中国科学院大学
University of Chinese Academy of Sciences

自然语言处理

第7讲 微调

王 石 资康莉 刘 瑜

2026年春季课程

https://ictkc.github.io/teaching/

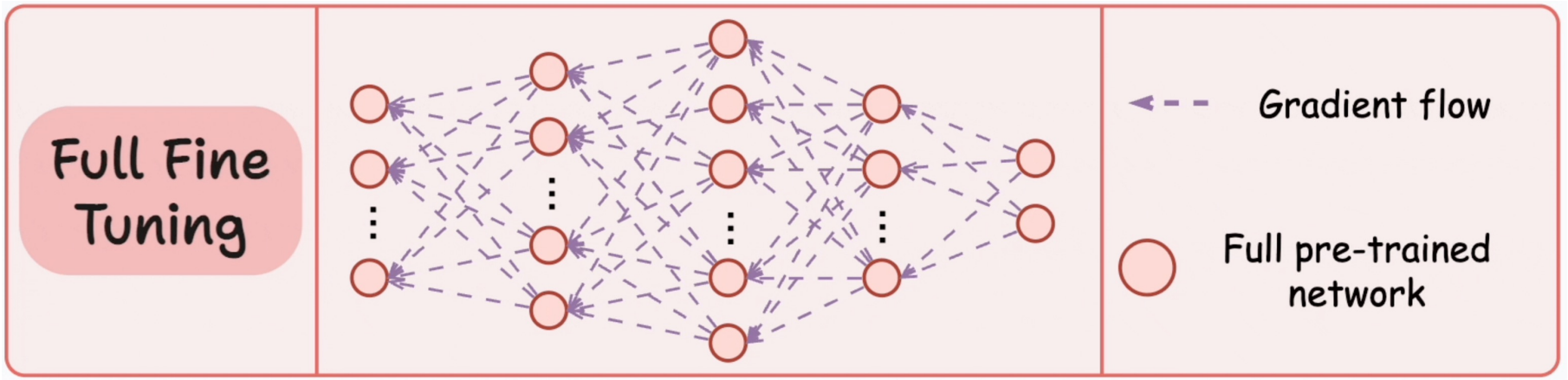
第7讲 微调

中国科学院大学
University of Chinese Academy of Sciences

目 录

全量微调

- 全量微调（Full Fine-Tuning）是指在微调过程中，更新预训练模型的**所有参数**，而非仅更新部分层或参数



全量微调：参数都得改

- 以语言模型为例，在微调过程中模型加载预训练参数Φ_0进行初始化，并通过最大化条件语言模型概率进行参数更新$\Phi_0 + \Delta\Phi$，即：

$$\max_{\Phi} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|y|} \log\left(P_{\Phi}(y_t|x, y_{<t})\right)$$

- 这种微调方式主要的缺点是我们学习到的参数增量$\Delta\Phi$的维度和预训练参数Φ_0是一致的，这种微调方式所需的资源很多，一般被称为**全量微调**

全量微调的类别

- 三种常见的类别形式

1. **标准全量微调（Standard FT）**

   全部参数参与梯度更新，适合数据量充足、资源充足

2. **分阶段全量微调（Layer-wise FT）**

   先训练顶层 → 再逐步解冻，相对稳定且一定程度缓解**灾难性遗忘**

3. **部分冻结全量微调（Hybrid FT）**

   冻结 embedding或底层，本质仍属于全参，因为大部分参数仍可训练

全量微调的优势

- 全量微调通过更新所有参数，赋予模型极强的任务适应能力，是追求极致性能时的首选策略

性能上限高

理论上可实现任务最优性能，充分挖掘训练数据中的信息

适配彻底

调整模型底层通用特征，使其更贴合特定任务，实现深度对齐

灵活性强

适用于各类复杂任务，尤其在任务与预训练目标差异较大时

全量微调的挑战与局限性

❑ 全量微调的四个核心问题：

计算成本极高

需巨大显存和算力，微调70B模型可能需要上百张顶级GPU支持

易过拟合

在小数据集上易过度学习数据特征，导致泛化能力显著下降

灾难性遗忘

针对新任务微调时，模型可能会遗忘预训练阶段习得的通用知识

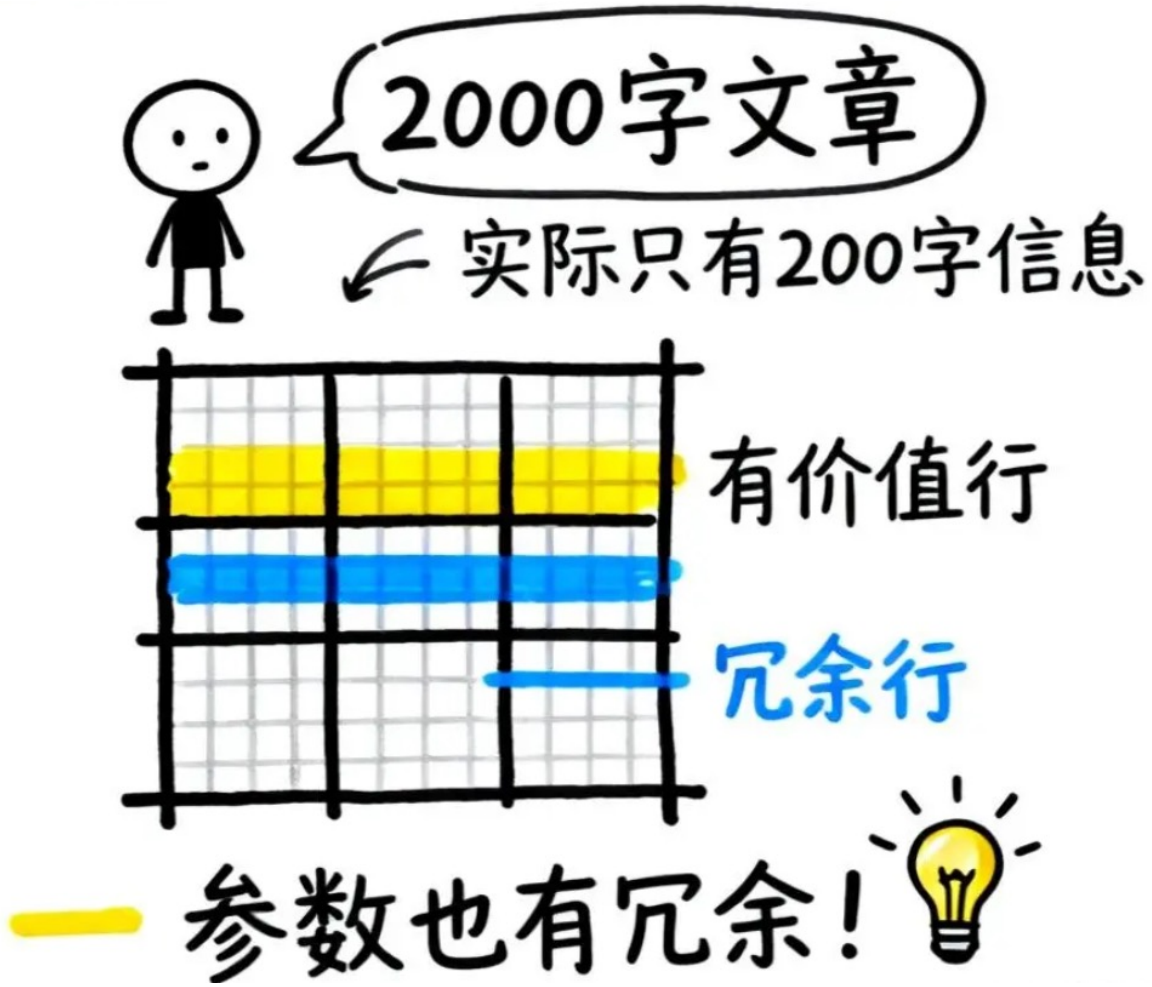
存储成本高

每个下游任务都需保存完整模型，随着任务增多存储开销剧增

中国科学院大学
University of Chinese Academy of Sciences

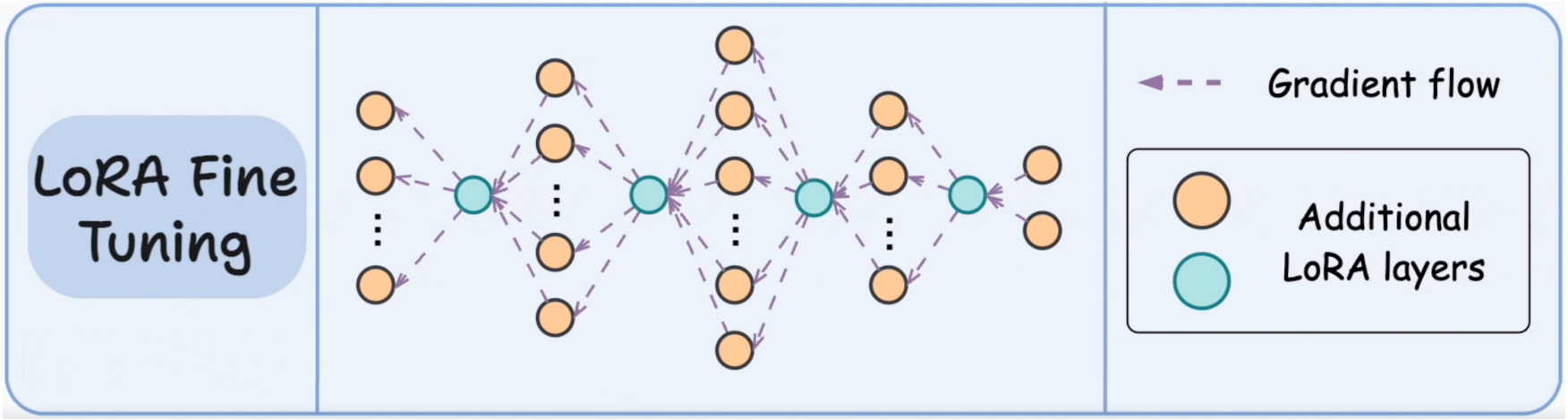
目 录

参数高效微调



参数高效微调

❑ 参数高效微调（Parameter-Efficient Fine-Tuning，PEFT）：
冻结预训练模型的大部分参数，仅更新或添加少量任务特定参数



参数高效微调的基本原理

- 研究者认为能用更少的参数表示上述要学习的参数增量$\Delta\Phi = \Delta\Phi(\Theta)$，其中$|\Theta| \ll |\Phi_0|$，原先寻找$\Delta\Phi$的优化目标变为寻找$\Theta$:

$$\max_{\Theta} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|y|} \log\left(p_{\Phi_0+\Delta\Phi(\Theta)}(y_t|x, y_{<t})\right)$$

- 这种仅微调一部分参数的方法称为**高效微调**。针对高效微调，研究者有很多的实现方式(如Adapter、prefixtuing、LoRA等)

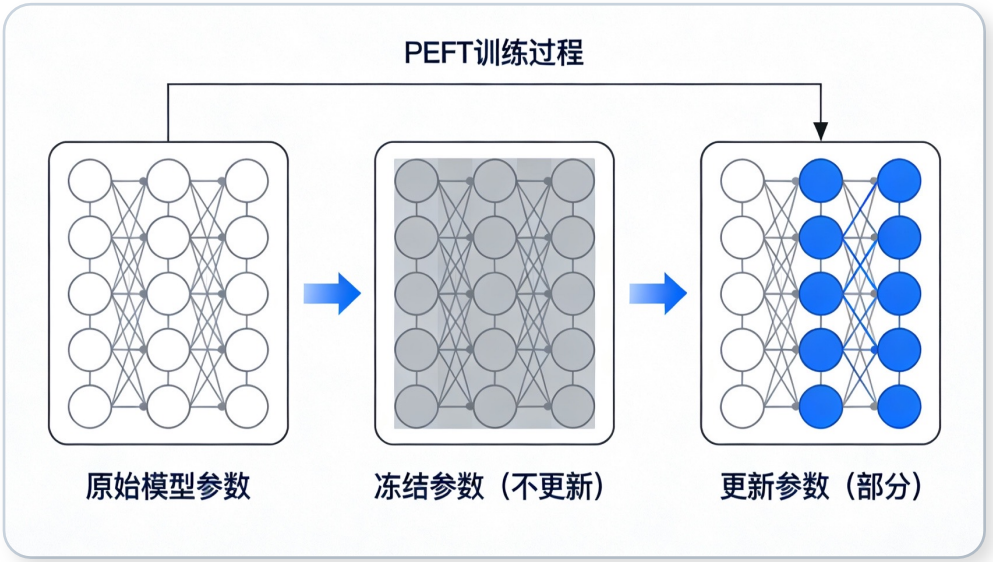
参数高效微调核心优势

❑ 参数高效微调的三个优点：

¥ **成本低廉**：仅训练少量参数，显存占用和算力需求显著降低，适配资源受限场景

泛化性强：保留预训练模型通用知识，避免灾难性遗忘，小样本下表现优异

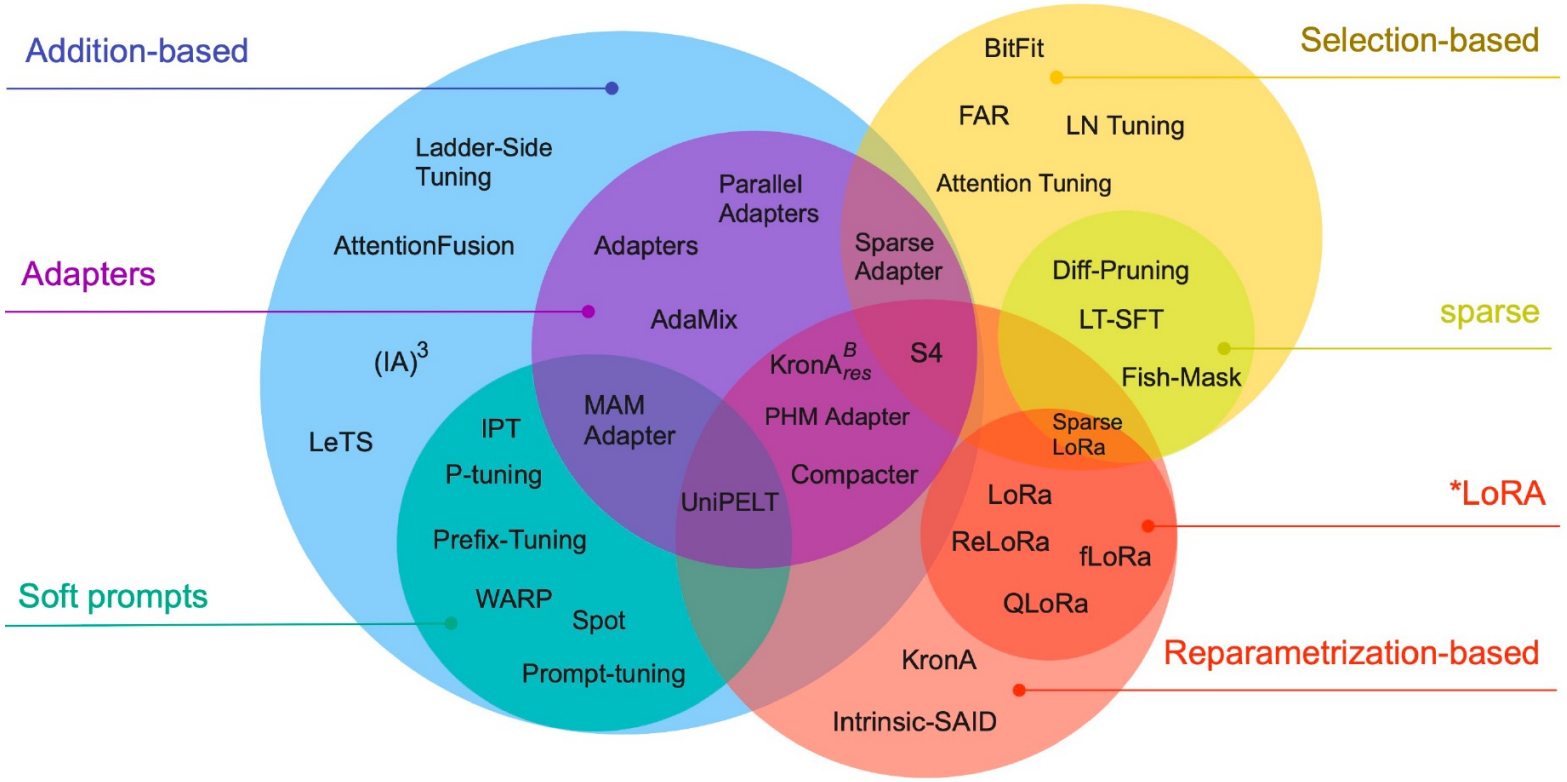
多任务适配：为不同任务训练独立小型模块，实现高效的模型复用与快速切换



参数高效微调方法分类

- 三类高效微调方法

1. 增加额外参数
   a. 适配器
   b. 软提示
2. 选取部分参数
3. 引入重参数化

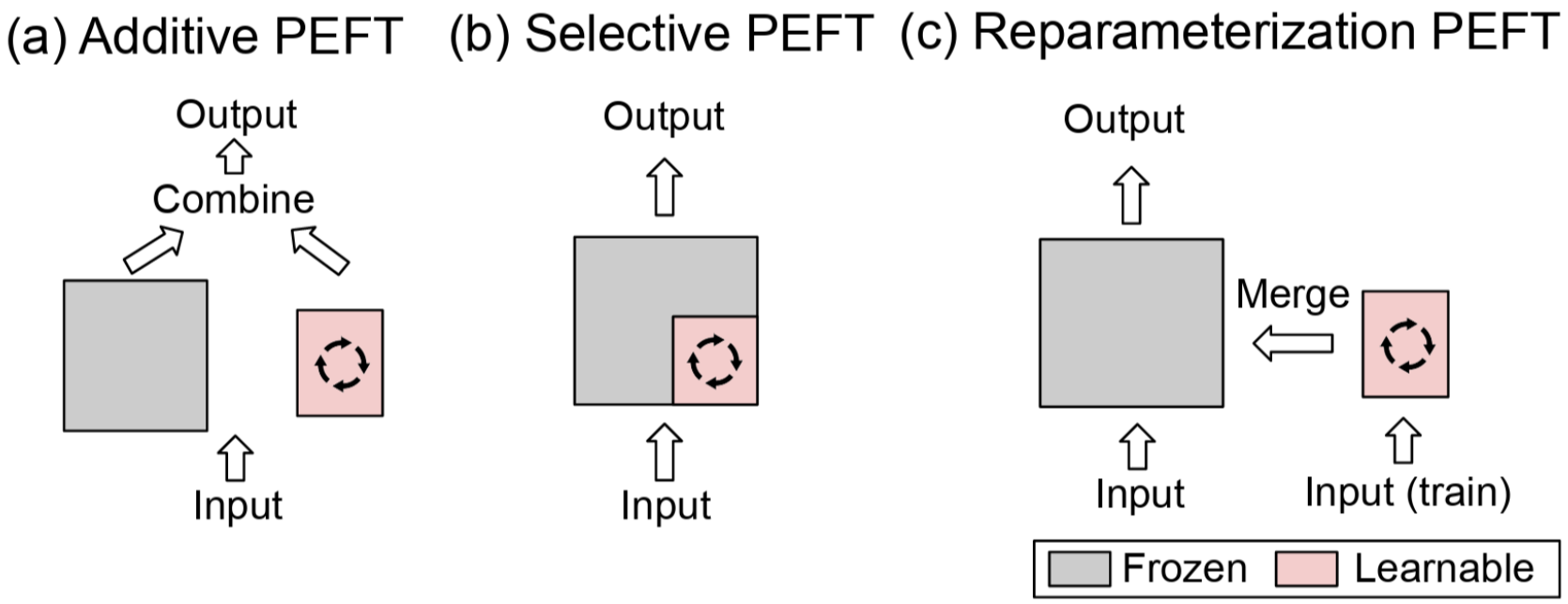


Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning. 2023.

参数高效微调模型架构

- 三类方法的模型结构

1. 增加额外参数　2.选择部分参数　3.引入重参数化



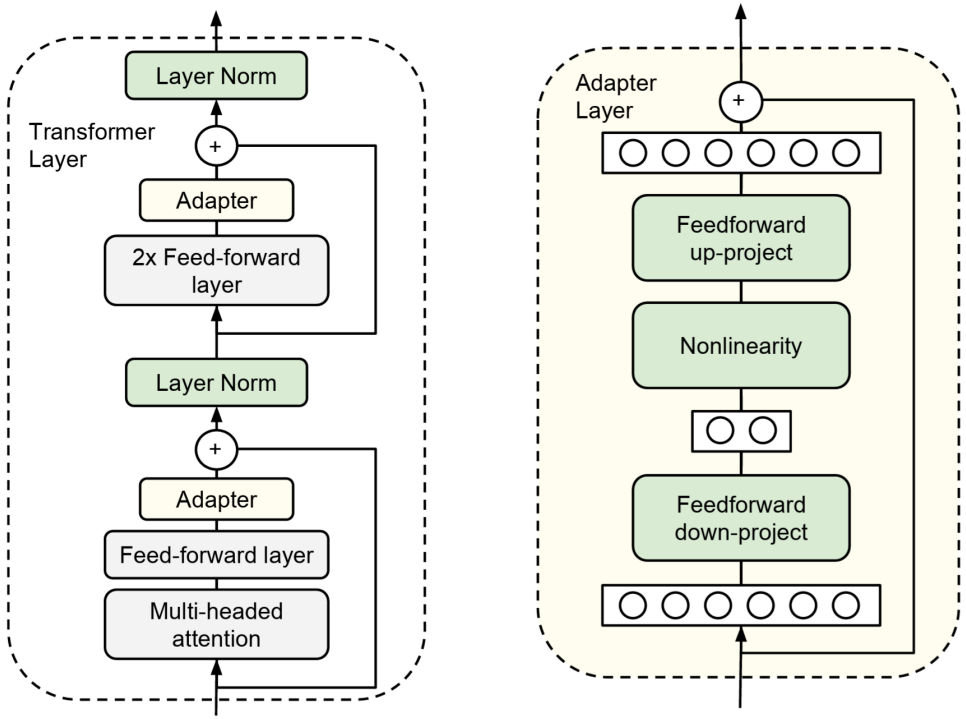
Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning. 2023.

中国科学院大学
University of Chinese Academy of Sciences

目 录

1.适配器方法-Adapter Tuning

❑ 针对每一个Transformer层，增加两个可训练的Adapter层，**通过训练这些模块来适配新任务**



- Adapter结构设计：

Adapter是一个 **下采样 + 非线性 + 上采样** 的瓶颈结构

$$h' = h + \sigma(W_{down} \cdot h) \cdot W_{up}$$

原始隐藏状态　降维（d → r）　升维（r → d）

通常，**瓶颈维度r ≪ d**，一般是几十到几百

Parameter-Efficient Transfer Learning for NLP. 2019.

1.适配器方法-Adapter Tuning

❑ 实验结果

1. Adapter仅需训练少量参数，即可接近全量微调效果
2. 瓶颈维度 r 与数据规模相关，维度在8-256之间，固定为 64 会略降性能

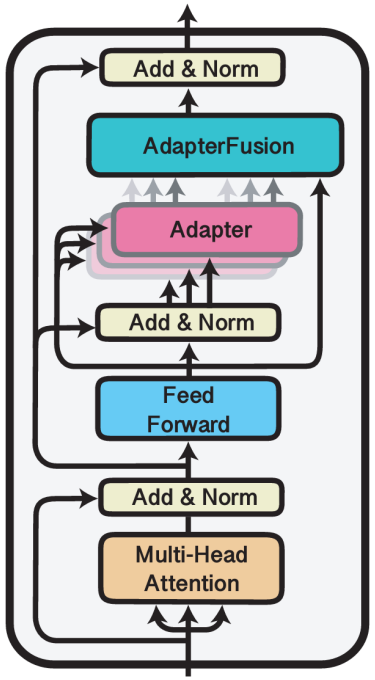
	Total num params	Trained params / task	CoLA	SST	MRPC	STS-B	QQP	MNLI_m	MNLI_mm	QNLI	RTE	Total
BERT_LARGE	9.0×	100%	60.5	94.9	89.3	87.6	72.1	86.7	85.9	91.1	70.1	80.4
Adapters (8-256)	1.3×	3.6%	59.5	94.0	89.5	86.9	71.8	84.9	85.1	90.7	71.5	80.0
Adapters (64)	1.2×	2.1%	56.9	94.2	89.6	87.3	71.8	85.3	84.6	91.4	68.8	79.6

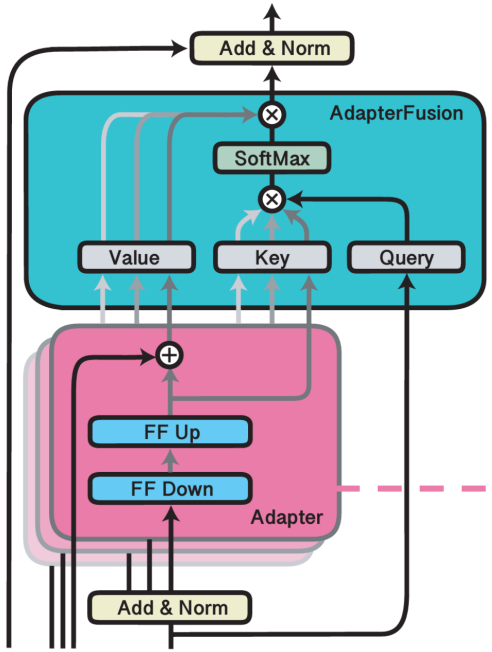
Table 1. Results on GLUE test sets scored using the GLUE evaluation server. MRPC and QQP are evaluated using F1 score. STS-B is evaluated using Spearman's correlation coefficient. CoLA is evaluated using Matthew's Correlation. The other tasks are evaluated using accuracy. Adapter tuning achieves comparable overall score (80.0) to full fine-tuning (80.4) using 1.3× parameters in total, compared to 9×. Fixing the adapter size to 64 leads to a slightly decreased overall score of 79.6 and slightly smaller model.

Parameter-Efficient Transfer Learning for NLP. 2019.

2.适配器方法-Adapter Tuning

❑ AdapterFusion 是 Adapter Tuning 的一个重要扩展，融合多个已训练好的 Adapter，实现更强的多任务迁移能力





- **整体流程（两阶段）**

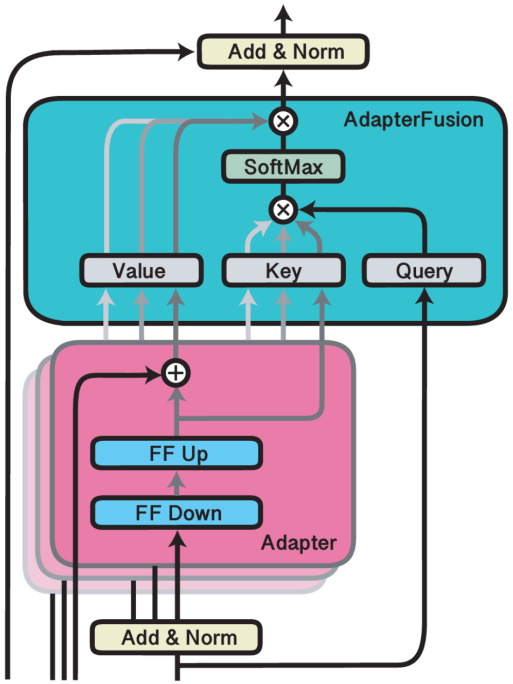
阶段二：知识组合阶段，训练AdapterFusion模块

阶段一：知识提取阶段，对每个任务分别训练 Adapter模块

AdapterFusion:Non-Destructive Task Composition for Transfer Learning. 2020.

2.适配器方法-Adapter Tuning

- 对于阶段一，在不同任务下引入各自的Adapter模块，用于学习特定任务的信息



- **两种训练方式：**

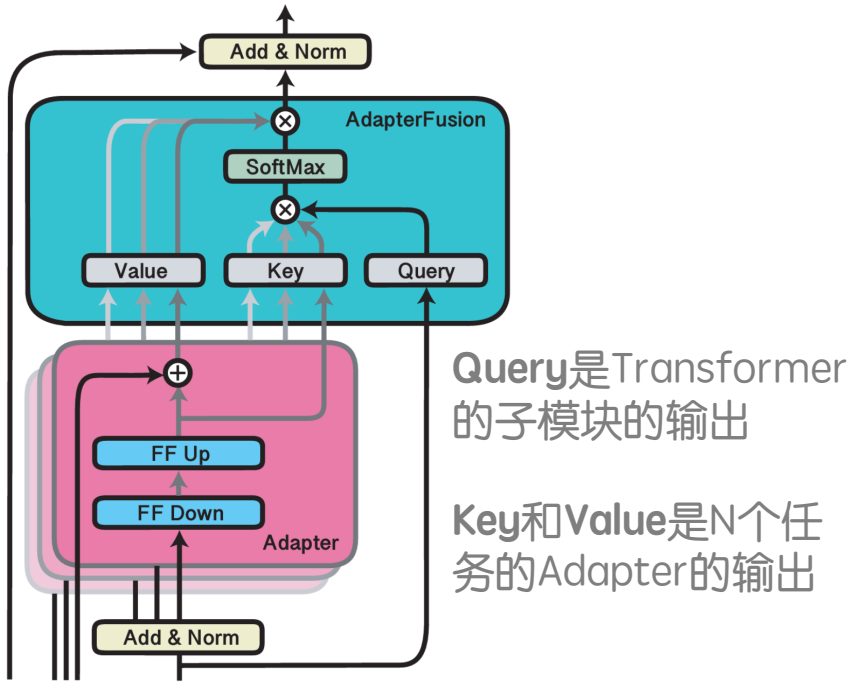
1. Single-Task Adapters(ST-A)：对于N个任务，模型分别独立进行优化，各个任务之间互不干扰，互不影响。

2. Multi-Task Adapters(MT-A)：N个任务通过多任务学习的方式，进行联合优化。

AdapterFusion:Non-Destructive Task Composition for Transfer Learning. 2020.

2.适配器方法-Adapter Tuning

- 对于阶段二，固定语言模型参数和特定任务的Adapter参数，引入一个共享多任务信息的结构AdapterFusion



AdapterFusion具体结构：

1. 每个 Adapter 输出一个表示：

$$h_1, h_2, \ldots, h_N$$

2. 通过注意力计算权重：

$$\alpha_i = \mathrm{softmax}(q \cdot k_i)$$

3. 加权融合：

$$h_{fusion} = \sum \alpha_i h_i$$

AdapterFusion:Non-Destructive Task Composition for Transfer Learning. 2020.

2.适配器方法-AdapterFusion

❑ 实验结果

Dataset	Head		Full		ST-A		MT-A		F. w/ ST-A		F. w/ MT-A	
MNLI	54.59		84.10		**84.32**		82.49	±0.49	84.28		83.05	
QQP	76.79		**90.87**		90.59		89.47	±0.60	90.71		90.58	
SST	85.17	±0.45	92.39	±0.22	91.85	±0.41	92.27	±0.71	92.20	±0.18	**93.00**	±0.20
WGrande	51.92	±0.35	60.01	±0.08	**61.09**	±0.11	57.70	±1.40	60.23	±0.31	59.32	±0.30
IMDB	85.05	±0.22	**94.05**	±0.21	93.85	±0.07	92.56	±0.54	93.82	±0.39	92.66	±0.32
HSwag	34.17	±0.27	**39.25**	±0.76	38.11	±0.14	36.47	±0.98	37.98	±0.01	37.36	±0.10
SocIQA	50.33	±2.50	62.05	±0.04	62.41	±0.11	61.21	±0.89	**63.16**	±0.24	62.56	±0.10
CosQA	50.06	±0.51	60.28	±0.40	60.01	±0.02	61.25	±0.90	60.65	±0.55	**62.78**	±0.07
SciTail	85.30	±2.44	94.32	±0.11	93.90	±0.16	94.53	±0.43	94.04	±0.23	**94.79**	±0.17
Argument	70.61	±0.59	76.87	±0.32	**77.65**	±0.34	75.70	±0.60	**77.65**	±0.21	76.08	±0.27
CSQA	41.09	±0.27	58.88	±0.40	58.91	±0.57	53.30	±2.19	59.73	±0.54	56.73	±0.14
BoolQ	63.07	±1.27	74.84	±0.24	75.66	±1.25	78.76	±0.76	76.25	±0.19	**79.18**	±0.45
MRPC	71.91	±0.13	85.14	±0.45	85.16	±0.52	81.86	±0.99	**90.29**	±0.84	84.68	±0.32
SICK	76.30	±0.71	87.30	±0.42	86.20	±0.00	88.61	±1.06	87.28	±0.99	**90.43**	±0.30
RTE	61.37	±1.17	65.41	±0.90	71.04	±1.62	77.61	±3.21	76.82	±1.68	**79.96**	±0.76
CB	68.93	±4.82	82.49	±2.33	86.07	±3.87	89.09	±1.15	**92.14**	±0.97	89.81	±0.99
Mean	64.17		75.51		76.05		75.80		**77.33**		77.06	

- Head：只训练分类头
- Full：全量微调
- ST-A：为每个任务独立训练 Adapter
- MT-A：对多个任务联合训练Adapter
- Fusion w/ ST-A 和 Fusion w/ MT-A

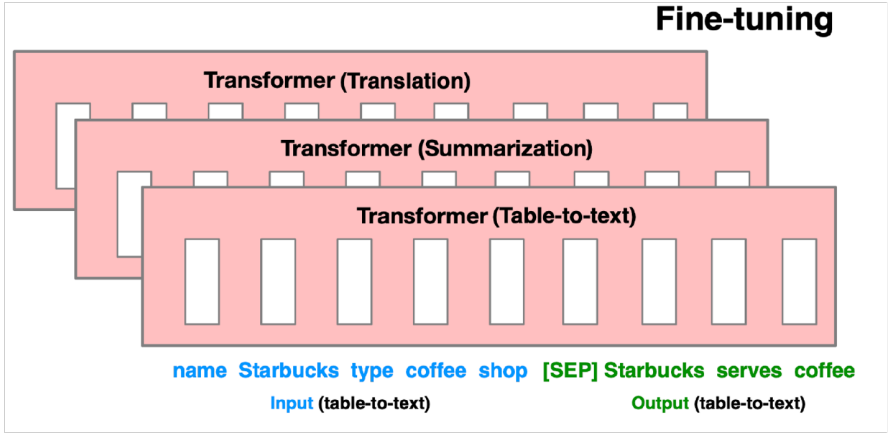
通过对比全量微调、Adapter Tuning、AdapterFusion三种方法，结果表AdapterFusion在整体性能表现优异

AdapterFusion:Non-Destructive Task Composition for Transfer Learning. 2020.

3.软提示方法-Prefix-tuning

❑ 方法介绍

Prefix-tuning在Transformer的每一层（注意不是只有输入层）中添加一个前缀提示，然后在模型训练时只针对前缀对于的结构进行微调



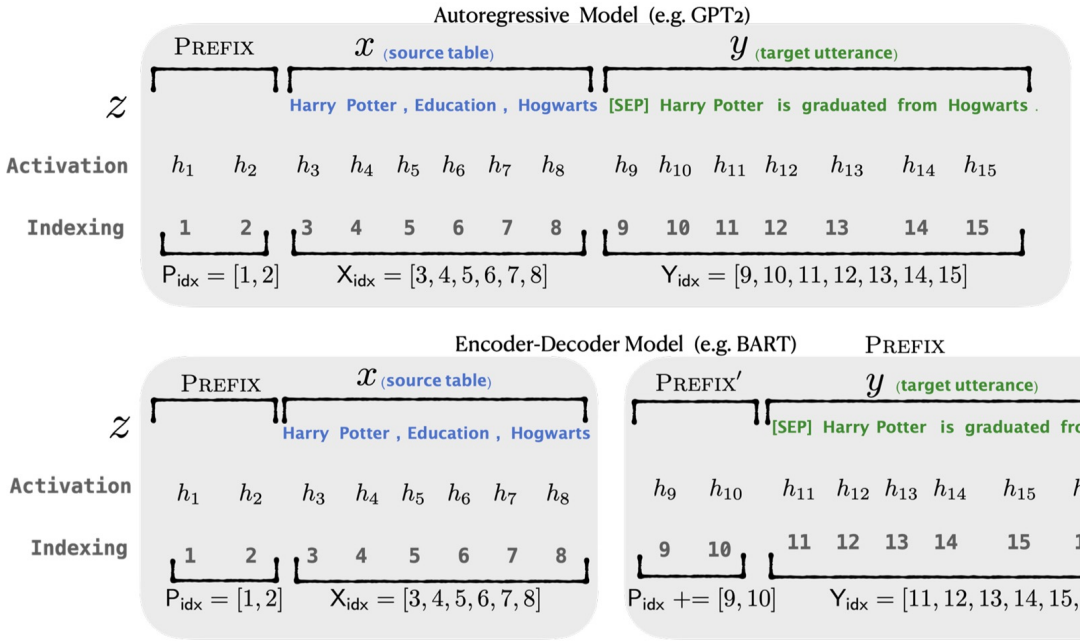
传统微调策略

连续提示优化策略

Prefix-tuning: Optimizing continuous prompts for generation. 2021.

3.软提示方法-Prefix-tuning

- 支持两种生成模型的架构：
  - 自回归（GPT系列等）：前缀的添加方式为z = [PREFIX;x;y]
  - Encoder-Decoder（BART、T5等）：z = [PREFIX;x;PREFIX';y]



Summarization Example

```
Article: Scientists at University College London discovered people
tend to think that their hands are wider and their fingers are
shorter than they truly are.They say the confusion may lie in the
way the brain receives information from different parts of the
body.Distorted perception may dominate in some people, leading to
body image problems ... [ignoring 308 words] could be very
motivating for people with eating disorders to know that there was
a biological explanation for their experiences, rather than
feeling it was their fault."

Summary: The brain naturally distorts body image -
a finding which could explain eating disorders like
anorexia, say experts.
```

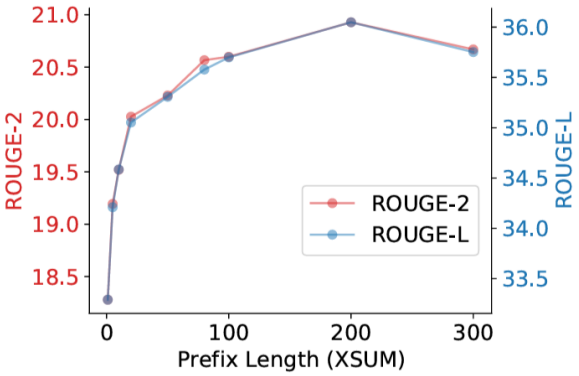
Table-to-text Example

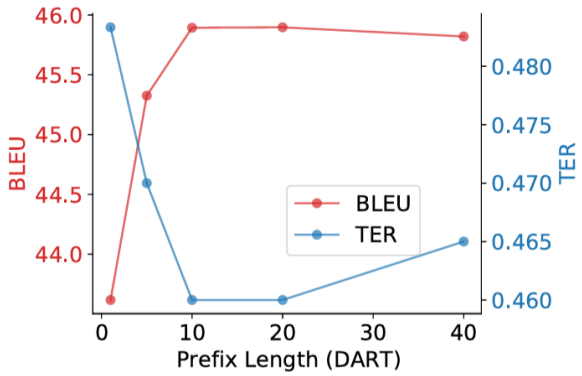
```
Table:  name[Clowns] customer-
rating[1 out of 5]  eatType[coffee
shop] food[Chinese] area[riverside]
near[Clare Hall]

Textual Description: Clowns is a
coffee shop in the riverside area
near Clare Hall that has a rating
1 out of 5 . They serve Chinese
food .
```

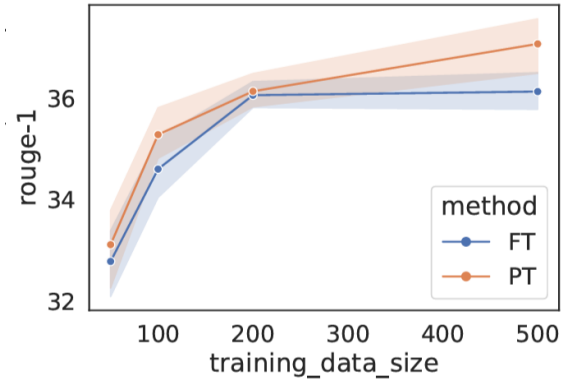
Prefix-tuning: Optimizing continuous prompts for generation. 2021.

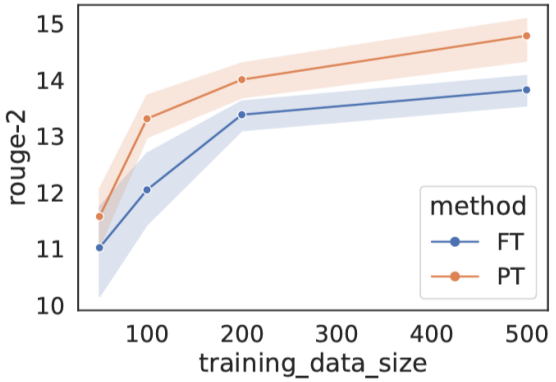
3.软提示方法-Prefix-tuning

❑ 实验结果









- 提示长度：

提示并不是越长越好，对于XSUM等摘要生成类任务，提示的最优长度是200，对于DART等table-to-text任务来说，提示的最优长度是10

- 低资源场景：

在低数据场景下，Prefix-tuning（橙色）不仅优于全量微调（蓝色），而且所需的参数量也要少得多

Prefix-tuning: Optimizing continuous prompts for generation. 2021.

3.软提示方法-Prefix-tuning

❑ 实验结果

	E2E				
	BLEU	NIST	MET	ROUGE	CIDEr
PREFIX	69.7	8.81	46.1	71.4	2.49
Embedding-only: EMB-{PrefixLength}					
EMB-1	48.1	3.33	32.1	60.2	1.10
EMB-10	62.2	6.70	38.6	66.4	1.75
EMB-20	61.9	7.11	39.3	65.6	1.85
Infix-tuning: INFIX-{PrefixLength}					
INFIX-1	67.9	8.63	45.8	69.4	2.42
INFIX-10	67.2	8.48	45.8	69.9	2.40
INFIX-20	66.7	8.47	45.8	70.0	2.42

Table 4: Intrinsic evaluation of Embedding-only (§7.2) and Infixing (§7.3). Both Embedding-only ablation and Infix-tuning underperforms full prefix-tuning.

- Full vs Embedding-only：
Full 是在Transformer的每一层都加入前缀特征，Embedding-only只在输入层加入可训练的参数
实验结果是 Embedding-only < Prefix tuning

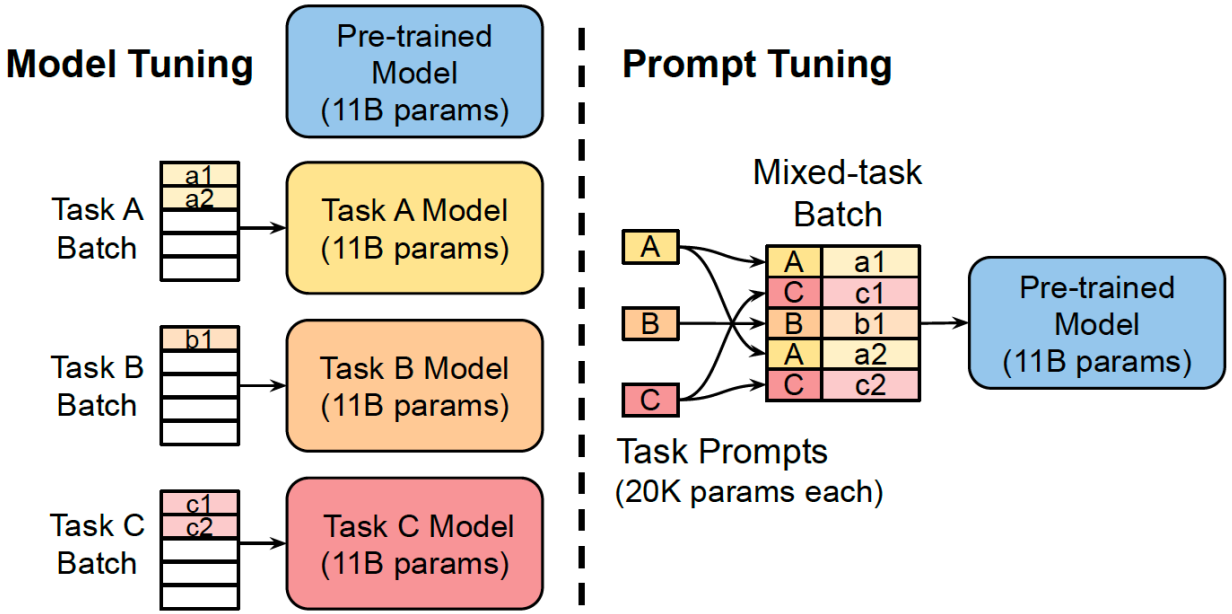
- 前缀（Prefix） vs 中缀（Infix）：
中缀是指在提示加在x和y中间（即z = [x;PREFIX;y]）
实验结果是中缀的效果略差于前缀

Prefix-tuning: Optimizing continuous prompts for generation. 2021.

4.软提示方法-Prompt tuning

❑ 方法介绍

Prompt tuning可看作是Prefix tuning的简化版：它仅在输入序列前拼接一段提示向量，且所有可训练参数都位于输入层



- Model Tuning方法

为每个下游任务创建完整预训练模型副本，且推断操作需分批次进行。（每个任务占110 亿个参数）

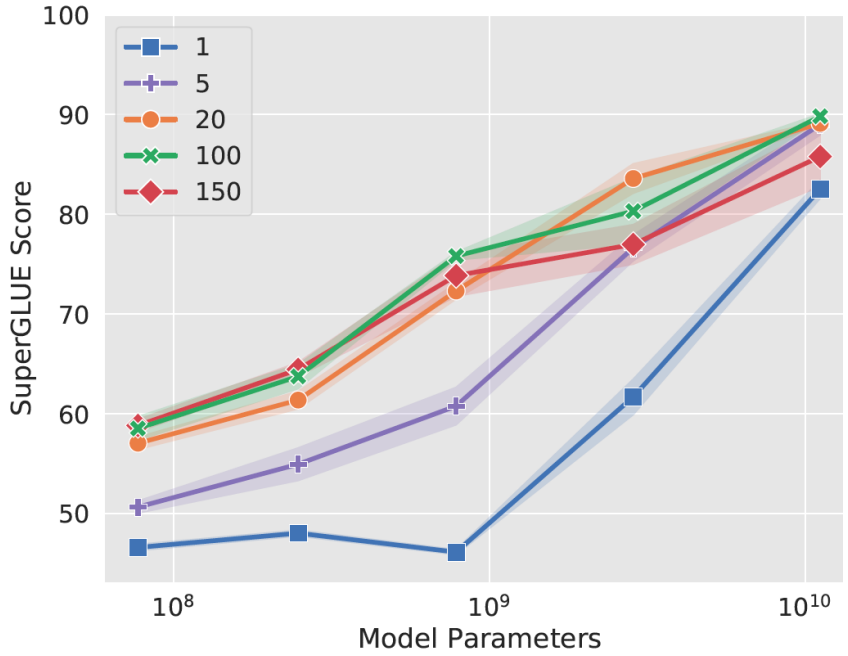
- Prompt tuning 方法

仅需为每个任务存储少量的提示词，并能够利用原始预训练模型实现多任务推断。（每个任务占20,480 个参数）

The power of scale for parameter-efficient prompt tuning. 2021.

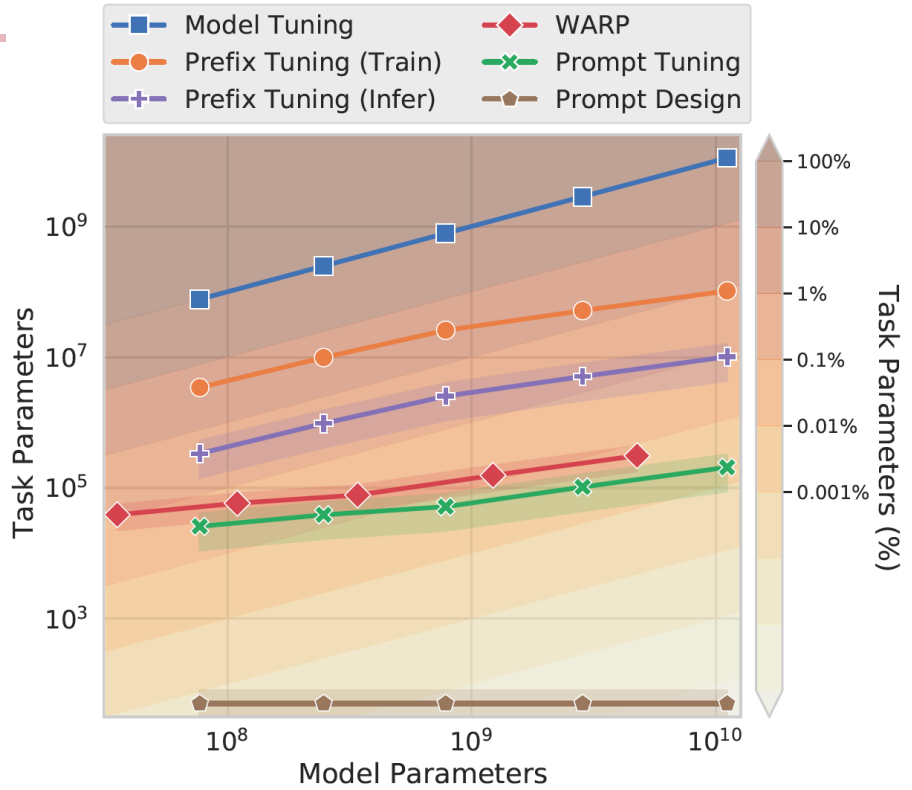
4.软提示方法–Prompt tuning

❑ 实验结果



(a) Prompt length

● 提示长度：
提示长度设置为20是比较好的，此外，当模型参数量达到了10亿量级，这些超参数不再重要



● 模型参数利用率：
Model Tuning：所有参数均为任务特定
Prefix Tuning：占用0.1% -1%参数
Prompt Tuning：低于0.01%参数

中国科学院大学
University of Chinese Academy of Sciences

目 录

BitFit （Bias-only Fine-tuning）

- BitFit仅通过更新语言模型中 bias （偏置）参数，是一种极简但高效的稀疏微调方法

对于全量微调：

会更新所有权重（weight）和偏置（bias）

对于BitFit：

❄️ 冻结所有权重矩阵（W）

🔧 只训练偏置项（b）

👉 本质：用极少量参数调整模型行为，而不是重塑模型结构

BitFit: Simple Parameter-efficient Fine-tuning or Transformer-based Masked Language-models. 2021.

BitFit （Bias-only Fine-tuning）

❑ 为什么BitFit只调bias也有效果?

- Transformer 中每一层通常包含：

  线性变换： $W_x + b$

  LayerNorm：也有 bias（偏移项）

- 调整 bias 的作用：

  改变激活分布（整体平移）

  微调决策边界

👉直观理解：不改变“方向”（权重），只做“平移校准”（bias）

BitFit: Simple Parameter-efficient Fine-tuning or Transformer-based Masked Language-models. 2021.

BitFit （Bias-only Fine-tuning）

❑ 实验结果

BitFit 在仅更新约 **0.1% 模型参数**的情况下，仍能取得较为出色的性能表现

		%Param	QNLI	SST-2	MNLI_m	MNLI_mm	CoLA	MRPC	STS-B	RTE	QQP	Avg.
	Train size		105k	67k	393k	393k	8.5k	3.7k	7k	2.5k	364k	
(V)	Full-FT†	100%	**93.5**	**94.1**	**86.5**	**87.1**	**62.8**	**91.9**	89.8	71.8	**87.6**	**84.8**
(V)	Full-FT	100%	91.7±0.1	93.4±0.2	85.5±0.4	85.7±0.4	62.2±1.2	90.7±0.3	**90.0±0.4**	**71.9±1.3**	87.5±0.4	84.1
(V)	Diff-Prune†	0.5%	**93.4**	**94.2**	**86.4**	**86.9**	63.5	91.3	89.5	71.5	**86.6**	**84.6**
(V)	BitFit	0.08%	91.4±2.4	93.2±0.4	84.4±0.2	84.8±0.1	**63.6±0.7**	**91.7±0.5**	**90.3±0.1**	**73.2±3.7**	85.4±0.1	84.2
(T)	Full-FT‡	100%	91.1	**94.9**	86.7	85.9	**60.5**	**89.3**	**87.6**	70.1	**72.1**	**81.8**
(T)	Full-FT†	100%	**93.4**	94.1	86.7	**86.0**	59.6	88.9	86.6	**71.2**	71.7	81.5
(T)	Adapters‡	3.6%	90.7	94.0	84.9	85.1	59.5	89.5	**86.9**	71.5	**71.8**	81.1
(T)	Diff-Prune†	0.5%	**93.3**	94.1	**86.4**	**86.0**	**61.1**	**89.7**	86.0	70.6	71.1	**81.5**
(T)	BitFit	0.08%	92.0	**94.2**	84.5	84.8	59.7	88.9	85.5	**72.0**	70.5	80.9

Table 1: BERT_LARGE model performance on the GLUE benchmark validation set (V) and test set (T). Lines with † and ‡ indicate results taken from Guo et al. (2020) and Houlsby et al. (2019) (respectively).

BitFit: Simple Parameter-efficient Fine-tuning or Transformer-based Masked Language-models. 2021.

中国科学院大学
University of Chinese Academy of Sciences

目录

1.LoRA （**Lo**w-**R**ank **A**daptation）

- LoRA认为：在下游任务微调过程中，模型的参数增量ΔW实际上是**低秩的**，并非真正的满秩矩阵

$$W_0 \rightarrow W_0 + \Delta W$$

- 这意味着参数中存在冗余或高度相关性，模型的有效更新可以用更低维的表示来刻画

LoRA的核心思想通过**低秩分解**近似权重更新，在冻结原模型的前提下，用极少参数实现高效微调

LoRA: Low-Rank Adaptation of Large Language Models. 2021.

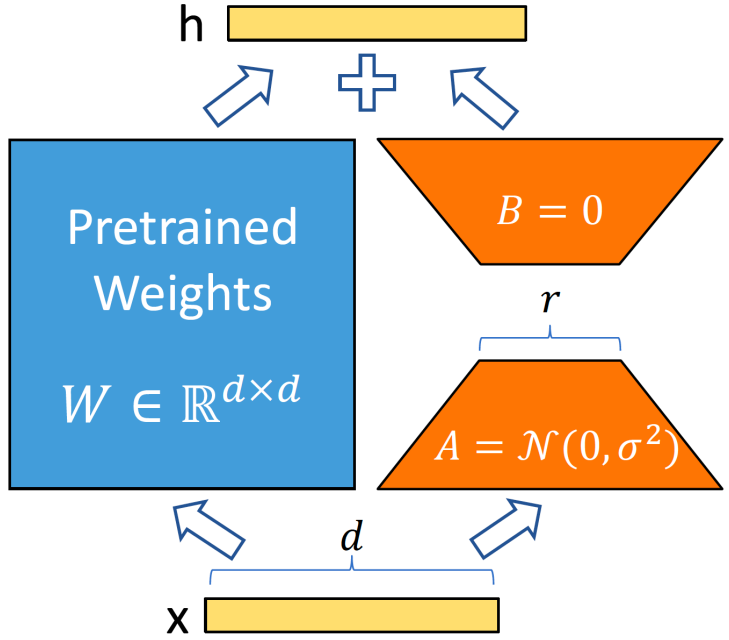
1. LoRA （**Lo**w-**R**ank **A**daptation）

❑ LoRA 在原有预训练权重上引入一个旁路结构，由两个低秩矩阵 A 和 B 组成

- 训练过程中冻结原模型参数，仅更新 A 和 B
- 训练完成后，将低秩增量 BA 与原始权重相加，即可得到微调后的模型参数：

$$h = W_0 x + \Delta W x = W_0 x + BAx$$

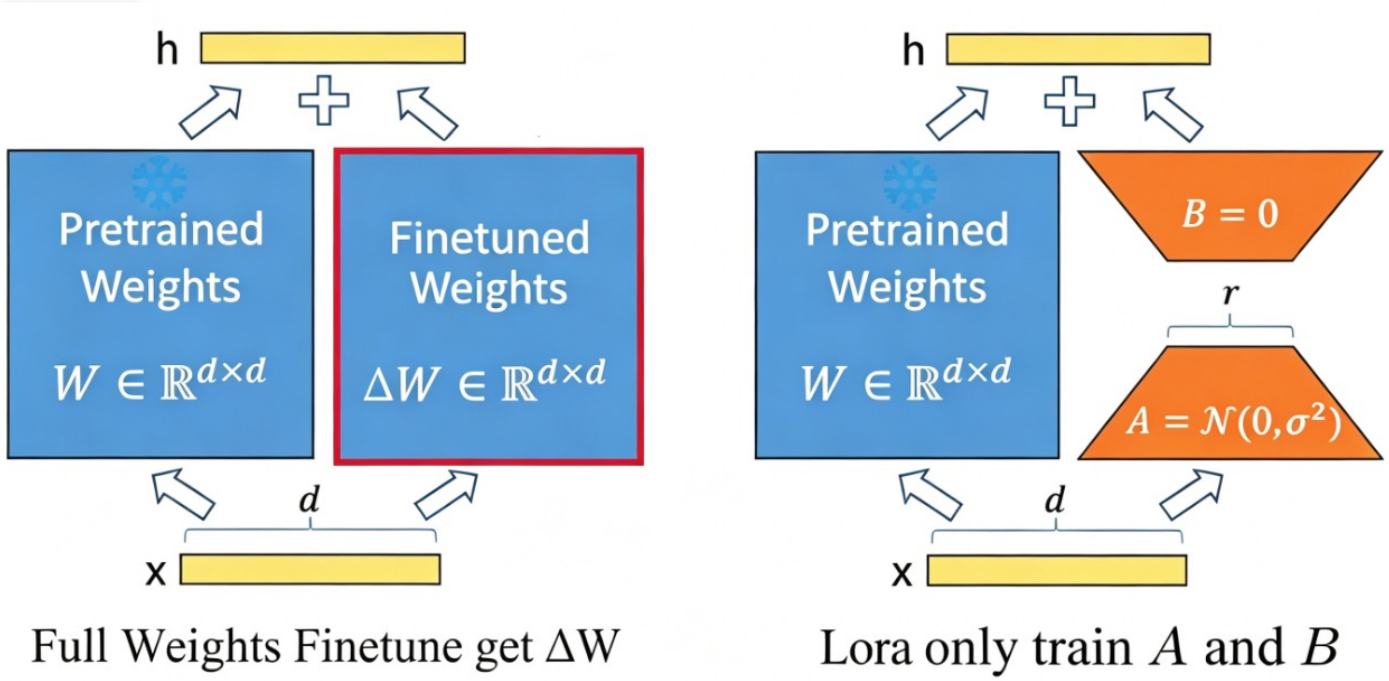
$W_0 \in \mathbb{R}^{d\times d}$　　$B \in \mathbb{R}^{d\times r}$　　$A \in \mathbb{R}^{r\times d}$



LoRA: Low-Rank Adaptation of Large Language Models. 2021.

1. LoRA （**Lo**w-**R**ank **A**daptation）

❑ LoRA 的形式化表示：$h = W_0 x + \Delta W x = W_0 x + BAx$



微调参数量大幅降低，由原来的d^2降为$2rd$（因为 $r \ll d$）

LoRA: Low-Rank Adaptation of Large Language Models. 2021.

1. LoRA （**Lo**w-**R**ank **A**daptation）

❑ 实验结果

在内容理解任务中，LoRA相较于其他微调方法表现最佳（基于RoBERTa）

Model & Method	# Trainable Parameters	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg.
RoB_{base} (FT)*	125.0M	**87.6**	94.8	90.2	**63.6**	92.8	**91.9**	78.7	91.2	86.4
RoB_{base} (BitFit)*	0.1M	84.7	93.7	**92.7**	62.0	91.8	84.0	81.5	90.8	85.2
RoB_{base} (Adpt^{D})*	0.3M	$87.1_{\pm.0}$	$94.2_{\pm.1}$	$88.5_{\pm1.1}$	$60.8_{\pm.4}$	$93.1_{\pm.1}$	$90.2_{\pm.0}$	$71.5_{\pm2.7}$	$89.7_{\pm.3}$	84.4
RoB_{base} (Adpt^{D})*	0.9M	$87.3_{\pm.1}$	$94.7_{\pm.3}$	$88.4_{\pm.1}$	$62.6_{\pm.9}$	$93.0_{\pm.2}$	$90.6_{\pm.0}$	$75.9_{\pm2.2}$	$90.3_{\pm.1}$	85.4
RoB_{base} (LoRA)	0.3M	$87.5_{\pm.3}$	$\mathbf{95.1}_{\pm.2}$	$89.7_{\pm.7}$	$63.4_{\pm1.2}$	$\mathbf{93.3}_{\pm.3}$	$90.8_{\pm.1}$	$\mathbf{86.6}_{\pm.7}$	$\mathbf{91.5}_{\pm.2}$	**87.2**
$\text{RoB}_{\text{large}}$ (FT)*	355.0M	90.2	**96.4**	**90.9**	68.0	94.7	**92.2**	86.6	92.4	88.9
$\text{RoB}_{\text{large}}$ (LoRA)	0.8M	$\mathbf{90.6}_{\pm.2}$	$96.2_{\pm.5}$	$\mathbf{90.9}_{\pm1.2}$	$\mathbf{68.2}_{\pm1.9}$	$\mathbf{94.9}_{\pm.3}$	$91.6_{\pm.1}$	$\mathbf{87.4}_{\pm2.5}$	$\mathbf{92.6}_{\pm.2}$	**89.0**
$\text{RoB}_{\text{large}}$ (Adpt^{P})†	3.0M	$90.2_{\pm.3}$	$96.1_{\pm.3}$	$90.2_{\pm.7}$	$\mathbf{68.3}_{\pm1.0}$	$\mathbf{94.8}_{\pm.2}$	$\mathbf{91.9}_{\pm.1}$	$83.8_{\pm2.9}$	$92.1_{\pm.7}$	88.4
$\text{RoB}_{\text{large}}$ (Adpt^{P})†	0.8M	$\mathbf{90.5}_{\pm.3}$	$\mathbf{96.6}_{\pm.2}$	$89.7_{\pm1.2}$	$67.8_{\pm2.5}$	$\mathbf{94.8}_{\pm.3}$	$91.7_{\pm.2}$	$80.1_{\pm2.9}$	$91.9_{\pm.4}$	87.9
$\text{RoB}_{\text{large}}$ (Adpt^{H})†	6.0M	$89.9_{\pm.5}$	$96.2_{\pm.3}$	$88.7_{\pm2.9}$	$66.5_{\pm4.4}$	$94.7_{\pm.2}$	$92.1_{\pm.1}$	$83.4_{\pm1.1}$	$91.0_{\pm1.7}$	87.8
$\text{RoB}_{\text{large}}$ (Adpt^{H})†	0.8M	$90.3_{\pm.3}$	$96.3_{\pm.5}$	$87.7_{\pm1.7}$	$66.3_{\pm2.0}$	$94.7_{\pm.2}$	$91.5_{\pm.1}$	$72.9_{\pm2.9}$	$91.5_{\pm.5}$	86.4
$\text{RoB}_{\text{large}}$ (LoRA)†	0.8M	$\mathbf{90.6}_{\pm.2}$	$96.2_{\pm.5}$	$\mathbf{90.2}_{\pm1.0}$	$68.2_{\pm1.9}$	$\mathbf{94.8}_{\pm.3}$	$91.6_{\pm.2}$	$\mathbf{85.2}_{\pm1.1}$	$\mathbf{92.3}_{\pm.5}$	**88.6**
DeB_{XXL} (FT)*	1500.0M	91.8	**97.2**	92.0	72.0	**96.0**	92.7	93.9	92.9	91.1
DeB_{XXL} (LoRA)	4.7M	$\mathbf{91.9}_{\pm.2}$	$96.9_{\pm.2}$	$\mathbf{92.6}_{\pm.6}$	$\mathbf{72.4}_{\pm1.1}$	$\mathbf{96.0}_{\pm.1}$	$\mathbf{92.9}_{\pm.1}$	$\mathbf{94.9}_{\pm.4}$	$\mathbf{93.0}_{\pm.2}$	**91.3**

1. LoRA （**Lo**w-**R**ank **A**daptation）

❑ 实验结果

在生成任务中，LoRA也取得了优异的效果（基于GPT-3模型）

Model&Method	# Trainable Parameters	WikiSQL Acc. (%)	MNLI-m Acc. (%)	SAMSum R1/R2/RL
GPT-3 (FT)	175,255.8M	**73.8**	89.5	52.0/28.0/44.5
GPT-3 (BitFit)	14.2M	71.3	91.0	51.3/27.4/43.5
GPT-3 (PreEmbed)	3.2M	63.1	88.6	48.3/24.2/40.5
GPT-3 (PreLayer)	20.2M	70.1	89.5	50.8/27.3/43.5
GPT-3 ($Adapter^H$)	7.1M	71.9	89.8	53.0/28.9/44.8
GPT-3 ($Adapter^H$)	40.1M	73.2	**91.5**	53.2/29.0/45.1
GPT-3 (LoRA)	4.7M	73.4	**91.7**	**53.8/29.8/45.9**
GPT-3 (LoRA)	37.7M	**74.0**	**91.6**	53.4/29.2/45.1

1. LoRA （**Lo**w-**R**ank **A**daptation）

- **权重矩阵探究：**LoRA仅应用于Attention模块中的4类权重矩阵，同时对 W_q和 W_v进行调整可获得最佳性能

	# of Trainable Parameters = 18M						
Weight Type	W_q	W_k	W_v	W_o	W_q, W_k	W_q, W_v	W_q, W_k, W_v, W_o
Rank r	8	8	8	8	4	4	2
WikiSQL ($\pm 0.5\%$)	70.4	70.0	73.0	73.2	71.4	**73.7**	**73.7**
MultiNLI ($\pm 0.1\%$)	91.0	90.8	91.0	91.3	91.3	91.3	**91.7**

Table 5: Validation accuracy on WikiSQL and MultiNLI after applying LoRA to different types of attention weights in GPT-3, given the same number of trainable parameters. Adapting both W_q and W_v gives the best performance overall. We find the standard deviation across random seeds to be consistent for a given dataset, which we report in the first column.

LoRA: Low-Rank Adaptation of Large Language Models. 2021.

1. LoRA （**Lo**w-**R**ank **A**daptation）

- **秩的选择**：相比增加r，保证权重矩阵种类的覆盖更加关键；单纯增大r并不一定能够提升效果。通常，r为4，8，16即可

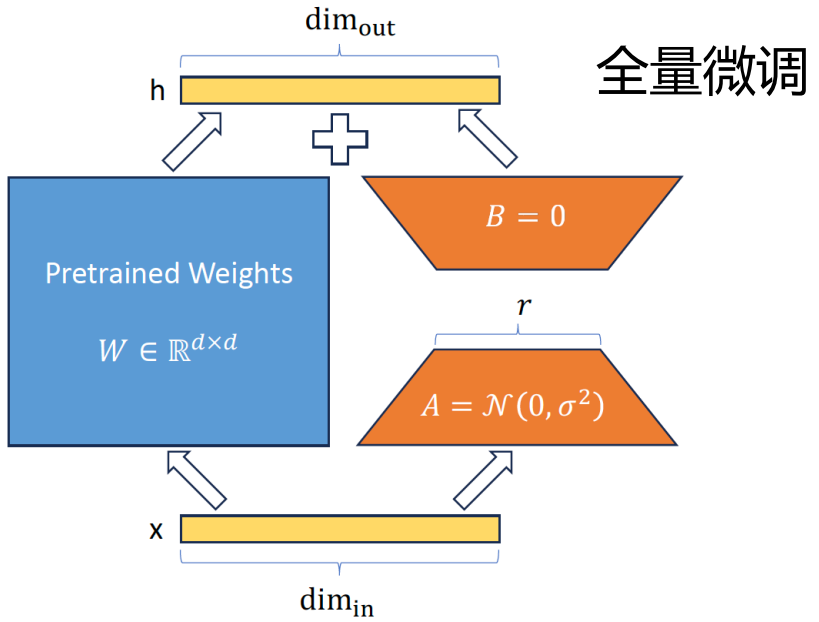
	Weight Type	$r=1$	$r=2$	$r=4$	$r=8$	$r=64$
WikiSQL(±0.5%)	W_q	68.8	69.6	70.5	70.4	70.0
	W_q, W_v	73.4	73.3	73.7	73.8	73.5
	W_q, W_k, W_v, W_o	74.1	73.7	74.0	74.0	73.9
MultiNLI (±0.1%)	W_q	90.7	90.9	91.1	90.7	90.7
	W_q, W_v	91.3	91.4	91.3	91.6	91.4
	W_q, W_k, W_v, W_o	91.2	91.7	91.7	91.5	91.4

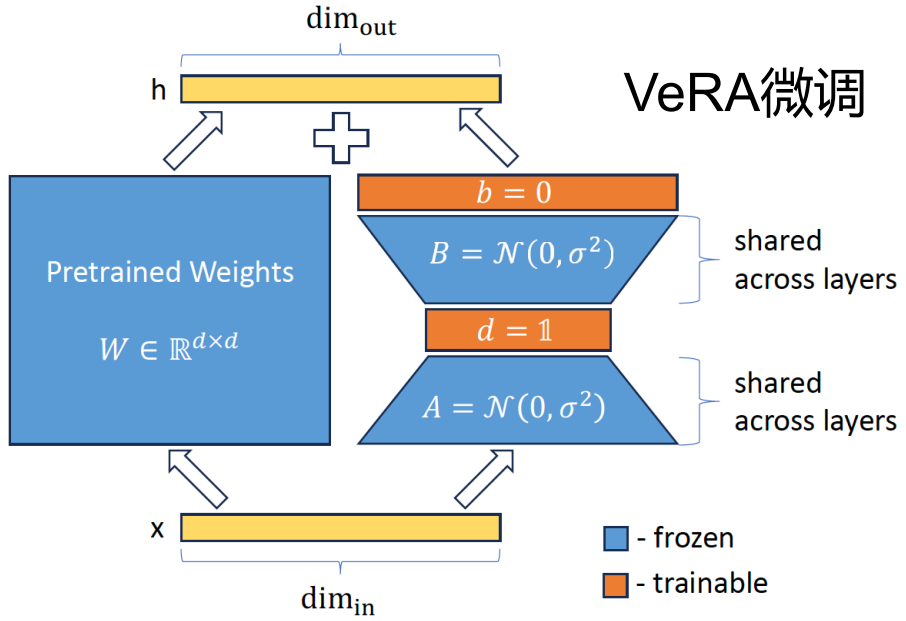
Table 6: Validation accuracy on WikiSQL and MultiNLI with different rank r. To our surprise, a rank as small as one suffices for adapting both W_q and W_v on these datasets while training W_q alone needs a larger r. We conduct a similar experiment on GPT-2 in Section H.2.

LoRA: Low-Rank Adaptation of Large Language Models. 2021.

2. VeRA（Vector-based Random Matrix Adaptation）

- VeRA是LoRA的轻量化，将LoRA中的矩阵学习问题转化为向量学习问题，其核心创新：**共享随机低秩矩阵 + 仅学习缩放向量**





$$h = W_0x + \Delta Wx = W_0x + \underline{BA}x,$$

$$h = W_0x + \Delta Wx = W_0x + \underline{\Lambda_b}B\underline{\Lambda_d}Ax$$

VeRA: Vector-based Random Matrix Adaptation. 2023.

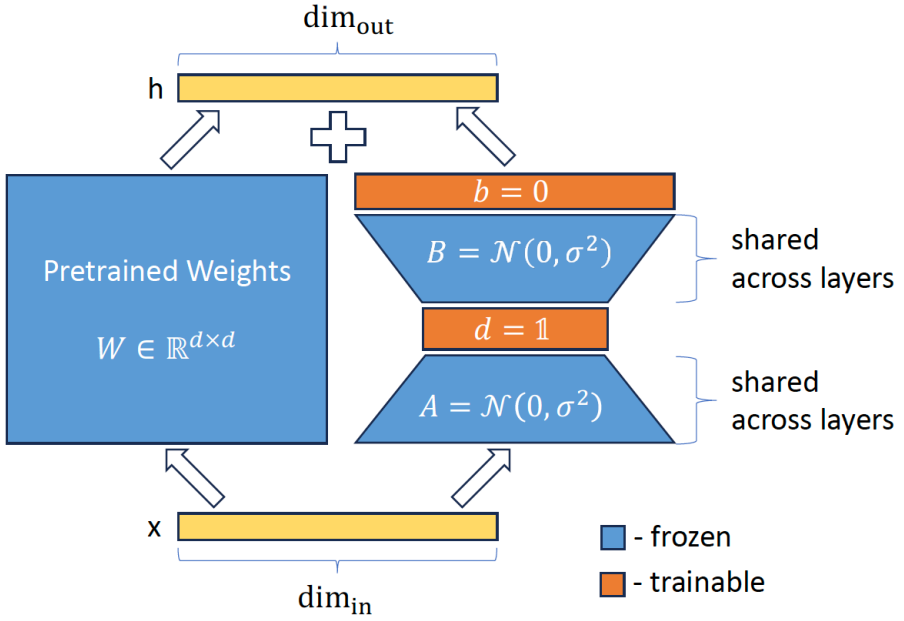
2. VeRA（Vector-based Random Matrix Adaptation）

- VeRA将低秩矩阵A和B随机初始化并固定，引入两个可训练的缩放向量b和d，其形式化表示：

$$W = W_0 + \Lambda_b B \Lambda_d A$$

$A \in \mathbb{R}^{r \times d}, B \in \mathbb{R}^{d \times r}$
随机初始化并冻结的矩阵，r是秩

$\Lambda_d \in \mathbb{R}^{d \times d}, \Lambda_b \in \mathbb{R}^{r \times r}$
可训练的对角矩阵，本质是向量$d \in \mathbb{R}^d, b \in \mathbb{R}^r$



参数量进一步降低，从LoRA的$2rd$降为$r + d$

VeRA: Vector-based Random Matrix Adaptation. 2023.

2. VeRA（**Ve**ctor-based Random Matrix Adaptation）

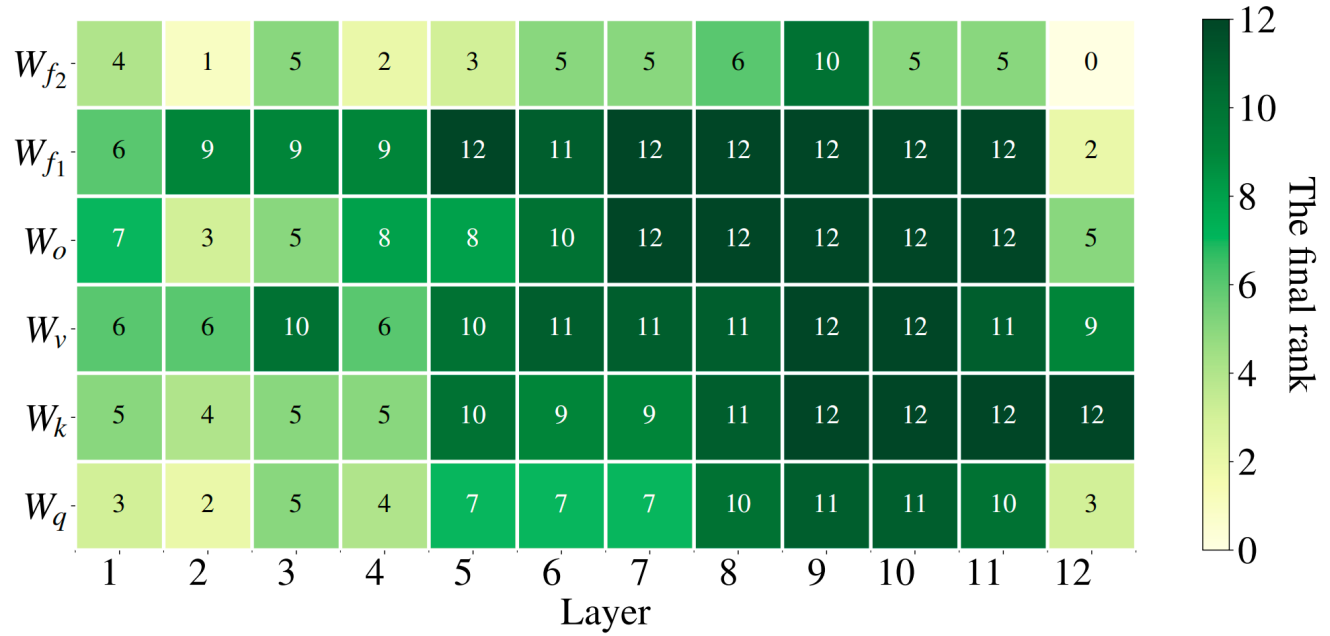
❑ 实验结果

VeRA与LoRA的性能相当，但参数量减少了一个数量级（约十倍）

	Method	# Trainable Parameters	SST-2	MRPC	CoLA	QNLI	RTE	STS-B	Avg.
RoBERTa BASE	FT	125M	94.8	90.2	63.6	92.8	78.7	91.2	85.2
	BitFit	0.1M	93.7	**92.7**	62.0	91.8	81.5	90.8	85.4
	Adpt^{D}	0.3M	$94.2_{\pm 0.1}$	$88.5_{\pm 1.1}$	$60.8_{\pm 0.4}$	$93.1_{\pm 0.1}$	$71.5_{\pm 2.7}$	$89.7_{\pm 0.3}$	83.0
	Adpt^{D}	0.9M	$94.7_{\pm 0.3}$	$88.4_{\pm 0.1}$	$62.6_{\pm 0.9}$	$93.0_{\pm 0.2}$	$75.9_{\pm 2.2}$	$90.3_{\pm 0.1}$	84.2
	LoRA	0.3M	$\mathbf{95.1}_{\pm 0.2}$	$89.7_{\pm 0.7}$	$63.4_{\pm 1.2}$	$\mathbf{93.3}_{\pm 0.3}$	$\mathbf{86.6}_{\pm 0.7}$	$\mathbf{91.5}_{\pm 0.2}$	**86.6**
	VeRA	**0.043M**	$94.6_{\pm 0.1}$	$89.5_{\pm 0.5}$	$\mathbf{65.6}_{\pm 0.8}$	$91.8_{\pm 0.2}$	$78.7_{\pm 0.7}$	$90.7_{\pm 0.2}$	85.2
RoBERTa LARGE	Adpt^{P}	3M	$96.1_{\pm 0.3}$	$90.2_{\pm 0.7}$	$\mathbf{68.3}_{\pm 1.0}$	$\mathbf{94.8}_{\pm 0.2}$	$83.8_{\pm 2.9}$	$92.1_{\pm 0.7}$	87.6
	Adpt^{P}	0.8M	$\mathbf{96.6}_{\pm 0.2}$	$89.7_{\pm 1.2}$	$67.8_{\pm 2.5}$	$\mathbf{94.8}_{\pm 0.3}$	$80.1_{\pm 2.9}$	$91.9_{\pm 0.4}$	86.8
	Adpt^{H}	6M	$96.2_{\pm 0.3}$	$88.7_{\pm 2.9}$	$66.5_{\pm 4.4}$	$94.7_{\pm 0.2}$	$83.4_{\pm 1.1}$	$91.0_{\pm 1.7}$	86.8
	Adpt^{H}	0.8M	$96.3_{\pm 0.5}$	$87.7_{\pm 1.7}$	$66.3_{\pm 2.0}$	$94.7_{\pm 0.2}$	$72.9_{\pm 2.9}$	$91.5_{\pm 0.5}$	84.9
	LoRA-FA	3.7M	96.0	90.0	68.0	94.4	86.1	92.0	87.7
	LoRA	0.8M	$96.2_{\pm 0.5}$	$90.2_{\pm 1.0}$	$68.2_{\pm 1.9}$	$\mathbf{94.8}_{\pm 0.3}$	$85.2_{\pm 1.1}$	$\mathbf{92.3}_{\pm 0.5}$	**87.8**
	VeRA	**0.061M**	$96.1_{\pm 0.1}$	$\mathbf{90.9}_{\pm 0.7}$	$68.0_{\pm 0.8}$	$94.4_{\pm 0.2}$	$\mathbf{85.9}_{\pm 0.7}$	$91.7_{\pm 0.8}$	**87.8**

3. AdaLoRA （**Ad**aptive **Lo**w-**R**ank Allocation）

- **AdaLoRA是一种自适应低秩微调方法，核心思想是动态为不同权重层分配不同秩，在固定总参数预算下实现性能最优**



AdaLoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning. 2023.

3. AdaLoRA （**Ad**aptive **Lo**w-**R**ank Allocation）

- AdaLoRA把增量矩阵ΔW表示为奇异值分解（SVD）的形式：

$$W = W_0 + \Delta W = W_0 + P\Lambda Q$$

$P \in \mathbb{R}^{d\times r}$ 左奇异向量　　$\Lambda \in \mathbb{R}^{r\times r}$ 奇异值对角阵　　$Q \in \mathbb{R}^{r\times d}$ 右奇异向量

- 计算奇异值的重要性：$\mathrm{I}_k = |\lambda_k| \cdot \|p_k\| \cdot \|q_k\|$
- I_k越大，第k个奇异值越重要，应该保留，否则剪纸

AdaLoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning. 2023.

3. AdaLoRA （**Ad**aptive **Lo**w-**R**ank Allocation）

- 自适应预算分配: 根据奇异值重要性，动态调整权重矩阵的秩
- 设总预算$b^{(t)}$(所有层总奇异值数），按立方衰减从初始$b^{(0)}$降到目标$b^{(T)}$：

$$b^{(t)} = \begin{cases} b^{(0)} & 0 \le t < t_i \\ b^{(T)} + \left(b^{(0)} - b^{(T)}\right)\left(1 - \frac{t - t_i - t_f}{T - t_i - t_f}\right)^3 & t_i \le t < T - t_f \\ b^{(T)} & \text{o.w.} \end{cases}$$

- 每步按I_k排序，保留前$b^{(t)}$ 个最重要分量，其余置 0 → 自适应分配秩

AdaLoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning. 2023.

3. AdaLoRA （**Ad**aptive **Lo**w-**R**ank Allocation）

❑ 实验结果

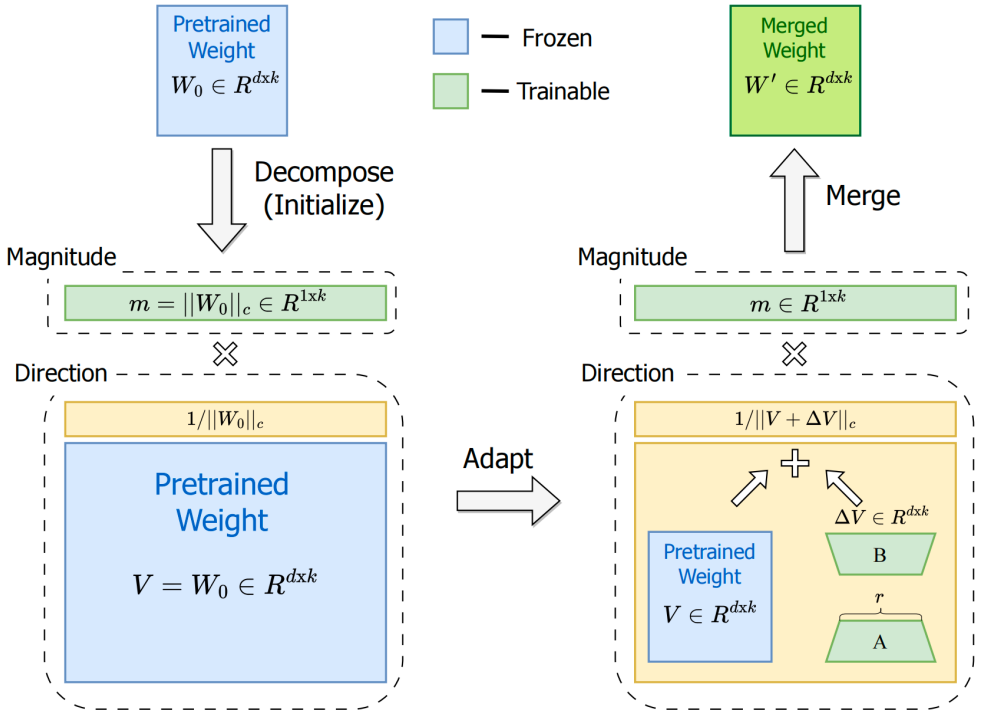
AdaLoRA在两个生成任务、不同参数量下均表现优异

# Params	Method	XSum	CNN/DailyMail
100%	Full FT	**45.49 / 22.33 / 37.26**	44.16 / 21.28 / 40.90
2.20%	LoRA	43.95 / 20.72 / 35.68	**45.03** / 21.84 / 42.15
	AdaLoRA	**44.72 / 21.46 / 36.46**	45.00 / **21.89 / 42.16**
1.10%	LoRA	43.40 / 20.20 / 35.20	44.72 / 21.58 / 41.84
	AdaLoRA	**44.35 / 21.13 / 36.13**	**44.96 / 21.77 / 42.09**
0.26%	LoRA	43.18 / 19.89 / 34.92	43.95 / 20.91 / 40.98
	AdaLoRA	**43.55 / 20.17 / 35.20**	**44.39 / 21.28 / 41.50**
0.13%	LoRA	42.81 / 19.68 / 34.73	43.68 / 20.63 / 40.71
	AdaLoRA	**43.29 / 19.95 / 35.04**	**43.94 / 20.83 / 40.96**

AdaLoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning. 2023.

4. DoRA （Weight-Decomposed LoRA）

❑ DoRA是一种权重分解低秩自适应方法，核心是**将预训练权重进行幅度-方向分解**，并对这两个部分进行独立更新



❑ **1. 权重矩阵W_0分解**：幅度向量m与单位方向矩阵V/ $\|V\|_c$，公式如下：

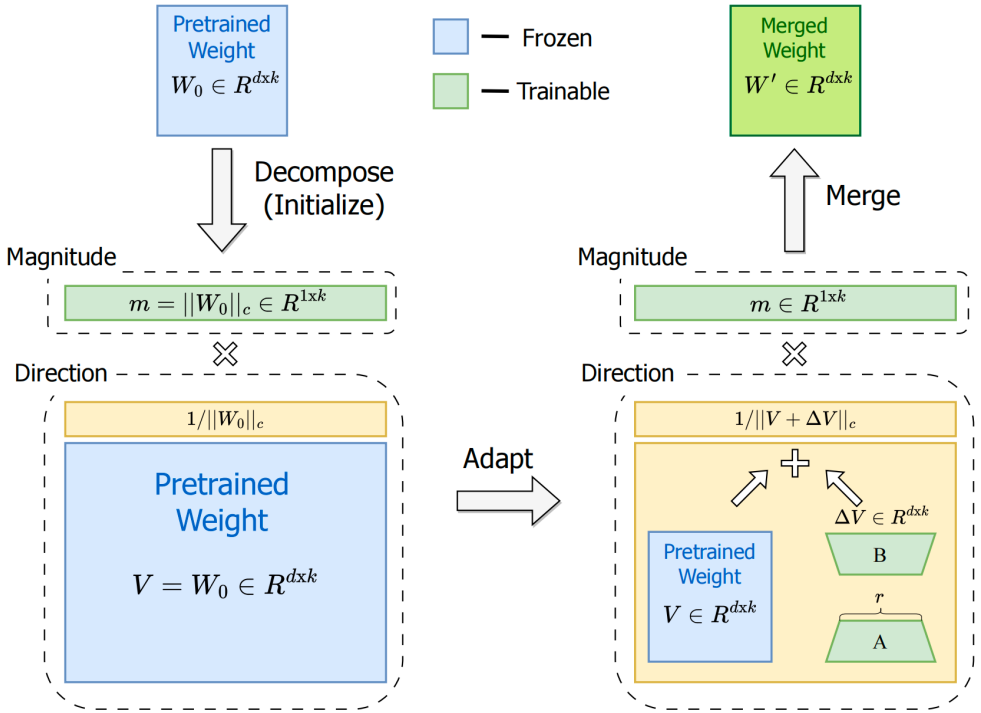
$$W_0 = m \cdot \frac{V}{\|V\|_c} = \|W_0\|_c \cdot \frac{W_0}{\|W_0\|_c}$$

- $m = \|W_0\|_c \in \mathbb{R}^{1\times k}$：按列 ℓ_2 范数构成的幅度向量，
- $V = W_0 \in \mathbb{R}^{d\times k}$：方向矩阵，初始化为原权重；
- $\|\cdot\|_c$：按列向量的 ℓ_2 范数（列范数）。

DoRA: Weight-Decomposed Low-Rank Adaptation. 2024.

4. DoRA （Weight-Decomposed LoRA）

❑ DoRA是一种权重分解低秩自适应方法，核心是**将预训练权重分解为幅度和方向**，并对这两个部分进行独立更新



❑ **2. 权重矩阵W_0更新**：幅度向量m可学习，方向矩阵用LoRA低秩更新，如下：

$$W' = \underline{m} \cdot \frac{V+\Delta V}{\|V+\Delta V\|_c} = \underline{m} \cdot \frac{W_0+\underline{BA}}{\|W_0+\underline{BA}\|_c}$$

- $\underline{m}$：可训练的幅度向量（仅训练这部分，参数量少）
- BA：LoRA 低秩增量，$B \in \mathbb{R}^{d\times r}$、$A \in \mathbb{R}^{r\times k}$
- $\|V+\Delta V\|_c$：更新后方向矩阵的列范数

DoRA: Weight-Decomposed Low-Rank Adaptation. 2024.

4. DoRA （Weight-Decomposed LoRA）

❑ 实验结果：DoRA在常识推理、视觉指令调优和图像/视频-文本理解等任务上，都显著优于LoRA

Model	PEFT Method	# Params (%)	BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Avg.
ChatGPT	-	-	73.1	85.4	68.5	78.5	66.1	89.8	79.9	74.8	77.0
LLaMA-7B	Prefix	0.11	64.3	76.8	73.9	42.1	72.1	72.9	54.0	60.6	64.6
	Series	0.99	63.0	79.2	76.3	67.9	75.7	74.5	57.1	72.4	70.8
	Parallel	3.54	67.9	76.4	78.8	69.8	78.9	73.7	57.3	75.2	72.2
	LoRA	0.83	68.9	80.7	77.4	78.1	78.8	77.8	61.3	74.8	74.7
	DoRA† (Ours)	0.43	70.0	82.6	79.7	83.2	80.6	80.6	65.4	77.6	**77.5**
	DoRA (Ours)	0.84	69.7	83.4	78.6	87.2	81.0	81.9	66.2	79.2	**78.4**
LLaMA-13B	Prefix	0.03	65.3	75.4	72.1	55.2	68.6	79.5	62.9	68.0	68.4
	Series	0.80	71.8	83	79.2	88.1	82.4	82.5	67.3	81.8	79.5
	Parallel	2.89	72.5	84.9	79.8	92.1	84.7	84.2	71.2	82.4	81.4
	LoRA	0.67	72.1	83.5	80.5	90.5	83.7	82.8	68.3	82.4	80.5
	DoRA† (Ours)	0.35	72.5	85.3	79.9	90.1	82.9	82.7	69.7	83.6	**80.8**
	DoRA (Ours)	0.68	72.4	84.9	81.5	92.4	84.2	84.2	69.6	82.8	**81.5**
LLaMA2-7B	LoRA	0.83	69.8	79.9	79.5	83.6	82.6	79.8	64.7	81.0	77.6
	DoRA† (Ours)	0.43	72.0	83.1	79.9	89.1	83.0	84.5	71.0	81.2	**80.5**
	DoRA (Ours)	0.84	71.8	83.7	76.0	89.1	82.6	83.7	68.2	82.4	**79.7**
LLaMA3-8B	LoRA	0.70	70.8	85.2	79.9	91.7	84.3	84.2	71.2	79.0	80.8
	DoRA† (Ours)	0.35	74.5	88.8	80.3	95.5	84.7	90.1	79.1	87.2	**85.0**
	DoRA (Ours)	0.71	74.6	89.3	79.9	95.5	85.6	90.5	80.4	85.8	**85.2**

4. DoRA （Weight-Decomposed LoRA）

- 图中展示了FT、LoRA与DoRA在查询权重矩阵上的变化趋势
- DoRA学习模式更加更接近全量微调（FT）的行为

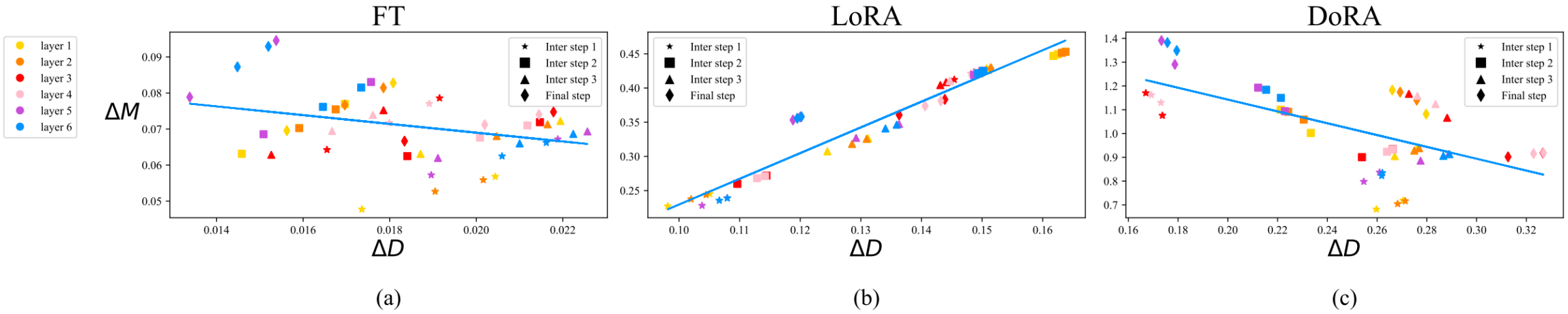
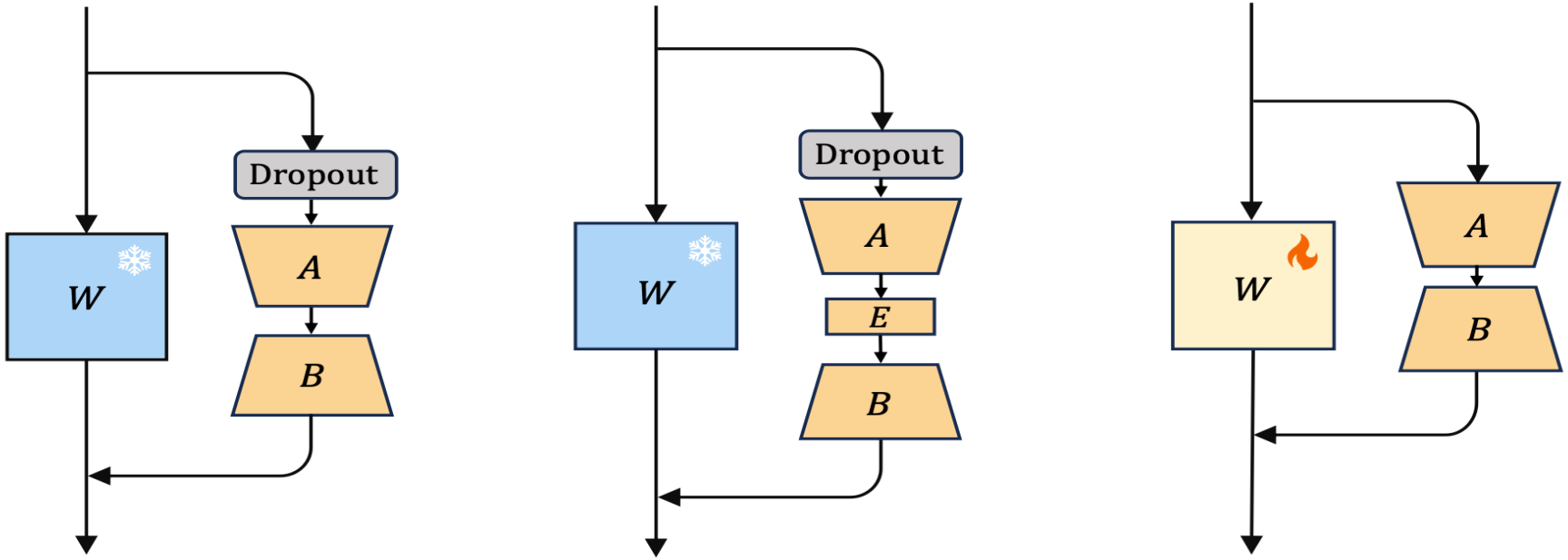


Figure 2. Magnitude and direction updates of (a) FT, (b) LoRA, and (c) DoRA of the query matrices across different layers and intermediate steps. Different markers represent matrices of different training steps and different colors represent the matrices of each layer.

DoRA: Weight-Decomposed Low-Rank Adaptation. 2024.

LoRA其他变体

- 此外，LoRA方法上还有许多不同的变体，主要有LoRA-FA、LoRa-drop、PiSSA、rsLoRA、Delta-LoRA、LoRA-GA等



(a) LoRA/DyLoRA　(b) AdaLoRA　(c) Delta-LoRA

目 录

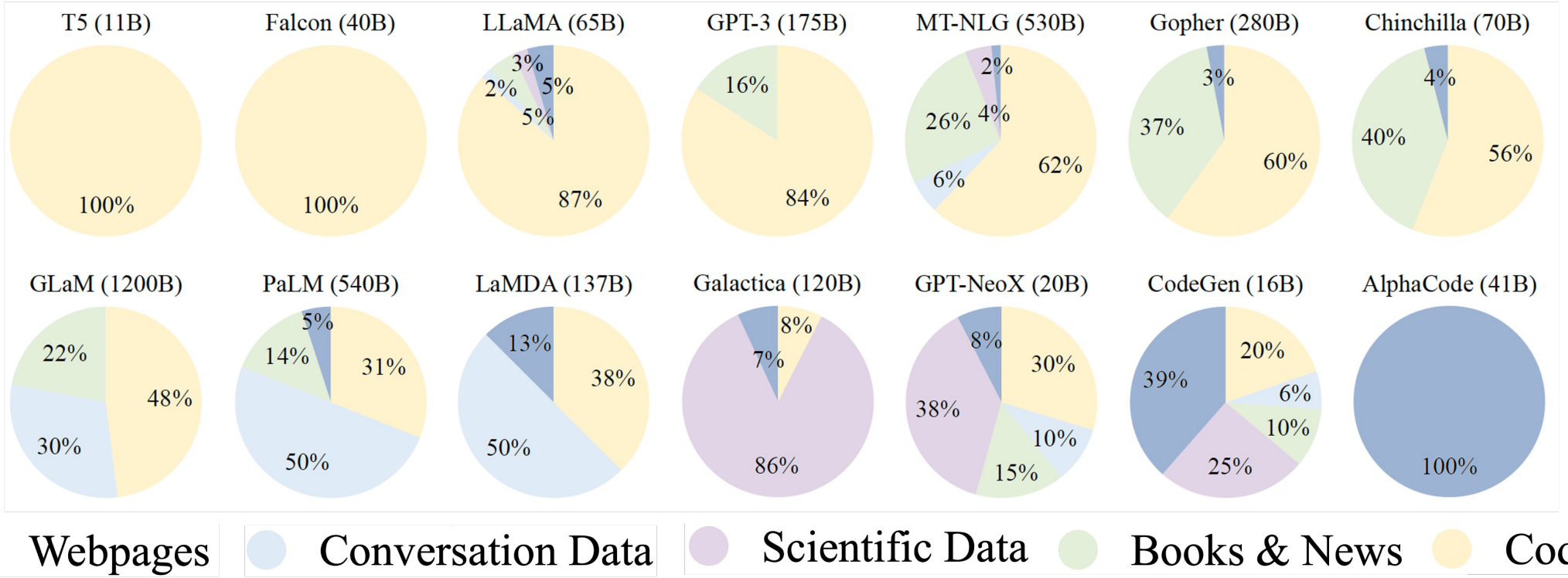
大模型三阶段训练流程

- **预训练**：在海量无标注文本上做自监督学习，掌握语言知识
- **指令微调**：用指令-回答数据训练，让模型学会按要求回答
- **对齐**：结合人类反馈优化输出，使结果更符合人类偏好

预训练　　指令微调　　强化学习

预训练数据

- **预训练数据**：来自互联网和公开语料（如网页、维基、新闻、代码等）的海量无标注文本



A Survey of Large Language Models. 2023.

预训练范式

- **自回归预测**：根据已有序列计算下一个token的概率分布

$$P(x_{T+1} \mid x_1, x_2, \ldots, x_T) = \mathrm{softmax}\big(f(x_1, x_2, \ldots, x_T)\big)$$

- **预训练目标**：在海量无标签文本上，最大化整个序列的联合概率

$$p(x_{1:T}) = \prod_{t=1}^{T} p(x_t \mid x_{<t}) \Leftrightarrow \max \prod_{t=1}^{T} p_\theta(x_t \mid x_{<t})$$

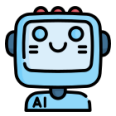
- **自监督范式**：不依赖人工标注、自动从原始文本构建训练信号

预训练效果：学到通用的词法、句法、语义及常识为后续指令微调和强化学习打下坚实基础

预训练不足

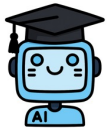
- 预训练模型虽然在理解语言方面表现出色，但并不总是能准确理解用户的具体指令或优先考虑用户的意图

根据以下需求，**用标准SQL语句查询用户表**(users)中age>30的记录，按signup_date 降序排序。**只输出SQL语句**



预训练模型

1.从用户表中选择age>30的记录，并按照signup_date 的降序排序。
2.从用户表中选择age>30的记录，并按照signup_date的升序排序，但只输出年龄大于30的记录。



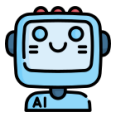
指令微调模型

```
SELECT* FROM users WHERE age >30
ORDER BY signup_date DESC;
```

预训练不足

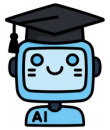
- 预训练模型虽然在理解语言方面表现出色，但并不总是能准确理解用户的具体指令或优先考虑用户的意图

在上海宝山，樱花进入盛花期，游人纷至沓来；在陕西西安，两公里樱花大道花海绵延；在河北沧州梨花盛开，春景醉人。**以上描述中地点有哪些?**



预训练模型

展现了中国各地春天各具特色的美景。在江苏苏州，园林桃花盛开，粉色花影映水，吸引游客驻足；在广东广州，紫荆花绽放，街道如诗如画；在四川成都…



指令微调模型

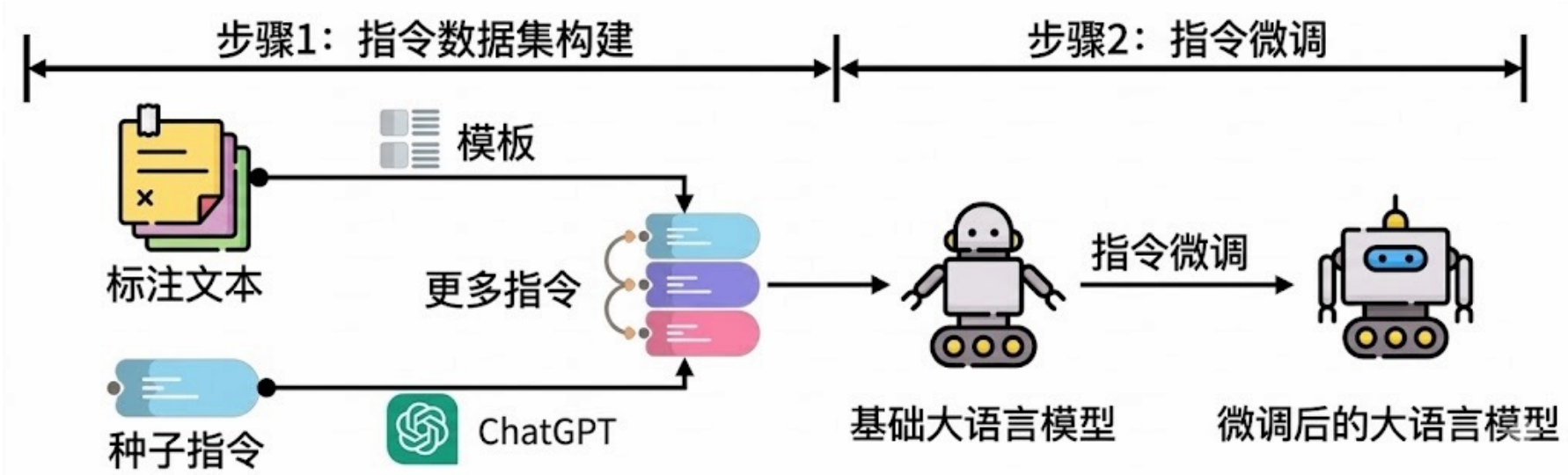
展示了中国不同地区春季的自然美景，包括上海宝山、陕西西安、河北沧州

为什么要指令微调?

- 预训练掌握语言和知识，但对指令理解不足、输出不够可控
- 指令微调后，模型能**准确执行任务、输出更稳定**，并更符合人类表达习惯

什么是指令微调?

- 指令微调（Instruction Tuning）是一种在预训练模型基础上，使用由“指令-响应对”标注数据集进行模型微调的技术
- **核心目标是让模型学会更好地理解和遵循人类指令**



A Survey on Post-training of Large Language Models. 2025.

指令微调的原理

- 指令微调：输入为 **指令 + 输入**（Instruction + Input），输出为 **回答**（Response）
- 优化目标：仅对 **回答部分** 计算损失：

$$\mathcal{L} = -\sum_{t=1}^{T} \log p_\theta(y_t \mid \text{Instruciton}, \text{Input}, y_{<t})$$

指令
请将下面句子翻译成中文

输入
I love NLP.

已生成的回答
我爱自然语言处理.

指令微调 vs 有监督微调

❑ 所属关系：指令微调是有监督微调的一种方式

指令微调 (Instruction Tuning)

核心目标：提升模型对**多样化指令**的理解能力，增强任务泛化性

训练重点：学习“理解人类意图”，并将知识应用于不同的下游任务

最终效果：在零样本(Zero-shot)和少样本(Few-shot)场景下表现出色

有监督微调 (SFT)

核心目标：专注于提升模型在**特定任务**上的性能和准确率

训练重点：深度学习特定任务的规则、数据模式和标准输出格式

最终效果：特定任务表现优异，但对未见过的新任务适应性较差

指令微调 vs 提示工程

❑ 本质区别：“改造模型”还是“善用模型”

指令微调 (Instruction Tuning)

通过更新权重，永久改变模型行为

方式 · 知识内化

利用反向传播更新模型权重，将新知识或能力“写入”模型参数中

阶段 · 模型训练期

属于模型构建阶段的操作，一旦训练完成，模型的底层逻辑即永久改变

提示工程 (Prompt Engineering)

通过优化输入，动态引导模型输出

方式 · 逻辑引导

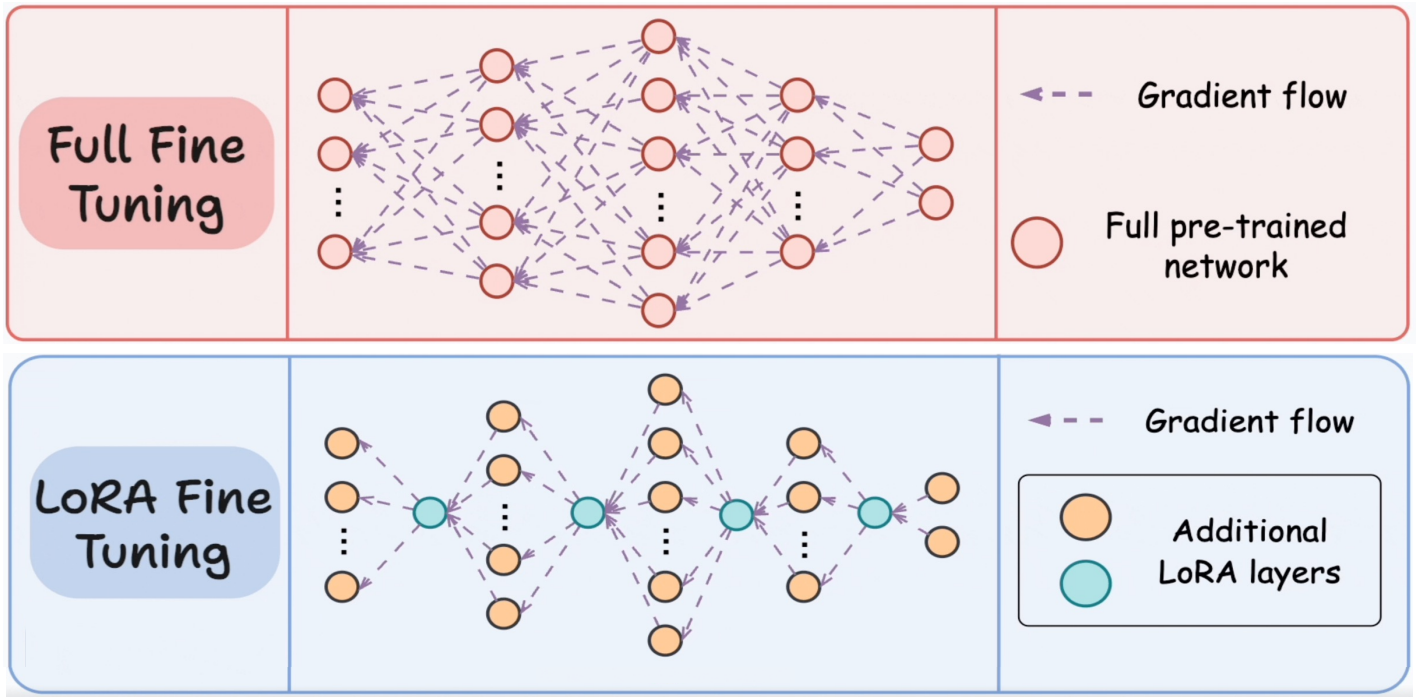
不修改模型参数，在推理阶段通过设计巧妙的指令来“引导”模型思考

阶段 · 模型推理期

属于模型应用阶段的技巧，模型本体保持不变，仅改变外部的输入交互方式

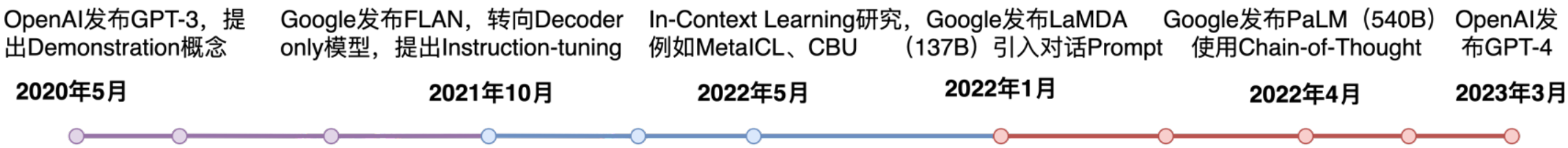
指令微调方法

- 全量微调：在训练过程中更新模型的全部参数
- 参数高效微调：只更新少量新增或特定参数（如LoRA等）



指令微调发展历程

❑ 自2021年Google明确提出指令微调概念以来，该技术迅速发展



1. 初期探索（2020–2021）

- 特点：利用上下文学习，让预训练模型更好地理解任务指令
- 代表工作：T5、GPT3

2. 系统指令微调（2021–2022）

- 特点：通过大规模指令-响应数据训练大模型，提高输出一致性
- 代表工作：FLAN、T0

3. 强化学习微调（2022–至今）

- 特点：结合强化学习、CoT，提升解决复杂问题的能力
- 代表工作：GPT-4

指令数据集的重要性

❑ 高质量的指令数据集是指令微调的基石，核心价值体现在：

决定模型能力上限：数据的质量和多样性直接决定了模型的知识边界与任务泛化能力

影响模型行为模式：低质量或带有偏见的数据会导致模型学习到错误逻辑和有害偏见

对齐人类意图的关键：包含人类偏好的指令数据，是实现人机对齐的核心途径

数据质量与模型性能的平衡关系

数据集格式

- 三个组成部分：I-I-O
  - Instruction：明确告诉模型需要执行的具体任务
  - Input：执行任务所需的上下文信息，可选
  - Output：期望模型生成的回答或结果
- 通常将数据表示为json格式

```
{
    "instruction": "请将下面的字符串反转",
    "input": "Hello World",
    "output": "dlroW olleH"
}
```

数据集质量

❑ 构建高质量数据集需关注以下五个核心维度：

完整性 Integrity

指令和响应在格式和内容上是否完整、清晰且易于被模型理解。

准确性 Accuracy

响应是否真实、客观，且正确地对应指令意图，避免虚构事实。

合理性 Rationality

指令上下文逻辑一致、连贯，推理过程符合常识与逻辑规则。

无害性 Harmlessness

数据内容必须安全合规，严禁包含歧视、暴力、违法、色情等有害信息，确保模型输出的安全性。

有用性 Helpfulness

回答需具备实质性的信息量，能切实解决用户问题，避免答非所问或简单敷衍的情况。

数据集多样性

❑ 数据多样性是提升模型泛化能力和零样本学习能力的核心

1. 任务类型多样性

覆盖摘要、翻译、问答、创作、代码生成等多种任务形式

2. 领域知识多样性

横跨科技、医疗、金融、教育、娱乐等垂直领域

3. 语言多样性

包含多语种语料，提升模型的跨语言理解与生成能力

4. 指令表述多样性

对同一任务目标使用不同的句式、关键词和表达方式进行描述，增强鲁棒性

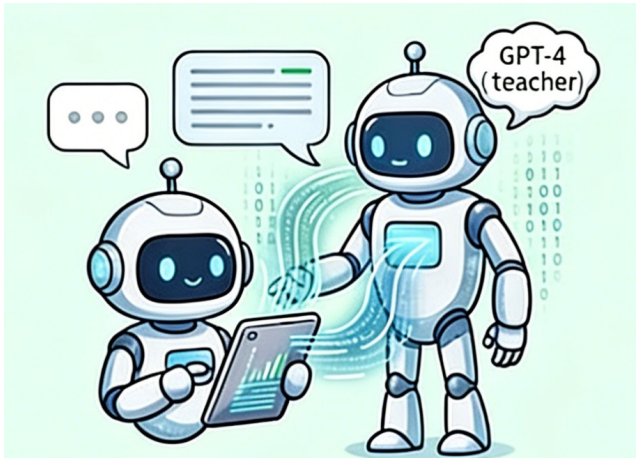
5. 风格多样性

涵盖正式书面语、日常口语、幽默调侃、专业术语等多种文本风格，提升适应性

指令数据集构造三类方法

方式1: 人工标注

由专家或众包人员手动创建高质量的“指令-响应对”，数据精准度最高



方式2: 蒸馏生成

利用强大的教师模型（如GPT-4）自动生成大量指令数据，效率高且成本低

方式3: 自我增强

模型通过自我生成、自我批判和迭代优化指令数据，实现能力的自我提升

人工标注方法

- 由领域专家或众包人员手动创建高质量的“指令-响应对”

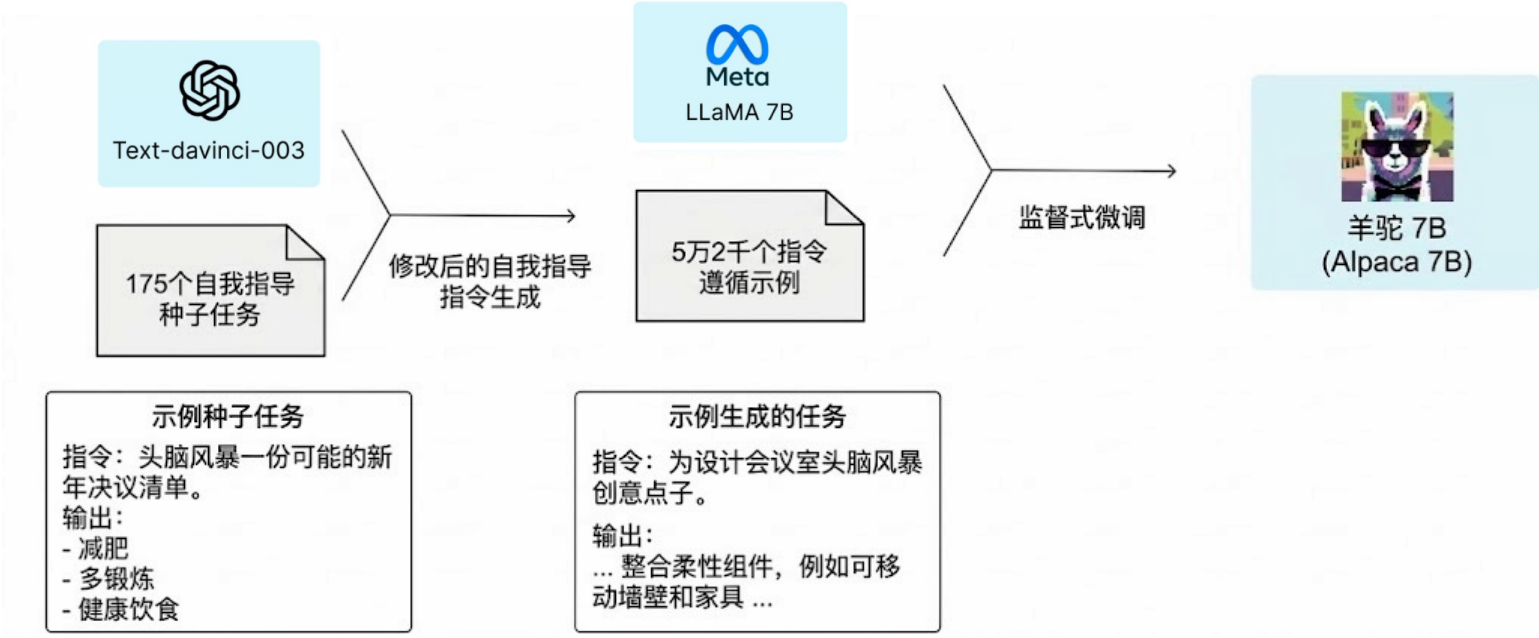
- 早期模型如**Flan**和**InstructGPT**的微调数据均采用此方法构建
- **优点：**数据质量最高，可控性强
- **缺点：**人力和时间成本极高，效率较低。规模难以扩大，难以满足大模型海量数据需求

Type	Dataset Name
Human-Crafted	UnifiedQA (Khashabi et al., 2020)[1]
	UnifiedSKG (Xie et al., 2022)[3]
	Natural Instructions (Honovich et al., 2022)[4]
	Super-Natural Instructions (Wang et al., 2022f)[5]
	P3 (Sanh et al., 2021)[6]
	xP3 (Muennighoff et al., 2022)[7]
	Flan 2021 (Longpre et al., 2023)[8]
	COIG (Zhang et al., 2023a)[9]
	InstructGPT (Ouyang et al., 2022)
	Dolly (Conover et al., 2023a)[22]
	LIMA (Zhou et al., 2023a)[18]
	ChatGPT (OpenAI, 2022)
	OpenAssistant (Köpf et al., 2023)[17]

蒸馏生成方法

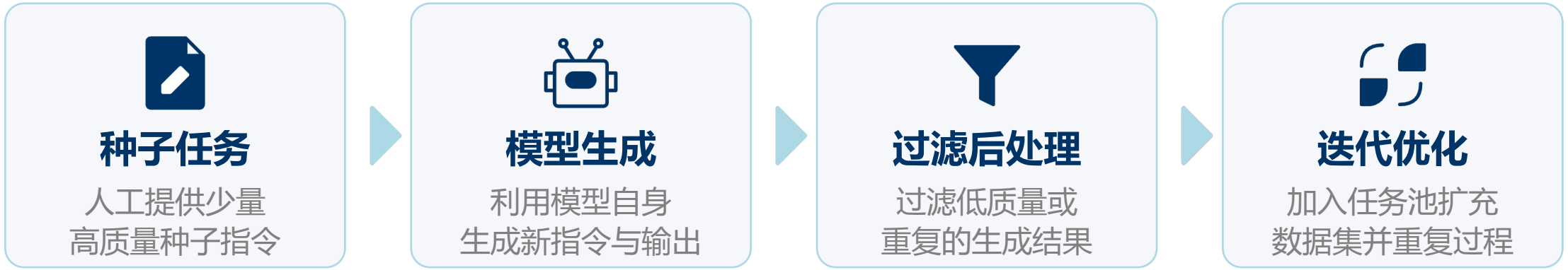
❑ 利用能力更强的“教师模型”，根据提示词自动生成大量“指令-响应对”数据，用于训练“学生模型”

- 优点：成本低、生成快，易规模化，覆盖多领域数据
- 缺点：质量依赖教师模型和提示设计，易引入偏见与错误



自我增强方法

- 模型通过**自我生成、自我批判和迭代优化**指令数据
- 优势：极大降低了对人工标注数据的依赖，实现了从“数据匮乏”到“数据丰富”的高效跨越



主流合成数据集（蒸馏&自我增强）

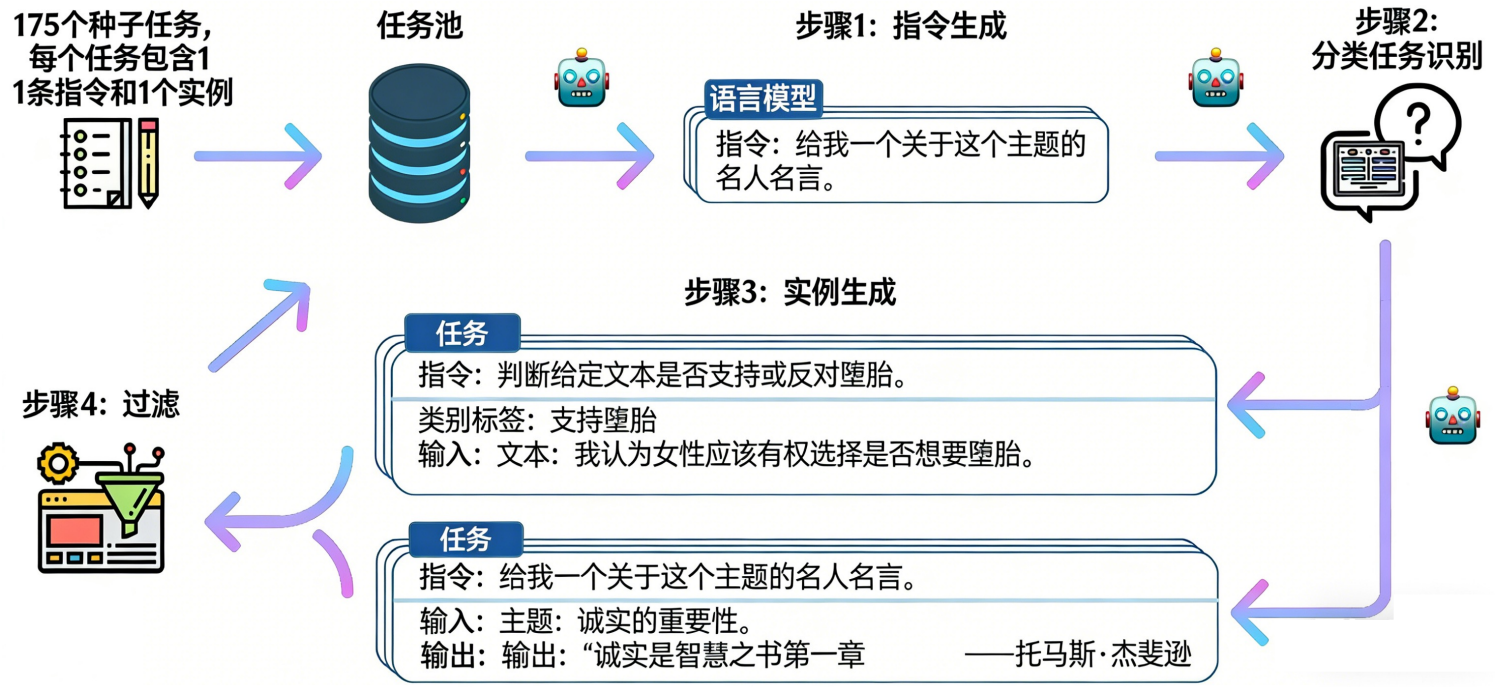
Type	Dataset Name	# of Instances	# of Lang	Construction	Open-source
Synthetic Data (Distillation)	OIG (LAION.ai, 2023)[2]	43M	En	ChatGPT (No technique reports)	Yes
	Unnatural Instructions (Honovich et al., 2022)[10]	240K	En	InstructGPT-Generated	Yes
	InstructWild (Xue et al., 2023)[12]	104K	-	ChatGPT-Generated	Yes
	Evol-Instruct / WizardLM (Xu et al., 2023a)[13]	52K	En	ChatGPT-generated	Yes
	Alpaca (Taori et al., 2023a)[14]	52K	En	InstructGPT-generated	Yes
	LogiCoT (Liu et al., 2023a)[15]	-	En	GPT-4-Generated	Yes
	GPT-4-LLM (Peng et al., 2023)[20]	52K	En&Zh	GPT-4-Generated	Yes
	Vicuna (Chiang et al., 2023)	70K	En	Real User-ChatGPT Conversations	No
	Baize v1 (Conover et al., 2023b)[21]	111.5K	En	ChatGPT-Generated	Yes
	UltraChat (Ding et al., 2023a)[16]	675K	En&Zh	GPT 3/4-Generated	Yes
	Guanaco (JosephusCheung, 2021)[19]	534,530	Multi	GPT (Unknown Version)-Generated	Yes
	Orca (Mukherjee et al., 2023)[23]	1.5M	En	GPT 3.5/4-Generated	Yes
	ShareGPT[24]	90K	Multi	Real User-ChatGPT Conversations	Yes
	WildChat[25]	150K	Multi	Real User-ChatGPT Conversations	Yes
	WizardCoder (Luo et al., 2023)	-	Code	LLaMa 2-Generated	No
	Magicoder (Wei et al., 2023b)[26]	75K/110K	Code	GPT-3.5-Generated	Yes
	WaveCoder (Yu et al., 2023)	-	Code	GPT 4-Generated	No
	Phi-1 (Gunasekar et al., 2023)[27]	6B Tokens	Code Q and A	GPT-3.5-Generated	Yes
	Phi-1.5 (Li et al., 2023i)	-	Code Q and A	GPT-3.5-Generated	No
	Nectar (Zhu et al., 2023a)[28]	183K	En	GPT 4-Generated	Yes
Synthetic Data (Self-Improvement)	Self-Instruct (Wang et al., 2022c)[11]	52K	En	InstructGPT-Generated	Yes
	Instruction Backtranslation (Li et al., 2023g)	502K	En	LLaMa-Generated	No
	SPIN (Chen et al., 2024b)[29]	49.8K	En	Zephyr-Generated	Yes

指令数据集时间线

- 指令数据集经历了从人工模板构建 → 模型生成扩展 →高质量自动化演进的逐步发展过程

Self-Instruct

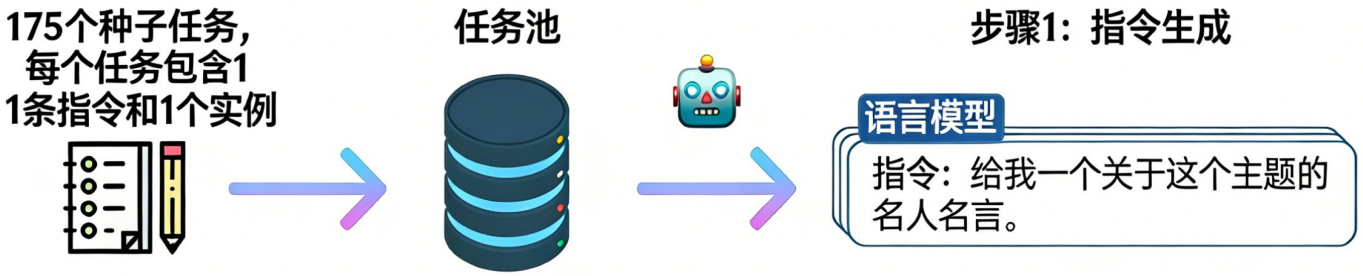
❑ Self-Instruct 的核心是**利用已有大模型生成更多指令-响应对**，形成一个自增强的数据集，用于指令微调



Self-Instruct: Aligning Language Model with Self Generated Instructions. 2022.

Self-Instruct

❑ Self-Instruct包括4个核心步骤：



步骤1：指令生成

- 先人工设计175条任务指令，每条配有（指令, 输入, 输出）或（指令, 输出），作为种子数据
- 然后从种子池随机抽取6条人工指令，再加2条之前生成的指令，共8条，按表6模板输入模型，生成新的指令

Self-Instruct: Aligning Language Model with Self Generated Instructions. 2022.

Self-Instruct

❑ Self-Instruct包括4个核心步骤：

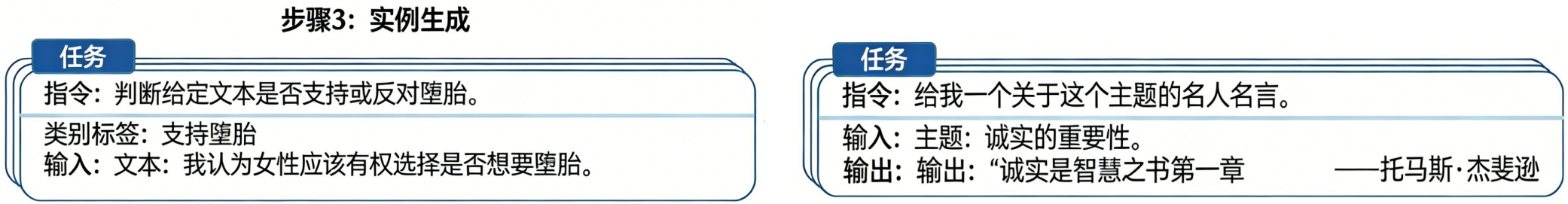
步骤2：分类任务识别

- 分类任务与非分类任务使用的prompt模版是不同的
- 从种子池随机选12条分类指令和19条非分类指令，加上新生成的指令，让模型判断新指令是否属于分类任务

Self-Instruct: Aligning Language Model with Self Generated Instructions. 2022.

Self-Instruct

❑ Self-Instruct包括4个核心步骤：



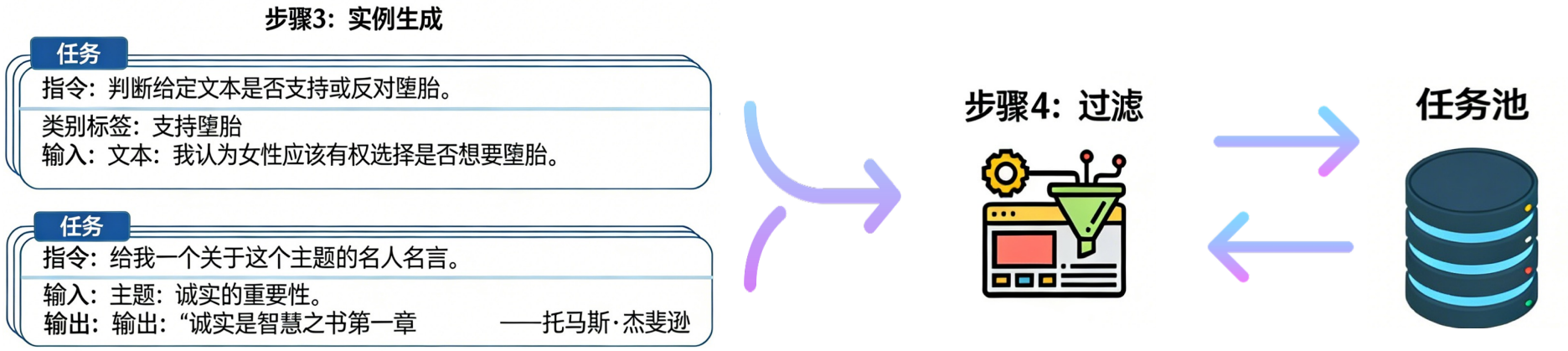
步骤3：实例生成

- 在给定指令之后，生成（输入, 输出）这个实例对时还有两种策略。
- 一种是先生成输入，后生成输出，即输入优先策略；另一种是先生成输出，后生成输入，即输出优先策略。

Self-Instruct: Aligning Language Model with Self Generated Instructions. 2022.

Self-Instruct

❑ Self-Instruct包括4个核心步骤：



步骤4：过滤

- **多样性：**新指令仅在与种子池指令的 ROUGE-L 小于 0.7 时才加入
- **适配性：**排除图像类无法处理的指令
- **正确性：**去掉输入相同但输出不同的实例

Self-Instruct: Aligning Language Model with Self Generated Instructions. 2022.

Self-Instruct

- 通过Self-Instruct方法生成**Alpaca数据集**

statistic	
# of instructions	52,445
- # of classification instructions	11,584
- # of non-classification instructions	40,861
# of instances	82,439
- # of instances with empty input	35,878
ave. instruction length (in words)	15.9
ave. non-empty input length (in words)	12.7
ave. output length (in words)	18.9

数据统计信息：总共生成了**52K**的指令，和**82K**的实例

数据多样性：分析最常用的动词-名词组合，反映了数据在意图和形式上的多样性

覆盖了写作、创作、查找、分类等几十种任务类型

Self-Instruct: Aligning Language Model with Self Generated Instructions. 2022.

Self-Instruct

❑ 数据集的实验结果

	Model	# Params	ROUGE-L
	Vanilla LMs		
	T5-LM	11B	25.7
	GPT3	175B	6.8
	Instruction-tuned w/o SUPERNI		
①	T0	11B	33.1
	GPT3 + T0 Training	175B	37.9
②	$GPT3_{SELF-INST}$ (Ours)	175B	39.9
②	$InstructGPT_{001}$	175B	**40.8**
	Instruction-tuned w/ SUPERNI		
	T*k*-INSTRUCT	11B	46.0
③	GPT3 + SUPERNI Training	175B	49.5
③	$GPT3_{SELF-INST}$ + SUPERNI Training (Ours)	175B	**51.6**

在数据集 SUPERNI 上的效果对比

实验结论：

① Self-Instruct能够给GPT3模型带来巨大的提升，大概33.1%

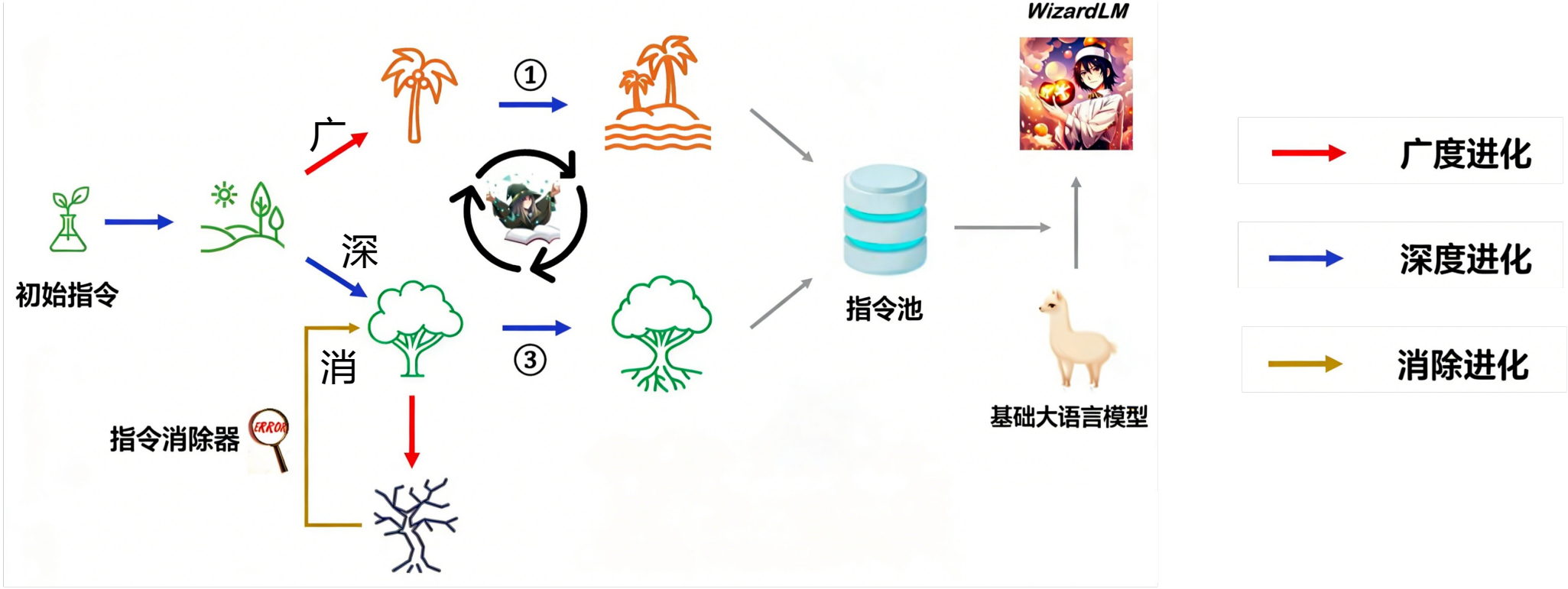
② 经过Self-Instruct之后的GPT3的效果接近 InstructGPT001

③ 对比最后两行，即使已经在同源的测评数据集 SUPERNI 上经过了微调之后，再使用 Self-Instruct 依然能够有提升

Self-Instruct: Aligning Language Model with Self Generated Instructions. 2022.

Evol-Instruct：指令进化

- Evol-Instruct一种用于自动构建高质量指令数据的进化式方法，旨在提升**数据多样性**与**复杂度，重点在数据质量而不是数量**



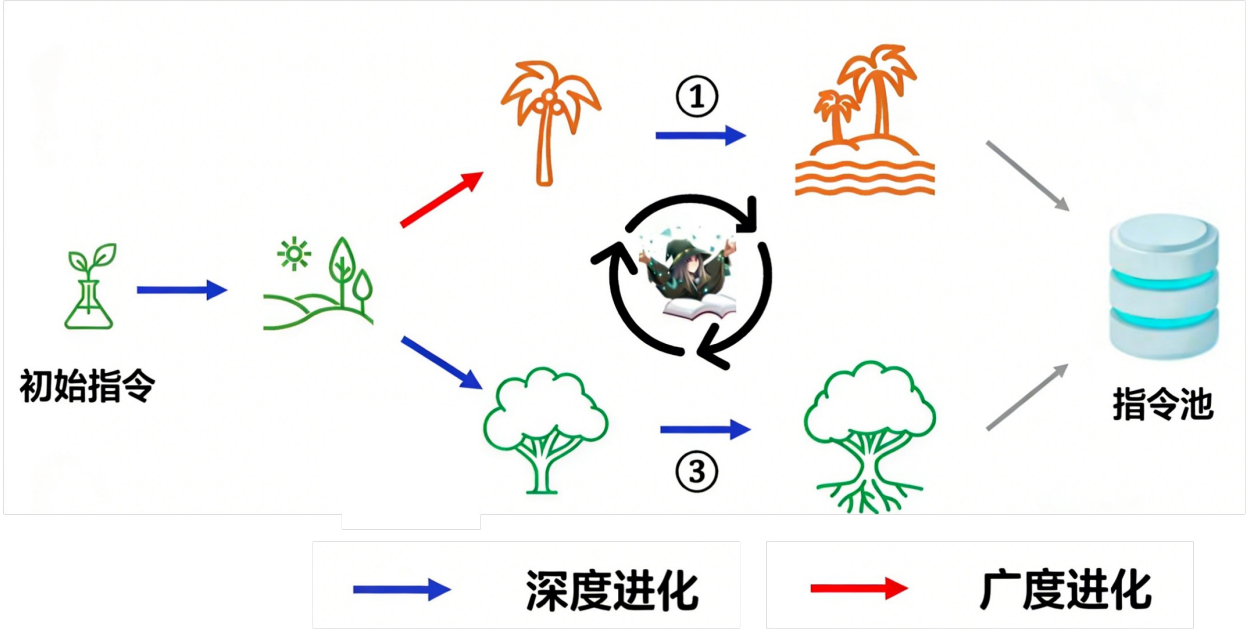
WizardLM: Empowering large pre-trained language models to follow complex instructions. 2023.

Evol-Instruct

❑ Evol-Instruct包含三个步骤：指令进化、响应生成、淘汰式进化

步骤1：指令进化

- 每一轮进化中，使用 LLM 在现有指令基础上：
  1. 深度进化：提升指令难度
  2. 广度进化：增加指令多样性
- **成功进化** → 新指令加入数据池
- **进化失败** → 原指令保留，下一轮重新处理



WizardLM: Empowering large pre-trained language models to follow complex instructions. 2023.

Evol-Instruct

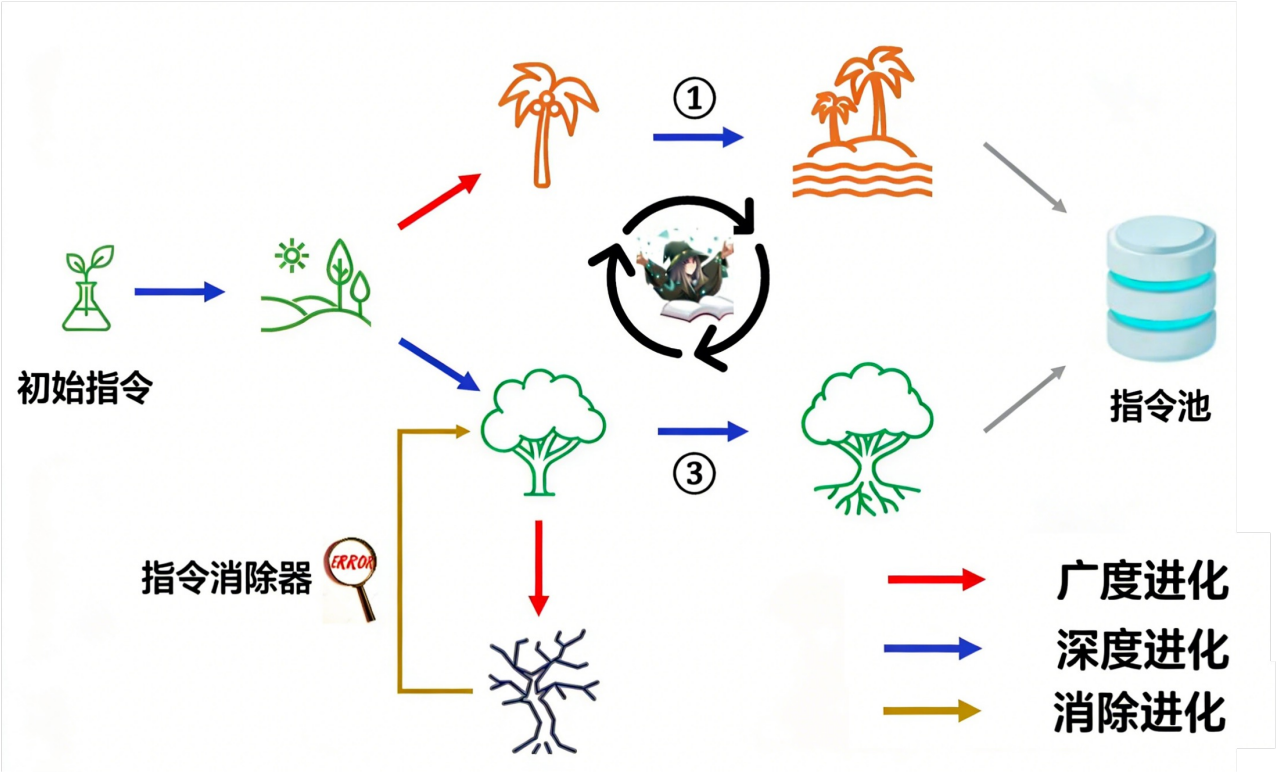
❑ Evol-Instruct包含三个步骤：指令进化、响应生成、淘汰式进化

步骤2：响应生成

- 将进化后的指令输入 LLM，生成高质量响应

步骤3：淘汰式进化

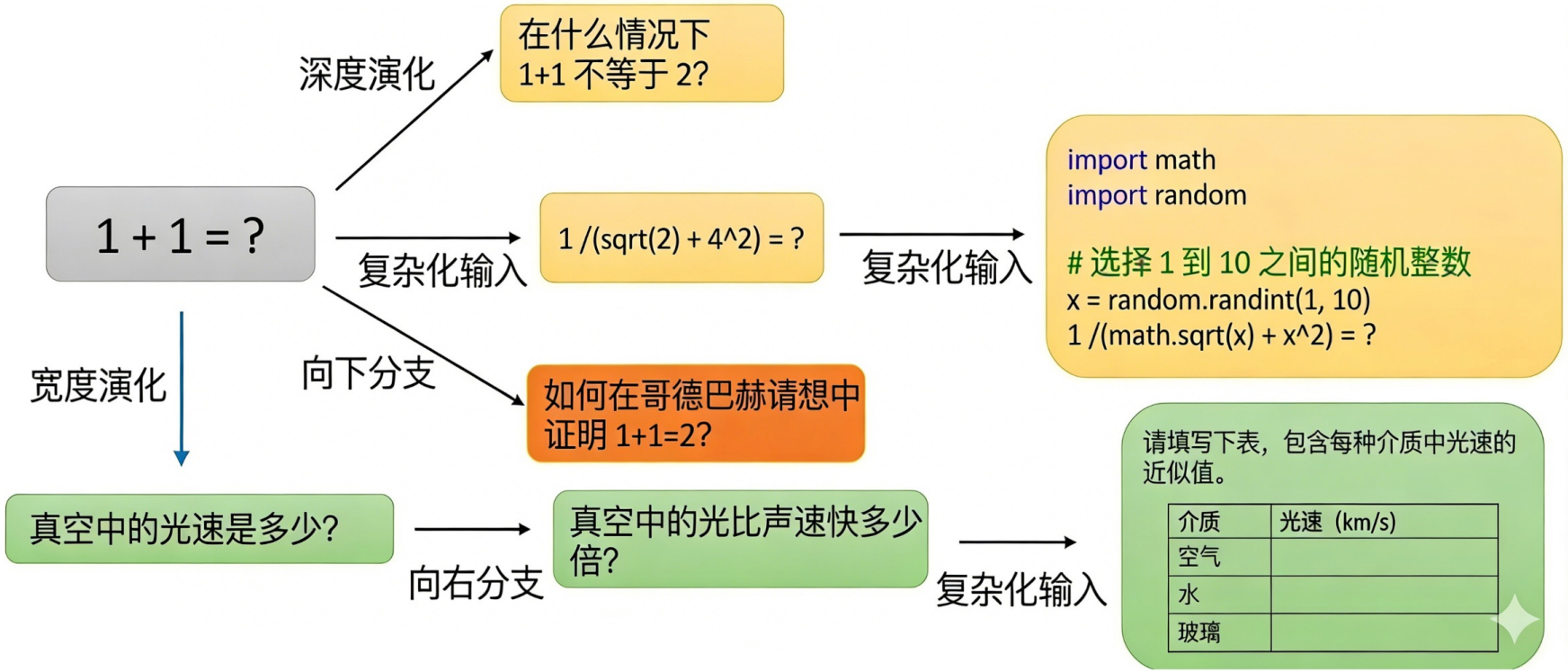
- 筛选不合格的指令数据，防止数据污染，例如LLM无法生成响应



WizardLM: Empowering large pre-trained language models to follow complex instructions. 2023.

Evol-Instruct

❑ Evol-Instruct指令进化的示例展示



WizardLM: Empowering large pre-trained language models to follow complex instructions. 2023.

Evol-Instruct

❑ 数据集分析及实验结果

以 Alpaca 的5万条数据为初始数据，经过四轮 ChatGPT 指令进化，生成 25 万条指令，并微调 LLaMA 得到 WizardLM 模型。

Model	Avg.	MMLU	ARC	HellaSwag	TruthfulQA	HumanEval	GSM8k	AlpacaEval	MT-Bench	WizardEval
ChatGPT-3.5	76.15	70.0	85.2	85.5	47.0	48.1	80.8	89.37	7.94	100.0
Alpaca-13b	43.44	46.63	51.20	76.31	41.62	9.2	8.35	33.25	4.78	76.6
Vicuna-13b	54.60	50.84	51.71	79.94	**52.68**	12.5	24.34	70.43	6.21	86.9
Baize-13b	51.46	49.72	56.91	79.29	47.88	14.6	8.95	66.96	5.75	81.3
CAMEL-13b	51.29	49.74	55.63	79.25	47.42	17.7	7.13	64.84	5.78	82.1
Tulu-13b	52.46	**53.19**	53.92	80.66	43.84	21.3	36.50	45.34	5.76	79.8
WizardLM-13b	**58.96**	52.92	**57.25**	**80.88**	50.55	**24.0**	**37.15**	**75.31**	**6.35**	**89.1**

Table 1: Performance comparison of ChatGPT-3.5, open-source baselines, and WizardLM-13b.

WizardLM: Empowering large pre-trained language models to follow complex instructions. 2023.

LIMA：少即是多

- LIMA探究是否可以仅依赖极少量高质量数据，就实现与现有商业对齐模型相当的效果?

LLaMA-Base

少量高质量数据

指令微调SFT

LLaMA++

LIMA: Less Is More for Alignment. 2023.

LIMA：少即是多

- 只用 1,000 条精心挑选的 prompt – response 示例，在LLaMa-65B基础模型上进行指令微调，不使用强化学习（例如RLHF）

Source	#Examples	Avg Input Len.	Avg Output Len.
Training			
Stack Exchange (STEM)	200	117	523
Stack Exchange (Other)	200	119	530
wikiHow	200	12	1,811
Pushshift r/WritingPrompts	150	34	274
Natural Instructions	50	236	92
Paper Authors (Group A)	200	40	334
Dev			
Paper Authors (Group A)	50	36	N/A
Test			
Pushshift r/AskReddit	70	30	N/A
Paper Authors (Group B)	230	31	N/A

LIMA: Less Is More for Alignment. 2023.

LIMA：少即是多

- LIMA证明了**“少即是多”的原则，** 少量高质量、高多样性的数据
  - 海量低质量数据

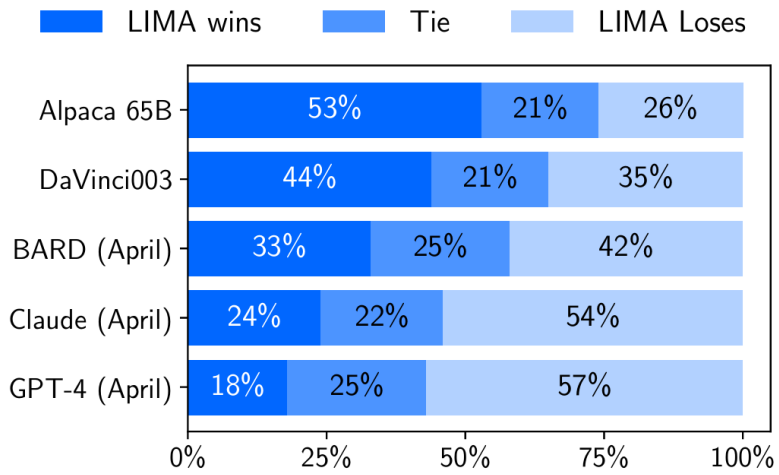


Figure 1: Human preference evaluation, comparing LIMA to 5 different baselines across 300 test prompts.

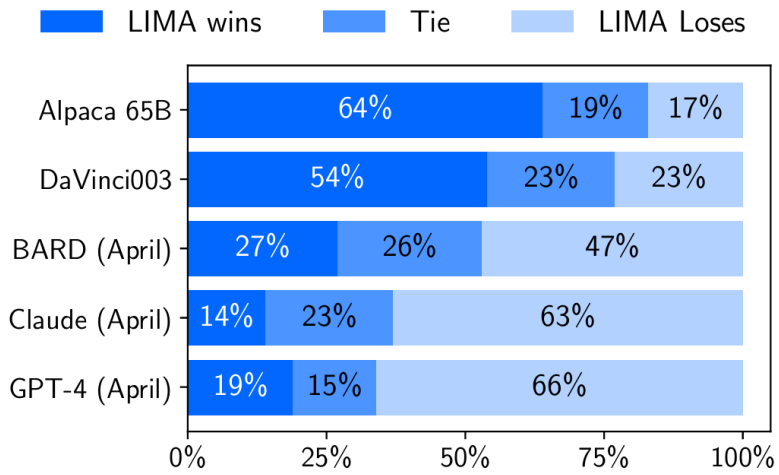


Figure 2: Preference evaluation using GPT-4 as the annotator, given the same instructions provided to humans.

LIMA: Less Is More for Alignment. 2023.

本节复习

- 全量微调
- 高效参数微调：Adapter、LoRA及变体
- 指令微调 Instruction Tuning
- 指令数据集构造：Self-Instruct、Evol-Instruct

参考文献

- A Survey of Large Language Models. 2023.
- A Survey on Post-training of Large Language Models. 2025.
- LIMA: Less Is More for Alignment. 2023.
- LoRA: Low-Rank Adaptation of Large Language Models. 2021.
- Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning. 2023.

致 谢

- 胡玥、曹亚男、方芳：国科大《自然语言处理基础》
- 曹亚男、任昱冰：国科大《深度学习与自然语言处理概述》


THANKS
https://ictkc.github.io/teaching/2026spring-nlp