4. 语境学习与模型预训练

本章首先介绍了什么是情境学习（in-context Learning），接着讨论了如何选择和设计示例，涵盖了单次提示（one-shot prompting）和少次提示（few-shot prompting），特别是引入了链式思维（chain of thought）思想。然后从数学角度解释了为什么情境学习在大规模语言模型（LLM）中是有效的，最后阐述了理解模型涌现能力（Emergent Abilities）的重要性。

此外，还介绍了训练大模型所面临的挑战，以及如何训练一个大模型的方法。具体内容包括数据并行、张量并行和参数并行等优化技术。说明了GLM-130B是如何训练出来的，以及LLAMA的训练过程，并强调了预训练时所采用的海量数据。

2024版本：唐杰、杜晋华、隋元培、张开元、赵若雯、张远达

## 4.1 引言

请考虑这样一句话“该公司预计其营业利润会有所改善。”

可以发现，这句话的情感是积极向上的。我们期待如果把这句话输入给大语言模型，它能够返回“积极（Positive）”这样的词汇。

然而，如果我们直接把这句话输入给大模型，可以预见的是，模型很难理解我们给出这句话的意图。它也许会表示赞同，也许会进行续写，也许会做出其他意料之外的回答。

那么我们如何让模型理解我们的意图，或者说任务需求呢？

一个直观的思路或许是预训练或者微调。我们只要在训练数据中加入足够多的对话情感识别数据，模型就有希望在遇见“该公司预计其营业利润会有所改善。”这句话时返回“积极（Positive）”。

不过，训练的开销可能是无法承受的。训练需要获取足够多的数据，消耗一定的计算资源和时间。此外，如果换成另外一个任务——例如预测这句话所描述事实的类别——那么就需要重新训练了，这些开销都是难以接受的。

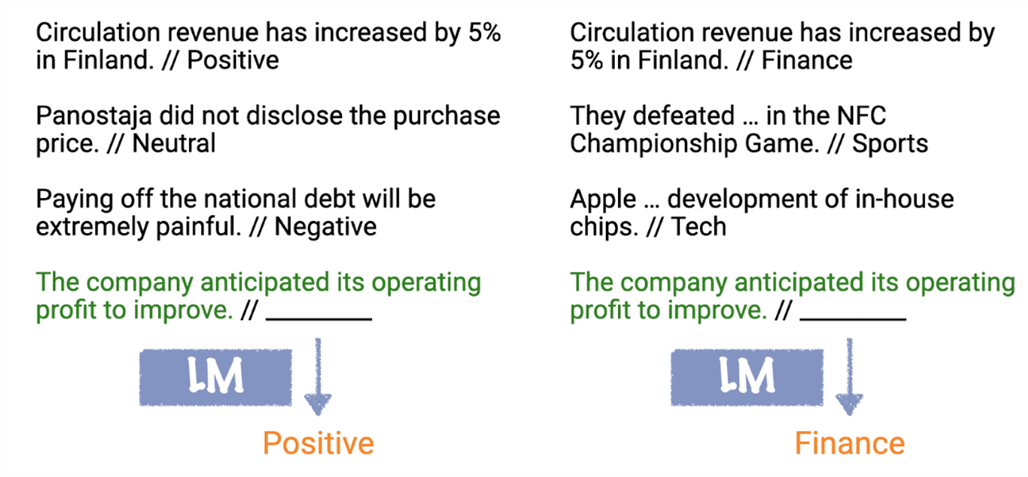
所以，单纯依赖预先的训练或者微调，无法涵盖所有的情形。对于这样需求频繁变化的、和语境（context）紧密相关的任务，模型是很难出色完成的。

既然某些问题需要依赖特定的语境，那么是否可以有这样一种方式，让模型通过上下文或语境判断此时应当做出的回复？这便是本章要介绍的**语境学习（in-context Learning)**

语境学习（in-context Learning)能**让模型在推理过程中学习**。通过简单的任务说明或少量的标签数据即可以灵活地处理不同的任务。

比如，当我们期待模型完成情感识别任务时，可以额外增添几个例子。能够让模型通过类比的方式把握任务内容。当我们期待模型完成语句分类任务时，同样可以使用其他的例子，让模型意识到需要输出语句对应的类别，而不是做出其他回复。

整个流程可以用下图表示



可以发现，对于同样的内容，通过不同的语境信息进行引导，模型便有可能通过语境内容进行学习，得出用户所期待的回复。

而这些例子只要和最后的“该公司预计其营业利润会有所改善”一并输入即可，无需额外的训练开销，对于频繁更改需求的场景，这显然是有巨大价值的。

除了语境学习之外，本章也会探讨大模型的预训练方法，并通过GLM-130B模型作为案例逐一解读大模型预训练的各个步骤。

## 4.2 语境学习的概念

实际上，引言中的例子里，我们已经把语境学习的相关过程已经涉及一二。

语境学习，实质上是一种通过类比进行学习的手段。使用时，只需要提供构造好的语境，模型便可以根据提供的语境学习得到任务目标以及任务意图。换言之，语境学习通过构造语境来辅助定义任务，以举例的方式让模型更好地完成特定的任务。

值得指出的是，语境学习的“学习”，与通常机器学习算法中的“学习”有些不同。通常意义下，机器学习中的“学习”需要经过数据训练，以反向传播等方式更新有关模型参数。而语境学习中的“学习”，并非要修改模型的参数，而是把“训练”要用的数据和待推理的问题合并作为输入，全部送入大模型中。

所以，语境学习**是在推理过程中的学习**，并不需要额外的训练阶段，从宏观模型使用上来看，与直接做推理并无二致。语境学习与传统意义上需要训练参数的学习相同之处在于，均需要构建一定的数据。只是前者无需训练过程。

提及数据，还应当说明的是，在传统机器学习中，往往需要构建足够多的数据集（这甚至可能需要百万条），才能获得优秀的效果；而语境学习中，仅需要很少量数据（往往个位数），甚至无需数据，便可以较好的达到所期待的目标。

使用少量数据进行语境学习的方法称为“few-shot”，而无需数据直接推理的方法称为“zero-shot”. 关于语境学习中，few-shot与zero-shot为什么有效的话题，我们放在后面的章节进行讨论。

## 4.3 语境构建

### 4.3.1 输入模板

构建语境时，可以包含如下内容：

①任务描述（可选）

②举例（作为样例）

③提示词（问题描述）

一个可行的例子如下图：



### 4.3.2 样例选取

样例选取应当兼顾正确性，多样性，同时应当考虑输入-输出映射的影响，以及语境示例的偏置。

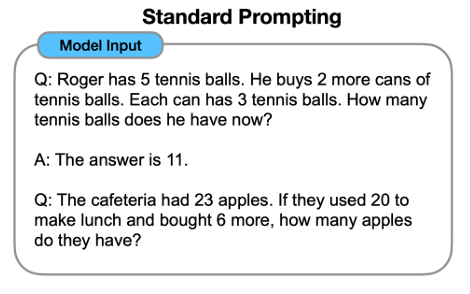
事实上，在这些因素中，保证样例-提示词数据分布的一致性是非常重要的。因为这能够把模型指引到正确的标签空间中。而更多的样例数据总体上能够带来收益，但这也并不绝对，更多的样例数据也可能带来负面作用（参考4.3.1中的实验分析），而且更多的样例也对模型的上下文提取能力做出了更高的要求。在这两点的基础上，如果样例能够提供正确的标签，那么这将给模型能力带来进一步提升。

除了上述技巧外，如何通过提示词的设计，让模型的语境学习推理能力获得更好的提升呢？思维链（Chain of thought）或许是一种可行的方法。

### 4.3.3 样例设计与思维链

#### 思维链

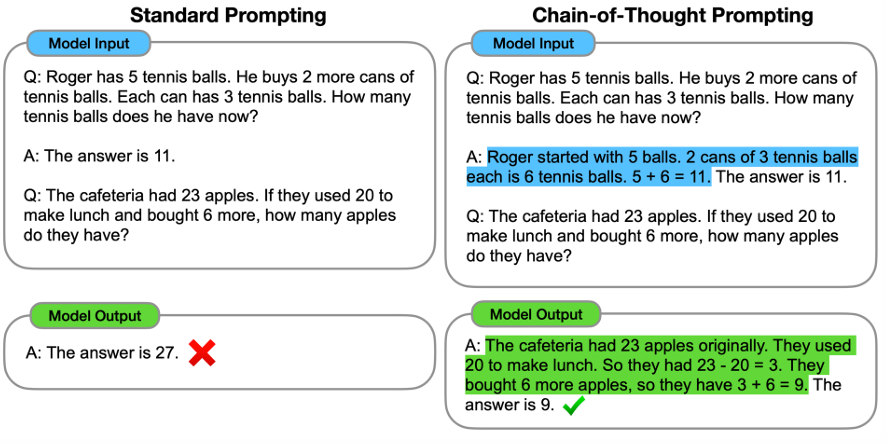
朴素的few-shot方法在需要强大推理能力的任务上表现不尽如人意。比如对于如下的场景：



提示词中是一道需要计算的数学题目，而在样例中，仅简单地提供了另外一道数学题的答案。可以想到的是，这种样例对模型语境学习的能力提升是十分有限的，模型难以从样例中获取较多的灵感来对提示词中的问题进行准确的回复。

而思维链的语境设计方式，则是通过自然语言的形式，对样例中的答案获得过程做出解释，或者说阐述了为什么该问题应当对应该答案。

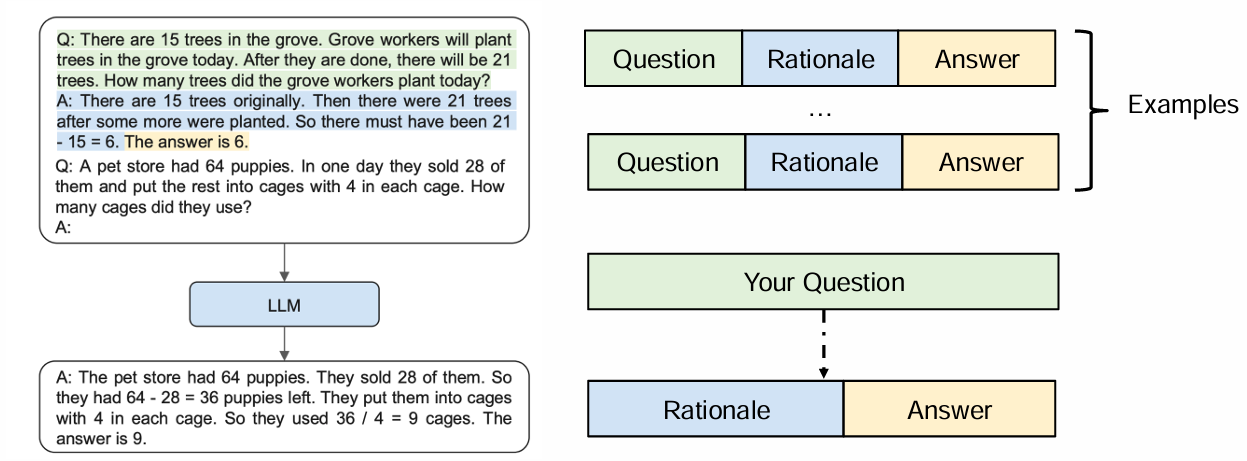
以上述数学题为例，在样例的答案环节增加计算过程，向模型解释为什么能够计算得到11这个答案。从实验结果来看，在语境中增加了思维链后的模型给出了正确的回复以及理由。



总而言之，思维链通过用“元数据”（即推理过程）做引导的方式，提升了语言模型处理更复杂问题的能力。

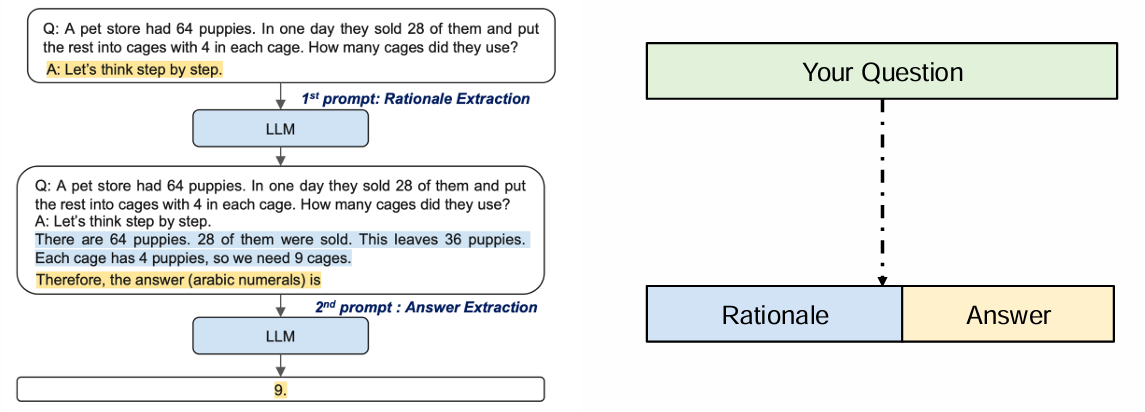
#### 思维链下的语境学习

在Few-shot方式下，使用思维链的语境构建示例和输入输出流程由下图所示：

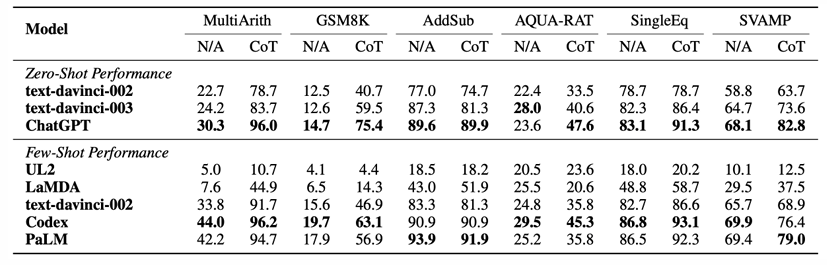


即先输入若干条样例，其中的每一条包含问题、理由（推理过程）、答案三个部分。最后输入向咨询的问题。期待情况下，模型会给出推理过程，以及相应的答案。

在Zero-shot方式下，可以通过在咨询的问题后添加“请一步一步地想”作为引导，以激活模型的推理能力。使用思维链的语境构建示例和输入输出流程可由下图所示：



实验验证表明了思维链方法的确可以激活模型地推理能力，并很大程度上提升了模型在多步骤推理任务上的表现。



## 4.4 语境学习的理解

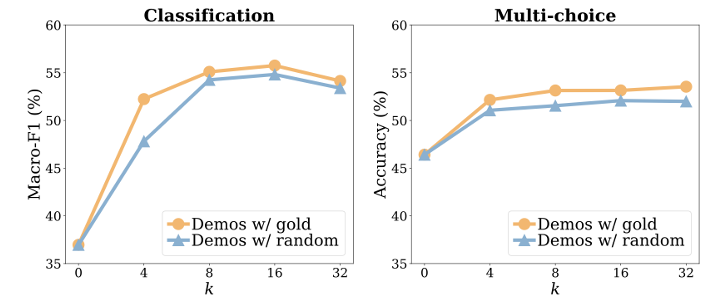
探究语境学习的效果受哪些因素影响。可能的因素包括：样例标签准确性，样例数据和训练数据的分布是否一致，样例数据的数量，输入语境格式等。

### 4.4.1.实验测试

Meta发表的文章Rethinking the Role of Demonstrations: What Makes In-Context Learning Work? 对语境学习有效的原因做出了探讨，并通过大量的实验进行了探究。



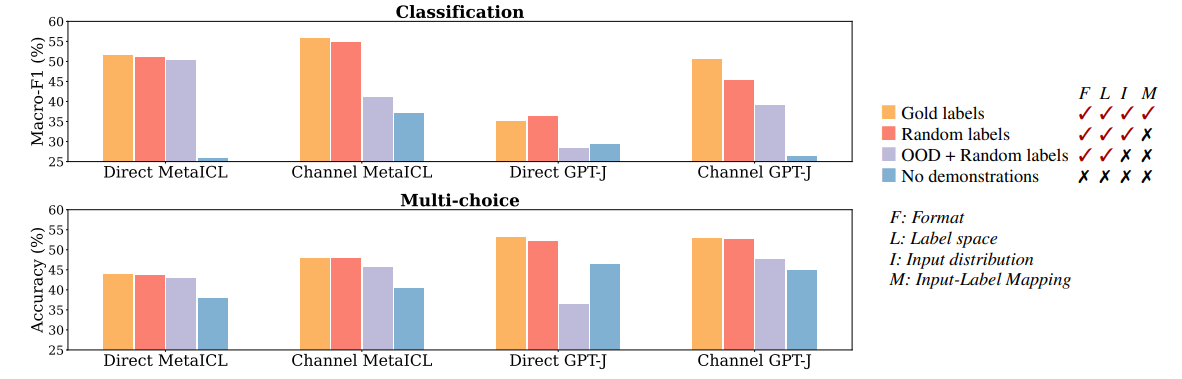
实验发现，即使样例的标签是随机生成的，大模型仍然能有一定概率分类正确。相比于提供正确的标签，精度上仅有细微的下降。而不提供标签的情况下，模型回答精度则下降较为明显。这种现象在不同大小的模型上均有所体现。



此外，样例的数目也会影响模型回答的准确程度。可以看到，无论是分类任务还是多选任务，即使仅仅使用4个样例，仍然能够大幅度提升模型的回答准确度。此外，当k大于等于8时，样例数目增长带来的提升就不那么明显了。同时，通过比较标签是随机的还是完全正确的这两条折线，依然可以发现样例标签的准确性似乎对模型能力的影响不是很大。

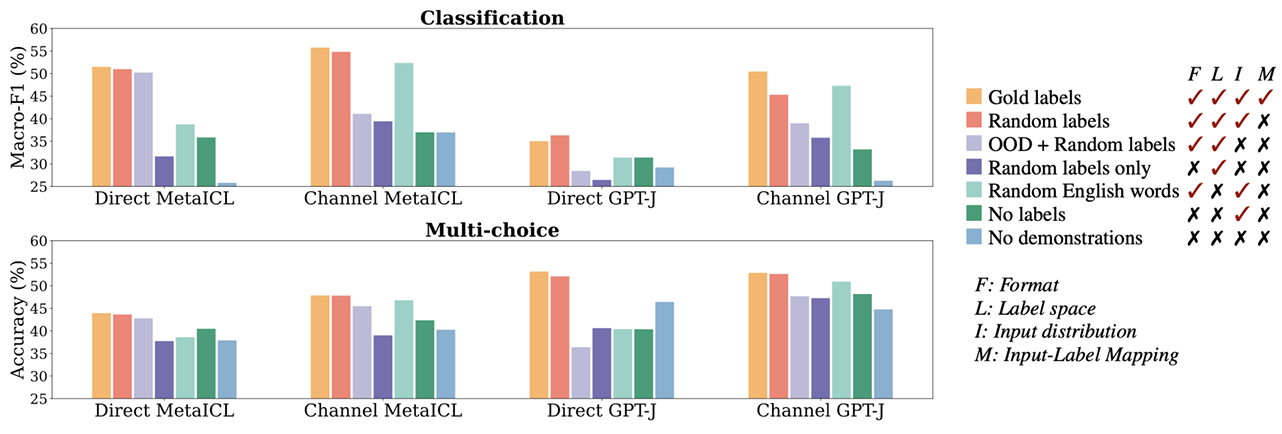
所以可以初步得出如下的结论：语境学习主要获益于对标签空间的监控，或者说激活了预训练/微调时标签空间相关的参数。而至于其他方面，比如标签的正确性等，则较容易从训练时习得的数据中还原出来。

进一步地，为了探究样例数据分布和提示词中的问题分布是否一致对模型预测准确度的影响，该论文使用分布上不同的数据作为样例进行了测试。实验结果如下（这里ppt上的图错了，我从原文找了图）



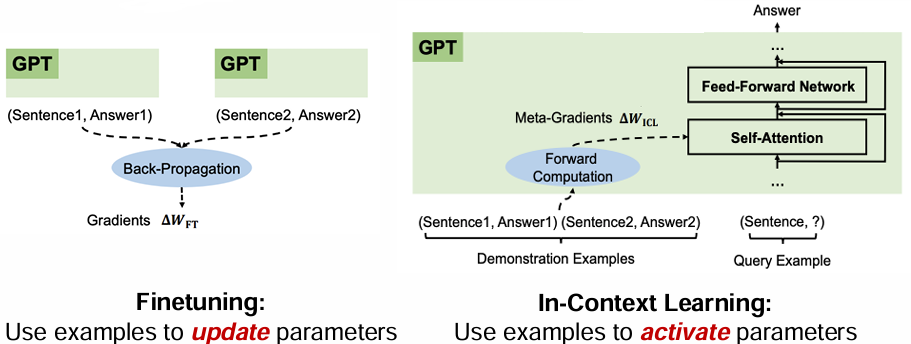
比较图中红色与浅紫色的柱形可以发现，在同样使用随机标签的情况下，使用和提示词数据分布不同（Out-of-Distribution,OOD)的数据作为样例，会导致精度的显著下降（除了MetaICL）。

随后，该文章探究了语境输入格式对回复精度的影响，从下图中可以看到，如果不保持规范的输入格式，整体上模型的表现会更差。



### 4.4.2 理论推导

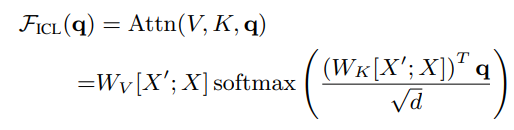
在进行理论推导前，首先让我们回顾语境学习和微调方式的关联性。



通过对比可以发现，微调实际上是用样例更新模型参数，而语境学习是用样例去激活模型参数。而微调之所以能成功，是因为反向传播时的梯度下降方法恰当地更新了模型的参数。而实际上，语境学习过程也可以类比梯度下降。

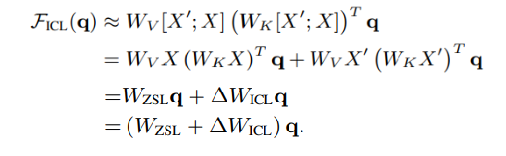
文章《Why can gpt learn in-context? language models implicitly perform gradient descent as meta-optimizers》对此进行了阐述。

语境学习中的注意力模块计算公式如下：



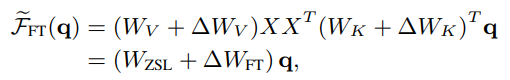
其中，X'为语境学习中输入的样例，X则为提示词中出现的问题。而q可以认为是当前阶段查询的token.

当忽略softmax函数以及标准化处理后，可以规约为下面的式子：



Wzsl表示Zero-shot涉及的权重，ΔWICL则表示通过增添样例带来的权重变化。

而微调后使用Zero-shot的计算方式可以写作下式：

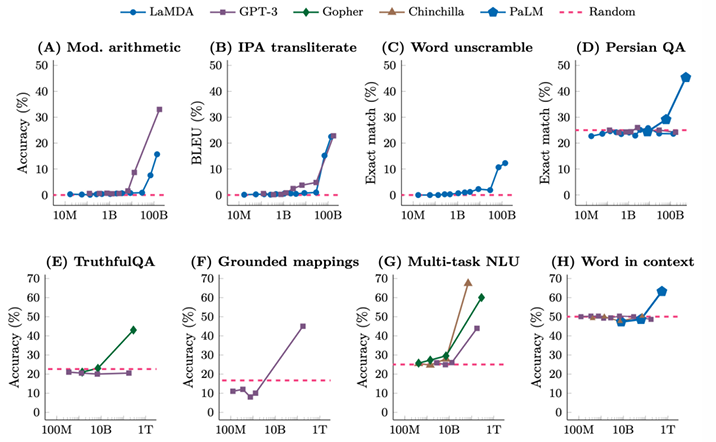


其中Wzsl仍然表示Zero-shot涉及的权重，ΔWFT则表示在初始模型基础上经过微调带来的权重变化。

通过类比观察可知，微调与语境学习在参数激活角度上是殊途同归的。所以，已知使用梯度下降的微调更新参数的方式能够带来性能上的提升，那么通过语境学习中的样例激活相关的参数便也同样能获得性能上的改善。

## 4.5 模型涌现能力

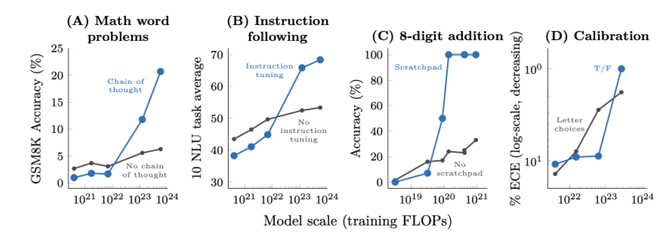
### 4.5.1 规模增长带来的收益



可以看到GPT-3和LaMDA等模型，随着模型和计算规模扩大，性能持续提高，在多个任务上展现出更强能力。此外，仍然可以发现，在某些任务上，模型在达到一定规模前性能随机，之后显著提升，且一个基础模型有潜力处理多种任务。

大模型（约100B参数或1023FLOPs）的这种能力被称作**涌现能力**，即在较小模型中不存在但在大模型中出现的能力。

对于数学能力，指令遵循，多步推理，程序执行等更困难的任务，模型数据量的增长依然可以带来性能上的提升，而且一定范围内，能力可以呈指数级上涨。这仍然是涌现能力的体现。

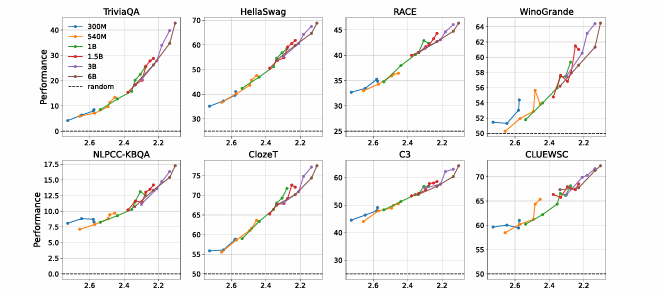


### 4.5.2 涌现能力的来源

可以看到，当模型参数规模上涨到一定程度时，在某些任务上的预测精度陡然提升，模型瞬间获得了解决这种问题的能力。此外，这种能力的获得也是不可预测的，发生精度陡然提升时所需要的模型参数随着任务的不同而不同。

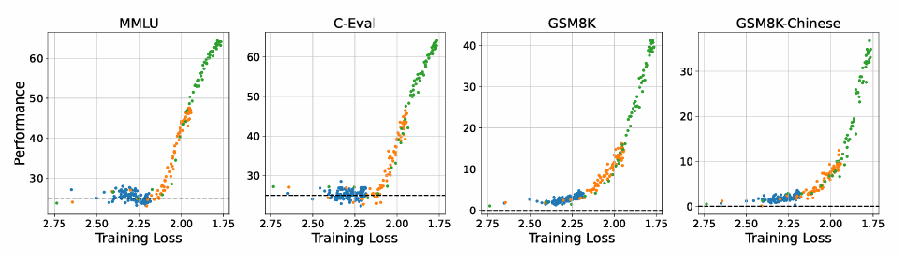
这种涌现能力是从何而来呢？

文章《Understanding Emergent Abilities of Language Models from the Loss Perspective》对此进行了讨论。



文章中，作者训练一系列具有不同大小和训练标记的语言模型，发现不同大小的数据点和训练token在很大程度上落在相同的趋势曲线上。换句话说，无论token计数和模型大小如何，具有相同预训练损失的语言模型在不同的语言、任务和提示格式上表现出相同的性能。

而参数量更大的模型能够将训练损失进一步降低，从而获得性能上的提升。



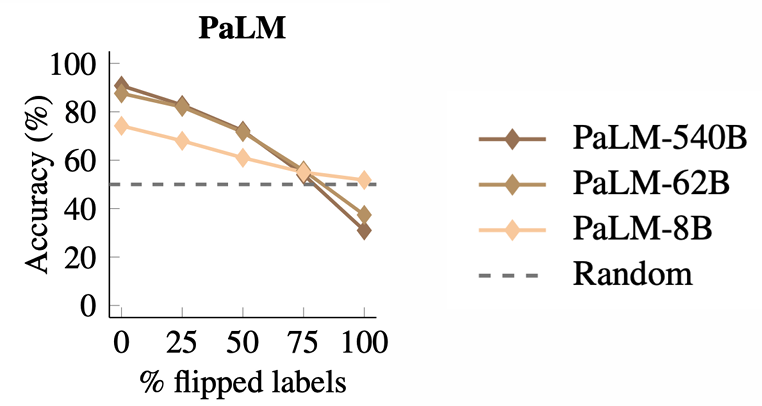
如上图所示，再进一步分析模型效果与训练损失，可以发现，训练前损失减小到2.2左右之前，三种不同规模的模型性能都在随机水平上。而当损失降到2.2及以下时，模型能力陡然攀升。受参数规模影响，蓝色点所代表的小模型无法进一步降低损失，停留在了随机预测的水平；而绿色、橙色代表的更大的模型则通过不断训练降低了损失，从而获得了涌现能力。

这同样说明在损失函数相同的情形下，不同规模的模型预测能力相近。而更大的模型能更好地学习训练数据，降低了损失，从而提升了性能。对涌现能力的认识，大大指导了模型训练的过程。

### 4.5.3 语境学习能力与规模

本节介绍了不同数据规模/参数规模情况下语境学习的能力，以及相关因素给预测精度带来的影响。

#### 1 标签翻转与参数量规模

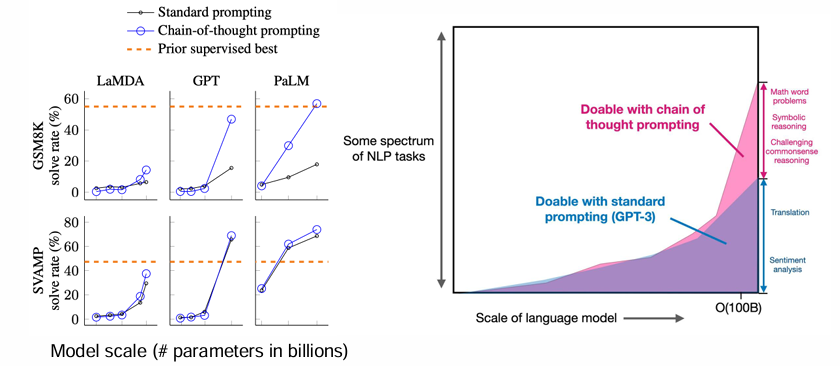


当语境中的样例标签发生翻转时（即二分类任务，标签取反），参数规模越大的模型的预测精度受影响越大。且当所有的标签全部翻转后，62B和540B的PaLM模型甚至预测精度低于50%的随机猜测准确度，而8B的小模型则仍然能够保持大于50%预测精度的水平。

这说明虽然模型具备一定的纠错能力，但参数量越大的模型越会优先考虑语境中提供的内容，故容易受到语境样例的错误标签影响。

#### 2 思维链与参数量规模

下图中的实验展示了随参数量规模增长，思维链方式带来的提升情况。可以看到，当参数量规模增长时，相比于普通的提示词设计方式（Standard prompt），思维链方式获得的提升更为显著。

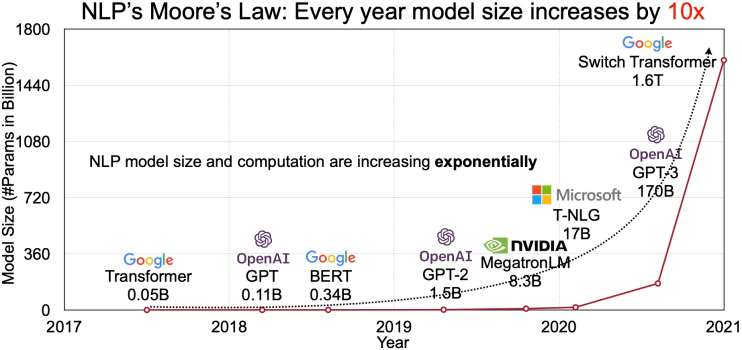


## 4.6 训练大模型的挑战

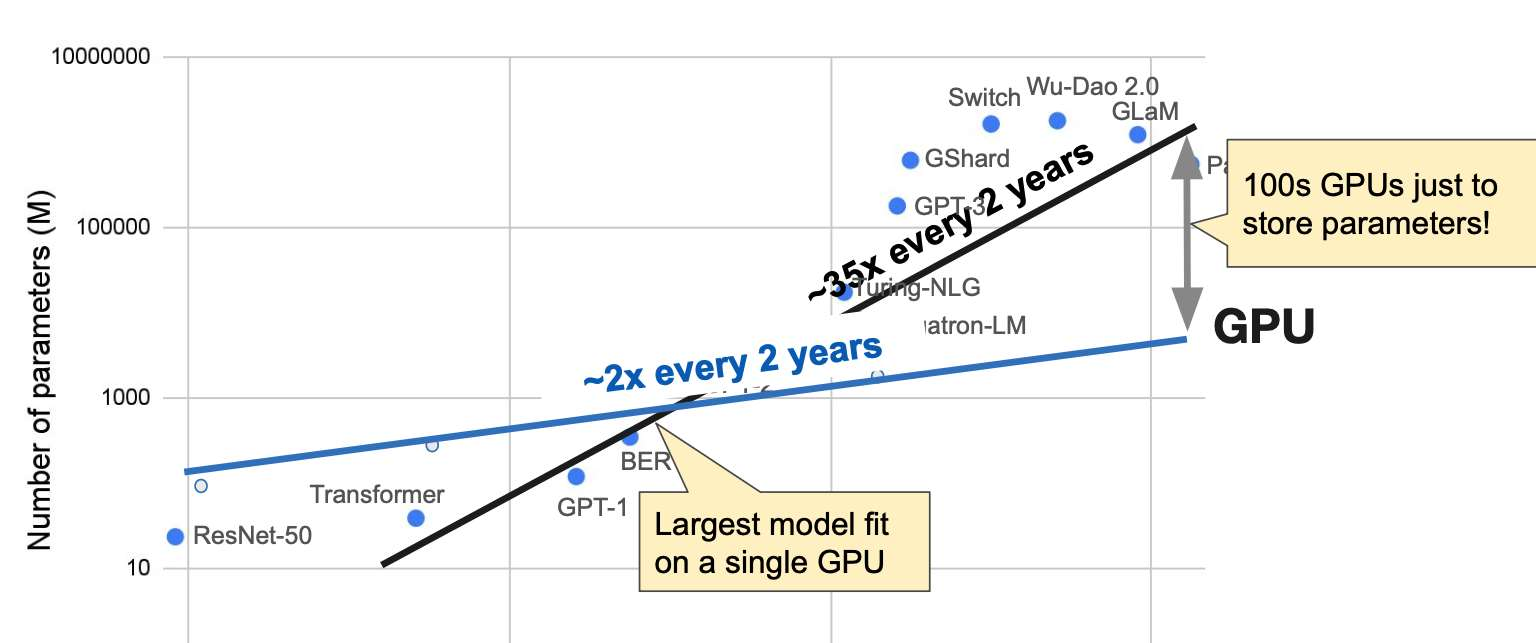
了解语境学习与大模型涌现能力后，接下来的章节内容将围绕大模型的训练展开。首先让我们先来了解训练大模型时面临的挑战。

1. **硬件资源限制**：

大语言模型呈现出快速增长态势，遵循类似摩尔定律，每年模型规模增长10倍，如GPT-3等模型规模不断扩大，促使人们追求更大模型训练。众多机构积极投入大模型研发，模型参数数量从2017年的0.05B迅速增长至2021年的175B及更高，计算量也呈指数级增加12。

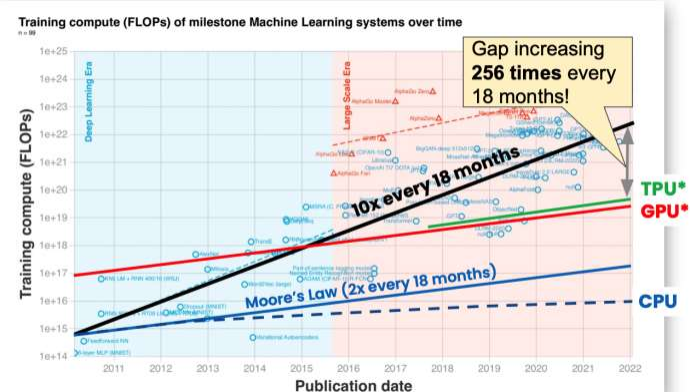


单个GPU内存与模型规模差距不断增大，如GPT-3的175B参数训练初始化需1.8TBGPU内存，远超单卡内存，导致需要大量GPU并行计算。



1. **训练成本高昂**：

训练大模型所需计算资源（FLOPS）不断增加，成本高昂。例如GPT-3使用10,000个V100 GPU，机器成本达460万美元，总成本估计460万美元；PaLM540B使用6144个TPUv4，计算量巨大，其计算量与传统硬件相比差距悬殊且随时间不断增大。



1. **多方面训练挑战**

* **内存挑战**：模型参数、梯度、优化器状态和激活值等占用大量内存，如BERT-Base模型中激活值在内存占比达87.6%，且在并行训练时模型权重和梯度会复制多次，优化器状态虽只保存一次，但整体内存需求仍很高。
* **性能挑战**：大模型训练计算密集，需要软硬件协同调优以提高集群性能，从而节省计算成本和时间。
* **通信挑战**：并行训练涉及多机通信，需根据模型大小、数据样本大小和集群拓扑特征合理设计并行策略，以提高通信效率，如张量并行在跨节点通信时受带宽限制。
* **维护挑战**：数千节点集群的软硬件运维难度大，需确保计算正确性、模型训练高可用性，以及在训练失败时及时恢复。

## **4.7 训练大模型的方法**

### **4.7.1模型内优化**

#### **混合精度训练**

* **FP 格式与优势：**采用16位格式（FP16/BF16）替代传统FP32格式，在 NVIDIA A100 TENSOR CORE GPU 上，与 FP32 相比速度可提升2-8倍。以A100为例，不同格式在计算性能（如TFLOPS）和内存带宽等方面存在差异，BF16和FP16在特定计算中表现出更高性能。
* **训练过程与稳定性：**在训练过程中，参数和数据使用FP16格式，在前向传播、反向传播和更新过程中进行精度转换和操作。前向传播中数据和激活值使用FP16，反向传播计算梯度后转换为FP32进行参数更新，同时利用动态损失缩放技术。然而，由于FP16数值范围有限，训练稳定性面临挑战，如优化器状态使用 FP16 可能导致不稳定并影响性能。随着技术发展，H100/H800支持Float-8，但如何在更有限数值范围下稳定训练仍需探索。

#### **2. 重计算**

* **原理与操作方式**：动态内存用于反向传播计算图且占比较大，通过重计算技术，在前向传播中丢弃激活值，在反向传播需要时重新计算，可有效节省内存。例如，普通前向-反向传播需保存所有激活状态，而梯度检查点技术几乎不保存激活状态，通过重新计算减少内存使用，但会增加计算量，不过其内存对层数不太敏感。具体操作中，如在某些模型训练中，可通过合理设计计算图，实现激活值的按需重计算，从而在一定程度上平衡内存和计算资源。

### **4.7.2 并行策略**

#### **1. 数据并行**

* **基本概念与实现方式**：数据并行通过复制模型到多个GPU并分片数据来增加批量大小，以利用更多数据加速训练。其包含参数服务器和All-Reduce两种方法。参数服务器从分布式数据加载器加载数据，执行前向和反向传播计算梯度后，将梯度推送到服务器，再从服务器拉取权重，通常每N次迭代异步更新，但存在负载均衡困难问题。All-Reduce方法同样从分布式数据加载器获取数据，计算梯度后通过 All-Reduce操作同步梯度，再从参数服务器拉取权重并本地更新，该方法通信均衡且易于实现（如 PyTorch DDP），但存在所有工作者冗余参数更新的缺点。
* **ZeRO 优化器**：针对纯数据并行中每个GPU维护冗余副本（梯度、优化器状态和模型权重）的问题，ZeRO优化器将所有参数分区到不同 GPU。具体分为三个阶段，优化器状态（Zero-1）、梯度（Zero-2）和模型权重（Zero-3）依次分区，根据不同阶段减少内存和通信成本。然而，随着模型增大，Zero-3虽可用于训练超大型模型（如100B参数模型权重约需200G内存），但会带来额外通信成本，且该成本与模型大小线性相关，当计算资源远小于通信开销时，这将成为沉重负担。

#### **2. 张量并行**

* **分割策略与计算过程**：当模型无法在单GPU运行时，张量并行将模型权重张量沿特定维度分割到多个GPU。例如，对于输入数据和模型权重，可将权重分割为（行并行与All-Reduce结合）或（列并行与All-Gather结合）的形式，在 Transformer 的多层感知机（MLP）的两个线性层和注意力机制（Attention）的四个线性层（query/key/value/output）中有具体应用方式。
* **局限性**：张量并行通常在节点内受限，如环All-Reduce受最小通信带宽限制，DGX-A100节点内带宽虽高，但跨节点做张量并行时带宽会降低（如降至 1/12）；同时，减小粒度会降低神经网络输入，降低硬件效率。

#### **3. 流水线并行**

* **工作原理与特点**：模型无法在单机运行时采用流水线并行，将模型分割成多阶段，阶段间传输输出。其优点是数据传输成本小，只在阶段边界通信阶段输出，但存在流水线气泡问题。例如，在普通流水线并行中，一次只有一个设备处于激活状态，流水线气泡百分比计算公式为（P-1）/P（假设P个阶段）。
* **改进方法GPipe**：通过流水线输入减少气泡，气泡百分比计算公式为（P-1）/（P-1+M）（P为阶段数，M为输入数），但需要存储所有M个微批次的中间激活值，更多微批次意味着更少气泡百分比但更多内存使用。
* **1F1B 调度**：尽早执行反向传播，具有相同延迟，内存仅与阶段数相关，最大每设备内存使用量计算公式为参数+激活值×微批次数量×设备数量。
* **交错 1F1B**：将神经网络切成更细粒度的阶段并分配给多个设备，以减少流水线气泡，提高流水线效率，但阶段之间通信开销更大。

## **4.8 GLM-130B模型案例**

### **4.8.1 训练挑战与应对**

* **工程挑战**：在训练GLM-130B过程中，面临多种硬件相关问题，如频繁和随机硬件故障、Megatron-DeepSpeed 3D流水线问题、CUDA内核效率问题、GPU内存溢出问题以及10K+线程TCP初始化和通信问题等，涉及硬件平台包括Hygon DCU、NVIDIA A100、Ascend 910、Sunway等。
* **算法挑战**：需要解决一系列算法相关问题以稳定训练，包括嵌入的梯度范数、Native Post-LN、Pre-LN3、Sandwich-LN4、数据加载器状态种子、Softmax/Attention中的计算精度选择等。

### **4.8.2 模型优势**

* **语言支持与性能表现**：具备双语支持（英语和中文），在性能上表现出色。英语方面，在LAMBADA数据集上优于GPT-3 175B（+4.0%）、OPT-175B（+5.5%）和 BLOOM-176B（+13.0%），在MMLU数据集上略优于GPT-3 175B（+0.9%）；中文方面，在7个零样本CLUE数据集上显著优于ERNIE TITAN 3.0 260B（+24.26%），在5个零样本FewCLUE数据集上（+12.75%）。
* **推理与可用性优势**：支持快速推理，在单个A100服务器上可通过SAT和FasterTransformer实现快速推理（最高快 2.5 倍），并且支持INT4权重量化，使其能够在4\*3090或8\*1080 Ti上进行推理，大大提高了模型的可用性。
* **可复现性与跨平台性**：具有可复现性，所有结果（30+任务）可通过开源代码和模型检查点轻松复现，同时具备跨平台特性，可在NVIDIA、Hygon DCU、Ascend 910和Sunway等多种硬件平台上进行训练和推理。

### **4.8.3 架构实验**

* **实验方法与技术**：实验涉及多种方法和技术，如DeepNorm（一种稳定训练 1000层Post-LN的方法）、旋转位置编码（RoPE，相对位置编码）和门控线性单元（GLU，替代前馈网络（FFN）层以稳定提高模型性能）。
* **实验结果与分析**：通过小规模实验对比不同架构在多个任务（如 RTE、COPA、BoolQ、WSC 等）上的性能表现，如不同架构下各任务的准确率等指标，为模型架构优化提供依据，确定更优的模型架构组合。

### **4.8.4 模型结构：前层归一化（Pre-LN）/后层归一化（Post-LN）**

* 原始的BERT模型[1]采用了后层归一化（Post-LN）。
* 然而，后层归一化容易导致训练不稳定，近年来模型通常采用前层归一化（Pre-LN）。
* 然而，前层归一化在处理超大规模（10B+参数）或多模态训练时也存在不稳定性。
* 三明治归一化（Sandwich-LN）[3]被提出用来缓解这种不稳定现象。

**前层归一化（Pre-LN）：**

* 容易训练，但在稳定训练后其性能不如后层归一化（Post-LN）[1]。
* 前层归一化路径恒定，有利于训练，但前几层的梯度较大，不利于微调。

**解决方案：**

* **DeepNet**[2]：调整残差误差，改变初始化方法 → 稳定1000层的后层归一化训练。
* **100B参数规模模型**：DeepNorm比三明治归一化（Sandwich LN）更稳定，因此选择DeepNorm。

### 4.8.5 模型架构：绝对位置编码（Absolute PE）/相对位置编码（Relative PE）

* **绝对位置编码：**
  + **正弦位置编码**[1]：基于固定公式生成的位置编码。
  + **可学习位置编码**[2]：通过训练优化生成的位置编码。
* **相对位置编码：**
  + **相对位置表示**[3]：在注意力机制中表示词对之间的相对距离。
  + **Transformer-XL**[4]：通过扩展上下文长度实现相对位置编码。

### 4.8.6 模型架构：ALiBi/RoPE

* **ALiBi（Attention with Linear Biases）：**
  + 向注意力分数矩阵中添加一个偏置矩阵。
  + BigScience项目发现，ALiBi显著提升了零样本学习性能，并被用于176B参数的模型。
* **RoPE（Rotary Position Embedding）：**
  + 通过绝对位置编码实现相对位置编码。
  + 已在EleutherAI和Google PaLM 530B中使用，以提高训练的稳定性。
  + 实验表明，RoPE对GLM模型更有效，并且更易于实现双向相对注意力。

### 4.8.7 **GLM-130B：核心技术**

* **架构：** 通用语言模型（General Language Model, GLM）。
* **组件改进：** 包括RoPE（旋转位置编码）、DeepNorm、GeGLU。
* **工程实现：**并行策略：数据并行、张量并行和流水线并行的三维并行策略。
  + 高效的多平台适配。
* **训练策略：**训练稳定性：大模型面临的最大挑战。
  + 梯度爆炸：嵌入层梯度缩减策略。
  + 注意力溢出：采用FP32进行Softmax计算以避免溢出。

## 4.9 并行策略：高效训练

* GPT-3模型需要2.8TB的GPU内存来存储训练状态和中间激活值。
* **挑战：** 如何高效训练超出单卡GPU内存（40GB）限制的模型？使用ZeRO优化器在数据并行组内共享优化器状态 → 节省约25%的内存。
* **模型并行：** 将模型参数分配到多个GPU上。**张量并行：** 将参数矩阵切分成多个部分，每个GPU计算其中一部分 → 需要额外通信来降低计算粒度。
* **流水线并行：** 将网络划分为多个并行阶段，每阶段分别计算 → 引入流水线气泡。
* **如何高效训练超大规模模型？ZeRO-3：** 在数据并行组中分配参数，并在计算前获取参数 → 增加额外通信时间。
* **策略总结：**张量并行随着模型规模增长线性扩展，但受限于单机大小（≤ 8）。其余部分通过调整微批次大小，使用流水线并行来减少气泡比例。

1. **其他优化：操作融合（Operator Fusion）：** 将多个元素级算法融合 → 提高计算速度约10%。
2. **流水线平衡：** 在流水线起始和结束阶段减少一层模型层，以平衡资源消耗 → 节省内存约10%。

* **测试集群配置：A100集群：** 96台DGX-A100机器，每台配备2个200GB IB网卡。
* **海光DCU：** 3000台机器，每台配备4张DCU卡和4个50GB IB接口。
* **神威处理器：** 8192个节点，每节点1个SW26010-PRO处理器。

结果： 估算GPT-3（175B参数、300B词汇量）规模模型的训练时间，硬件利用率提高了25%。

* **扩展性实验（在海光DCU上）：**不同模型规模下的训练性能和硬件利用率记录：较小模型（1.7B参数）：利用率为31.83%。
  + 超大模型（301.1B参数）：利用率逐渐提高至53.33%。
* **结论：**模型规模越大，矩阵乘法占比越高，硬件利用率越高。
  + 通过合理的并行策略（例如模型分割和批量大小控制），有效降低流水线气泡比例。
* **总体结论：**超线性扩展性：模型越大，计算效率越高。
* **模型计算特性：**随着模型规模增加，矩阵乘法的比例也增加。
  + 合理的并行策略（模型并行划分和批量大小优化）可以进一步提升流水线并行的效率。

## 4.10 GLM-130B训练策略

* **架构：** GLM（通用语言模型）。**组件改进：** 包括RoPE、DeepNorm和GeGLU。**工程实现：**三维并行策略：数据并行、张量并行和流水线并行。
* 高效适配多种平台。
* **训练策略：**提高训练稳定性：是大模型面临的最大挑战。
* 梯度爆炸：通过嵌入层梯度缩减策略解决。
* 注意力溢出：采用FP32计算Softmax避免溢出。

**最致命的挑战：稳定性**

* **权衡：** 稳定性（慢）和效率（不稳定）之间的选择。
* **现有解决方案：OPT-175B：** 手动调整学习率，跳过损坏数据 → 但性能下降。
  + **BLOOM-176B：** 使用嵌入层归一化和BF16 → 性能下降且支持平台有限。

**混合精度训练**

* **前向与后向传播：** 参数、激活和梯度均使用FP16。
* **更新阶段：** 参数、动量和方差切换为FP32。
* **动态损失缩放：** 通过缩放损失避免数值问题。
* **挑战：**异常前向传播可能导致后向传播崩溃（例如损失剧增）。

**GLM-130B：稳定性策略**

* **注意力分数：** Softmax在FP32中计算以避免溢出。
* **嵌入层梯度缩减（EGS）：**使用公式对嵌入层梯度进行缩减,嵌入层的梯度可能比其他层大很多。

**百亿参数规模模型**

* **示例：** GLM-130B，公开的双语预训练模型。

**训练基础设施**

* **计算：**使用高达16,000张H100 GPU（80GB HBM3，700W TDP）。
  + 每台服务器配备8张GPU，通过NVLink连接。
* **存储：**Tectonic分布式文件系统，240PB容量。
  + 持续吞吐量：2TB/s；峰值吞吐量：7TB/s。
  + **主要挑战：** 检查点保存和存储写入负载高峰。
* **网络：**采用基于RoCE的三层Clos网络连接24,000张GPU。
  + 提高负载均衡和拥塞控制能力。

## 4.11 LLAMA 3.1：并行策略

* **GPU分组策略：**依次分配为：张量并行（TP）、上下文并行（CP）、流水线并行（PP）和数据并行（DP）。
* **优化：**减少批量大小限制带来的瓶颈。
  + 平衡每个阶段的计算和内存使用。

**LLAMA 3.1：并行策略**

* **流水线并行改进：**提供更灵活的批量大小设置。
  + 减少流水线气泡，通过削减首尾阶段的Transformer层实现平衡。
  + 采用异步点对点通信，提升并行效率。
* **上下文并行：**提高内存效率，支持超长序列（长达128K）。
  + 通过分割输入序列改善负载均衡，支持多种注意力掩码。
* **流水线并行改进的挑战：批量大小限制：** 无法灵活调整大规模训练的批量大小。
* **内存不平衡：** 第一阶段的嵌入层消耗较多内存。
* **计算不平衡：** 最后一阶段的输出和损失计算消耗较多计算资源。
* **解决方案：**调整每个阶段的Transformer层分布，减少不平衡现象。
* 在一个流水线排名中增加交错调度，提高效率。
* **上下文并行：**改善内存效率，支持长达128K的超长序列训练。
* 输入序列分割为多个块（2 × CP），每个上下文并行节点接收两个块以平衡负载。
* **改进：** 在本地计算注意力输出之前，先完成所有的全收集（all-gather）操作。
* **优势：**更容易支持各种注意力掩码。
  + 由于分组查询注意力机制（GQA），减少了全收集操作的延迟。
* **训练稳定性：**使用FP32梯度累积并减少梯度散射。
* 对多次前向模块（如视觉输入）采用FP32梯度。
* **集群稳定性：**自动干预机制处理训练中断问题（54天内466次中断，78%为硬件问题）。
* 减少启动时间、跟踪通信事件、自动超时和重启。
* **环境影响：**GPU动态电压和频率调整导致的昼夜间吞吐量波动（1-2%）。

## 4.12 大语言模型背后的数据

* **重要性：**训练大语言模型需要“原始文本”，涵盖多领域、多体裁、多语言的广泛内容。
* 主要来源包括：**网络：** 例如，Google搜索索引包含100PB的数据。
  + **私有数据集：** 比如沃尔玛每小时生成2.5PB的数据。

**数据源**

* **常见数据集：**Common Crawl。
  + WebText 和 OpenWebText。
  + Colossal Clean Crawled Corpus (C4)。
  + GPT-3 数据集。
  + The Pile。

**Common Crawl**

* **简介：** Common Crawl 是一个非盈利组织，提供免费的网页快照。
* **规模：** 2021年4月快照的数据量为320TB。
* **应用：** 被用于训练诸如T5、GPT-3和Gopher等模型。

**Common Crawl 的表示偏差**

* **观察：**大规模数据仍存在对某些群体的代表性不足。
  + 互联网数据更倾向于代表发达国家的年轻用户。
  + 例如：Reddit 用户中67%为男性，64%年龄在18-29岁。
    - Wikipedia 的编辑者中只有8.8%-15%为女性。
* **结论：** 需要充分理解并记录用于大语言模型训练的数据集组成。

**WebText**

* **简介：** 用于GPT-2训练的WebText数据集，目标是获取高质量且多样化的数据。
* **构建过程：**抓取Reddit中收到至少3个点赞的链接。
  + 去除Wikipedia相关内容以用于评估。
  + 最终得到40GB的文本数据。
* **注意：** OpenAI 未公开WebText数据集。

**WebText和OpenWebText**

* **OpenWebText：** WebText的开源复现版本。
* **构建过程：**从Reddit提交数据集中提取所有URL。
  + 使用Facebook的fastText工具过滤非英语内容。
  + 移除近似重复内容。
  + 最终得到38GB的文本。

**WebText 和 OpenWebText 的毒性分析**

* **数据：**OpenWebText中2.1%的内容毒性评分≥50%。
  + WebText中4.3%的内容毒性评分≥50%。
  + 与新闻可靠性呈负相关（Spearman ρ=−0.35）。
  + 3%的OpenWebText来自被禁或隔离的子Reddit。
* **结论：** 可构建数据集评估大模型的毒性分数。

**Colossal Clean Crawled Corpus (C4)**

* **简介：** C4 是一个更大的数据集，用于训练T5模型。
* **构建过程：**使用2019年4月的Common Crawl快照。
  + 移除“敏感词”、代码和非英语文本。
  + 最终得到806GB的文本（1560亿个标记）。

**C4 的分析**

* **发现：**来自patents.google.com的数据占据了很大比例。
  + 65%的页面来自互联网档案，其中92%为过去十年的内容。
  + 美国托管的页面占比51.3%，但印度的页面较少（尽管英语使用者很多）。
  + 存在系统性错误，如OCR生成的文本或自动翻译的外国语言。

**基准数据污染**

* **问题：**当评估语言模型能力时，基准数据是否出现在训练数据中会影响结果。
  + **污染类型：**输入和输出同时出现在训练数据中。
    - 仅输入出现在训练数据中。
* **注意：** 污染与数据存储格式（如JSON）无关。

**C4中的危害**

* **表示性危害：**不同群体（如犹太人、阿拉伯人）之间的积极情绪占比存在差异。
  + 不同媒体来源（如《纽约时报》和半岛电视台）中差异程度不同。
* **资源分配危害：**性取向相关术语（如女同性恋、男同性恋）更可能被过滤。
  + 某些方言（如非洲裔英语）更可能被过滤。

**GPT-3 数据集**

* **构建过程：**从Common Crawl中选择与WebText相似的子集。
  + 通过二元分类器预测文档与WebText的相似性。
  + 进行模糊去重，扩展数据来源，最终包含多个高质量文本。

**The Pile**

* **简介：** EleutherAI开发的高质量数据集，用于语言建模。
* **组成：** 包括825GB的英语文本和22个高质量数据集来源。

**主要收获**

* 现有网络和私有数据量庞大，但训练效果并不完全取决于数据量。
* 数据过滤和整理（如C4、OpenWebText、The Pile）至关重要，但可能引入偏差。
* 发展高质量非网络数据集是一个有前景的方向。
* 必须仔细记录和检查数据集内容以确保透明性和可用性。

**数据集文档化**

* **重要性：**数据集文档化可以提高透明性和可追踪性。
  + 可参考其他领域（如电子元件的数据表和食品营养标签）制定标准。
  + 数据集的创建者和用户可以从中受益。

**数据集文档化（持续）**

* **核心问题：**动机：为什么创建这个数据集？
  + 组成：数据实例代表了什么？
  + 收集过程：如何获取和清理数据？
  + 使用：数据是否已被用于其他任务？是否存在禁用的使用场景？
  + 分发与维护：如何分发并支持后续更新？
* **示例：** The Pile 数据集的文档提供了详细的透明性。

**数据集文档化**

* **文档化的重要性：**数据集文档化可确保透明性，为数据集的创建和使用提供规范。
* **影响力：Datasheets for Datasets**（Gebru et al., 2018）：提供关于文档化的社区规范。
  + **Data Statements**（Bender & Friedman, 2018）：为语言数据集设计的相关框架。
* **核心目的：创建者：** 在创建数据集时反思决策及可能的社会偏见和危害。
  + **使用者：** 了解数据集适合和不适合的使用场景。

**数据集文档示例问题**

* **动机：**创建数据集的目的是什么？
* **组成：**数据集的实例代表了什么？是否缺失某些信息？
* **收集过程：**数据是如何获取的？数据收集过程中的参与者有哪些？
* **预处理/清理/标注：**数据是否经过预处理、清理或标注？使用的工具是否公开？

**数据集文档示例问题（续）**

* **用途：**数据集是否已被用于某些任务？是否有不适合的任务？
* **分发：**数据集如何分发？是否有支持/托管机制？
* **维护：**数据集是否会更新（例如修正标注错误、添加新实例或删除实例）？
* **示例：The Pile：** 通过详细的文档提供数据透明性和使用建议。

**总结**

* **扩展模型规模的关键：**重计算技术和并行化策略。
  + 数据并行和ZeRO优化器。
  + 模型并行、张量并行和流水线并行。
* **超大模型的实际训练：**架构设计、训练语料和稳定性至关重要。
* **大语言模型背后的数据：**数据源（如Common Crawl、WebText等）。
  + 数据集文档化对于透明性和规范性的重要性。

**训练超大语言模型**

* 提供大规模语言模型训练的全貌，包括架构、数据和训练策略。
* 数据集透明化和训练基础设施的优化是大规模模型发展的关键。

## 附录：

1. **Chinchilla scaling law**: Chinchilla scaling law 是由 DeepMind 提出的，是一种在模型大小、数据量和计算资源之间进行权衡的新的方法论。它建议，对于大规模预训练语言模型，如果固定计算资源的预算，通过增加训练数据量而不进一步增大模型尺寸可能会获得更好的性能（即应该更注重总的 token 数而不是过度增大模型参数）。

2. **Fineweb 预训练用数据，目前最大 15T token**:Fineweb 是指精细化筛选和构建的互联网文本数据集合，用于训练大型语言模型。数据集的规模（15T tokens）指的是输入给模型以进行学习的文本单元（tokens）的总数，通过这样的规模可以训练出性能优越的语言模型。

3. **Common Crawl**:Common Crawl 是一个公开的网络抓取项目，提供了大量从世界各地的网页中提取的原始文本数据。该数据集常用于训练大规模的语言模型，因为它包含大量种类丰富的文本内容，覆盖了多种语言和不同领域。

4. **从HTML中提取文本数据: trafilatura\inscriptis**:Trafiltura 和 Inscriptis 是两种从HTML网页中提取纯文本的工具。Trafiltura 是一个高级的网页爬取和内容提取工具，能够有效过滤广告等无关信息。Inscriptis 专注于将 HTML 格式良好的文本提取为干净的纯文本，用于进一步处理。

5. **Qurating 比较两个网页哪个更好**:Quarating 通常涉及对两个网页进行质量评估和比较，选择其中信息更可靠、准确和相关的网页。这样的评估可能依赖于一系列预定义的标准，例如网页内容的全面性、准确性、来源的可信度等。

6. **Minerva 用于提取数学的**:Minerva 是专门用于处理和生成数学表达及内容的工具，可能包括符号识别、方程式识别和处理等。此外，Minerva 也可能涉及从文本中准确提取和解读数学相关内容的功能。

7. **Openwebmath 训练一个分类器判断是否和数学相关**:Openwebmath 是指使用来自开放网络数据集构建和训练的分类器，用于判断文本内容是否与数学相关。这个过程通常包括训练数据的收集、特征提取、模型选择和优化等步骤。

8. **Automathtext 用大模型来分类与数学相关**:Automathtext 是指使用大型预训练语言模型（如 GPT-3 等）来自动分类和识别文本内容是否与数学相关。这包括对输入文本进行特征提取，通过模型进行分类器训练，并进行预测，以确定文本的相关性。