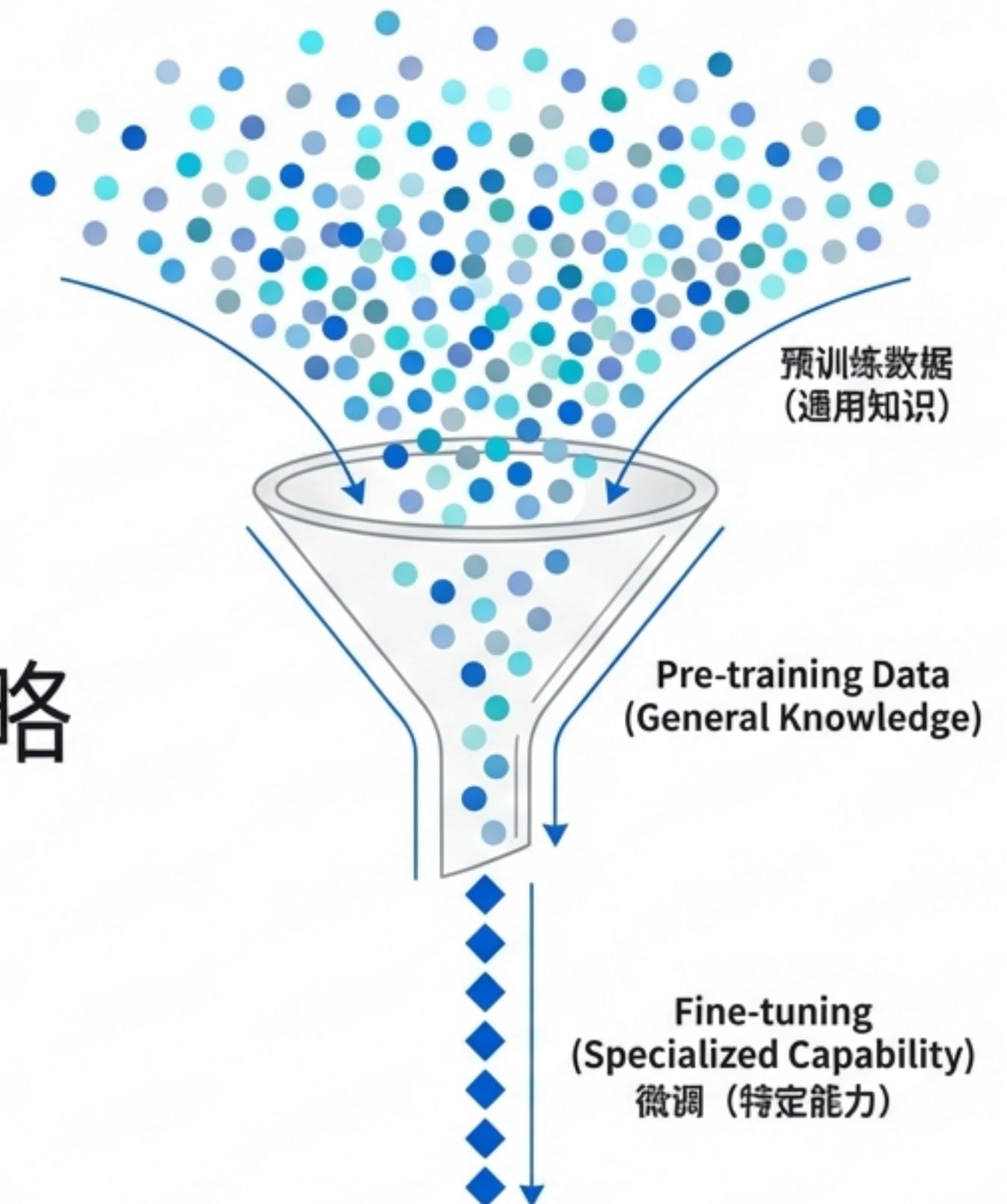


第6部分：总结与拓展

从分类微调实战到前沿训练策略

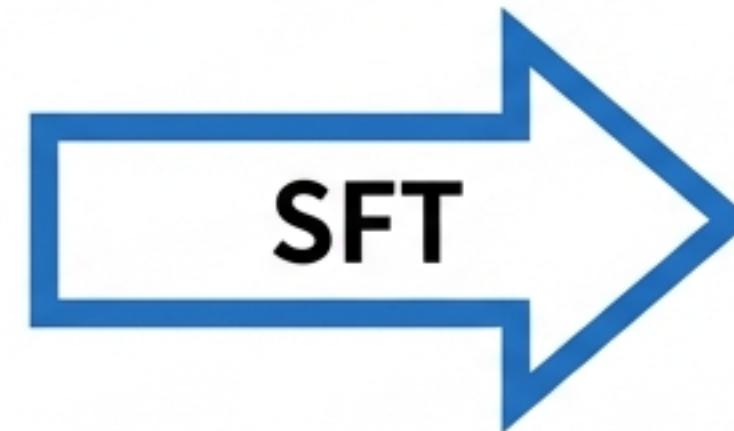


什么是监督微调 (SFT)?

预训练 (Pre-training)



监督微调 (Fine-tuning)



- 无监督学习 (Unsupervised)
- 预测下一个 Token
- 获得通用语言能力

- 有监督学习 (Supervised)
- 学习特定输入-输出对
- 获得特定任务能力

为什么需要 SFT?为什么需要 SFT?

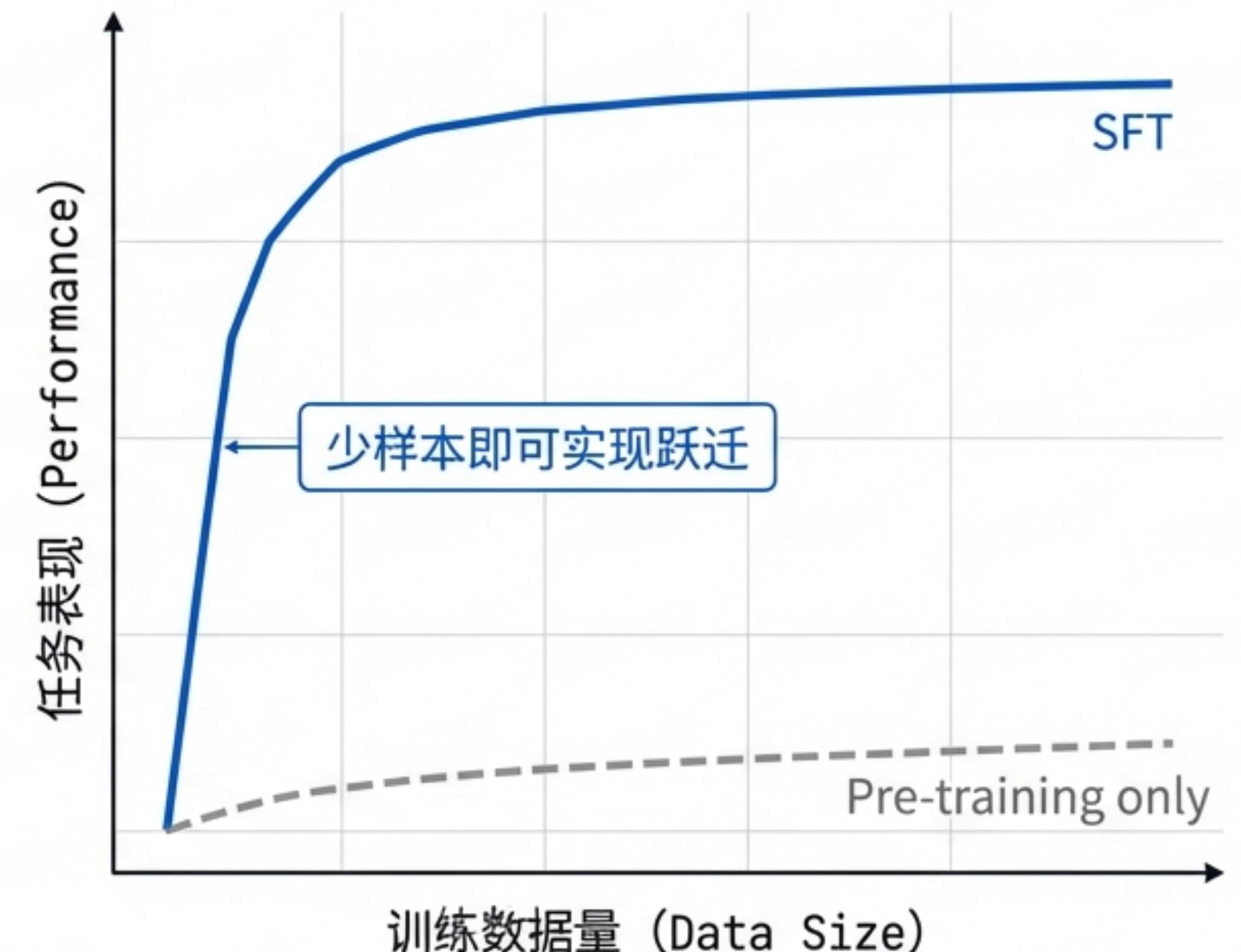
JetBrains Mono 弥补通用模型在特定领域的不足

通用模型的局限:

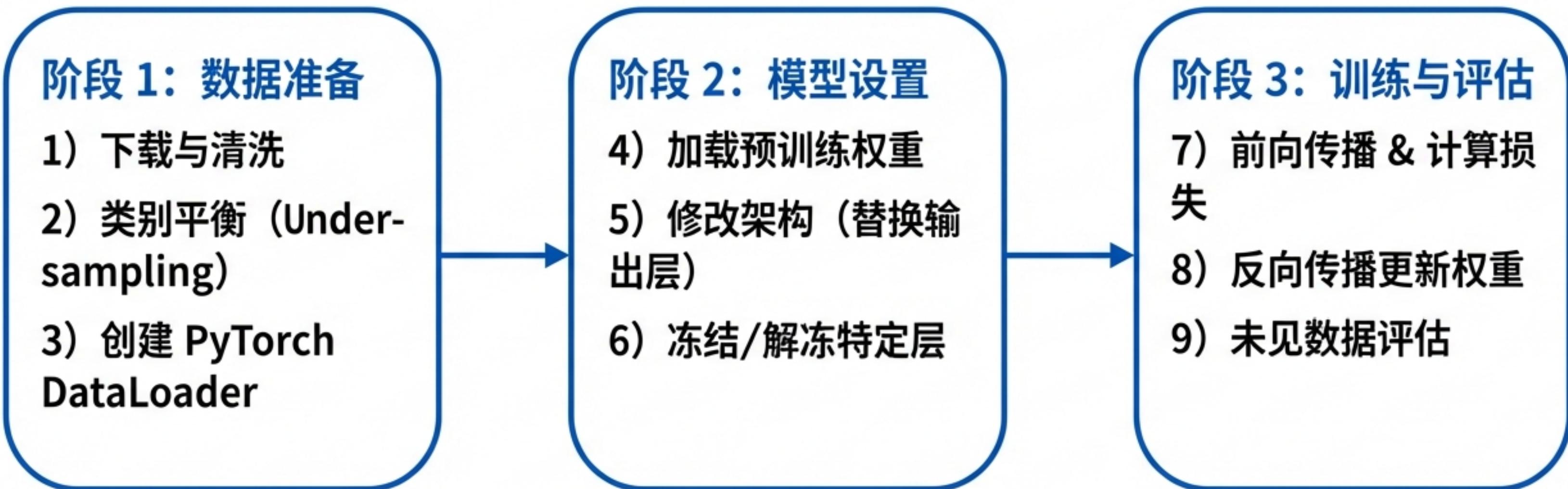
- 虽然博学，但不懂特定领域的术语、格式或指令。

SFT 的优势:

1. 效率 (Efficiency) : 仅需少量样本 (几百到几千条) 即可显著提升性能。
2. 定制化 (Customization) : 适用于情感分析、垃圾邮件检测、医疗问答等。

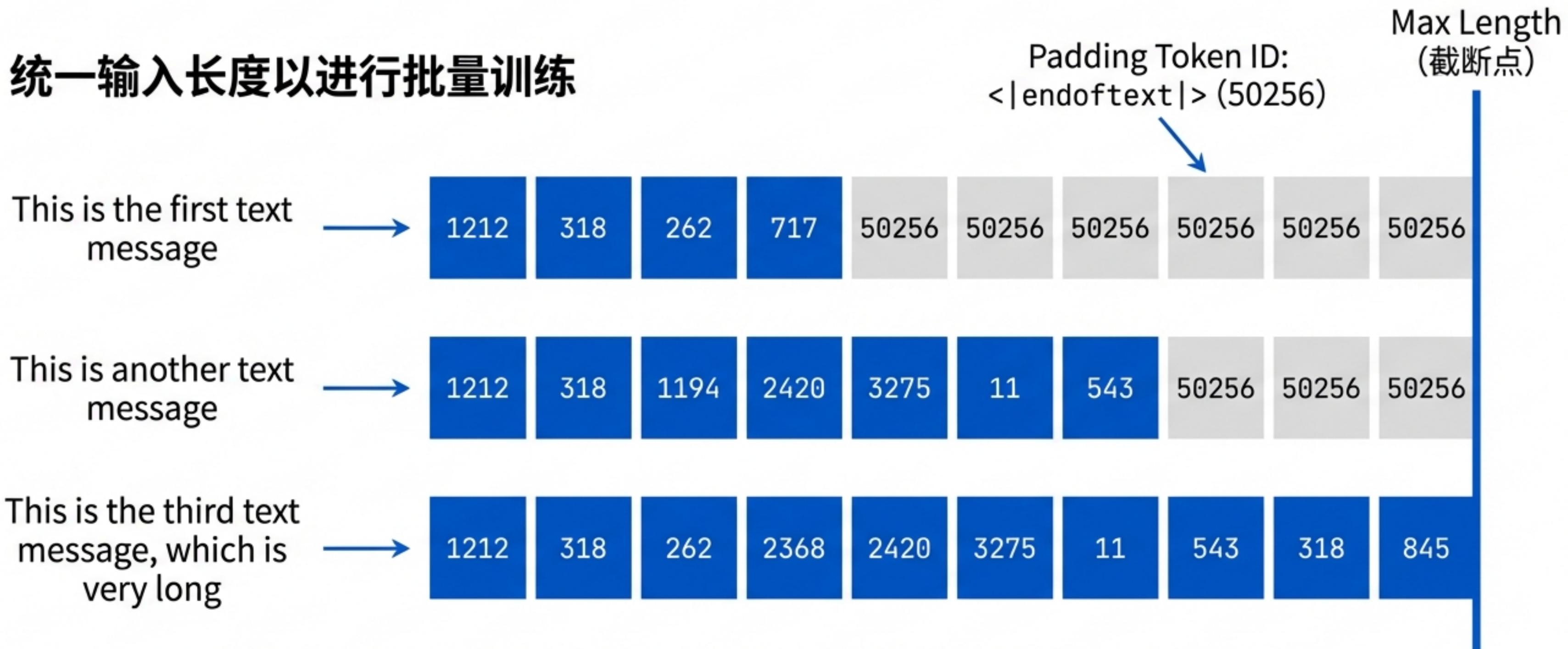


微调全流程回顾



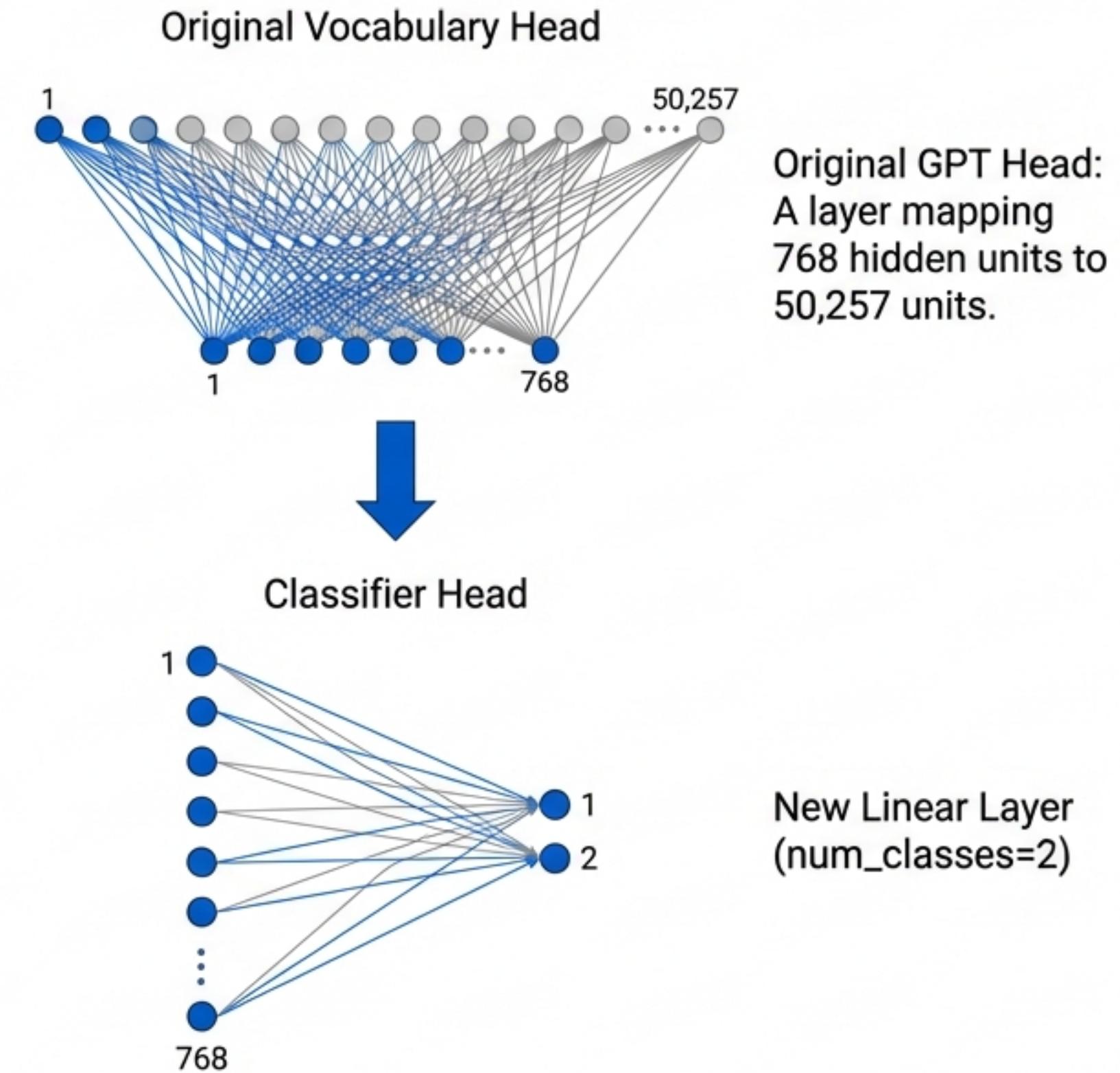
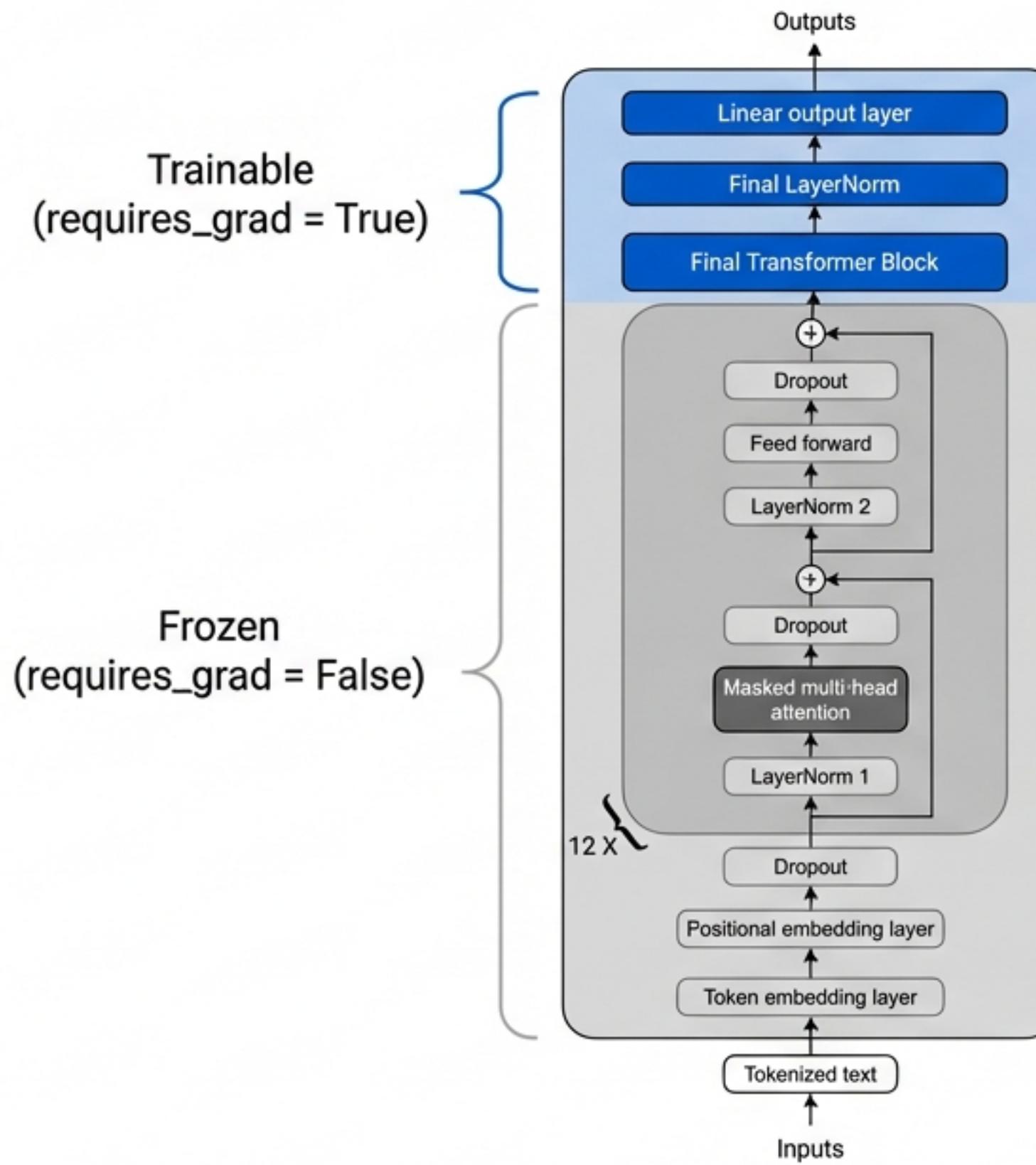
关键技术点 1：数据填充（Padding）

统一输入长度以进行批量训练



注意：在计算 Attention 时，需使用 Attention Mask 忽略填充位置。

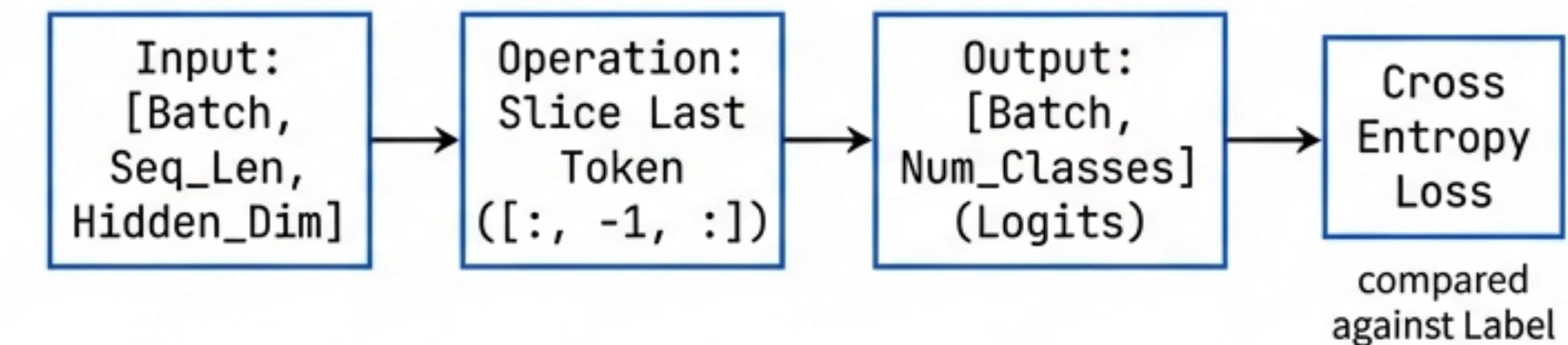
关键技术点 2：架构修改



关键技术点 3：预测与损失计算

为什么只关注最后一个 Token?

	Do	you	have	time
Do	1.0			0.27
you	0.55	1.0		0.24
have	0.45	0.49	1.0	0.24
time	0.58	0.32	0.25	0.25



因果注意力 (Causal Attention)：最后一个 Token 聚合了全序列的信息。

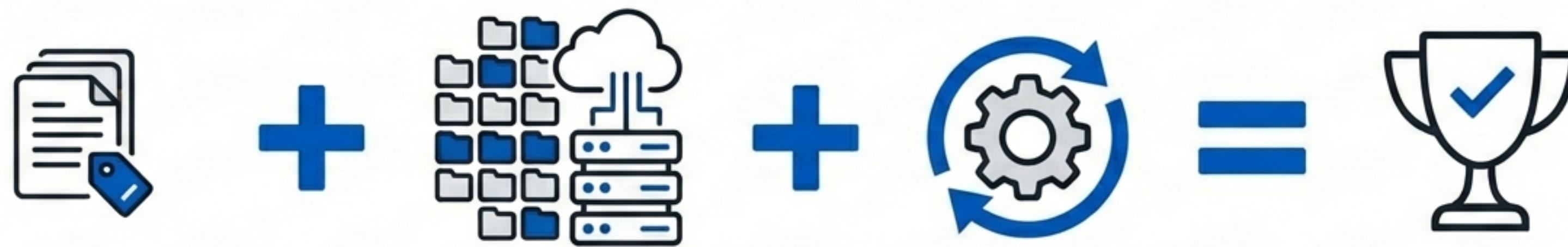
预测逻辑：Label = argmax(logits)

拓展：微调训练策略对比

全参数微调 (Full Parameter)	部分参数微调 (Partial/Frozen)	参数高效微调 (PEFT / LoRA)
		
<ul style="list-style-type: none">机制：更新所有权重优点：性能上限高缺点：显存消耗极大，易灾难性遗忘	<ul style="list-style-type: none">机制：冻结大部分层（本章方法）优点：速度快，显存少缺点：仅适合简单任务	<ul style="list-style-type: none">机制：冻结原模型，仅训练低秩适配器优点：极高参数效率(<1%)，工业界主流 <p>✓ 状态：  Recommended</p>

挑战 1：缺少标签怎么办？

Few-shot 场景下的解决方案

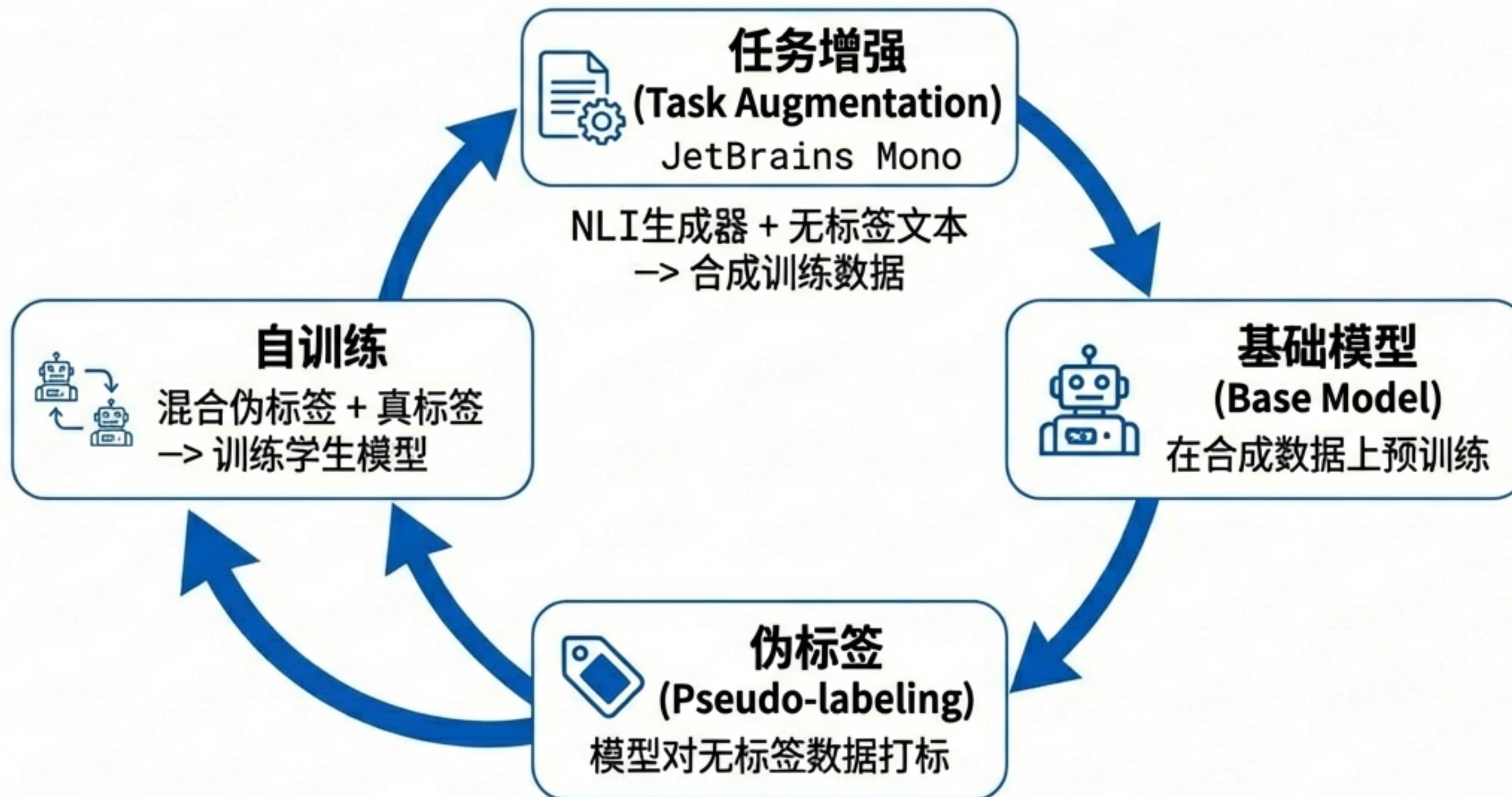


引入 STraTA: Self-Training with Task Augmentation

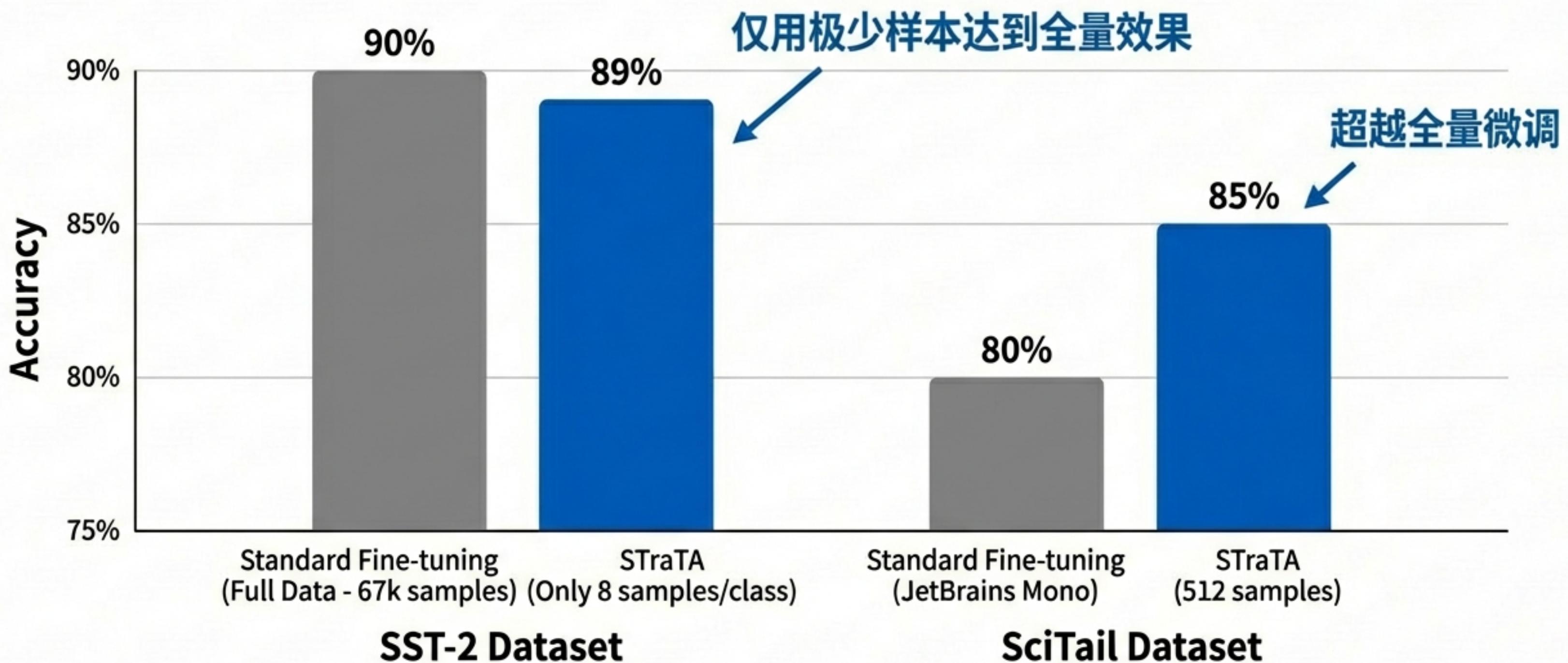
核心思路：利用无标签数据“无中生有”地创造训练信号。

来源：EMNLP 2021 Paper

STraTA 详解：任务增强与自训练



STrTA 的效果验证

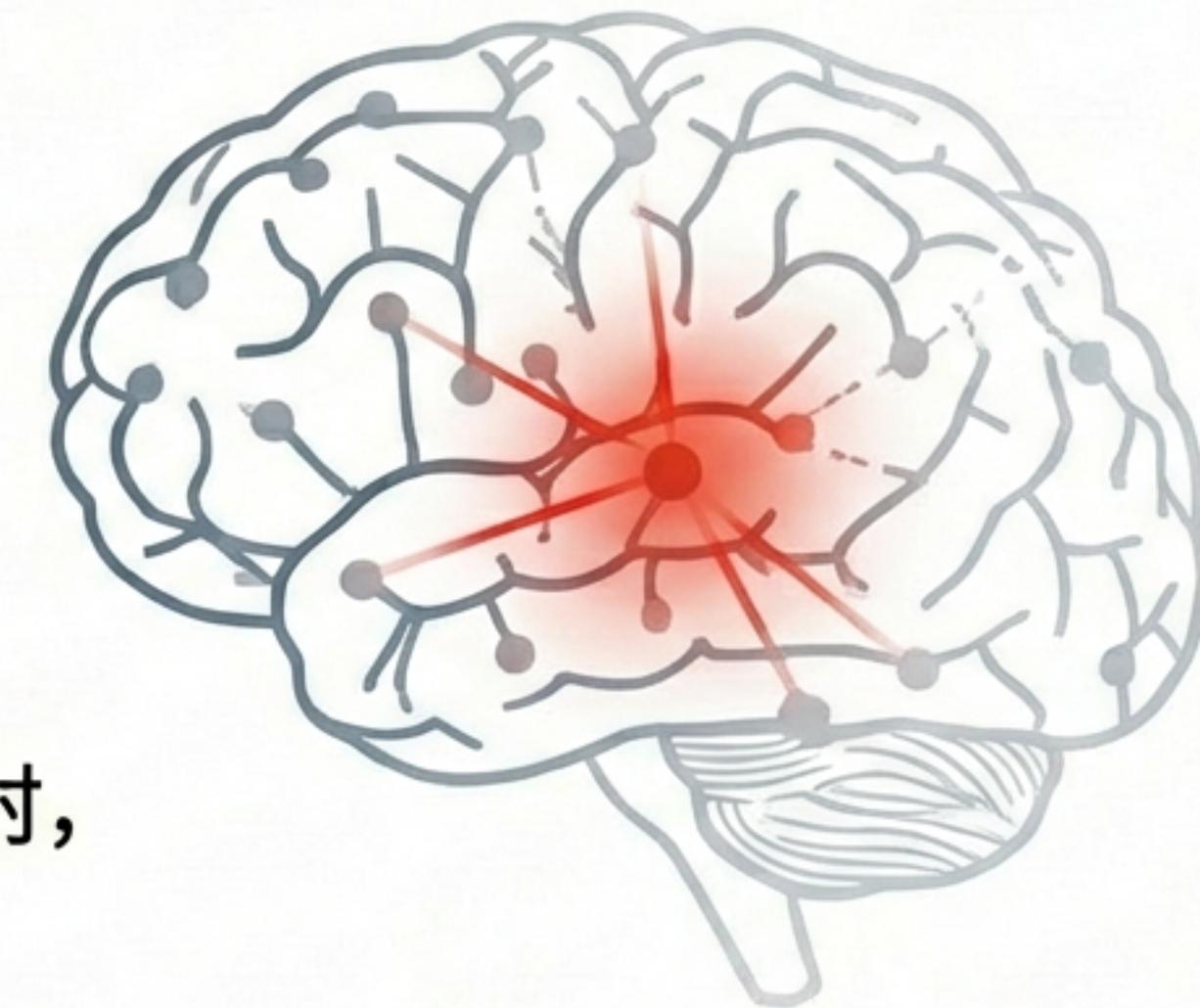


挑战 2：单一任务微调的隐患

The Single-Task Trap

灾难性遗忘 (Catastrophic Forgetting)

当模型过度适应特定任务（如只做垃圾邮件分类）时，它会修改关键权重，导致丧失原有的通用能力（如写诗、逻辑推理）。



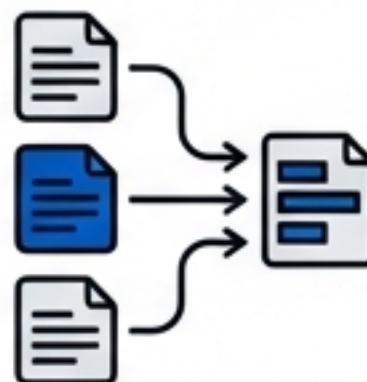
"Fine-tuning on a single task... can lead to the model forgetting how to do other things." — Andrew Ng

如何应对灾难性遗忘？

多任务微调

Multi-task Instruction Tuning

不要只训练一个任务。在混合数据集上训练，并保留部分通用数据。

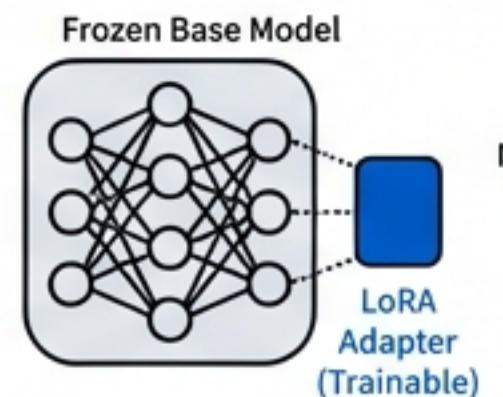


```
train(tasks=['task_a',  
           'task_b',  
           'general'],  
      mix_ratio=0.8)
```

参数高效微调

PEFT / LoRA

冻结原模型权重，只训练外挂参数。原能力被完美保留。

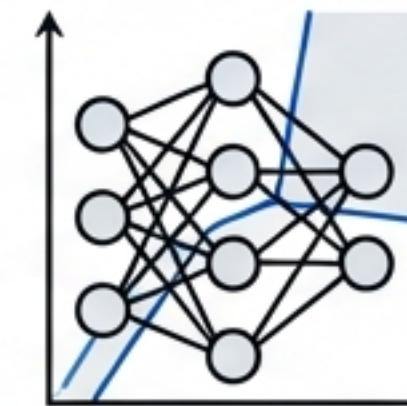


```
model = peft_model(  
                    base_model,  
                    lora_config)
```

正则化

Regularization

限制权重更新的幅度，防止模型参数发生剧烈突变。



```
optimizer =  
          AdamW(lr=1e-5,  
                 weight_decay=0.1)
```

总结与下一步

本章核心回顾 (Key Takeaways)

- ✓ SFT 核心：数据填充 + 架构修改 + Last Token Loss
- ✓ 数据稀缺：使用 STraTA（合成数据 + 自训练）
- ✓ 模型健康：警惕灾难性遗忘，使用 PEFT

