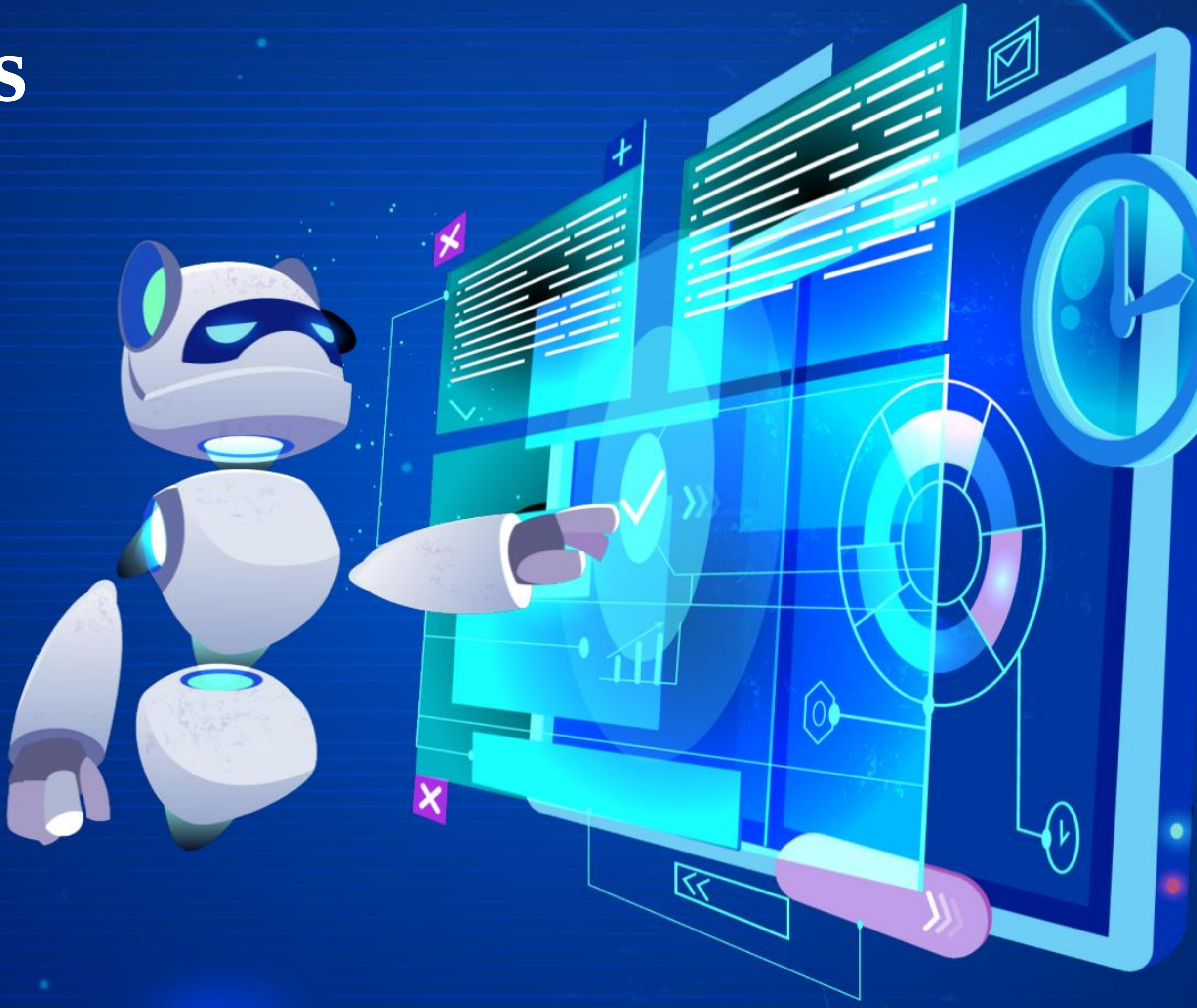
Current Best Practices for Training LLMs from Scratch

从零开始构建大语言模型的关键要点



UTC NETWORK PTE LTD

WWW.UTC.GROUP

Table of Contents 目录

Introduction

Although we're only a few years removed from the transformer breakthrough, LLMs have already grown massively in performance, cost, and promise. But many of the critical details and key decision points are often passed down by word of mouth.

The goal of this white paper is to distill the best practices for training your own LLM for scratch. We'll cover everything from scaling and hardware to dataset selection and model training, letting you know which tradeoffs to consider and flagging some potential pitfalls along the way. This is meant to be a fairly exhaustive look at the key steps and considerations you'll make when training an LLM from scratch.

The first question you should ask yourself is whether training one from scratch is right for your organization. As such, we'll start there:

引言

虽然我们距离 Transformer 的突破只有几年，但大型语言模型（LLMs）在性能、成本和潜力方面已经有了巨大的提升。但很多关键细节和关键决策点通常都是口口相传的。

本白皮书的目标是提炼出训练自己的大型语言模型（LLM）的最佳实践。我们将覆盖从缩放和硬件配置到数据集选择和模型训练的所有方面，以帮助您了解需要权衡的因素，并在这个过程中指出一些潜在的问题。这份白皮书旨在深入探讨从零开始训练 LLM 时需要考虑的关键步骤和注意事项。

您应该首先考虑的问题是，从零开始训练一个大型语言模型是否适合您的组织。因此，我们将从这里开始：

BUILD VS. BUY PRE-TRAINED LLM MODELS

Before starting LLM pre-training, the first question you need to ask is whether you should pre-train an LLM by yourself or use an existing one. There are three basic approaches:

Option 1: Use the API of a commercial LLM.

自建与采购预训练过的 LLM 模型

在开始 LLM 的预训练之前，你需要问的第一个问题是，你是应该自己预训练一个 LLM，还是使用一个现有的 LLM。有三种基本方法：

选项 1：使用商业 LLM 的 API。

Option 2: Use an existing open-sourced LLM.

选项 2：使用现有的开源 LLM。

Option 3: Pre-train an LLM by yourself or with consultants.

选项 3：通过自己或与商业伙伴一起对 LLM 进行训练。

That said, there are a lot of details to consider when making your choice. Here are the pros, cons, and applicable scenarios for each option:

也就是说，在你做出选择时，有很多细节需要考虑。以下是每种选项的优点、缺点和适用的场景：

	Option 1:Use the API of a commercial LLM	Option 2: Use an existing open-sourced LLM	Option 3: Pre-train an LLM by yourself or with consultants
	选项 1：使用商业 LLM	选项 2：使用开源 LLM	选项 3：通过自己或与商业伙伴一起对 LLM 进行训练
Pros	• Requires the least LLM training technical skills. • Minimum upfront training/exploration cost, given main cost incurs at inference time. • The least data-demanding option. Only a few examples (or no examples) are needed for models to perform inference. • Can leverage the best-performing LLMs in the market and build a superior experience. • Reduce time-to-market of your apps and de-risk your project with a working LLM model.	• A good way to leverage what LLMs have learned from a vast amount of internet data and build on top of it without paying for the IP at inference. • Compared to option one, you are less dependent on the future direction of LLM service providers and thus have more control regarding roadmap & backwards compatibility. • Compared to option three, you have a much faster time-to-value given you are not building LLMs from scratch, also leading to less data, training time, training budget needed.	• Compared to options one and two, you have the most control of your LLM`s performance and future direction, giving you lots of flexibility to innovate on techniques and/or customize to your downstream tasks. • Gain full control of training datasets used for the pre-training, which directly impacts model quality, bias, and toxicity issues. In comparison, those issues are less controllable in option one or two. • Training your own LLM also gives you a deep moat: Superior LLM performance either across horizontal use cases or tailored to your vertical, allowing you to build a sustaining advantage especially if you create a positive data/feedback loop with LLM deployments.

优点	• 需要最少的 LLM（大型语言模型）训练技术技能。 • 最小的前期训练/实验成本，因为主要成本发生在推理时期。 • 需求最少的数据选项。只需要一些示例（或没有示例）即可让模型执行推理。 • 可以利用市场上性能最佳的 LLM，并构建出卓越的体验。 • 通过成熟的 LLM 模型，缩短了您的应用发布时间，并降低了项目风险。	• 这是一种很好的方法，可以充分利用 LLM（大型语言模型）从大量互联网数据中学到的知识，并在推理时不支付知识产权费用。 • 与选项 1 相比，您对 LLM 服务提供商未来方向的依赖较小，因此在路线图和向后兼容性方面具有更多的控制权。 • 与选项 3 相比，由于不需要从头构建 LLM，因此可以更快地实现价值，同时也需要更少的数据、训练时间和训练预算。	• 与选项 1 和选项 2 相比，您对 LLM 的性能和未来方向具有最多的控制权，使您能够在技术方面进行创新或根据下游任务进行定制。 • 获得预训练数据集的完全控制权，这直接影响模型的质量、偏见和毒性问题。相比之下，在选项 1 或选项 2 中，这些问题较难控制。 • 自行训练 LLM 还使您拥有深厚的护城河：无论是横向用例还是专门针对您的行业，都可以获得卓越的 LLM 性能，尤其是如果您在 LLM 部署中创建了积极的数据/反馈循环，可以构建持续的竞争优势。
Cons	• Commercial LLM services can get expensive with a high volume of fine-tuning or inference tasks. It comes down to LLM total–cost–of- ownership (TCO) amortized to each inference. • Many industries/use cases forbid the use of commercial LLM services as sensitive data / PII data cannot be seen by the service for compliance (healthcare use cases, for example). • If building external apps, you`ll need to find other moats and de-risk your business if you`re highly reliant on external LLM service technology. • Less flexible downstream: Doesn’t support edge inference, limited ability to customize the model (fine-tuning	• Not as demanding as building your own, but still requires lots of domain expert skills to train, fine-tune, and host an open-sourced LLM. LLM reproducibility is still a significant issue so the amount of time and work needed cannot be underestimated. • Slower time-to-market and less agile if you are building downstream apps, due to a more vertical tech stack. • Open-sourced models typically lag performance compared to commercial models by months/years. If your competitor leverages commercial models, they have an advantage on LLM tech and you’ll need to find other competitive advantages.	• Very expensive endeavor with high risks. Need cross-domain knowledge spanning from NLP/ML, subject matter expertise, software and hardware expertise. If not done well, you could end up in a situation where you’ve spent thousands or even millions of dollars with a suboptimal model. Mistakes, especially late into training stages, are hard to fix / unwind. • Less efficient than option two. Option two leverages existing LLMs, learning from an entire internet’s worth of data and can provide a solid starting point. With option 3, you start from scratch and need lots of high-quality / diverse datasets for your models to gain generalized capabilities.

	gets expensive), limited ability for ongoing model improvements.		
缺点	• 大量的微调或推理任务可能会使商业 LLM 服务的成本变得昂贵，最终会分摊到每次推理的成本上。 • 许多行业/用例禁止使用商业 LLM 服务，因为敏感数据/个人身份信息数据有严格的合规要求（例如，医疗保健用例）。 • 如果构建外部应用程序，您需要寻找其他护城河，并降低对外部 LLM 服务技术的高度依赖，以减少业务风险。 • 下游灵活性较差：不支持边缘推理，定制模型的能力有限（微调成本高昂），持续改进模型的能力有限。	• 不像自行构建那么苛刻，但仍然需要大量专家技能来训练、微调和托管开源 LLM。LLM 的可重现性仍然是一个重要问题，因此不能低估所需的时间和工作量。 • 如果您正在构建下游应用程序，由于技术堆栈更为垂直，所以上市时间较长，灵活性较差。 • 与商业模型相比，开源模型的性能通常会滞后数月/数年。如果您的竞争对手利用商业模型，他们在 LLM 技术上具有优势，那您需要找到其他竞争优势。	• 这是一个成本高昂且风险很大的努力。需要跨足 NLP/ML、专业知识、软件和硬件领域的知识。如果做得不好，您可能会陷入一种局面，即花费了成千上万甚至数百万美元，但模型效果并不理想。特别是在训练后期，错误很难修复/撤销。 • 比选项 2 效率低。选项 2 利用现有的 LLM，从整个互联网的数据中学习，并提供坚实的起点。而在选项 3 中，您需要从零开始，需要大量高质量/多样化的数据集，使您的模型获得广义能力。
When to consider each option	• Best if you either have less technical teams but want to leverage LLM techniques to build downstream apps, or you want to leverage the best-in-class LLMs for performance reasons (outsourcing the LLM tech). • Good if you have very limited training datasets and want to leverage an LLM`s capability to do zero/few-shot learning. • Good for prototyping apps and exploring what is possible with LLMs.	• Between options two and three, if you aren’t trying to change the model architecture, it is almost always better to either directly take an existing pre-trained LLM and fine-tune it or take the weights of an existing pre-trained LLM as a starting point and continue pre-training. The reason is because a good pre-trained LLM like GPT-NeoX has already seen a vast amount of data and thus has learned general capabilities from the data. You can leverage that learning especially if your training dataset is not huge or diverse. • Another typical scenario is that you operate in a regulatory environment or have user / sensitive data that cannot be fed to commercial LLM services. Or you need edge	• Best if you need to change model architecture or training dataset from existing pre-trained LLMs. For example, if you want to use a different tokenizer, change the vocabulary size, or change the number of hidden dimensions, attention heads, or layers. • Typically, in this case the LLM is a core part of your business strategy & technological moat. You are taking on some or a lot of innovations in LLM training, and have a large investment appetite to train and maintain expensive models on an ongoing basis. • Typically, you have or will have lots of proprietary data associated with your LLM to create a con-

		deployment of the model for latency or locational reasons.	tinuous model improvement loop for sustainable competitive advantage.
三种不同模型如何选择？	• 如果您的技术团队较少，但希望利用 LLM 技术构建下游应用程序，或者出于性能原因希望利用最佳 LLM 技术（外包 LLM 技术），则这是最佳选择。 • 如果您拥有非常有限的训练数据集，并希望利用 LLM 的能力进行零/少次尝试学习，则这是一个不错的选择。 • 用于原型设计应用程序并探索 LLM 的潜力也是不错的选择。	• 在选项 2 和选项 3 之间，如果您不打算更改模型架构，更好的选择是直接使用现有的预训练 LLM 并进行微调，或者使用现有的预训练 LLM 权重作为起点继续进行预训练。原因是，像 GPT-NeoX 这样的优秀预训练 LLM 已经看到了大量的数据，因此已经从数据中学到了通用能力。如果您的训练数据集不是很大或不太多样化，您可以利用这种学习。 • 另一种典型情况是，您在监管环境中运营，或者拥有不能提供给商业 LLM 服务的用户/敏感数据。或者出于延迟或地理原因，您需要在边缘部署模型。	• 如果您需要从现有的预训练 LLM 中更改模型架构或训练数据集，则这是最佳选择。例如，如果您想使用不同的分词器，更改词汇大小，或更改隐藏维度、注意力头或层数。 • 通常情况下，LLM 是您业务战略和技术护城河的核心部分。您正在承担一些或大量的 LLM 训练创新，并且具有大量的投资意愿，以持续训练和维护昂贵的模型。 • 通常情况下，您拥有或将拥有大量与您的 LLM 相关的专有数据，以创建连续的模型改进循环，从而获得可持续的竞争优势。

It is also worth mentioning that if you only have a very targeted set of use cases and don’t need the general-purpose capabilities or generative capabilities from LLMs, you might want to consider training or fine-tuning a much smaller transformer or other much simpler deep learning models. That could result in much less complexity, less training time, and less ongoing costs.

同样值得一提的是，如果您只有一组非常有针对性的需求，而不需要来自 LLM 的通用功能或生成功能，那么您可能需要考虑训练或微调一个小得多的模型或其他更简单的深度学习模型。这可能会降低复杂性、训练时间和持续成本。

THE SCALING LAWS

缩放定律

Before you dive into training, it’s important to cover how LLMs scale. Understanding scaling lets you effectively balance the size and complexity of your model and the size of the data you’ll use to train it.

在开始训练之前，必须了解 LLM 如何扩展。了解缩放可以让您有效地平衡模型的大小和复杂程度，以及用于训练模型的数据大小。

Some relevant history here: OpenAI originally introduced "the LLM scaling laws" in 2020. They suggested that increasing model size was more important than scaling data size. This held for about two years before DeepMind suggested almost the polar opposite: that previous models were significantly undertrained and that increasing your foundational training datasets actually leads to better performance.

这里有一些相关的历史：OpenAI 最初在 2020 年提出了“LLM 缩放定律”。他们认为，扩大模型规模比扩大数据规模更重要。在 DeepMind 提出几乎截然相反的观点之前，这一观点维持了大约两年时间：以前的模型明显训练不足，增加基础训练数据集实际上会带来更好的性能。

That changed in 2022. Specifically, DeepMind put forward an alternative approach in their Training Compute-Optimal Large Language Models paper. They found that current LLMs are actually significantly undertrained. Put simply: these large models weren’t trained on nearly enough data.

这种情况在 2022 年发生了变化。具体来说，DeepMind 在他们的论文《训练计算最优的大型语言模型》中提出了另一种方法。他们发现，目前的 LLM 实际上训练严重不足。简单地说：这些大型模型没有经过足够的数据训练。

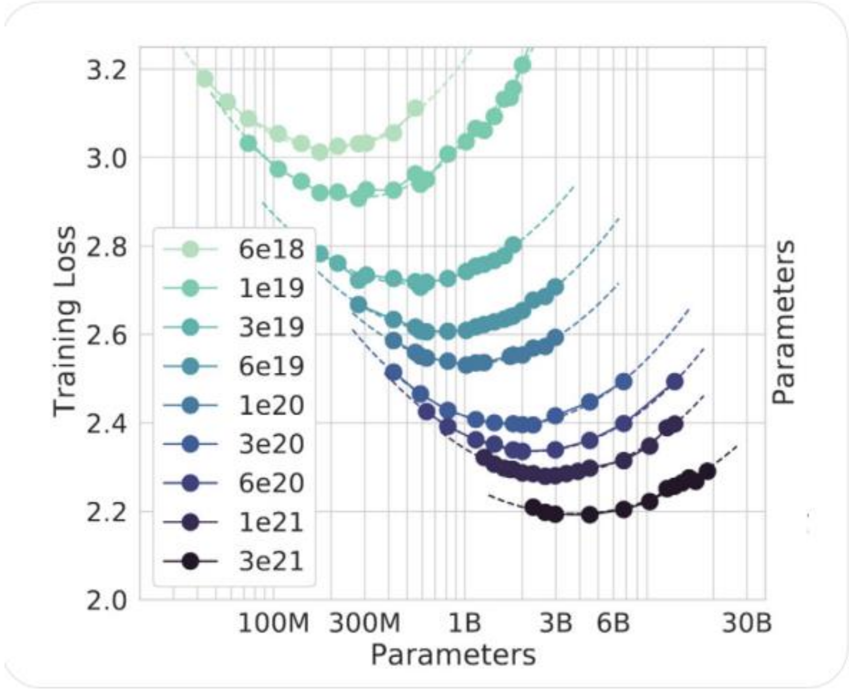
Deepmind showcased this with a model called Chinchilla, which is a fourth the size of the Gopher model above but trained on 4.6x more data. At that reduced size but with far more training data, Chinchilla outperformed Gopher and other LLMs.

DeepMind 通过一个名为 Chinchilla 的模型来展示这一点，该模型只有上面 Gopher 模型大小的四分之一，但是训练的数据是其 4.6 倍。尽管模型大小减小，但由于拥有更多的训练数据，Chinchilla 的表现超过了 Gopher 和其他 LLM。

DeepMind claims that the model size and the number of training tokens* should instead increase at roughly the same rate to achieve optimal performance. If you get a 10x increase in compute, you should make your model 3.1x times bigger and the data you train over 3.1x bigger; if you get a 100x increase in compute,

DeepMind 声称，模型大小和训练分词（词元 Token）*的数量应当大致以相同的速率增加，以达到最佳性能。如果你的计算能力增加了 10 倍，那么你应该让你的模型变得大 3.1 倍，并且训练的数据也增加 3.1 倍；如果你的计算能力增加了 100 倍，你应该让你的模型变得大 10 倍，并且训练的数据

you should make your model 10x bigger and your data 10x bigger.

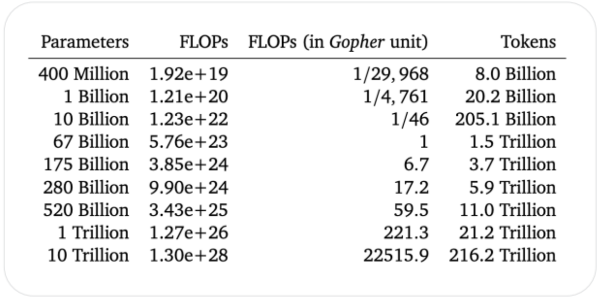


To the left of the minima on each curve, models are too small, a larger model trained on less data would be an improvement. To the right of the minima on each curve, models are to a large, a smaller model trained on more data would be an improvement. The best models are at the minima.

在每条抛物线顶点的左侧，模型太小了，一个更大的模型在较少的数据上训练会是一种改进。在每条抛物线顶点的右侧，模型太大了，一个较小的模型在更多的数据上训练会是一种改进。最佳的模型位于抛物线顶点上。

In summary, the current best practices in choosing the size of your LLM models are largely based on two rules: 1) Decide on your dataset and find the Chinchil-

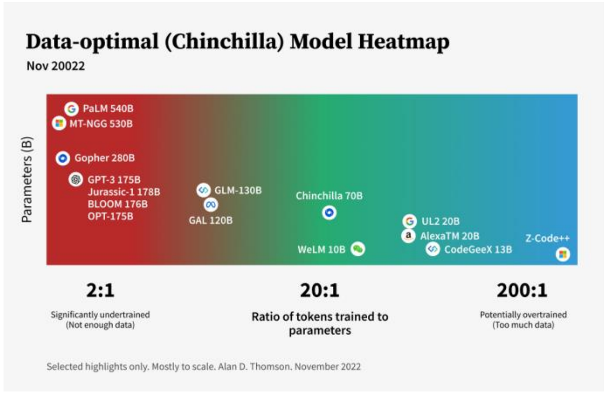
也增加 10 倍。



Parameters	FLOPs	FLOPs (in *Gopher* unit)	Tokens
400 Million	1.92e+19	1/29, 968	8.0 Billion
1 Billion	1.21e+20	1/4, 761	20.2 Billion
10 Billion	1.23e+22	1/46	205.1 Billion
67 Billion	5.76e+23	1	1.5 Trillion
175 Billion	3.85e+24	6.7	3.7 Trillion
280 Billion	9.90e+24	17.2	5.9 Trillion
520 Billion	3.43e+25	59.5	11.0 Trillion
1 Trillion	1.27e+26	221.3	21.2 Trillion
10 Trillion	1.30e+28	22515.9	216.2 Trillion

Estimated optimal training FLOPs and training tokens for various model sizes

针对不同模型规模的最佳训练 FLOP 和训练标记的估计值



Data/compute-optimal (Chinchilla) heat map, Chinchilla data-optimal scaling laws

数据/计算最优（钦奇拉）热图，钦奇拉数据最优缩放规律

总之，目前选择 LLM 模型大小的最佳做法主要基于两条规则：1）依据您的数据集决定，并根据数据大小找到钦奇拉最佳模型大小（或在数据收集限

la-optimal model size based on data size (or close to Chinchilla-optimal within the boundary of your data collection limitation). 2) Determine the data and model size combination that's best for your model, based on your training compute budget and inference latency requirements.

制范围内接近钦奇拉最佳模型大小）。2）根据训练计算预算和推理延迟要求，确定最适合模型的数据和模型大小组合。

HARDWARE

硬件

It should come as no surprise that pre-training LLMs is a hardware-intensive effort. The following examples of current models are a good guide here:

预训练 LLM 是一项硬件密集型工作，这一点不足为奇。以下模型的示例就是很好的指南：

PaLM (540B, Google): 6144 TPU v4 chips used in total, made of two TPU v4 Pods connected over data center network (DCN) using a combination of model and data parallelism.

PaLM（540B，谷歌）：总共使用了 6144 个 TPU v4 芯片，由两个通过数据中心网络（DCN）连接的 TPU v4 Pods 组成，采用了模型和数据并行相结合的方法。

OPT (175B, Meta AI): 992 80GB A100 GPUs, utilizing fully shared data parallelism with Megatron-LM tensor parallelism.

OPT（175B，Meta AI）：992 个 80GB A100 GPU，•利用完全共享的数据并行与 Megatron-LM 的张量并行。

GPT-NeoX (20B, EleutherAI): 96 40GB A100 GPUs in total.

GPT-NeoX（20B，EleutherAI）：共 96 个 40GB A100 GPU。

Megatron-Turing NLG (530B, NVIDIA & MSFT): 560 DGX A100 nodes, each cluster node has 8 NVIDIA 80-GB A100 GPUs.

Megatron-Turing NLG（530B、NVIDIA 和 MSFT）：560 个 DGX A100 节点，每个集群节点有 8 个英伟达 80GB 内存 A100 GPU。

Training LLMs is challenging from an infrastructure perspective for two big reasons. For starters, it is simply no longer possible to fit all the model parameters in the memory of even the largest GPU (e.g. NVIDIA 80GB-A100), so you'll need some parallel architecture here. The other challenge is that a large number of compute operations can result in unrealistically long training times if you aren't concurrently optimizing your algorithms, software, and hardware stack (e.g.

从基础架构的角度来看，训练 LLM 具有挑战性，主要有两个原因。首先，即使是最大的 GPU（如英伟达 80GB-A100）的内存也不可能容纳所有模型参数，因此您需要一些并行架构。另一个挑战是，如果不同时优化算法、软件和硬件堆栈，大量的计算操作会导致不切实际的超长训练时间（例如，使用单个 V100 NVIDIA GPU 训练具有 175B 参数的 GPT-3 需要约 288 年时间）。

training GPT-3 with 175B parameters would require about 288 years with a single V100 NVIDIA GPU).

Memory vs. Compute Efficiency

To achieve the full potential of thousands of distributed GPUs, it is crucial to design parallelism into your architecture to balance memory and compute efficiency.

内存与计算效率

要充分发挥数千个分布式 GPU 的潜力，在架构中设计并行性以平衡内存和计算效率至关重要。

Memory efficiency

Training a LLM requires terabytes of aggregate memory for model weights, gradients, and optimizer states - far beyond what is available on a single GPU. One typical mitigation strategy is gradient accumulation, in which the full training batch is split into micro-batches that are processed in sequence with their resulting gradients accumulated before updating the model weights. That means your training batch size can scale without increasing the peak resident activation memory.

内存效率

训练 LLM 需要数 TB 的总内存来存储模型权重、梯度和优化器状态，这远远超出了单个 GPU 的可用容量。一种典型的缓解策略是梯度累积，即在更新模型权重之前，将整个训练批次拆分成微批次依次处理，并累积由此产生的梯度。这意味着您可以在不增加常态活跃内存的峰值的情况下扩大训练批次规模。

Compute efficiency

While large GPU clusters can have thousands of high-throughput GPUs, achieving high compute efficiency at this scale is challenging. A large batch size can be an effective way to increase compute efficiency, because it increases the arithmetic intensity of a GPU kernel and helps amortize the time spent stalled on communication and synchronization. However, using too large of a batch size can have negative effects on the model quality. While parallelization is paramount, there are many different ways to do it. We’ll get into the most common in our next section.

计算效率

虽然大型 GPU 集群可以拥有成千上万个高吞吐量 GPU，但在这种规模下实现高计算效率却极具挑战性。大的批处理规模可以有效提高计算效率，因为它可以增加 GPU 内核的运算强度，并有助于摊销在通信和同步上耗费的时间。然而，使用过大的批处理规模可能会对模型质量产生负面影响。虽然并行化是最重要的，但有许多不同的方法。我们将在下一节介绍最常见的方法。

Techniques for Parallelization

Parallelization refers to splitting up tasks and distributing them across multiple processors or devices, such as GPUs, so that they can be completed simultaneously. This allows for more efficient use of compute resources and faster completion times compared to running on a single processor or device. Parallelized training across multiple GPUs is an effective way to reduce the overall time needed for the training process. There are several different strategies that can be used to parallelize training, including gradient accumulation, micro-batching, data parallelization, tensor parallelization and pipeline parallelization, and more. Typical LLM pre-training employs a combination of these methods. Let's define each:

Data Parallelism

Data parallelism is the best and most common approach for dealing with large datasets that cannot fit into a single machine in a deep learning workflow.

More specifically, data parallelism divides the training data into multiple shards (partitions) and distributes them to various nodes. Each node first works with its local data to train its sub-model, and then communicates with the other nodes to combine their results at certain intervals in order to obtain the global model.

The parameter updates for data parallelism can be either asynchronous or synchronous.

The advantage of this method is that it increases compute efficiency and that it is relatively easy to implement. The biggest downside is that during the backward pass you have to pass the whole gradient to all other GPUs. It also replicates the model and optimizer across all workers which is rather memory inefficient.

并行技术

并行是指将任务拆分并分配到多个处理器或 GPU 等设备上，以便同时完成训练任务。与在单个处理器或设备上运行相比，这样可以更有效地利用计算资源，加快完成时间。在多个 GPU 上进行并行化训练是实现目标减少时间的有效方法。有几种不同的策略可用于并行化训练，包括梯度累积、微批处理、数据并行化、张量并行化和流水线并行化等等。典型的 LLM 预训练采用了这些方法的组合。让我们分别定义一下：

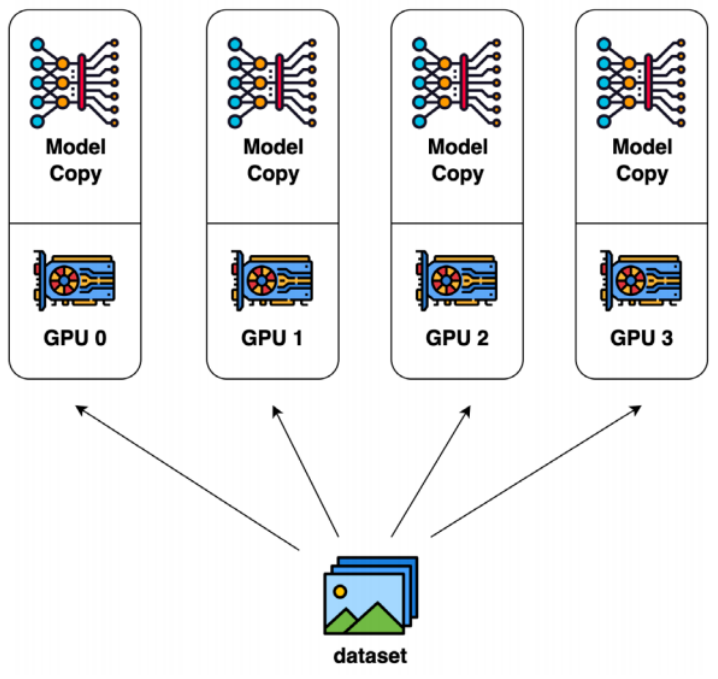
数据并行

数据并行是处理深度学习工作流程中单台机器无法处理的大型数据集的最佳和最常用的方法。

更具体地说，数据并行将训练数据分成多个分区，并将其分配给不同的节点。每个节点首先使用本地数据训练自己的子模型，然后与其他节点通信，在一定时间间隔内合并它们的结果，以获得全局模型。

数据并行的参数更新可以是异步的，也可以是同步的。

这种方法的优点是可以提高计算效率，而且相对容易实现。最大的缺点是在后向传递过程中，必须将整个梯度传递给所有其他 GPU。此外，它还会在所有工作站中复制模型和优化器，内存效率相当低。



图：数据并行示意图

Tensor Parallelism

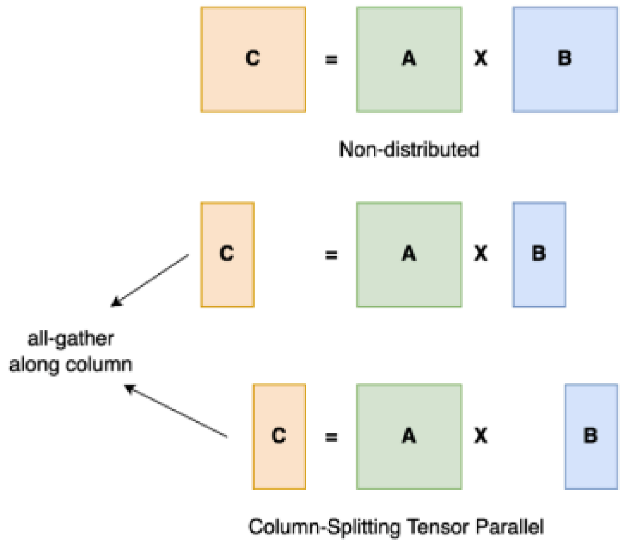
Tensor parallelism divides large matrix multiplications into smaller sub-matrix calculations which are then executed simultaneously using multiple GPUs.

This allows for faster training times due to its asynchronous nature and the ability to reduce communication overhead between nodes. The benefit of this method is that it is memory-efficient. The downside, however, is that it introduces additional communication of activations in each forward & backward propagation, and therefore requires high communication bandwidth to be efficient.

张量并行

张量并行将大型矩阵乘法划分为较小的子矩阵计算，然后使用多个 GPU 同时执行。

这种方法具有异步性，能够减少节点之间的通信开销，因此训练时间更短。这种方法的优点是节省内存。但缺点是，它在每次前向和后向传播中都会引入额外的激活通信，因此需要较高的通信带宽才能实现高效训练。



Pipeline parallelism and model parallelism

管道并行和模型并行

Pipeline parallelism improves both the memory and compute efficiency of deep learning training by partitioning the layers of a model into stages that can be processed in parallel.

管道并行通过将模型层划分为可并行处理的阶段，提高了深度学习训练的内存和计算效率。

This helps with overall throughput speeds significantly while adding the smallest communication overhead. You can think of pipeline parallelism as "inter-layer parallelism" (where tensor parallelism can be thought of as "intra-layer parallel-ism"). Similar to pipeline parallelism, model parallelism is when you split the model among GPUs and use the same data for each model; so each GPU works on a part of the model rather than a part of the data. The downside of pipeline and model parallelism is that it cannot scale infinitely given that the degree of pipeline parallelism is bounded by the depth of the model.

这有助于显著提高整体吞吐速度，同时增加最小的通信开销。你可以把流水线并行看作 “层间并行”（而张量并行可以看作“层内并行”）。与流水线并行类似，模型并行是指在 GPU 之间拆分模型，并在每个模型中使用相同的数据；因此每个 GPU 只处理模型的一部分，而不是数据的一部分。流水线并行和模型并行的缺点是无法无限扩展，因为流水线并行的程度受模型深度的限制。

As mentioned at the start of this section, it’s not uncommon for teams to leverage a combination of parallelism techniques during training. For example, PaLM (Google Brain, 2022) and OPT (Meta AI, 2022) both used a combination of tensor model parallelism and data parallelism.

NVIDIA approached things a little differently in the Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM paper. They proposed a PTD-P technique that combines pipeline, tensor, and data parallelism to achieve state-of-the-art computational performance (52% of peak device throughput) on 1000s of GPUs.

Specifically, PTD-P leverages a combination of pipeline parallelism across multi-GPU servers, tensor parallelism within a multi-GPU server, and data parallelism to practically train models with a trillion parameters. The method also employs graceful scaling in an optimized cluster environment with high-bandwidth links between GPUs on the same server and across servers.

Using these techniques to train LLMs requires not only the highest-performing GPUs to be efficient, but also needs high-bandwidth networking for optimal communication—InfiniBand is often used to move data between nodes.

But this of course comes with a cost. Leveraging thousands of high-performing GPUs and high-bandwidth networks to train LLMs is infrastructure-intensive. For example, a back-of-the-envelope calculation estimated that the cost of the PaLM model (540B, Google) might be as high as $23MM (see detailed analysis).

To implement distributed deep learning training systems, software toolkits such

正如本节开头提到的，团队在训练过程中综合利用并行技术的情况并不少见。例如，PaLM(Google Brain, 2022)和 OPT(Meta AI, 2022)都结合使用了张量模型并行和数据并行。

英伟达在《利用 Megatron-LM 在 GPU 集群上进行高效大规模语言模型训练》一文中采用了略微不同的方法。他们提出了一种 PTD-P 技术，该技术结合了流水线、张量和数据并行性，以实现在 1000 个 GPU 上实现最先进的计算性能（设备峰值吞吐量的 52%）。

具体来说，PTD-P 综合利用了多 GPU 服务器之间的流水线并行、多 GPU 服务器内部的张量并行和数据并行，可实际训练具有万亿个参数的模型。该方法还在优化的集群环境中，利用同一服务器和跨服务器 GPU 之间的高带宽连接，实现了优美的扩展。

使用这些技术来训练 LLM 不仅需要性能最高的 GPU 效率，而且还需要高带宽的网络来实现最佳的通信——InfiniBand 通常用于在节点之间移动数据。

但这当然是有代价的。利用数以千计的高性能 GPU 和高带宽网络来训练 LLM 需要大量基础设施。例如，根据反向计算估计，PaLM (540B, Google)的成本可能高达 2300 万 美元（见详细分析）。

要实现分布式深度学习训练系统，通常需要分布式 TensorFlow、Torch Disas Distributed TensorFlow, Torch Distributed, Horovod, and libraries such as DeepSeed and Megatron are often needed. There is implementation complexity here so it requires system expertise if you're going to be successful.

tributed、Horovod 等软件工具包以及 DeepSeed 和 Megatron 等库。这里存在实施的复杂性，因此要想取得成功，就需要系统方面的专业知识。

In addition, the following techniques and strategies are commonly employed to achieve parallelism:

此外，还有以下技术和战略通常用于实现并行性：

Gradient accumulation

梯度累积

Gradient accumulation involves adding up gradients from multiple batches before performing one weight update step on all accumulated gradients at once.

梯度累积是将多个批次的梯度相加，然后对所有累积梯度一次性执行一个权重更新步骤。

This approach reduces communication overhead between GPUs by allowing them to work independently on their own local batch of data until they have synchronized with each other again, after accumulating enough gradients for a single optimization step.

这种方法通过减少 GPU 之间的通信开销，允许它们在自己的本地批数据上独立工作，直到它们在积累足够的梯度后，再次彼此同步。

Asynchronous stochastic gradient descent optimization

异步随机梯度下降优化

Asynchronous stochastic gradient descent optimization methods can also be employed when performing model optimization over multiple GPUs.

在多个 GPU 上进行模型优化时，也可以采用异步随机梯度下降优化方法。

This method uses small subsets (microbatches) of data from each node instead of loading all data at once, which helps reduce memory requirements while still allowing for fast convergence rates due to its asynchronous nature. It works like this:

这种方法使用每个节点的小数据子集（微批次），而不是一次性加载所有数据，这有助于减少内存需求，同时由于其异步性质，仍可实现快速收敛率。其工作原理如下：

- First, we fetch the most up-to-date parameters of the model needed to process

首先，我们从参数服务器获取处理当前迷你批次所需的最新模型参数。

the current mini-batch from the parameter servers.

- We then compute gradients of the loss with respect to these parameter.

然后，我们计算损失相对于这些参数的梯度。

- Finally, these gradients are sent back to the parameter servers, which then updates the model accordingly.

最后，这些梯度被送回参数服务器，然后相应地更新模型。

Micro-batching

微批处理

Micro-batching combines small mini-batches into larger ones so that more batches can be processed in less time and with fewer synchronization points between devices during back propagation operations. It has become increasingly popular for training very large models across many GPUs due to its ability to reduce memory consumption and improve scalability. Overall, microbatching is an effective way to leverage distributed deep learning techniques when dealing with very large datasets or models that require significant amounts of processing power.

微批处理将小批量组合成更大的批量，这样在反向传播操作中就可以在更短的时间内处理更多的批量，并减少设备之间的同步点。由于它能够减少内存消耗和提高可伸缩性，它在许多 GPU 上训练非常大的模型已经变得越来越受欢迎。总的来说，在处理需要大量处理能力的非常大的数据集或模型时，微批处理是利用分布式深度学习技术的一种有效方法。

Now that we’ve gone through scaling, hardware, and some techniques for parallelizing your training runs, let’s look at what your LLM will actually learn from: data.

现在我们已经经历了缩放、硬件和一些并行化训练运行的技术，让我们看看 LLM 将从中学到什么：数据。

DATASET COLLECTION

数据集整合

Bad data leads to bad models. But careful processing of high-quality, high-volume, diverse datasets directly contributes to model performance in

糟糕的数据会导致糟糕的模型。但仔细处理高质量、大量、多样的数据集直接有助于模型在下游任务中的性能以及模型收敛。

downstream tasks as well as model convergence.

Dataset diversity is especially important for LLMs. That’s because diversity improves the cross-domain knowledge of the model, as well as its downstream generalization capability. Training on diverse examples effectively broadens the ability of your LLM to perform well on myriad nuanced tasks.

数据集多样性对于 LLM 尤为重要。这是因为多样性可以提高模型的跨领域知识，以及其下游泛化能力。通过对不同示例的训练，可以有效提高 LLM 的能力，使其能够出色地完成各种细微的任务。

A typical training dataset is comprised of textual data from diverse sources, such as crawled public data, online publication or book repositories, code data from GitHub, Wikipedia, news, social media conversations, etc.

典型的训练数据集由不同来源的文本数据组成，如抓取的公共数据、在线出版物或书籍库、GitHub 的代码数据、维基百科、新闻、社交媒体对话等。

For example, consider The Pile. The Pile is a popular text corpus created by EleutherAI for large-scale language modeling. It contains data from 22 data sources, coarsely broken down into five broad categories:

例如，The Pile。The Pile 是 EleutherAI 为大规模语言建模而创建的流行文本语料库。它包含来自 22 个数据源的数据，粗略地分为五大类：

Academic Writing: PubMed Abstracts and PubMed Central, arXiv, FreeLaw, USPTO Backgrounds, PhilPapers, NIH Exporter.

学术写作：PubMed 摘要和 PubMed Central、arXiv、FreeLaw、美国专利商标局背景资料、PhilPapers、NIH Exporter。

Online or Scraped Resources: CommonCrawl, OpenWebText2, Stack Exchange, Wikipedia.

在线或抓取资源：CommonCrawl、OpenWebText2、Stack Exchange、维基百科。

Prose: BookCorpus2, Bibliotik, Project Gutenberg.

散文：BookCorpus2, Bibliotik, 古腾堡计划。

Dialog: YouTube subtitles, Ubuntu IRC, OpenSubtitles, Hacker News, Europarl.

对话：YouTube 字幕， Ubuntu IRC, OpenSubtitles, Hacker News, Europarl。

Miscellaneous: GitHub, the DeepMind Mathematics dataset, Enron emails.

杂项：GitHub、DeepMind 数学数据集、安然 Enron 电子邮件。

Note that The Pile dataset is one of the very few large-scale text datasets that is free for the public. For most of the existing models like GPT-3, PaLM, and Galactica, their training and evaluation datasets are not publicly available. Given the large scale effort it takes to compile and pre-process these datasets for LLM training, most companies have kept them in-house to maintain competitive ad-

请注意，公共数据集是极少数的对公众免费的大型文本数据集之一。对于大多数现有的模型，如 GPT-3、PaLM 和 Galactica，它们的训练和评估数据集是不公开的。考虑到为 LLM 训练编写和预处理这些数据集所需的大规模努力，大多数公司都将它们保留在内部，以保持竞争优势。这使得像 The Pile 这样的数据集和来自 AllenAI 的一些数据集对于公共的大规模 NLP 研究目的

vantage. That makes datasets like The Pile and a few datasets from AllenAI extremely valuable for public large-scale NLP research purposes.

非常有价值。

Another thing worth mentioning is that, during dataset collection, general data can be collected by non-experts but data for specific domains normally needs to be collected or consulted by subject matter experts (SMEs), e.g. doctors, physicists, lawyers, etc. SMEs can flag thematic or conceptual gaps that NLP engineers might miss. NLP engineers should also be heavily involved at this stage given their knowledge of how a LLM "learns to represent data" and thus their abilities to flag any data oddities or gaps in the data that SMEs might miss.

另外值得一提的是，在数据集收集过程中，一般数据可以由非专家收集，但特定领域的数据通常需要收集或咨询领域专家（SME），如医生、物理学家、律师等。领域专家可以指出 NLP 工程师可能会忽略的主题或概念上的漏洞。鉴于 NLP 工程师了解 LLM "学习如何表示数据"，因此他们也应在这一阶段大力参与，这样他们才有能力标出领域专家可能会忽略的任何数据怪异之处或数据缺陷。

Once you’ve identified the dataset(s) you’ll be using, you’ll want to prepare that data for your model. Let’s get into that now:

确定要使用的数据集后，您需要为模型准备数据。让我们现在就开始探讨吧：

DATASET PRE-PROCESSING

数据集预处理

In this section, we’ll cover both data adjustments (like deduplication and cleaning) and the pros and cons of various tokenization strategies. Let’s start with the former:

在本节中，我们将介绍数据调整（如去重和清洗）以及各种词元化策略的利弊。让我们从前者开始：

Data sampling:

数据集处理

To ensure training data is high-quality and diverse, several pre-processing techniques can be used before the pre-training steps:

为了确保训练数据的高质量和多样化，在预训练步骤之前可以使用几种数据预处理技术：

Data sampling

数据取样

Certain data components can be up-sampled to obtain a more balanced data distribution. Some research down-samples lower-quality dataset such as unfiltered web crawl data. Other research up-samples data of a specific set of domains depending on the model objectives.

可以对某些数据组分层面进行上采样，以获得更均衡的数据分布。有些研究对质量较低的数据集（如未经过滤的网络抓取数据）进行下采样。其他研究会根据模型的目标对特定领域的数据进行上采样。

There are also advanced methods to filter high-quality data such as using a trained classifier model applied to the dataset. For example, the model Galactica by Meta AI is built purposefully for science, specifically storing, combining and reasoning about scientific knowledge.

还有一些过滤高质量数据的先进方法，例如使用训练有素的模型对数据集数据进行过滤。例如，Meta AI 的 Galactica 模型就是专门为科学而设计的，特别是存储、组合和推理科学知识。

Due to its goals, the pre-training dataset is composed of high-quality data mainly from science resources such as papers, textbooks, lecture notes, encyclopedias. The dataset is also highly curated, for example, with task-specific datasets to facilitate composition of this knowledge into new task contexts.

由于其目标，预训练数据集主要由来自科学资源（如论文、教科书、讲义、百科全书）的高质量数据组成。此外，数据集还经过精心策划，例如，包含特定任务的数据集，以方便将这些知识融入新的任务情境中。

Data cleaning

数据清理

Normally, data cleaning and reformatting efforts are applied before training. Examples include removing boilerplate text and removing HTML code or markup. In addition, for some projects, fixing misspellings, handling cross-domain homographs, and/or removing biased / harmful speech are performed to improve model performance. For other projects, these techniques are not used under the idea that models should see the fair representation of the real world and learn to deal with misspellings and toxicity as a part of the model capabilities.

通常情况下，数据清洗和重新格式化工作是在训练之前进行的。例如，删除模板文本和 HTML 代码或标记。此外，在某些项目中，还需要修正拼写错误、处理跨域同音异义词，和/或删除有偏差/有害的言语，以提高模型性能。对于其他项目，这些技术并未被使用，原因是我们认为，模型应该看到真实世界的客观表现，作为模型能力的一部分，学会处理包括拼写错误和毒性在内的这些问题。

Non-standard textual components handling

处理非标准文本组件

In some cases, it is important to convert non-standard textual components into texts, e.g. converting emoji into their text equivalent: This conversion can be

在某些情况下，将非标准文本组件转换成文本非常重要，例如将表情符号转换成对应的文本。当然，这种转换可以通过编程完成。

done programmatically, of course.

Data deduplication

Some researchers see significant benefits from deduplicating training data. Fuzzy deduplication methods such as locality sensitive hashing (LSH) are commonly used here. See Deduplicating Training Data Makes Language Models Betterpaper to understand details regarding deduplication.

重复数据删除

一些研究人员认为数据去重可以带来巨大的好处。模糊数据去重方法（如定位敏感哈希算法 (LSH)）常用于此。有关数据去重的详细信息，请参阅《重复训练数据使语言模型更完善》一文。

Downstream task data removal

Data leakage happens when the data you are using to train happens to have the information you are trying to predict. Downstream task data removal methods (such as n-grams) are needed to remove training data also present in the evaluation dataset.

删除下游任务数据

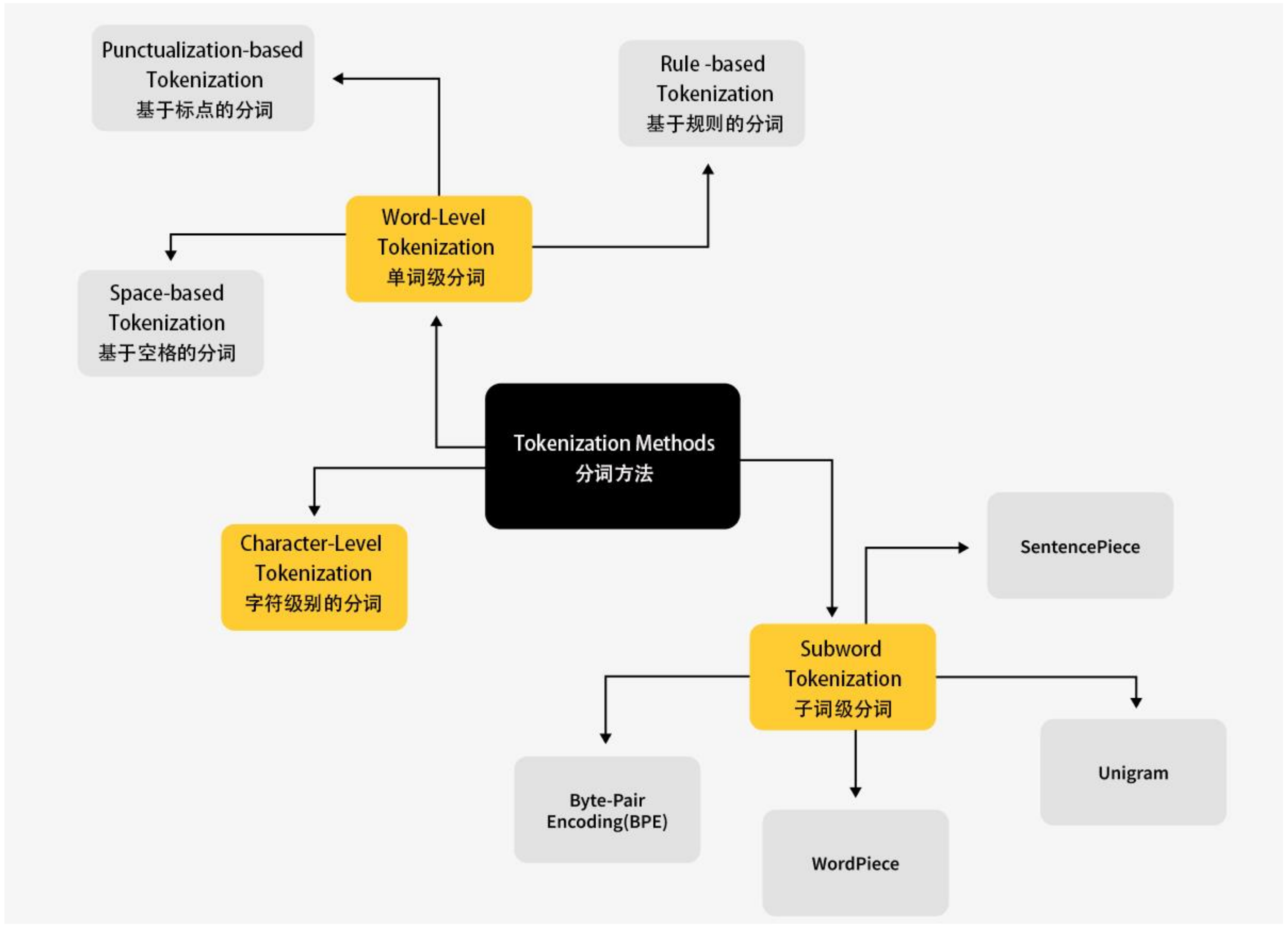
当你用来训练的数据中恰好有你试图预测的信息时，就会发生数据泄露。需要使用下游任务数据移除方法（如 n-grams）来移除评估数据集中的训练数据。

Tokenization

Tokenization is the process of encoding a string of text into transformer-readable token ID integers. Most state-of-the-art LLMs use subword-based tokenizers like byte-pair encoding (BPE) as opposed to word-based approaches. We'll present the strengths and weaknesses of various techniques below, with special attention to subword strategies as they're currently the most popular versus their counterparts.

词元化

词元化（分词，Tokenization）是将文本字符串编码成转换器 Transformer 可读的词元 ID 整数的过程。大多数先进的 LLM 都使用基于子词的分词器，如字节对编码 (BPE)，而不是基于单词的方法。下面我们将介绍各种技术的优缺点，并特别关注子词策略，因为它们是目前最流行的同类技术。



Punctualization-based Tokenization 基于标点的分词
Rule -based Tokenization 基于规则的分词
Word-Level Tokenization 单词级分词
Space-based Tokenization 基于空格的分词
Tokenization Methods 分词方法
Character-Level Tokenization 字符级别的分词
SentencePiece
Subword Tokenization 子词级分词
Byte-Pair Encoding(BPE)
WordPiece
Unigram

Summary of the most used tokenization methods, Two minutes NLP — A Taxonomy of Tokenization Methods

最常用的词元化方法汇总，两分钟 NLP -词元化方法分类学

Tokenization Methods	Word-based tokenization	Character-based tokenization	Subword-based tokenization
标记化方法	基于单次的词元化	基于字符的词元化	基于子词的词元化
Example Tokenizers	Space tokenization (split sentences by space); rule-based tokenization (e.g. Moses, spaCy).	Character tokenization (simply tokenize on every character).	Byte-Pair Encoding (BPE); WordPiece; Sentence-Piece; Unigram (tokenizing by parts of a word vs. the entirety of a word; see table above).
分词符号示例	空格词元化（用空格分割句子）；基于规则的词元化（如 Moses、spaCy）。	字符词元化（只需对每个字符进行词元化处理）。	字节对编码 (BPE)；单词片段； 句子片段；单字符（按单词的部分与单词的全部进行标记化；见上表）。
Considerations	Downside: Generates a very large vocabulary leading to a huge embedding matrix as the input and output layer; large number of out-of-vocabulary (OOV) tokens; and different meanings of very similar words. Transformer models normally have a vocabulary of less than 50,000 words, especially if they are trained only on a single language.	Lead to much smaller vocabulary; no OOV (out of vocabulary) tokens since every word can be assembled from individual characters. Downside: Generates very long sequences and less meaningful individual tokens, making it harder for the model to learn meaningful input representations. However, if character-based tokenization is used on non-English language, a single character could be quite information rich (like “mountain” in Mandarin).	Subword-based tokenization methods follow the principle that frequently used words should not be split into smaller subwords, but rare words should be decomposed into meaningful subwords Benefit: Solves the downsides faced by word-based tokenization and character-based tokenization and achieves both reasonable vocabulary size with meaningful learned context-independent representations.
考虑因素	缺点：产生的词汇量非常大，导致庞大的嵌入矩阵作为输入和输出层；大量表外词汇（OOV）词元；以及非常多相似不同意义词的标记。 转换器模型的词汇量通常少于 5 万个单词，尤其是在只用一种语言 进行训练的情况下。	词汇量大大减少；由于每个单词都可以由单个字符组合而成， 因此不存在 OOV 标记词。 缺点：产生的序列很长但意义不大的单个词元。这样，模型就很难学习到有意义的输入表征。但是，如果在非英语语言中使用基于字符的标记化技术，单个字符的信息量可能会很大（如汉语中的"山"）。	基于子词的词元化方法遵循的原则是，常用词不应拆分成更小的子词，但罕见词应分解成有意义的子词。 优势：解决了基于单词的词元化和基于字符的词元化所面临的弊端，并且既实现了合理的词汇量，又实现了有意义的与上下文无关的学习表征。

In other words, the choice of tokenization technique depends on the specific task and language being analyzed.

换句话说，词元化技术的选择取决于具体任务和分析语言。

Word-based tokenization is simple and efficient but can be limited in its ability to handle complex languages.

基于单词的词元化简单而高效，但在处理复杂语言方面能力有限。

Character-based tokenization can be useful for languages without distinct word boundaries.

基于字符的词元化对于没有明显词界的语言非常有用。

Subword-based tokenization, including BPE, wordpiece tokenization, sentence-piece tokenization, and unigram tokenization, is particularly useful for handling complex morphology and out-of-vocabulary words.

基于子词的词元化，包括 BPE、词块词元化、句子块词元化和单字块词元化，对于处理复杂的词形和词汇表以外的词特别有用。

Let’s look at those subword-based methods in a little more detail:

让我们来详细了解一下这些基于子词的方法：

Subword-based Tokenization Methods	Byte-Pair Encoding (BPE)	WordPiece	Unigram	SentencePiece
基于子词的词元化方法	字节对编码（BPE）	WordPiece	Unigram	SentencePiece
Description	One of the most popular subword tokenization algorithms. The Byte-Pair-Encoding works by starting with characters, while merging those that are the most frequently seen together, thus creating new tokens. It then works iteratively to build new tokens out of the most frequent pairs it sees in a corpus. BPE is able to build words it has never seen by using multiple subword tokens, and thus requires smaller vocabularies, with less	Very similar to BPE. The difference is that WordPiece does not choose the highest frequency symbol pair, but the one that maximizes the likelihood of the training data once added to the vocabulary (evaluates what it loses by merging two symbols to ensure it’s worth it).	In contrast to BPE / WordPiece, Unigram initializes its base vocabulary to a large number of symbols and progressively trims down each symbol to obtain a smaller vocabulary. It is often used together with SentencePiece.	The left 3 tokenizers assume input text uses spaces to separate words, and therefore are not usually applicable to languages that don’t use spaces to separate words (e.g. Chinese). SentencePiece treats the input as a raw input stream, thus including the space in the set of characters to use. It then uses the BPE / Uni-

	chances of having "unk" (unknown) tokens.			gram algorithm to construct the appropriate vocabulary.
说明	最流行的子词词元化算法之一。该算法字节对编码 BPE 的工作原理是从字符开始，合并同时出现频率最高的字符，从而创建新的词元。然后，它将在语料库中出现频率最高的词对进行迭代，创建新的词元。BPE 能够通过使用多个子词词元来构建它从未见过的词，因此所需的词汇量较小，而降低带有"unk"（未知）词元出现的几率。	与 BPE 非常相似。不同之处在于，WordPiece 不会选择频率最高的符号对，而是选择在添加到词汇表后能最大化训练数据相似度的符号对（评估合并两个符号所造成的损失，以确保合并是值得的）	与 BPE / WordPiece 不同的是，Unigram 将其基本词库初始化为大量符号，并逐步减少每个符号的数量以获得更小的词汇量。它通常与 SentencePiece 一起使用。	左侧 3 种词元化方法假定输入文本使用空格分隔单词，因此通常不适用于不使用空格分隔单词的语言（如中文）。SentencePiece 将输入视为原始输入流，因此在字符集中包含了空格。然后，它使用 BPE / Unigram 算法来构建适当的词汇。
Considerations	BPE is particularly useful for handling rare and out-of-vocabulary words since it can generate subwords for new words based on the most common character sequences. Downside: BPE can result in subwords that do not correspond to linguistically meaningful units.	WordPiece can be particularly useful for languages where the meaning of a word can depend on the context in which it appears.	Unigram tokenization is particularly useful for languages with complex morphology and can generate subwords that correspond to linguistically meaningful units. However, unigram tokenization can struggle with rare and out-of-vocabulary words.	SentencePiece can be particularly useful for languages where the meaning of a word can depend on the context in which it appears.
考虑因素	BPE 可以根据最常见的字符序列为新词生成子词，因此在处理罕见词和词汇表以外的词时特别有用。 缺点：BPE 可能会产生与语言意义不符的子词单位。	对于单词的含义可能取决于其出现的语境的语言，WordPiece 尤其有用。	Unigram 单字元词元化尤其适用于语态复杂的语言，并能生成子词与语言意义单元相对应。然而在处理罕见词和词汇表以外 OOV 的词时可能会遇到困难。	SentencePiece 对于一些语言尤其有用，因为在这些语言中，一个词的含义可能取决于它出现的语境。
Transformers using this tokenization method	GPT-2, GPT3, Roberta	BERT, DistilBERT, Electra	Unigram is not used directly for any of the transformer models. Instead, it's used together with SentencePiece.	ALBERT, XLNet, Marian, T5
使用这种词元化方法的转换器模型	GPT-2, GPT3, Roberta	BERT, DistilBERT, Electra	Unigram 并不直接用于任何转换器模型。相反，它与 SentencePiece 一起使用。	ALBERT, XLNet, Marian, T5

Note: there is also recent work proposing a token-free model (ByT5) that learns from raw bytes. The benefit is that these models can process text in any language out of the box, that they work better with corpora with a large quantity of OOV (out-of-vocabulary) words/tokens and are more robust to noise, and that they minimize technical debt by removing complex and error-prone text preprocessing pipelines. The downside is that they are in general less accurate than tokenization-based models. More research in this area is needed to determine how promising this direction is.

After tokenization, we usually want to consider a few extra steps, namely padding and truncation. Padding is a strategy to ensure tensors are rectangular by adding a special padding token to shorter sentences so all inputs have a uniform tensor shape. In contrast, truncation shortens the length of sequences for the ones that are too long for a model to handle.

Note: there is an inherent limitation for subword tokenizer based LLMs that is directly related to the tokenizer technique. Due to that subword-based tokenizer's token granularity is between word and character, LLMs don't see letters and words as humans do. Instead, it sees "tokens," which are chunks of characters. An example due to this inherent limitation is that they have trouble merging characters.

PRE-TRAINING STEPS

Training a multi-billion parameter LLM is usually a highly experimental process

注：最近也有研究提出了一种从原始字节学习的无分词模型(ByT5)。这样做的好处是，这些模型可以直接处理任何语言的文本，而且在有大量 OOV 的语料库中效果更好。对噪声具有鲁棒性，通过消除复杂和易出错的文本预处理流水线，从而最大限度地减少了技术债务。缺点是，相较于基于词元化的模型，它们的准确性一般。要确定这一方向的前景如何，还需要在这一领域开展更多的研究。

注意：基于子词的词元器的 LLM 有一个固有的限制，这与词元器技术直接相关。由于基于子词的词元器的标记粒度介于单词和字符之间，因此 LLM 不会像人类那样看到字母和单词。相反，它看到的是 "词元"，即字符块，这是其固有的局限性，一个例子就是 LLM 在合并字符时会遇到困难。

在词元化（分词）之后，我们通常还要考虑一些额外的步骤，即填充和截断。填充是一种确保张量为矩形的策略，通过在较短的句子中添加特殊的填充标记，使所有输入的张量形状一致。与此相反，截断会缩短序列的长度，以避免模型无法处理过长的序列。

预训练步骤

训练千亿参数 LLM 通常是一个高度实验性的过程，需要进行大量的试错。

with lots of trial and error. Normally, the team would start with a much smaller model size, make sure it’s promising, and scale up to more and more parameters. Keep in mind that as you scale, there will be issues that require addressing which simply won’t be present when training on smaller data sizes.

通常情况下，团队会从规模小得多的模型开始，确保其前景良好，然后将规模扩大到越来越多的参数。请留意，随着规模的扩大，会出现一些在较小的数据规模上进行训练时不会出现而需要解决的问题。

Let’s look at some common pre-training steps, starting with architecture.

让我们从架构入手，看看一些常见的预训练步骤。

Model Architecture

模型架构

To reduce risk of training instabilities, practitioners often start with the model architecture and hyperparameters of a popular predecessor such as GPT-2 and GPT-3 and make informed adjustments along the way to improve training efficiency, scale the size of the models (in both depth and width), and improve performance. Two examples:

为了降低训练不稳定的风险，实践者通常会从 GPT-2 和 GPT-3 等流行前代的模型架构和超参数开始，并在此过程中做出明智的调整，以提高训练效率、扩大模型规模（深度和宽度）并提高性能。举两个例子：

GPT-NeoX-20B (20B, EleutherAI) originally took GPT-3’s architecture and made these changes:

GPT-NeoX-20B（20B，EleutherAI）最初采用了 GPT-3 的架构，并做了上述改动：

• Rotary embedding used for the first 25% of embedding vector dimensions instead of learned positional embeddings, to balance performance and computational efficiency.

• 旋转嵌入用于嵌入向量维度的前 25%，而不是学习的位置嵌入，以平衡性能和计算效率。

• Parallel attention combined with feed forward layers instead of running them in series, primarily for computing efficiency purposes.

• 并行注意与前馈层相结合，而不是串联运行，主要是为了提高计算效率。

• While GPT-3 uses alternating dense and sparse layers, GPT-NeoX exclusively uses dense layers to reduce implementation complexity.

• GPT-3 交替使用密集层和稀疏层，而 GPT-NeoX 则只使用密集层，以降低实现的复杂性。

OPT-175B (175B, Meta AI) also built on GPT-3 and adjusted:

• Batch size for increased computational efficiency.

• Learning rate schedule. Specifically, it follows a linear learning rate (LR) schedule, warming up from 0 to the maximum learning rate over the first 2000 steps in OPT-175B, or over 375M tokens in the smaller baselines, then decaying down to 10% of the maximum LR over 300B tokens. A number of mid-flight changes to LR were also required.

• Token amount. OPT-175B, despite the same model size as GPT-3 (175B) was trained on a much smaller dataset of 180B tokens (as compared to the 300B tokens by GPT-3).

Experiments and Hyperparameter Search

As we mentioned above, typical pre-training involves lots of experiments to find the optimal setup for model performance.

Experiments can involve any or all of the following: weight initialization, positional embeddings, optimizer, activation, learning rate, weight decay, loss function, sequence length, number of layers, number of attention heads, number of parameters, dense vs. sparse layers, batch size, and dropout.

A combination of manual trial and error of those hyperparameter combinations and automatic hyperparameter optimization (HPO) are typically performed to find the optimal set of configurations to achieve optimal performance. Typical hyperparameters to perform automatic search on: learning rate, batch size, dropout, etc.

OPT-175B（175B，Meta AI）也是基于 GPT-3 并进行了调整：

• 批量大小，提高计算效率。

• 学习率计划。具体来说，它遵循线性学习率（LR）计划，在 OPT- 175B 的前 2000 个步骤中从 0 升温到最大学习率，或在较小基线中超过 3.75 亿个代币，然后在超过 3 亿个代币时衰减到最大 LR 的 10%。在训练过程中，还需要对 LR 进行多次修改。

• 令牌数量。尽管 OPT-175B 的模型大小与 GPT-3 相同（175B），但其训练数据集却小得多，只有 180B（相比之下，GPT-3 的数据集为 300B ）。

实验和超参数搜索

如上所述，典型的预训练包括大量实验，以找到模型性能的最佳设置。

实验可以涉及以下任何一项或全部内容：权重初始化、位置嵌入、优化器、激活、学习率、权重衰减、损失函数、序列长度、层数、注意力头数、参数量、密集层与稀疏层、批量大小和退出。

通常会对这些超参数组合进行人工试错和自动超参数优化（HPO），以找到实现最佳性能的最优配置集。进行自动搜索的典型超参数包括：学习率、批量大小、退出率等。

Hyperparameter search is an expensive process and is often too costly to perform at full scale for multi-billion parameter models.

超参数搜索是一个昂贵的过程，对于数十亿参数的模型来说，进行全面搜索往往成本太高。

It’s common to choose hyperparameters based on a mixture of experiments at smaller scales and by interpolating parameters based on previously published work instead of from scratch.

通常的做法是，根据较小规模的混合实验来选择超参数，并根据以前发表的工作来内插参数，而不是从头开始。

In addition, there are some hyperparameters that need to be adjusted even during a training epoch to balance learning efficiency and training convergence. Some examples:

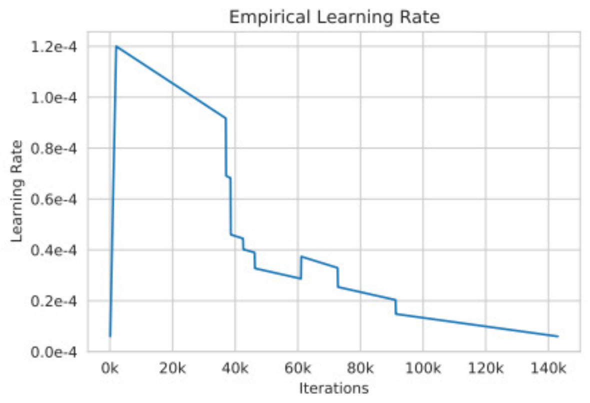
此外，有些超参数甚至需要在训练期间进行调整，以平衡学习效率和训练收敛性。举几个例子：

• Learning rate: can increase linearly during the early stages, then decay towards the end.

• 学习率：在早期阶段可呈线性增长，但在后期阶段会逐渐下降。

• Batch size: it’s not uncommon to start with smaller batch sizes and gradually ramp up to larger ones.

• 批量大小：从较小的批量开始，逐渐增加到较大的批量，这种情况并不少见。



You'll want to do a lot of this early in your pre-training process. This is largely because you'll be dealing with smaller amounts of data, letting you perform more experiments early versus when they'll be far more costly down the line.

在预训练过程中，您需要尽早进行大量实验。这主要是因为您需要处理的数据量较小，因此您可以在早期进行更多实验，而不是在后期进行成本更高的实验。

Before we continue, it's worth being clear about a reality here: you will likely run into issues when training LLMs. After all: these are big projects and like anything sufficiently large and complicated, things can go wrong.

在我们继续之前，有必要明确一个现实：在训练 LLM 时，您很可能会遇到问题。毕竟，这些都是大项目，就像任何足够庞大和复杂的事情一样，可能会出错。

Hardware Failure

硬件故障

During the course of training, a significant number of hardware failures can occur in your compute clusters, which will require manual or automatic restarts. In manual restarts, a training run is paused, and a series of diagnostics tests are conducted to detect problematic nodes. Flagged nodes should then be cordoned off before you resume training from the last saved checkpoint.

在训练过程中，计算集群可能会出现大量硬件故障，需要手动或自动重启。手动重启时，训练运行会暂停，并进行一系列诊断测试以检测有问题的节点。然后，在从上次保存的检查点恢复训练之前，应封锁标记节点。

Training Instability

训练的不稳定性

Training stability is also a fundamental challenge. While training the model, you may notice that hyperparameters such as learning rate and weight initialization directly affect model stability. For example, when loss diverges, lowering the learning rate and restarting from an earlier checkpoint might allow the job to recover and continue training.

训练稳定性也是一个基本挑战。在训练模型时，您可能会注意到学习率和权重初始化等超参数会直接影响模型的稳定性。例如，当损失发散时，降低学习率并从较早的检查点重新开始可能会使作业恢复并继续训练。

Additionally, the bigger the model is, the more difficult it is to avoid loss spikes during training. These spikes are likely to occur at highly irregular intervals, sometimes late into training.

此外，模型越大，在训练过程中就越难避免损失峰值。这些峰值很可能会以极不规则的间隔出现，有时甚至会在训练后期出现。

There hasn't been a lot of systematic analysis of principled strategy to mitigate spikes. Here are some best practices we have seen from the industry to effectively get models to converge:

• **Batch size:** in general, using the biggest batch size that your GPU allows you to use is the best policy here.

• **Batch Normalization**: Normalizing the activations within a mini-batch can speed up convergence and improve model performance.

• **Learning Rate Scheduling:** A high learning rate can cause the loss to oscillate or diverge, leading to loss spikes. By scheduling the learning rate to decrease over time, you can gradually decrease the magnitude of updates to the model's parameters and improve stability. Common schedules include step decay, where the learning rate is decreased by a fixed amount after a fixed number of steps, and exponential decay, where the learning rate is decreased by a fixed factor each step. Note that it is not really possible to know ahead of time what LR to use, but you can use different LR schedules to see how your model responds. See more details here.

• **Weight Initialization**: Properly initializing the weights can help the model converge faster and improve performance. For example, it is common to use small Gaussian noise or, in the case of Transformers, the T-Fixup initialization. Techniques that can be used for weight initialization include random initialization, layer-wise initialization, and initialization using pre-trained weights.

• **Model training starting point:** Using a pre-trained model that is trained on related tasks as a starting point can help the model converge faster and improve

目前还没有对缓解峰值的原则性策略进行大量系统分析。以下是我们在业内看到的一些有效使模型趋同的最佳实践：

• **批次大小：**一般来说，在 GPU 允许的范围内使用最大的批次大小是上策。

• **批次归一化：**对迷你批次中的激活值进行归一化处理，可以加快收敛速度，提高模型性能。

• **学习率调度：**高学习率会导致损耗振荡或发散，造成损耗峰值。通过安排学习率随时间的推移而降低，可以逐步减少模型参数更新的幅度，提高稳定性。常见的计划包括阶跃衰减，即学习率为指数衰减是指学习率在固定步数后以固定系数递减。需要注意的是，要提前知道使用什么 LR 是不可能的，但您可以使用不同的 LR 计划来观察模型的反应。

• **权重初始化：**正确初始化权重可以帮助模型更快收敛并提高性能。例如，通常使用小的高斯噪声或者，就 Transformers 而言，T-Fixup 初始化。权重初始化可使用的技术包括随机初始化、分层初始化和使用预训练权重的初始化。

• **模型训练起点：**使用在相关任务上训练过的预训练模型作为起点，可以帮助模型更快地收敛并提高性能。

performance.

• **Regularization:** Regularization techniques, such as dropout, weight decay, and L1/L2 regularization, can help the model converge better by reducing overfitting and improving generalization.

• **正则化**：正则化技术（如滤除、权重衰减和 L1/L2 正则化）可以减少过拟合，提高泛化效果，从而帮助模型更好地收敛。

• **Data Augmentation:** Augmenting the training data by applying transformations can help the model generalize better and reduce overfitting.

• **数据增强**：通过应用变换来增强训练数据，可以帮助模型更好地泛化，减少过拟合。

• **Hot-swapping during training:** Hot-swapping of optimizers or activation functions are sometimes used during LLM training to fix issues as they appear during the process. It sometimes requires a team on it almost 24/7 trying various heuristics to train further.

• **训练期间的热插拔**：在 LLM 训练中，有时会使用优化器或激活函数的热插拔来修复在训练过程中出现的问题。它有时需要一个团队全天候的尝试各种启发式方法来进一步训练。

• **Other simple strategies mitigating the instability issue when encountered:** Restart training from a previous checkpoint; skip some data batches that were seen during the spike (the intuition is that pikes occur due to the combination of specific data batches with a particular model parameter state).

• **遇到不稳定问题时，还有其他简单的缓解策略**：从上一个检查点重新开始训练；跳过尖峰期间出现的某些数据批次（直觉认为尖峰的出现是由于特定数据批次与特定模型参数状态的结合）。

Note: most of the above model convergence best practices not only apply to transformer training, but also apply in a broader deep learning context across architectures and use cases.

注：上述大多数模型融合最佳实践不仅适用于 Transformers 训练，也适用于跨架构和场景的更广泛的深度学习环境。

Finally, after your LLM training is completed, it is very important to ensure that your model training environment is saved and retained in that final state. That way, if you need to re-do anything or replicate something in the future you can because you have the training state preserved.

最后，在完成 LLM 训练后，确保模型训练环境被保存并保留在最终状态是非常重要的。这样，如果您将来需要重新做任何事情或复制某些东西，您都可以做到，因为您保存了训练状态。

A team could also try some ablation studies. This allows you to see how pulling parts of the model out might impact performance. Ablation studies can allow you to massively reduce the size of your model, while still retaining most of a mod-

团队还可以尝试一些烧蚀研究。这允许您看到提取模型的某些部分可能会如何影响性能。烧蚀研究可以让你大幅缩小你的模型的大小，同时仍然保留一

el’s predictive power.

MODEL EVALUATION

Typically, pre-trained models are evaluated on a diverse language model datasets to assess their ability to perform logical reasoning, translation, natural language inference, question answering, and more.

Machine learning practitioners coalesced around a variety of standard evaluation benchmarks. A few popular examples include:

- Open-Domain Question Answering tasks: TriviaQA, Natural Questions, Web Questions.
- Cloze and Completion tasks: LAMBADA, HellaSwag, StoryCloze.
- Winograd-style tasks: Winifred, WinoGrande.
- Common Sense Reasoning: PIQA, ARC, OpenBookQA.
- In-context Reading Comprehension: DROP, CoQA , QuAC, SQuADv2, RACE, SuperGLUE.
- Natural Language Inference (NLI): SNLI, QNLI.
- Reasoning tasks: Arithmetic reasoning tasks.
- Code tasks: HumanEval, MBPP (text-to-code); TransCoder (code-to-code).
- Translation tasks: Translation BLEU score on WMT language pairs.

个模型的大部分预测能力。

模型评估

通常，预先训练好的模型会在不同的语言模型数据集上进行评估，以评估其执行逻辑推理、翻译、自然语言推理、问题解答等的能力。

机器学习从业者围绕各种标准评估基准展开讨论。一些流行的例子包括：

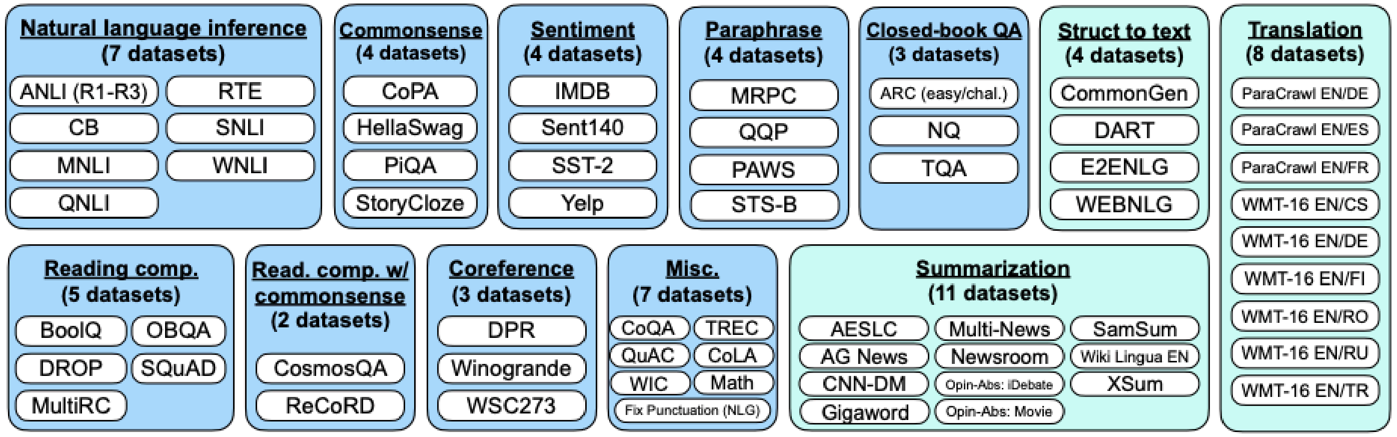
- 开放域问题解答任务：琐事问答、自然问题、网络问题
- 猜词和完成任务：LAMBADA、HellaSwag、StoryCloze。
- 维诺格拉德式任务：Winograd, WinoGrande。
- 常识推理：PIQA、ARC、OpenBookQA。
- 上下文阅读理解：DROP、CoQA、QuAC、SQuADv2、RACE、SuperGLUE。
- 自然语言推理 (NLI)：SNLI, QNLI。
- 推理任务：算术推理任务。
- 编码任务：HumanEval、MBPP（文本到代码）；TransCoder（代码到代码）。
- 翻译任务：WMT 语言对的翻译 BLEU 分数。

• BIG-bench: A collaborative benchmark aimed at producing challenging tasks for large language models, including 200+ tasks covering diverse textual tasks and programmatic tasks.

• BIG-bench：一个旨在为大型语言模型提供挑战性任务的协作基准，包括 200 多个任务，涵盖各种文本任务和程序任务。

• LM Evaluation Harness: A library for standardized evaluation of autoregressive LLMs across 200+ tasks released by EleutherAI. It has gained popularity because of its systematic framework approach and robustness.

• LM 评估工具包：用于对 200 多项任务发布的自回归 LLM 进行标准化评估的库，由 EleutherAI 提供。它因其系统的框架方法和鲁棒性而广受欢迎。



Here is a summary of typical language tasks (NLU tasks in blue NLG tasks in teal):

典型语言任务的总结（蓝色的 NLU 任务；蓝绿色的 NLG 任务）

Another evaluation step is n-shot learning. It’s a task-agnostic dimension, and refers to the number of supervised samples (demonstrations) you provide to the model right before asking it to perform a given task. N-shots are normally provided via a technique called prompting. You’ll often see n-shot bundled into the following three categories:

另一个评估步骤是 n—样本学习。这是一个与任务无关的维度，指的是在要求模型执行给定任务之前向其提供的监督样本（演示）数量。N 次样本通常是通过一种称为提示的技术来提供的。你经常会看到 n—样本被捆绑到以下三个类别：

• Zero-shot: refers to evaluation on any tasks without providing any supervised samples to the model at inference time.

• 零样本：指在推理时不对模型提供任何监督样本的任何任务进行评估。

• One-shot: similar to few-shot but with n=1, evaluation where one supervised sample is provided to the model at inference time.

• 单次样本：类似于少量样本，但使用 n=1，评估，在推理时提供一个监督样本给模型。

• Few-shot: refers to evaluation where a few supervised samples are provided to the model at inference time (e.g. 5 samples provided -> 5-shot).

• 少量样本：指在推理时向模型提供一些监督样本的评估（例如，提供的 5 个样本—> 5-shot）。

Evaluation typically involves both looking at benchmarking metrics of the above tasks and more manual evaluation by feeding the model with prompts and looking at completions for human assessment. Typically, both NLP Engineers and subject matter experts (SMEs) are involved in the evaluation process and assess the model performance from different angles:

评估通常包括查看上述任务的基准指标，以及通过向模型提供提示和查看完成情况进行人工评估。通常，NLP 工程师和领域专家 (SME) 都会参与评估过程，并从不同角度对模型性能进行评估：

NLP engineers: people with a background in NLP, computational linguistics, prompt engineering, etc., who can probe and assess the model's semantic and syntactic shortcomings and come up with model failure classes for continuous improvement. A failure class example would be "the LLM does not handle arithmetic with either integers (1, 2, 3, etc.) nor their spelled-out forms: one, two, three."

NLP 工程师：具有 NLP、计算语言学、提示工程等背景的人员，他们可以探究和评估模型的语义和句法缺陷，并提出模型失败类别，以便不断改进。一个失败类别的例子是："LLM 不能处理整数（1、2、3 等）的算术运算，也不能处理它们的拼写形式：1、2、3"。

Subject matter experts (SMEs): In contrast to the NLP engineers, the SMEs are asked to probe specific classes of LLM output, fixing errors where necessary, and "talking aloud" while doing so. The SMEs are required to explain in a step-by-step fashion the reasoning and logic behind their correct answer versus the incorrect machine-produced answer.

领域专家（SMEs）：与 NLP 工程师相比，领域专家被要求对特定类别的 LLM 输出进行调查，在必要时修复错误，并在此时知无不言。主题专家被要求以一步一步的方式解释他们的正确答案和机器产生的错误答案背后的推理和逻辑。

BIAS AND TOXICITY

There are potential risks associated with large-scale, general purpose language models trained on web text. Which is to say: humans have biases, those biases make their way into data, and models that learn from that data can inherit those biases. In addition to perpetuating or exacerbating social stereotypes, you want to ensure your LLM doesn't memorize and reveal private information.

It's essential to analyze and document such potential undesirable associations and risks through transparency artifacts such as model cards.

Similar to performance benchmarks, a set of community developed bias and toxicity benchmarks are available for assessing the potential harm of LLM models. Typical benchmarks include:

Hate speech detection: The ETHOS dataset can help measure the ability of LLM models to identify whether or not certain English statements are racist or sexist.

Social bias detection: CrowSPairs is a benchmark aiming to measure intrasentence level biases in 9 categories: gender, religion, race/color, sexual orientation, age, nationality, disability, physical appearance, and socioeconomic status; StereoSet benchmark measure stereotypical bias across 4.

Toxic language response: The RealToxicityPrompts dataset helps evaluate if and how models use toxic language.

Dialog safety evaluations: The SaferDialogues benchmark measures how unsafe

偏差和毒性

在网络文本上训练的大规模通用语言模型存在潜在风险。也就是说：人类有偏差，这些偏差会进入数据，而从数据中学习的模型会继承这些偏差。除了会延续或加剧社会成见外，您还希望确保您的 LLM 不会记忆和泄露私人信息。

有必要通过模型卡等透明度工具来分析和记录这些潜在的不良关联和风险。

与性能基准类似，一套共同开发的偏差和毒性基准可用于评估 LLM 模型的潜在危害。典型基准包括：

仇恨言论检测：ETHOS 数据集有助于衡量 LLM 模型识别某些英语言论是否具有种族主义或性别歧视的能力

社会偏见检测：CrowSPairs 是一个基准，旨在测量 9 个类别的句子内部偏见：性别、宗教、种族/肤色、性取向、年龄、国籍、残疾、外貌和社会经济地位；StereoSet 基准测量 4 个类别的刻板偏见：性别、宗教、种族/肤色、性取向、年龄、国籍、残疾、外貌和社会经济地位。

有毒语言反应：RealToxicityPrompts 数据集有助于评估模型是否以及如何使用有毒语言。

对话安全评估：SaferDialogues 基准衡量了模型响应的不安全程度，分为四

a model’s response is, stratified across four levels: safe, realistic, unsafe, and adversarial.

个等级：安全、现实、不安全和对抗。

To date, most analysis on existing pre-trained models indicate that internet-trained models have internet-scale biases. In addition, pre-trained models generally have a high propensity to generate toxic language, even when provided with a relatively innocuous prompt, and adversarial prompts are trivial to find.

迄今为止，对现有预训练模型的大多数分析表明，互联网训练模型存在互联网规模的偏差。此外，预先训练的模型通常很容易产生有毒语言，即使是相对无害的提示语，而对抗性提示语也很容易找到。

Bias and Toxicity Mitigation

减少偏差和毒性

So how do we fix this? Here are a few ways to mitigate biases during and after the pre-training process:

那么，我们该如何解决这个问题呢？以下是一些在预训练过程中和之后减少偏差的方法：

Training set filtering: Here, you want to analyze the elements of your training dataset that show evidence of bias and simply remove them from the training data.

训练集过滤：在此，您要分析训练数据集中显示出以下证据的元素偏差，只需将它们从训练数据中删除即可。

Training set modification: This technique doesn’t filter your training data but instead modifies it to reduce bias. This could involve changing certain gendered words (from policeman to policewoman or police officer, for example) to help mitigate bias.

修改训练集：这种技术不会过滤训练数据，而是修改数据以减少偏差。这可能涉及更改某些性别词（例如，从警察改为女警或警官），以帮助减少偏差。

Additionally, you can mitigate bias after pre-training as well:

此外，您还可以在预训练后减少偏差：

Prompt engineering: The inputs to the model for each query are modified to steer the model away from bias (more on this later).

提示工程：对每个查询的模型输入进行修改，以避免模型出现偏差（稍后详述）。

Fine-tuning: Take a trained model and retrain it to unlearn biased tendencies.

微调：对训练有素的模型进行再训练，以消除偏差倾向。

Output steering: Adding a filtering step to the inference procedure to re-weigh

输出转向：在推理过程中添加一个过滤步骤来重新权衡输出值，并引导输出

output values and steer the output away from biased responses.

远离有偏倚的响应。

INSTRUCTION TUNING

指令调优

At this point, let’s assume we have a pre-trained, general purpose LLM. If we did our job well, our model can already be used for domain-specific tasks without tuning for few-shot learning and zero-shot learning scenarios. That said, zero-shot learning is in general much worse than its few-shot counterpart in plenty of tasks like reading comprehension, question answering, and natural language inference. One potential reason is that, without few-shot examples, it’s harder for models to perform well on prompts that are not similar to the format of the pre-training data.

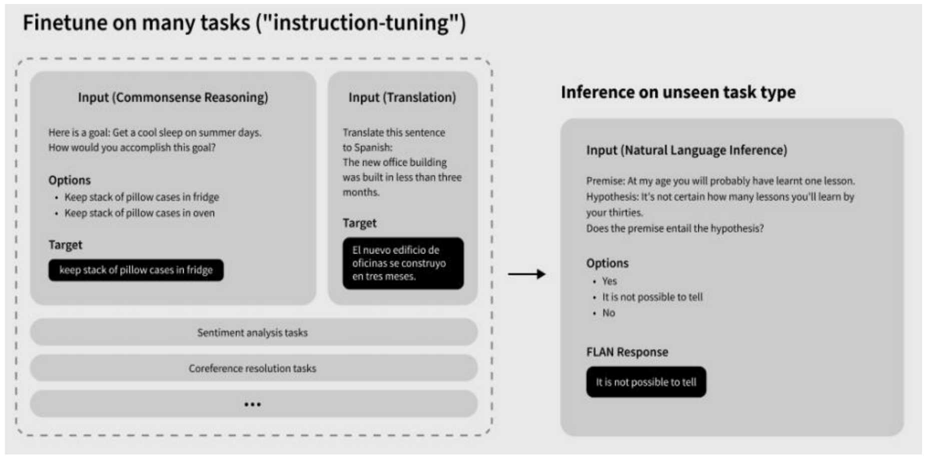
在这一点上，让我们假设我们有一个预先训练好的通用 LLM。如果我们做得很好，我们的模型已经可以用于特定领域的任务，而无需针对少样本训练和零样本训练场景进行调整。尽管如此，在阅读理解、问题解答和自然语言推理等大量任务中，零样本学习的效果总体上要比少样本学习差得多。其中一个潜在的原因是，如果没有少量实例，模型就很难在与预训练数据格式有差异的提示上表现出色。

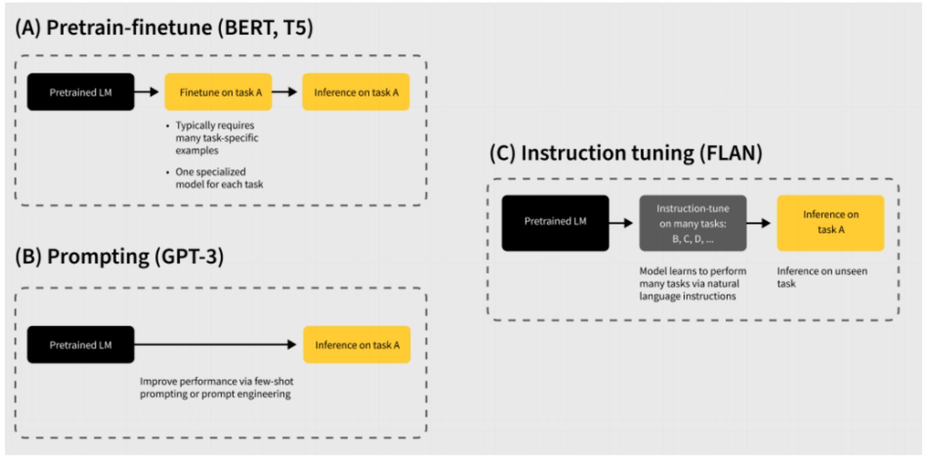
To solve this issue, we can use instruction tuning. Instruction tuning is a state-of-the-art fine-tuning technique that fine-tunes pre-trained LLMs on a collection of tasks phrased as instructions. It enables pre-trained LLMs to respond better to instructions and reduces the need for few-shot examples at prompting stage (i.e. drastically improves zero-shot performance).

为了解决这个问题，我们可以使用指令调优技术。指令调优是一种最先进的微调技术，它可以在一组以指令形式表述的任务上对预先训练好的 LLM 进行微调。它能使预先训练好的 LLM 对指令做出更好的响应，并减少提示阶段对少量示例的需求（即大幅提高零样本性能）。

Instruction tuning has gained huge popularity in 2022, given that the technique considerably improves model performance without hurting its ability to generalize. Typically, a pre-trained LLM is tuned on a set of language tasks and evaluated on its ability to perform another set of language tasks unseen at tuning time, proving its generalizability and zero-shot capability. See illustration below:

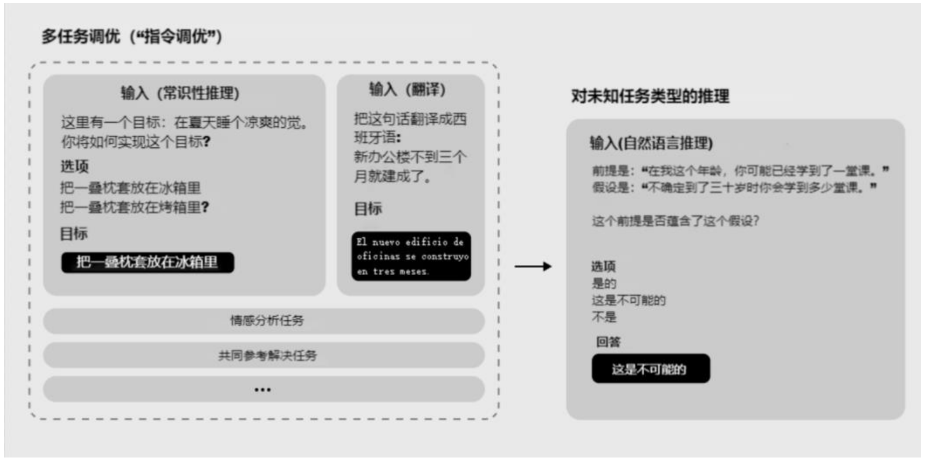
2022 年，指令调优技术大受欢迎，因为该技术能在不损害模型泛化能力的情况下显著提高模型性能。通常情况下，预训练好的 LLM 会在一组语言任务上进行调整，并根据其在微调时执行另一套未见过的语言任务的能力，证明了其通用性和零误差能力。参见下图：

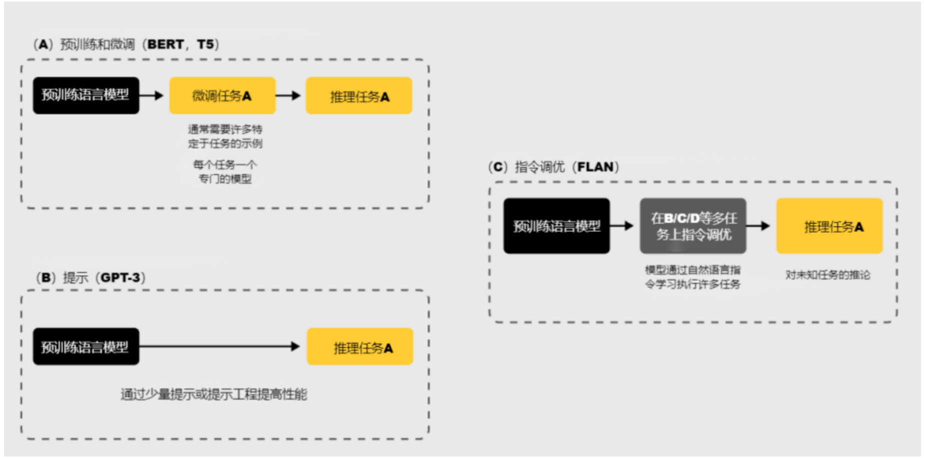

Finetune on many tasks ("instruction-tuning")
Input (Commonsense Reasoning)
Here is a goal: Get a cool sleep on summer days. How would you accomplish this goal?
Options
Keep stack of pillow cases in fridge
Keep stack of pillow cases in oven
Target
keep stack of pillow cases in fridge
Input (Translation)
Translate this sentence to Spanish:
The new office building was built in less than three months.
Target
El nuevo edificio de oficinas se construyo en tres meses.
Sentiment analysis tasks
Coreference resolution tasks
...
Inference on unseen task type
Input (Natural Language Inference)
Premise: At my age you will probably have learnt one lesson.
Hypothesis: It's not certain how many lessons you'll learn by your thirties.
Does the premise entail the hypothesis?
Options
Yes
It is not possible to tell
No
FLAN Response
It is not possible to tell


(A) Pretrain-finetune (BERT, T5)
Pretrained LM
Finetune on task A
Inference on task A
Typically requires many task-specific examples
One specialized model for each task
(B) Prompting (GPT-3)
Pretrained LM
Inference on task A
Improve performance via few-shot prompting or prompt engineering
(C) Instruction tuning (FLAN)
Pretrained LM
Instruction-tune on many tasks: B, C, D, ...
Inference on task A
Model learns to perform many tasks via natural language instructions
Inference on unseen task

A few things to keep in mind about instruction tuning:

• Instruction tuning tunes full model parameters as opposed to freezing a part of them in parameter-efficient fine tuning. That means it doesn't bring with it the


多任务调优（“指令调优”）
输入（常识性推理）
这里有一个目标：在夏天睡个凉爽的觉。你将如何实现这个目标?
选项
把一叠枕套放在冰箱里
把一叠枕套放在烤箱里?
目标
把一叠枕套放在冰箱里
输入（翻译）
把这句话翻译成西班牙语:
新办公楼不到三个月就建成了。
目标
El nuevo edificio de oficinas se construyo en tres meses.
情感分析任务
共同参考解决任务
...
对未知任务类型的推理
输入(自然语言推理)
前提是：“在我这个年龄，你可能已经学到了一堂课。”
假设是：“不确定到了三十岁时你会学到多少堂课。”
这个前提是否蕴含了这个假设?
选项
是的
这是不可能的
不是
回答
这是不可能的



关于指令调优，有几点需要注意：

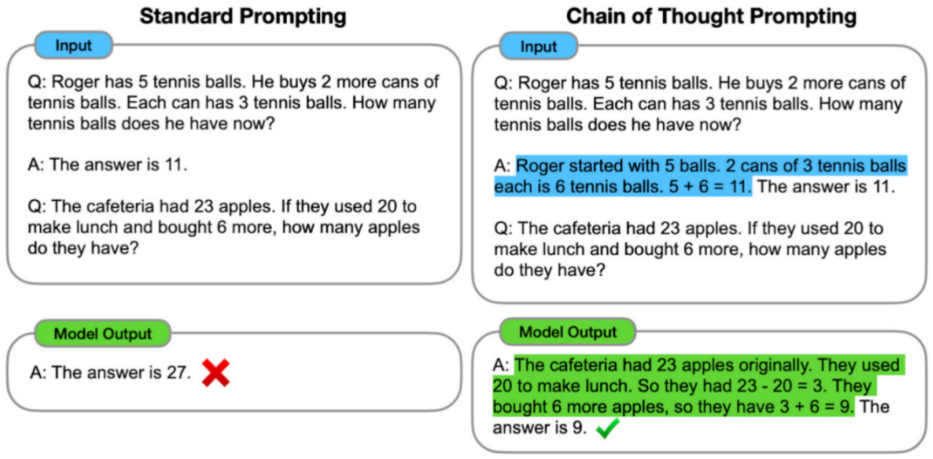
• 指令调优是对全部模型参数进行调整，而不是在参数效率微调中冻结部分参数。这意味着它不会有参数效率微调所带来的成本优势。然而，与参数效

cost benefits that come with parameter-efficient fine tuning. However, given that instruction tuning produces much more generalizable models compared to parameter-efficient fine tuning, instruction-tuned models can still serve as a general- purpose model serving multiple downstream tasks. It often comes down to whether you have the instruction dataset available and training budget to perform instruction tuning.

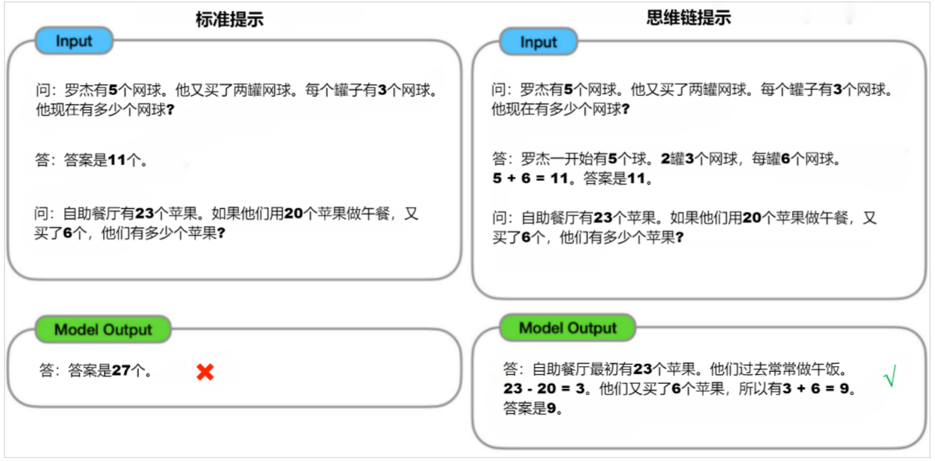
率微调相比，指令微优产生的模型更具通用性，因此指令微调模型仍可作为通用模型服务于多个下游任务。这通常取决于你是否有可用的指令数据集和训练预算来进行指令调优。

• Instruction tuning is universally effective on tasks naturally verbalized as instructions (e.g., NLI, QA, translation), but it is a little trickier for tasks like reasoning. To improve for these tasks, you'll want to include chain-of-thought examples during tuning.

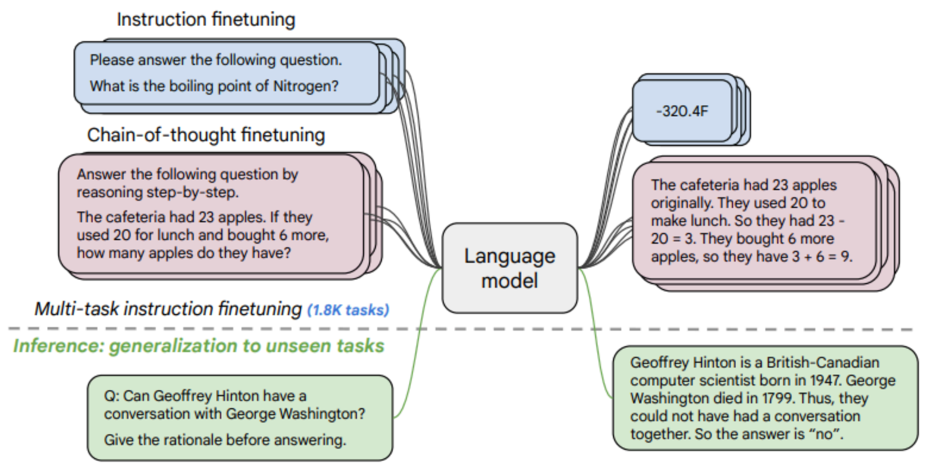
• 指令调优对自然口头化为指令的任务（如 NLI、QA、翻译）普遍有效，但对推理等任务则比较棘手。要改进这些任务，您需要在调整过程中加入思维链示例。



Standard Prompting
Input
Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: The answer is 11.
Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?
Model Output
A: The answer is 27. ✗
Chain of Thought Prompting
Input
Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.
Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?
Model Output
A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✓



标准提示
Input
问：罗杰有5个网球。他又买了两罐网球。每个罐子有3个网球。他现在有多少个网球?
答：答案是11个。
问：自助餐厅有23个苹果。如果他们用20个苹果做午餐，又买了6个，他们有多少个苹果?
Model Output
答：答案是27个。 ✗
思维链提示
Input
问：罗杰有5个网球。他又买了两罐网球。每个罐子有3个网球。他现在有多少个网球?
答：罗杰一开始有5个球。2罐3个网球，每罐6个网球。5 + 6 = 11。答案是11。
问：自助餐厅有23个苹果。如果他们用20个苹果做午餐，又买了6个，他们有多少个苹果?
Model Output
答：自助餐厅最初有23个苹果。他们过去常常做午饭。23 - 20 = 3。他们又买了6个苹果，所以有3 + 6 = 9。答案是9。 √


Instruction finetuning

Please answer the following question.
What is the boiling point of Nitrogen?

-320.4F

Chain-of-thought finetuning

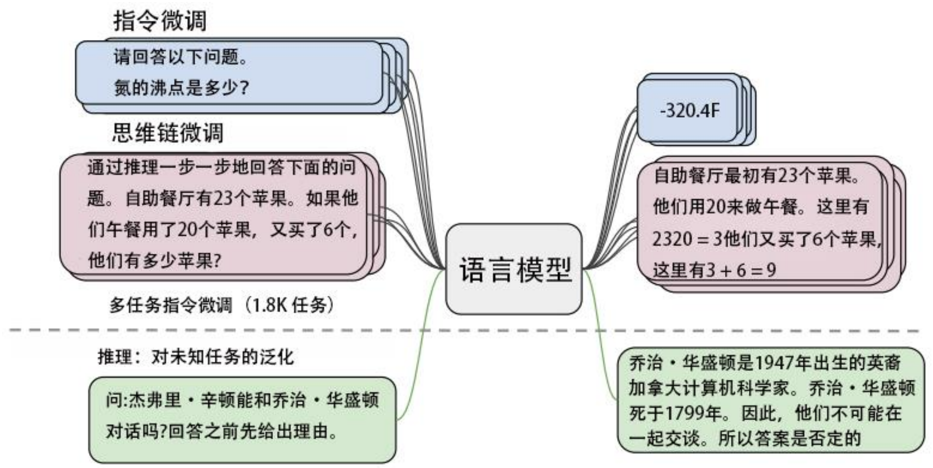
Answer the following question by reasoning step-by-step.
The cafeteria had 23 apples. If they used 20 for lunch and bought 6 more, how many apples do they have?

The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9.

Language model
Multi-task instruction finetuning *(1.8K tasks)*
Inference: generalization to unseen tasks

Q: Can Geoffrey Hinton have a conversation with George Washington?
Give the rationale before answering.

Geoffrey Hinton is a British-Canadian computer scientist born in 1947. George Washington died in 1799. Thus, they could not have had a conversation together. So the answer is "no".


指令微调

请回答以下问题。
氮的沸点是多少?

-320.4F

思维链微调

通过推理一步一步地回答下面的问题。自助餐厅有23个苹果。如果他们午餐用了20个苹果，又买了6个，他们有多少苹果?

自助餐厅最初有23个苹果。他们用20来做午餐。这里有2320＝3他们又买了6个苹果，这里有3＋6＝9

问:杰弗里・辛顿能和乔治・华盛顿对话吗?回答之前先给出理由。

乔治・华盛顿是1947年出生的英裔加拿大计算机科学家。乔治・华盛顿死于1799年。因此，他们不可能在一起交谈。所以答案是否定的

REINFORCEMENT LEARNING THROUGH HUMAN FEEDBACK (RLHF)

RLHF is an extension of instruction tuning, with more steps added after the instruction tuning step to further incorporate human feedback.

As discussed above, pre-trained LLMs often express unintended behaviors such as making up facts, generating biased or toxic responses, or simply not following user instructions. This is because the objective for many recent large LMs – i.e. predicting the next token on a webpage from the internet – is rather different from the objective "follow the user's instructions safely."

RLHF behaves how its name suggests it would. Here, we incorporate human feedback about a model's outputs given certain prompts. Those opinions about whether the quality of the outputs are then used as additional data points to im-

通过人类反馈强化学习 (RLHF)

RLHF 是指令调优的扩展，在指令调优步骤之后添加了更多的步骤，以进一步合并人工反馈。

如上所述，预训练的 LLM 常常会表现出非故意的行为，如捏造事实、产生有偏差或有毒的回应，或者干脆不遵循用户指令。这是因为最近许多大型 LLM 的目标——即从互联网上预测网页上的下一个标记——与"安全遵循用户指令"的目标大相径庭。

RLHF 的行为如其名称所示。在这里，我们将人类对模型输出结果的反馈意见纳入到特定的提示中。这些关于输出质量的意见将作为额外的数据点来改进模型的整体性能。

prove the model’s overall performance.

OpenAI has had some recent success here with InstructGPT. It’s essentially a pre-trained model GPT-3, fine-tuned with RLHF. In fact, their recent ChatGPT model also leverages RLHF on a more advanced GPT model series (referred to as GPT-3.5).

OpenAI 最近在 InstructGPT 上取得了一些成功。它本质上是一个预训练过的 GPT-3 模型，用 RLHF 进行了微调。事实上，他们最近的 ChatGPT 模型也利用了一个更高级的 GPT 模型系列（称为 GPT-3.5）上的 RLHF。

More granularly, RLHF generally works like this:

更具体的是，RLHF 通常的工作原理是这样的：

• Step 1: Instruction tuning – just collect a dataset of labeler demonstrations of the desired model behavior, and use them to fine-tune the pre-trained LLM using supervised learning.

• 步骤 1：指令调优——只需收集标注者演示所需的模型行为的数据集，然后利用这些数据集通过监督学习对预训练的 LLM 进行微调。

• Step 2: Collect a dataset of comparisons between model outputs, where labelers indicate which output they prefer for a given input. Then, train a reward model to predict the human-preferred output.

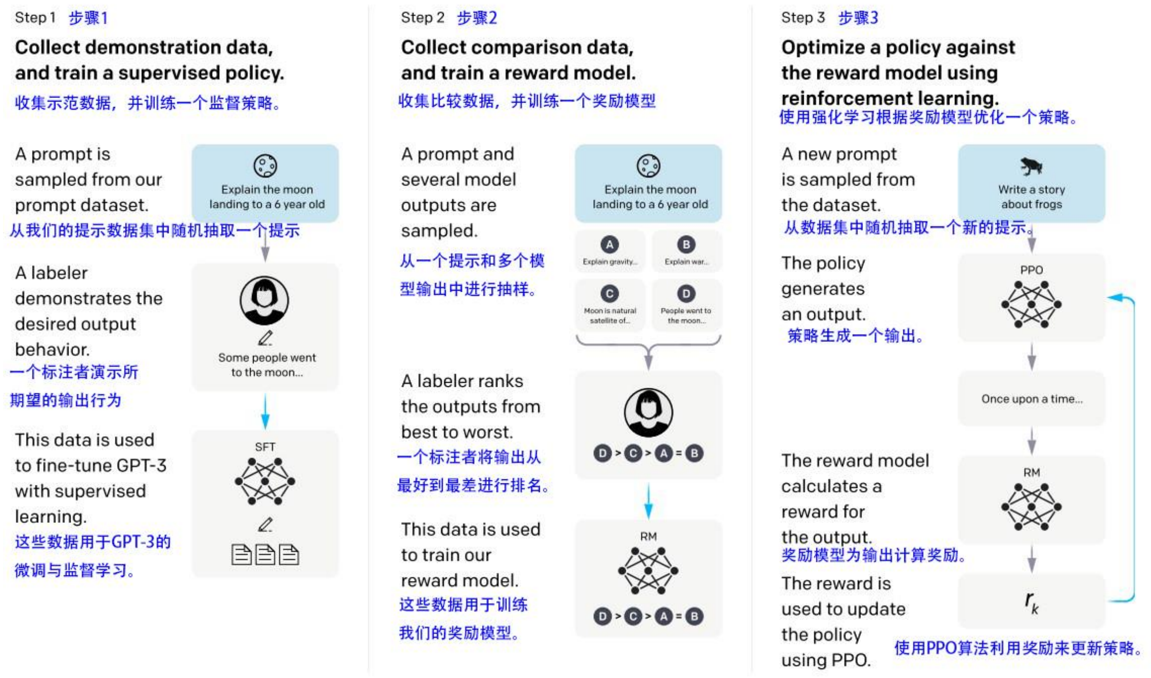
• 步骤 2：收集模型输出比较的数据集，在该数据集中，标注者会指出对于给定的输入，他们更喜欢哪种输出。然后，训练奖励模型来预测人类偏好的输出。

• Step 3: Take the trained reward model and optimize a policy against the reward mode using reinforcement learning.

• 步骤 3：采用训练好的奖励模型，利用强化学习优化针对奖励模式的策略。

Steps 2 and 3 can be iterated continuously. More comparison data is collected on the current best policy, which is used to train a new reward model and then a new policy. See below for RLHF process demonstration.

步骤 2 和步骤 3 可以不断重复。收集更多关于当前最佳结果的比较数据，用于训练新的奖励模型，然后再训练新的政策。请看下面的 RLHF 流程演示。


Step 1 步骤1
Collect demonstration data, and train a supervised policy.
收集示范数据，并训练一个监督策略。
A prompt is sampled from our prompt dataset.
从我们的提示数据集中随机抽取一个提示
Explain the moon landing to a 6 year old
A labeler demonstrates the desired output behavior.
一个标注者演示所期望的输出行为
Some people went to the moon...
This data is used to fine-tune GPT-3 with supervised learning.
这些数据用于GPT-3的微调与监督学习。
SFT
Step 2 步骤2
Collect comparison data, and train a reward model.
收集比较数据，并训练一个奖励模型
A prompt and several model outputs are sampled.
从一个提示和多个模型输出中进行抽样。
Explain the moon landing to a 6 year old
A Explain gravity...
B Explain war...
C Moon is natural satellite of...
D People went to the moon...
A labeler ranks the outputs from best to worst.
一个标注者将输出从最好到最差进行排名。
D > C > A = B
This data is used to train our reward model.
这些数据用于训练我们的奖励模型。
RM
D > C > A = B
Step 3 步骤3
Optimize a policy against the reward model using reinforcement learning.
使用强化学习根据奖励模型优化一个策略。
A new prompt is sampled from the dataset.
从数据集中随机抽取一个新的提示。
Write a story about frogs
The policy generates an output.
策略生成一个输出。
PPO
Once upon a time...
The reward model calculates a reward for the output.
奖励模型为输出计算奖励。
RM
The reward is used to update the policy using PPO.
r_k
使用PPO算法利用奖励来更新策略。

Adiagram illustrating the three steps of our method:(1)supervised fi-ne-tuning(SFT).(2) reward model(RM) training, and(3) reinforcement leaming via pradimal policy optimization(PPO) on this reward model, Training language models to follow instructions. with human feedback.

图说明了我们的三个步骤：(1)监督微调（SFT），(2)奖励模型（RM）训练，(3)通过该奖励模型上的近端策略优化（PPO）强化学习，训练语言模型遵循人类反馈的指令。

To date, RLHF has shown very promising results with InstructGPT and ChatGPT, bringing improvements in truthfulness and reductions in toxic output generation while having minimal performance regressions compared to the pre-trained GPT.

迄今为止，RLHF 在 InstructGPT 和 ChatGPT 上取得了非常令人鼓舞的成果，与预训练的 GPT 相比，RLHF 提高了真实性，减少了有毒输出的生成，同时性能退步极小。

Note that the RLHF procedure does come with the cost of slightly lower model performance in some downstream tasks - referred to as the alignment tax. Com-

需要注意的是，RLHF 程序的代价是模型在某些下游任务中的性能略有降低，这被称为 "对齐税" 。像 Scale AI、Labelbox.Net 这样的公司都在使用

panies like Scale AI, Labelbox.

Surge, and Label Studio offer RLHF as a service so you don't have to handle this yourself if you're interested in going down this path. But research has shown promising results using RLHF techniques to minimize the alignment cost to increase its adoption, so it's absolutely worth considering.

RLHF。

Surge 和 Label Studio 将 RLHF 作为一项服务提供，因此，如果您有兴趣走这条路，就不必亲自处理。但研究表明，使用 RLHF 技术最大限度地降低对齐成本以提高其采用率的结果很有希望，因此绝对值得考虑。

Conclusion

结论

Whether it's OpenAI, Cohere, or open-source projects like EleutherAI, cutting-edge large language models are built on Weight & Biases. Our platform enables collaboration across teams performing the complex, expensive work required to train and push these models to production, logging key metrics, versioning datasets, enabling knowledge sharing, sweeping through hyperparameters, and a whole lot more. LLM training is complex and nuanced and having a shared source of truth throughout the lifecycle of a model is vital for avoiding common pitfalls and understanding performance every step of the way.

无论是 OpenAI、Cohere 还是 EleutherAI 等开源项目，最前沿的大型语言模型都是在 Weight & Biases 上构建的。我们的平台可以让各团队协作执行复杂而昂贵的工作，这些工作包括训练这些模型并将其推向生产、记录关键指标、数据集版本化、知识共享、超参数扫描等。LLM 训练在模型的整个生命周期中，拥有一个共享的真相来源对于避免常见陷阱和了解每一步的性能都至关重要。

We'd like to also send a hearty thanks to OpenAI, Deepmind, Meta, and Google Brain. We referenced their research and breakthroughs frequently in this white paper and their contributions to the space are already invaluable. If you're interested in learning more about how W&B can help, please reach out and we'll schedule some time. And if you have any feedback we'd love to hear those too.

我们还要向 OpenAI、Deepmind、Meta 和 Google Brain 表示衷心的感谢。我们在本白皮书中经常提到他们的研究和突破，他们对该领域的贡献已经非常宝贵。如果您有兴趣进一步了解 W&B 可以提供哪些帮助，请联系我们，我们会安排时间。如果您有任何反馈意见，我们也非常乐意听取。

请关注UTC微信公众号

咨询单位：W&B 、UTC

UTC NETWORK PTE LTD

www.utc.group