这些被蒸馏的模型是否在除代码和数学之外的领域仍然有很高的蒸馏价值，其二则是通过生成思维链进行蒸馏的方法是否对其它领域也适用。

论文	更强的大模型	下游模型	评测任务	借助监督标签
WizardLM(2023)	GPT-3.5	LLaMA-7B	Evol-Instruct、Vicuna	否
Orca(2023)	GPT-3.5、GPT-4	LLaMA-13B	BBH、AGIEval等	是
WizardCoder(2023b)	GPT-3.5	StarCoder-15B	HumanEval、MBPP等	否
phi-1(2023)	GPT-3.5	-	HumanEval、MBPP	否
MetaMath(2023)	GPT-3.5	LLaMA-2系列	GSM8K、Math	是
Mammoth(2023)	GPT-4	LLaMA-2系列	GSM8K、MATH等	否
MathCoder(2023a)	GPT-4	LLaMA-2系列	GSM8K、MATH等	是
WaveCoder(2024)	GPT-3.5、GPT-4	StarCoder-15B等	HumanEval、MBPP等	否
OpenMathInstruct(2024)	Mixtral 8x7B	LLaMA系列	GSM8K、MATH	是
Orca-Math(2024)	GPT-4	Mistral-7B	GSM8K	否
Xwin-Math(2024)	GPT-4	LLaMA-2 7B	GSM8K、MATH等	否
Mammoth2(2024)	Mixtral-8x22B等	Mistral-7B等	GSM8K、MATH等	否

Table 1: 基于蒸馏的合成数据工作

3 基于模型自我进化

除借助更强大模型合成数据的思路外，也有大量的工作着眼于利用较弱模型自身的能力，使用EM算法或强化学习等算法，通过框架和合成方法的设计帮助模型进行自我进化(Self-Evolution)。从方法和原理上看，这类工作与上一节基于蒸馏的方法没有本质区别，但这类方法由于不借助更强的大模型，因此具有更高的理论上限，同时也能避免一些法律许可相关的问题[3]，因此我们单独把这一类方法列为一节进行讨论。

在STaR方法 (Zelikman et al., 2022)中，作者利用GPT-J (Wang and Komatsuzaki, 2021)模型来生成大量的推理路径，并将这些推理路径用作训练数据微调模型自身。由于这个方法是在有监督标签的数据集上，因此判断这些推理路径是否正确时可以采用对比最终答案和标准答案的方式。除了直接使用对比答案的方法，在Self-Improve (Huang et al., 2022)中作者提出了通过采样生成多次答案，然后进行多数投票的方式选择可能的答案，并将此作为可用的训练数据用于训练模型。此外，指令回译 (Li et al., 2023a)方法也可以在不需要标准答案的情况下合成数据，具体来说，模型通过反向利用指令数据集，即将回答作为输入，将指令作为输出训练了一个指令生成模型，得到该生成模型后可以将大量的自然文本作为潜在的回答，并让该回译模型生成潜在的指令，这种方式得到的指令和回答数据质量可能参差不齐，因此论文提到也需要借助指令数据集训练一个正向模型，该正向模型可以用来评估生成的指令和回答的质量是否可用，并从中筛选出高质量的部分。筛选出来的数据可以继续用于迭代训练回译模型和正向模型，经过多轮迭代的模型可以取得良好的指令跟随能力。

受Alpha-Zero (Silver et al., 2017)的启发，Alpha-Math (Chen et al., 2024)构建了一个由蒙特卡洛树搜索（MCTS）驱动的推理算法，在采样推理路径时通过MCTS充分激发模型的推理潜能，通过策略和价值网络的协同使用进行节点的增长。其中策略网络为原始的模型，而价值网络通过在原始模型上添加一个带有tanh激活函数得到。在搜索过程中，论文将MCTS推理过程简化为步级别的束搜索（Step-level Beam Search）操作，迭代地从最优的步骤中采样多组方案，使用价值网络对不同的采样方案进行评估并得到更新后的最优步骤。通过使用价值模型对不同的推理路径进行评估，模型的推理表现得到了显著的提升。

除了以上这些显式通过大模型生成数据的自我进化方法，模型可以通过将上一代模型用于下一代模型的数据生产实现不同代模型间的逐步提升。例如在LLaMA-3 [4]中Meta的研究员利

[3]OpenAI的法律条款中规定不能使用ChatGPT生成的数据训练基础模型

[4]https://ai.meta.com/blog/meta-llama-3/

用LLaMA-2模型 (Touvron et al., 2023b)来识别高质量语料，实验发现LLaMA-2可以可靠地为文本质量分类器生产训练数据。通过上一代模型为下一代模型提供助力的方式应该具备广阔的前景。

目前来说，基于模型自我迭代增强的数据合成方法较少，未来有很大的研究空间。并且基于已有的方法来看，其效果随着迭代次数的增加并没有特别好的规模（Scaling）效应，未来如何提高迭代方案的性能上限是一个值得研究的问题。更多相关的探讨可以参阅论文 (Burns et al., 2023)及引用了该论文的后续论文。

4 基于工具的合成数据

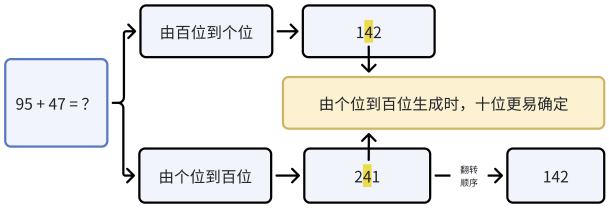
有许多工作在合成数据的过程中使用了外部工具，如代码解释器、计算器、推理引擎和抽象语法树等。外部工具可以为合成数据提供额外的信息。工具所提供的信息可能是不充分的或冗余的，但一般是客观、真实的。这些信息往往并不足以直接作为合适的训练数据，但能在合成数据的过程中为模型提供信息增益，从而合成更有效，更准确的数据。

在使用语言模型合成代码数据的工作中，代码解释器是一个天然的辅助工具。代码解释器的执行结果能带来额外的信息增益，这部分信息可以用于筛选合成数据的正确性，或者作为推理过程的中间结果。MAmmoTH (Yue et al., 2023)调用模型对已有数据集中的问题生成Python代码，并调用代码解释器返回代码的执行结果，通过比较代码执行结果与标准答案的异同，即可筛选出正确的代码数据。而MathCoder (Wang et al., 2023a)和OpenMathInstruct (Toshniwal et al., 2024)等工作使模型在推理时生成含有内嵌代码段的题解，在得到代码段的执行结果后模型将继续自回归的推理过程。Cai等人 (2024a)使语言模型充当工具制作者，为特定任务编写Python函数作为工具，同时为函数编写相应的测例。在工具通过所有测例后，另一个LLM将作为工具调用者解决实际问题。还有一些工作利用代码解释器搭建了交互式的框架 (Shypula et al., 2024; Yang et al., 2023a; Shinn et al., 2023)，使用强化学习等算法，在模型生成代码数据后，利用代码解释器提供的信息对合成数据进行分类或更正，进而迭代优化模型。Ni等人(2024)在代码修复任务中通过使用注释的方式将调试信息加入代码中，使得模型可以基于调试信息修正代码中的Bug。

尽管大模型在文字生成方面取得了良好的效果，但是它们却不能很好地计算基础的四则运算。为了让大模型能够在数值计算的时候减少错误，过去的方法尝试了通过规则生成大量的四则运算等式以及基础数学等式作为训练数据 (Yuan et al., 2023; Yang et al., 2023b)，这类数据的一些示例如图 4(a)所示。但这类方法主要依靠让模型背下来所有的计算，不太具备泛化计算的能力。人类在计算这类复杂运算时一般会通过草稿纸的形式，而非靠记下所有的四则运算，受此启发，Lee等人 (2023)提出通过Scratchpad的形式计算复杂运算，一个简单的示意如图 4(b)所示，通过加入对任务的理解，可以极大降低模型学习的难度，结合领域知识对于用好工具合成数据很重要。

运算类型	示例
四则运算	$7 + 1825.4 = 1832.4$
乘方运算	$7^3 = 343$
对数运算	$\ln 1523 = 7.328437$
三角运算	$\cot(313\pi/306) = 13.890726$

(a) 遍历各类运算



(b) 改变计算方式减少学习难度

Figure 4: 遍历四则运算与Scratchpad示例

借助数学工具，不但可以合成简单的四则运算，甚至可以合成比较复杂的数学推理数据。Lean (de Moura and Ullrich, 2021)是一个功能强大的交互式定理证明器和编程语言，主要用于形式化数学和计算机科学的证明，通过Lean可以实现自动验证数学证明。因此可以通过将大量数学证明问题转换为Lean的形式，并让语言模型使用Lean语言生成大量证明，通过Lean的定理验证器从中选择推理正确的数据，我们就可以生产出大量的严格正确的数学证明。通过这种方式大模型已经能够完成一些国际数学奥林匹克大赛的题目 (Ying et al., 2024; Xin et al., 2024)。除此之外，AlphaGeometry还通过学习几何推理引擎合成大量的高难度集合证明题，在

国际数学奥林匹克竞赛（IMO）的几何证明题目上取得了惊艳的结果 (Trinh et al., 2024b)。几何推理引擎能够在给定前提的基础上，利用已知的欧式几何规则，反复生成新的结论直到所有结论穷尽，即生成给定前提的推理闭包。然而，仅仅依靠几何推理引擎并不能够解决大部分证明问题，启发式的辅助点添加策略是必不可少的，而AlphaGeometry使用语言模型来替代了原先启发式添加辅助点的策略。由以上的工作可以看出，为了更好地利用各类工具，合成数据需要各领域相关的专业知识，大模型未来的数据合成需要和不同行业的专家进行合作。

使用工具合成数据最大的优势是数据的正确性比较容易有保证，例如四则运算和定理构造等都是基于符号化推理得到的结果，因此其结果出错概率很低。但如何比较好地利用工具，如何挑选用于合成数据的工具，这些需要相关的领域知识。

5 讨论

上述的各种方法只是合成数据方法的一部分，从以上的方法描述中不难看出，合成数据中两个比较关键的问题是：数据质量和数据多样性。对于提高合成数据的质量的方法，一般有以下四种：（1）对照生成答案与标准答案，只采用两者一致的合成数据；（2）采用多次投票的方案，采用获取票数最多的答案；（3）采取更强大模型，如GPT-4；（4）采用外部工具。为了提高合成数据的多样性，可以采用的方案有（1）调整模型生成时的温度等系数，使得生成更多样；（2）通过从预训练语料中挖掘；（3）通过限制生成条件的方式。

在实际应用中，为了获得高质量的合成数据，在成本可控的情况下，应尽可能使用性能更强的模型，或者通过迭代使用自合成的数据对较弱的模型进行优化，使其在所需领域的性能逐渐接近强大模型；对于输入文本x，应当选用质量较高的数据，并在使用前进行筛选，以排除那些质量低下、无意义的文本；对于先验知识t，可以利用数据集的标注来监督合成数据的输出y；此外，还应尝试利用各类工具，例如代码解释器，以减少模型生成发生错误的可能性，或者在提示词（Prompt）中为模型提供额外的先验知识来增加模型输出好的回答的概率。为了使合成数据具有多样性，应尽可能尝试从广泛的预训练语料中获取相关的种子数据，根据这些种子数据进行数据扩增；同时调整大模型推理时的随机性参数，或使用不同的大模型进行数据合成。

但当前合成数据仍然面临着以下的限制，（1）大量依赖更强的语言模型，例如GPT-4，通过蒸馏更强的闭源语言模型可以快速提升模型的能力，但是对于我们进一步实现通用人工智能（Artificial General Intelligence，简称AGI）帮助有限，同时构造大量语料的花费较高，例如使用GPT-4o构造1B的训练语料需要花费1.5万美金左右 [5]；（2）目前的合成数据方案较多只是针对某个特定能力，合成的数据量级大多在1B词元以下，与动辄上T量级的预训练数据需求存在几个数据量级的差距。大部分的合成数据方法不具备规模化的前景，同时在单一的领域上进行训练也可能导致模型在其它领域的能力下降 (Gudibande et al., 2023)；（3）方法主要集中在数学、代码类数据上，这些技术方法能否泛化到真实场景中的数理推理和代码生成是个尚待研究的问题，同时对于那些答案不唯一或者比较复杂的场景，可能需要发掘更多方法来进行合成数据质量控制；（4）在长序列语境下的合成数据较少，近期大模型的长语境处理能力给大模型带来了很大的想象空间，但目前高质量长序列训练数据非常缺乏 (Lv et al., 2024)，合成高质量长语境数据将有助于提高模型的对应能力。

6 总结

在本文中，我们介绍了三种常见的大模型数据合成方法。基于强大模型蒸馏的方法是目前采用最多的数据合成方法，但这类方法依赖于强大模型，尽管取得了很好的实际效用，但从进一步突破模型能力上限的角度作用有限。基于模型自我进化的方法有较高的研究价值，但目前相关工作较少。基于工具的方法可以保证合成数据的质量，但是高质量、多样的工具数据依赖于领域知识的注入，未来大模型领域需要和各行各业的科研工作者一道合作补充这类数据。

致谢

本文受上海人工智能实验室资助。

[5] https://openai.com/api/pricing/

参考文献

Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, et al. 2023. Gemini: A family of highly capable multimodal models. *CoRR*, abs/2312.11805.

AI Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, et al. 2023. Qwen technical report. *CoRR*, abs/2309.16609.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, et al. 2020. Language models are few-shot learners. *CoRR*, abs/2005.14165.

Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, et al. 2023. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. *CoRR*, abs/2312.09390.

Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. 2024a. Large language models as tool makers.

Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, et al. 2024b. Internlm2 technical report. *CoRR*, abs/2403.17297.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. 2024. Alphamath almost zero: process supervision without process.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, et al. 2023. Palm: Scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24:240:1–240:113.

Common crawl web archive. http://commoncrawl.org. Accessed: 2024-05-30.

Leonardo de Moura and Sebastian Ullrich. 2021. The lean 4 theorem prover and programming language. Lecture Notes in Computer Science, pages 625–635. Springer.

Bosheng Ding, Chengwei Qin, Linlin Liu, Yew Ken Chia, Shafiq Joty, et al. 2023. Is gpt-3 a good data annotator?

Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, et al. 2021. A survey of data augmentation approaches for NLP. Findings of ACL, pages 968–988. Association for Computational Linguistics.

Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. Specializing smaller language models towards multi-step reasoning. In *International Conference on Machine Learning*, pages 10421–10430. PMLR.

Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *Int. J. Comput. Vis.*, 129(6):1789–1819.

Arnav Gudibande, Eric Wallace, Charlie Snell, Xinyang Geng, Hao Liu, et al. 2023. The false promise of imitating proprietary llms. *CoRR*, abs/2305.15717.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, , et al. 2023. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*.

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2022. Large language models are reasoning teachers. *arXiv preprint arXiv:2212.10071*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, et al. 2022. Training compute-optimal large language models. *CoRR*, abs/2203.15556.

Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, et al. 2022. Large language models can self-improve. *arXiv preprint arXiv:2210.11610*.

Kevin Maik Jablonka, Philippe Schwaller, Andres Ortega-Guerrero, and Berend Smit. 2024. Leveraging large language models for predictive chemistry. *Nature Machine Intelligence*, pages 1–9.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, et al. 2020. Tinybert: Distilling BERT for natural language understanding. Findings of ACL, pages 4163–4174. Association for Computational Linguistics.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, et al. 2016. Fasttext.zip: Compressing text classification models.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, et al. 2020. Scaling laws for neural language models. *CoRR*, abs/2001.08361.

Nayoung Lee, Kartik Sreenivasan, Jason D. Lee, Kangwook Lee, and Dimitris Papailiopoulos. 2023. Teaching arithmetic to small transformers. *CoRR*, abs/2307.03381.

Yujia Li, David H. Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, et al. 2022. Competition-level code generation with alphacode. *CoRR*, abs/2203.07814.

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Luke Zettlemoyer, et al. 2023a. Self-alignment with instruction backtranslation. *arXiv preprint arXiv:2308.06259*.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, et al. 2023b. Textbooks are all you need II: phi-1.5 technical report. *CoRR*, abs/2309.05463.

Chen Li, Weiqi Wang, Jingcheng Hu, Yixuan Wei, Nanning Zheng, et al. 2024. Common 7b language models already possess strong math capabilities. *arXiv preprint arXiv:2403.04706*.

Wing Lian, Bleys Goodson, Eugene Pentland, Austin Cook, Chanvichet Vong, et al. 2023. Openorca: An open dataset of gpt augmented flan reasoning traces. https://https://huggingface.co/Open-Orca/OpenOrca.

Yih-Kai Lin, Chu-Fu Wang, Ching-Yu Chang, and Hao-Lun Sun. 2021. An efficient framework for counting pedestrians crossing a line using low-cost devices: the benefits of distilling the knowledge in a neural network. *Multim. Tools Appl.*, 80(3):4037–4051.

Ruibo Liu, Jerry Wei, Fangyu Liu, Chenglei Si, Yanzhe Zhang, et al. 2024. Best practices and lessons learned on synthetic data for language models.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, et al. 2023. The flan collection: Designing data and methods for effective instruction tuning. Proceedings of Machine Learning Research, pages 22631–22648. PMLR.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, et al. 2023a. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, et al. 2023b. Wizardcoder: Empowering code large language models with evol-instruct. In *The Twelfth International Conference on Learning Representations*.

Kai Lv, Xiaoran Liu, Qipeng Guo, Hang Yan, Conghui He, et al. 2024. Longwanjuan: Towards systematic measurement for long text quality. *CoRR*, abs/2402.13583.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2022. Teaching small language models to reason. *arXiv preprint arXiv:2212.08410*.

Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. 2024. Orca-math: Unlocking the potential of slms in grade school math. *arXiv preprint arXiv:2402.14830*.

Niklas Muennighoff, Alexander M. Rush, Boaz Barak, Teven Le Scao, Nouamane Tazi, et al. 2023. Scaling data-constrained language models.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, et al. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*.

Ansong Ni, Miltiadis Allamanis, Arman Cohan, Yinlin Deng, Kensen Shi, et al. 2024. Next: Teaching large language models to reason about code execution.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. 2023. Are emergent abilities of large language models a mirage?

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, et al. 2023. Reflexion: Language agents with verbal reinforcement learning.

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. 2022. Distilling reasoning capabilities into smaller language models. *arXiv preprint arXiv:2212.00193*.

Alexander Shypula, Aman Madaan, Yimeng Zeng, Uri Alon, Jacob Gardner, et al. 2024. Learning performance-improving code edits.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, et al. 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR*, abs/1712.01815.

Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, et al. 2024. Beyond human data: Scaling self-training for problem-solving with language models.

Demin Song, Honglin Guo, Yunhua Zhou, Shuhao Xing, Yudong Wang, et al. 2024. Code needs comments: Enhancing code llms with comment augmentation.

Shubham Toshniwal, Ivan Moshkov, Sean Narenthiran, Daria Gitman, Fei Jia, et al. 2024. Openmathinstruct-1: A 1.8 million math instruction tuning dataset. *arXiv preprint arXiv:2402.10176*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, et al. 2023a. Llama: Open and efficient foundation language models.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Trieu H. Trinh, Yuhuai Wu, Quoc V. Le, He He, and Thang Luong. 2024a. Solving olympiad geometry without human demonstrations. *Nat.*, 625(7995):476–482.

Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. 2024b. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482.

Eric Wallace, Mitchell Stern, and Dawn Song. 2020. Imitation attacks and defenses for black-box machine translation systems. pages 5531–5546. Association for Computational Linguistics.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax, May.

Ke Wang, Houxing Ren, Aojun Zhou, Zimu Lu, Sichun Luo, Weikang Shi, Renrui Zhang, Linqi Song, Mingjie Zhan, and Hongsheng Li. 2023a. Mathcoder: Seamless code integration in llms for enhanced mathematical reasoning. *arXiv preprint arXiv:2310.03731*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, et al. 2023b. Self-instruct: Aligning language models with self-generated instructions. In *The 61st Annual Meeting Of The Association For Computational Linguistics*.

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, et al. 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models.

Huajian Xin, Daya Guo, Zhihong Shao, Zhizhou Ren, Qihao Zhu, Bo Liu, Chong Ruan, Wenda Li, and Xiaodan Liang. 2024. Deepseek-prover: Advancing theorem proving in llms through large-scale synthetic data. *arXiv preprint arXiv:2405.14333*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, et al. 2023. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, et al. 2024. A survey on knowledge distillation of large language models. *CoRR*, abs/2402.13116.

John Yang, Akshara Prabhakar, Karthik Narasimhan, and Shunyu Yao. 2023a. Intercode: Standardizing and benchmarking interactive coding with execution feedback.

Zhen Yang, Ming Ding, Qingsong Lv, Zhihuan Jiang, Zehai He, et al. 2023b. GPT can solve mathematical problems without a calculator. *CoRR*, abs/2309.03241.

Jiacheng Ye, Jiahui Gao, Qintong Li, Hang Xu, Jiangtao Feng, et al. 2022. Zerogen: Efficient zero-shot learning via dataset generation.

Huaiyuan Ying, Shuo Zhang, Linyang Li, Zhejian Zhou, Yunfan Shao, et al. 2024. Internlm-math: Open math large language models toward verifiable reasoning. *CoRR*, abs/2402.06332.

Longhui Yu, Weisen Jiang, Han Shi, YU Jincheng, Zhengying Liu, et al. 2023. Metamath: Bootstrap your own mathematical questions for large language models. In *The Twelfth International Conference on Learning Representations*.

Zhaojian Yu, Xin Zhang, Ning Shang, Yangyu Huang, Can Xu, et al. 2024. Wavecoder: Widespread and versatile enhanced instruction tuning with refined data generation.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, and Songfang Huang. 2023. How well do large language models perform in arithmetic tasks? *CoRR*, abs/2304.02015.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, et al. 2023. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*.

Xiang Yue, Tuney Zheng, Ge Zhang, and Wenhu Chen. 2024. Mammoth2: Scaling instructions from the web.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488.

Di Zhang, Wei Liu, Qian Tan, Jingdan Chen, Hang Yan, et al. 2024. Chemllm: A chemical large language model. *CoRR*, abs/2402.06852.