



DaSE
Data Science
& Engineering

指令微调与人类对齐

梁远远

2024-06-03

深度学习课程

内容目录

一、指令微调

- 指令微调的定义与作用
- 指令微调与传统微调的对比
- 指令微调的流程
- 指令数据构造方法
- 指令微调的策略

二、人类对齐

- 人类对齐的概念与目的
- 人类对齐的原则：3H标准
- 人类反馈强化学习算法（RLHF）
- 非RL方法的对齐方法

思考？

问题一：大语言模型经过预训练已经学到了充分的知识，为什么还需要指令微调？

问题二：指令微调是大语言模型独有的嘛？小语言模型也需要吗？

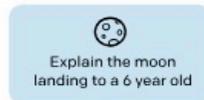
问题三：人类对齐要解决的什么大语言模型的什么问题？

InstructGPT-大模型对齐人类指令的步骤

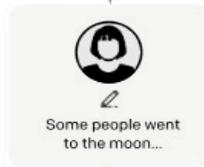
Step 1

Collect demonstration data, and train a supervised policy.

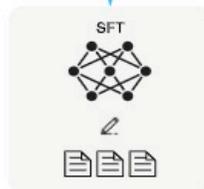
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



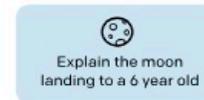
This data is used to fine-tune GPT-3 with supervised learning.



Step 2

Collect comparison data, and train a reward model.

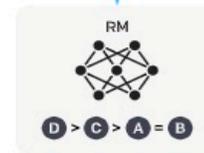
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



Step 3

Optimize a policy against the reward model using reinforcement learning.

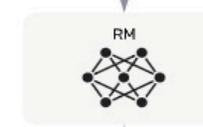
A new prompt is sampled from the dataset.



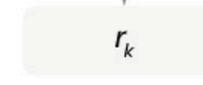
The policy generates an output.



The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.



[1]Training language models to follow instructions with human feedback.(2022)

指令微调的定义

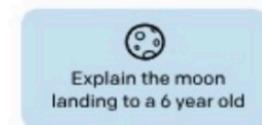
指令微调（Instruction Tuning）是22年谷歌ICLR论文[1]中提出这个概念，指使用自然语言形式的数据对已经预训练好的大型语言模型进行参数微调，使模型适应特定任务或领域。微调主要目的是，完成知识注入、指令对齐。指令微调也被称为有监督微调（Supervised Fine-tuning）或多任务提示训练（Multitask Prompted Training）。

[1]Finetuned language models are zero-shot learners.(2021).

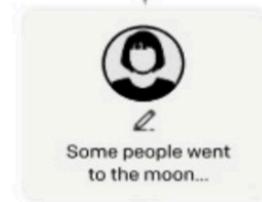
Step 1

**Collect demonstration data,
and train a supervised policy.**

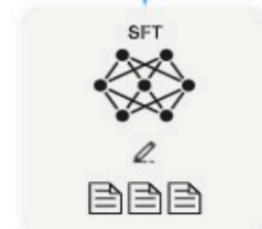
A prompt is
sampled from our
prompt dataset.



A labeler
demonstrates the
desired output
behavior.

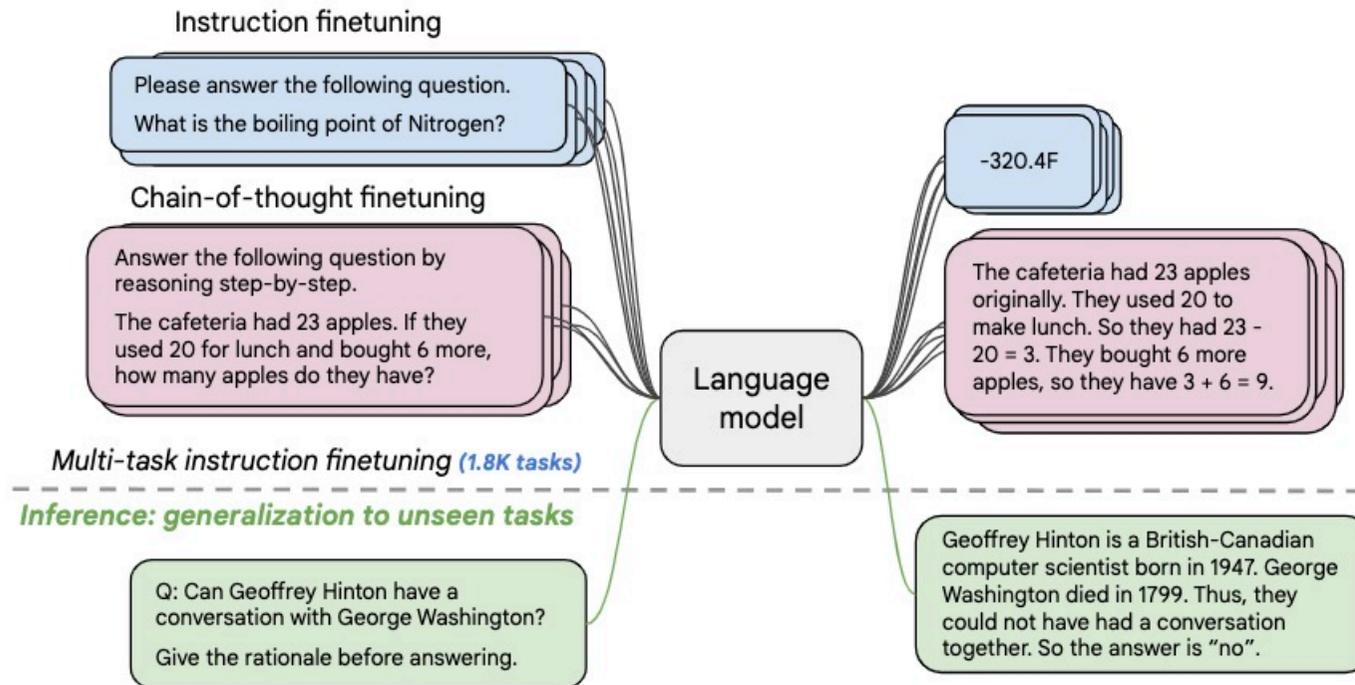


This data is used
to fine-tune GPT-3
with supervised
learning.



指令微调的作用

经过指令微调后，大语言模型能够展现出较强的指令遵循能力（Instruction Following），可以通过零样本学习的方式解决多种下游任务,同时也能够解决其他未见过的NLP任务。（知识来自于pretrain阶段，指令微调通常只不过是为了让LLM显示地输出已具备知识。）



[1]Scaling Instruction-Finetuned Language Models.(2022).

指令微调与传统微调的对比

在传统微调中，语言模型预测样本的标签，而在指令微调中，语言模型回答指令集中的问题。两者之间的差别：

一、目标

传统微调：在特定任务或数据集上进一步优化预训练模型的性能。

指令微调：使模型能够更好地理解和执行自然语言指令。

二、数据集

传统微调：通常使用单一任务的专用数据集，例如情感分析、命名实体识别等。数据量可能较小，但高度特定于目标任务。

指令微调：使用包含各种任务和指令的大规模多任务数据集。这些数据集涵盖了多个领域和任务类型。数据量通常非常大，涵盖广泛的指令和任务示例。

三、应用场景

传统微调：适用于需要在特定任务上获得最佳性能的应用，例如医学影像分析、语音识别等。

指令微调：适用于需要处理多种任务和指令的自然语言处理应用，例如智能助理、聊天机器人等

传统微调与指令微调的对比

NLI

Original sample:

Text 1: "No Weapons of Mass Destruction Found in Iraq Yet. "

Text 2: "Weapons of Mass Destruction Found in Iraq. "

Label: $1 \in \{0,1\}$ ("not entailment")

NLI

Instruction + Text:

No Weapons of Mass Destruction Found in Iraq Yet. Question: Does this imply that "Weapons of Mass Destruction Found in Iraq."? Yes or no?
Answer: No

NLI:自然语言推理

SC

Original sample:

Premise: "My body cast a shadow over the grass. " Question: cause

Choice 1: "The sun was rising. "

Choice 2: " The grass was cut. "

Label: $0 \in \{0,1\}$ ("Choice 1")

SC

Instruction + Text:

My body cast a shadow over the grass.
This happened because...
Help me pick the more plausible option:
- The sun was rising.- The grass was cut.
Answer: The sun was rising.

SC:句子分类

指令微调的流程

1、准备数据集

a. 收集多任务数据

收集包含多种任务的数据集，例如文本分类、翻译、摘要生成、问答等。

确保数据集中每个任务都附有清晰的自然语言指令，这些指令可以描述任务本身或具体要求。

b. 数据预处理

对数据进行清洗和格式化，确保所有指令和任务数据的一致性。

将任务和指令组合成训练样本，确保模型能够理解指令并执行相应任务。

2、加载预训练模型

选择一个预训练语言模型作为基础模型，例如Llama、Baichuan、Chatglm等。

加载预训练模型的参数，作为指令微调的起点。

3、进行微调训练

将每个任务的数据转换为模型可以处理的格式。例如，将指令和输入文本拼接在一起作为模型的输入。

然后使用传统的微调的方法进行训练和更新模型参数。

4、评估与调优

使用独立的测试集对微调后的模型进行最终评估。

确保模型在未见过的数据上有良好的泛化性能。

指令数据的构造方法

指令数据的格式：

指令-必要部分，描述任务的定义和要求。

输入-可选部分，任务的输入，可以不存在。

输出-必要部分，描述任务的输出label。

```
Instruction: Given an address and city, come up with the zip code.  
Input:  
Address: 123 Main Street, City: San Francisco  
Output: 94105
```



```
Instruction: I am looking for a job and I need to fill out an application form. Can you please help me complete it?  
Input:  
Application Form:  
Name: _____ Age: _____ Sex: _____  
Phone Number: _____ Email Address: _____  
Education: _____ ...  
Output:  
Name: John Doe Age: 25 Sex: Male  
Phone Number: ...
```



问答任务

- 指令：`"回答以下问题："`
- 输入：`"Python是一种什么样的编程语言？"`
- 输出：`"Python是一种高级编程语言。"`

翻译任务

- 指令：`"将以下句子从英语翻译成法语："`
- 输入：`"Hello, how are you?"`
- 输出：`"Bonjour, comment ça va?"`

训练过程

- 输入：`"回答以下问题： Python是一种什么样的编程语言？"`
输出：`"Python是一种高级编程语言。"`
- 输入：`"将以下句子从英语翻译成法语： Hello, how are you?"`
输出：`"Bonjour, comment ça va?"`

指令数据的构造方法

- 基于现有NLP任务数据集构建

因为现有的NLP任务大多是{input,output}的格式，所以把现有的NLP任务数据集添加instruction就构成了指令数据集。

- 基于日常对话数据构建

日常对话有问有答，基本是符合指令+input+output的格式，所以可以很方便日常对话数据中提取有意义的指令数据。

- 基于合成的指令数据构建

比较典型的有：

Self-Instruct

Evol-Instruct

UltraChat等等

基于合成的指令数据构建-Self-Instruct

Self-Instruct 动机：

人工标注指令数据集一般有两个步骤：

- 1.创造各种新颖的任务，即 instruction；
- 2.对上述每个任务编写正确的答案，即 complement；

这种人工标注数据的方式成本很高，而且很倾向于主流的NLP任务，没有涵盖真正的各种任务和描述它们的不同方式。鉴于这些限制，就需要继续提高指令调整模型的质量。可以借助大语言模型（例如 ChatGPT）所具备的数据合成能力，通过迭代的方法高效地生成大量的指令微调数据。Self-Instruct方法由此而来，是一种有效的模型蒸馏的方法。

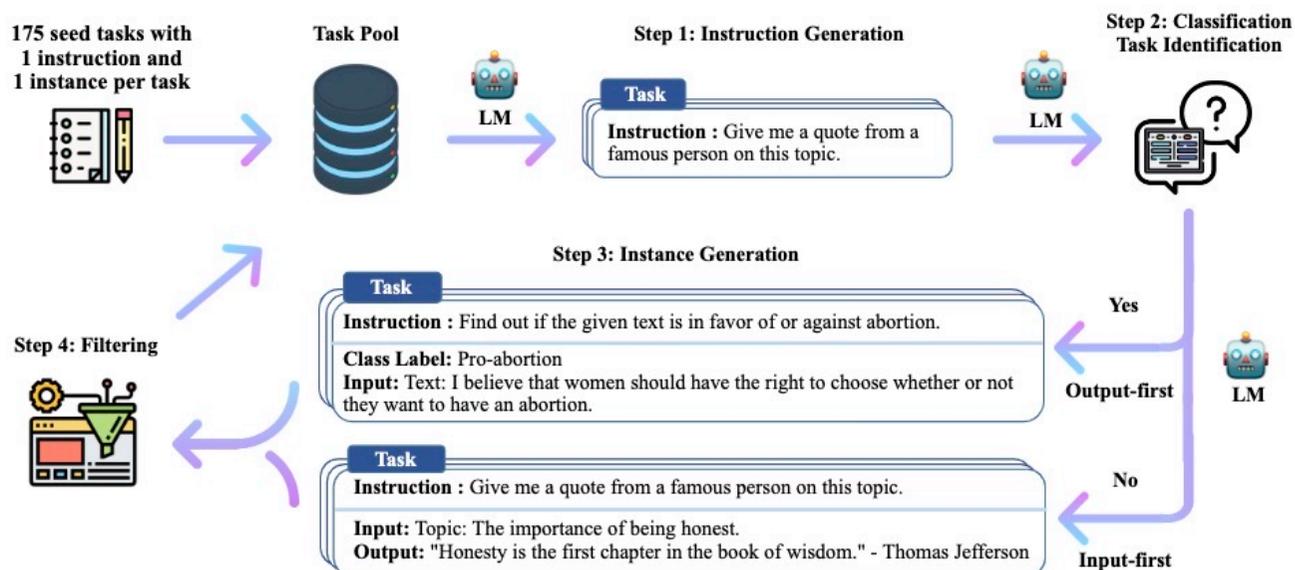
[1]Self-Instruct: Aligning Language Model with Self Generated Instructions. (2022)

基于合成的指令数据构建-Self-Instruct

该方法首先构建了 175 条高质量且多样的指令数据作为初始任务池，之后经由两个主要步骤生成指令微调数据。

(1) 指令数据生成：从任务池中随机选取少量指令数据作为示例，并针对 ChatGPT 设计精细指令来提示模型生成新的指令数据。

(2) 过滤与后处理：剔除低质量或者重复的生成实例，从而保证指令数据的多样性与有效性。常见的过滤方法包括：去除与任务池中指令相似度过高的指令、语言模型难以生成回复的指令、过长或过短的指令以及输入或输出存在重复的实例。



基于合成的指令数据构建-Self-Instruct

Self-Instruct方法中文命令实例如下：

```
你被要求提供 10 个多样化的任务指令。这些任务指令将被提供给 GPT 模型。  
以下是你提供指令需要满足的要求：  
1. 尽量不要在每个指令中重复动词，要最大化指令的多样性。  
2. 使用指令的语气也应该多样化。例如，将问题与祈使句结合起来。  
.....（省略后续要求）  
下面是 10 个任务指令的列表：  
### 指令：将 85 华氏度转换为摄氏度。  
### 输出：85 华氏度等于 29.44 摄氏度。  
### 指令：是否有科学无法解释的事情？  
### 输出：有很多科学无法解释的事情，比如生命的起源、意识的存在.....  
.....（省略上下文示例）
```

基于合成的指令数据构建-Self-Instruct

数据质量评估：

随机抽取200条指令，并给每个指令随机抽取一个实例，然后人工对该指令和实例进行标注评估，评估结果如右表所示。

可以看出：

- ✓ 生成的指令有含义，能表示有效一个任务的占比为92%；
- ✓ 给每个指令生成合适的输入的占比为79%；
- ✓ 生成的输出是指令和输入的正确结果的占比为58%；
- ✓ 指令、输入、输出，这三个字段全对的占比为54%；

由此可得，使用Self-Instruct生成的数据集还是有一些噪音的，另外Self-Instruct 生成的实例可能过于简单或缺乏多样性。

| Quality Review Question | Yes % |
|---|-------|
| Does the instruction describe a valid task? | 92% |
| Is the input appropriate for the instruction? | 79% |
| Is the output a correct and acceptable response to the instruction and input? | 58% |
| All fields are valid | 54% |

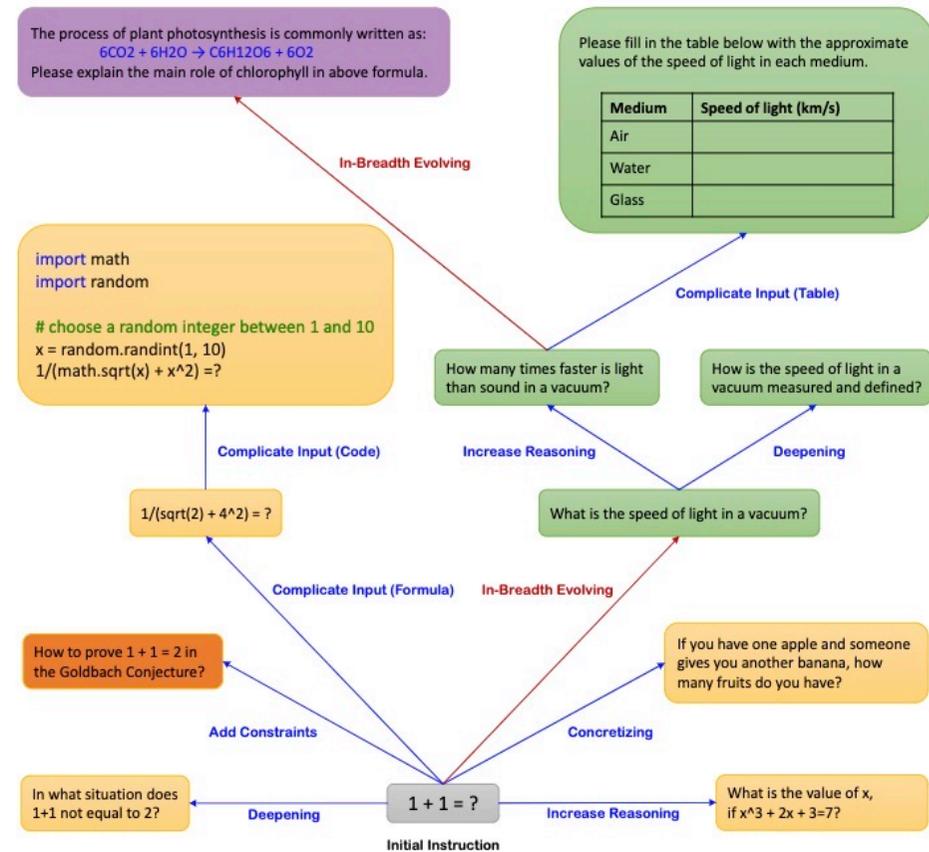
基于合成的指令数据构建-Evol-Instruct

Self-Instruct 生成的实例可能过于简单或缺乏多样性，Evol-Instruct提出一种改进的指令数据合成方法。该方法通过基于深度和广度的演化来提高实例的复杂性和多样性。

Evol-Instruct 演化过程：

(1) 指令演化：分为深度演化和广度演化。深度演化通过五种特定类型的提示（添加约束、深化、具体化、增加推理步骤以及使输入复杂化）使得指令变得更加复杂与困难；而广度演化旨在扩充指令主题范围、指令涉及的能力范围以及整体数据集的多样性。

(2) 数据后处理：主要使用了如下的规则进行处理：使用 ChatGPT 比较演化前后的指令，移除 ChatGPT 认为差异很小的指令；移除大模型难以响应的指令，如响应中包含“sorry”或响应长度过短；移除仅包含标点符号和连词的指令或回复。



[1] WizardLM: Empowering Large Language Models to Follow Complex Instructions. (2023)

基于合成的指令数据构建-Evol-Instruct

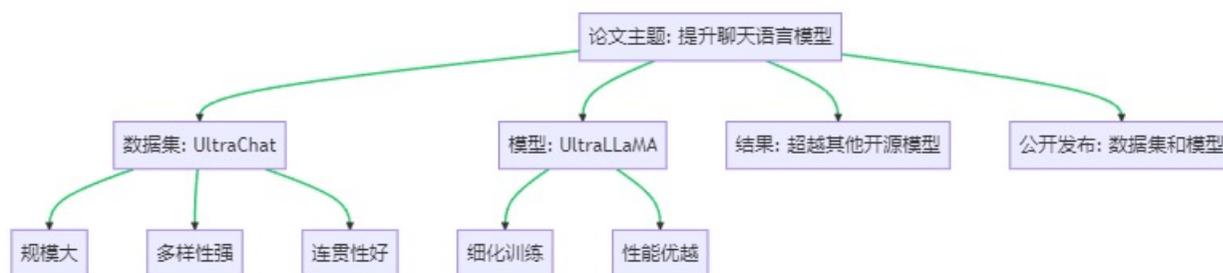
中文命令实例如下：

```
我希望您充当指令重写器。  
您的目标是将给定的提示重写为更复杂的版本，使得著名的 AI 系统（例如 ChatGPT 和 GPT-4）更难处理。  
但重写的提示必须是合理的，且必须是人类能够理解和响应的。  
您的重写不能省略 # 给定提示 # 中表格和代码等非文本部分。  
您应该使用以下方法使给定的提示复杂化：  
请在 # 给定提示 # 中添加一项约束或要求。  
你应该尽量不要让 # 重写提示 # 变得冗长，# 重写提示 # 只能在 # 给定提示 # 中  
添加 10 到 20 个单词。  
# 重写提示 # 中不允许出现“# 给定提示 #”、“# 重写提示 #”字段。  
# 给定提示 #：{需要重写的指令}  
# 重写提示 #
```


基于合成的指令数据构建-UltraChat

UltraChat 动机：

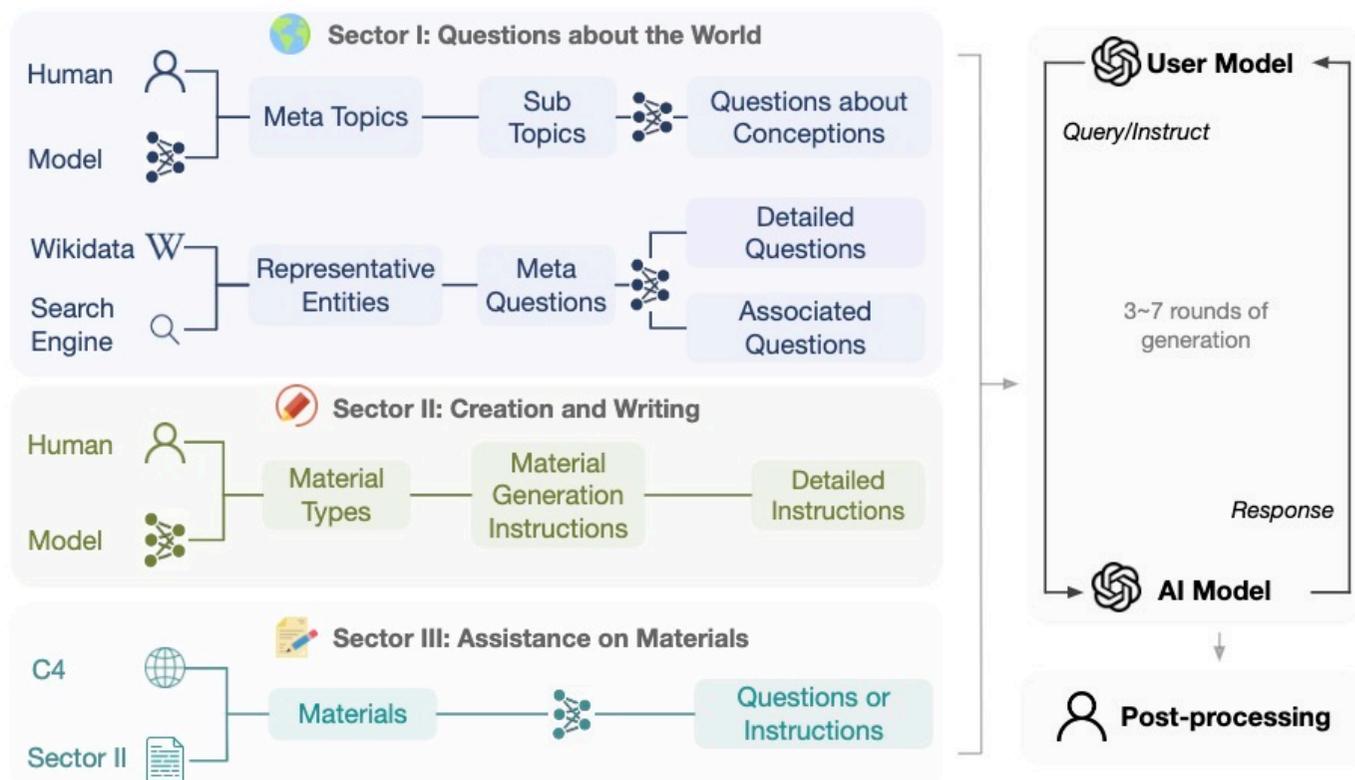
这个工作旨在提高开源模型的性能上限，提供了一个系统设计的、多样化的、信息丰富的、大规模的教学对话数据集UltraChat。UltraChat包含150万个高质量的多轮对话，并涵盖了广泛的主题和指令。UltraChat的统计分析揭示了其在尺度、平均长度、多样性、一致性等各种关键指标上的优势，巩固了其作为领先的开源数据集的地位。



[1]Enhancing Chat Language Models by Scaling High-quality Instructional Conversations. (2023)

基于合成的指令数据构建-UltraChat

UltraChat的总体思路是使用单独的LLM来生成开场白、模拟用户和响应查询。UltraChat的三个方案：关于世界的问题、写作和创作、对现有材料的协助都有特点的设计，如下图：



基于合成的指令数据构建-UltraChat

关于世界的问题

这部分数据主要关注的是现实世界中存在的概念、对象和实体。请求ChatGPT生成30个涵盖我们日常生活各个方面，具有代表性和多样性的元主题，如下图：

| | | |
|---|---|--|
|  Technology |  Health and wellness |  Travel and adventure |
|  Food and drink |  Art and culture |  Science and innovation |
|  Fashion and style |  Relationships and dating |  Sports and fitness |
|  Nature and the environment |  Music and entertainment |  Politics and current events |
|  Education and learning |  Money and finance |  Work and career |
|  Philosophy and ethics |  History and nostalgia |  Social media and communication |
|  Creativity and inspiration |  Personal growth and development |  Spirituality and faith |
|  Pop culture and trends |  Beauty and self-care |  Family and parenting |
|  Entrepreneurship and business |  Literature and writing |  Gaming and technology |
|  Mindfulness and meditation |  Diversity and inclusion |  Travel and culture exchange |

构建过程：

- 首先，根据这些元主题生成了1100多个子主题；同时从维基数据中收集了最常用的10,000个现实世界的命名实体，比如人物、地点、事件等。
- 再为每个子主题设计了最多10个具体的问题；每个实体设计了5个基本问题，10个具体问题和20个扩展问题。
- 然后使用Turbo API为10个问题中的每一个生成新的相关问题。想用这些问题来创建对话，所以从大约500,000个问题中筛选和抽样了一些作为对话的开头。
- 最后对200k个特定问题和250k个一般问题以及50k个元问题进行采样，并迭代地生成多轮的对话。

基于合成的指令数据构建-UltraChat

关于写作和创作

这部分的目的是根据用户的指示，自动生成不同类型的写作用文本，使用ChatGPT使其根据用户的指示，生成20种不同类型的写作用文本，比如故事、诗歌、论文等。如下图：

| | | |
|--|---|---|
|  Articles and Blog Posts |  Job Application Material |  Stories |
|  Legal Documents and Contracts |  Poems |  Educational Content |
|  Screenplays |  Scripts for Language Learning |  Technical Documents and Reports |
|  Marketing Materials |  Social Media Posts |  Personal Essays |
|  Emails |  Scientific Papers and Summaries |  Speeches and Presentations |
|  Recipes and Cooking Instructions |  News Articles |  Song Lyrics |
|  Product Descriptions and Reviews |  Programs and Code | |

构建过程：

- 对于每种类型的写作，生成200条不同的prompt，让AI助手生成文本材料，其中80%的指令被进一步扩展和细化。
- 将生成的指令作为初始输入，分别生成2-4轮的对话。

基于合成的指令数据构建-UltraChat

关于对现有材料的协助

这部分的目的是根据现有的文本材料，生成不同类型的任务，比如改写、翻译、总结等。使用的Template如下：

Templates for concatenation

```
{text}\n{instruction}
{text} {instruction}
{instruction} Answer according to: {text}
{text} Based on the passage above, {instruction}
{instruction}: {text}
Given the text: {text}\n{instruction}
{instruction}\nGenerate according to: {text}
```

构建过程：

- 从C4数据集中提取了约10w种不同的材料。为每种类型设计了一些关键字。
- 用ChatGPT为每份材料生成最多5个问题说明。
- 将每个问题指令的材料与一组手动设计的模板结合起来，作为用户的初始输入，开始与 AI 助手的对话。
- 得到了50万个对话开头，每个对话开头包含了一个文本片段和一个任务指令。
- 对于每个输入，生成 2~4 轮对话。

基于合成的指令数据构建-UltraChat

数据质量评估

UltraChat数据集是一个大规模的多模态对话数据集，它包含了超过100万个对话，每个对话平均包含3.8轮对话。其中不仅包含了文本信息，还包含了音频、视频和屏幕共享数据。

UltraChat与其他几个指令数据集进行统计分析比较，结果下表所示。

| Dataset | #Dialogue | Avg. #Turns | Avg. Dialog Length (by token) | Avg. Utt. Length (by token) | Lexical Diversity (↑) | Topic Diversity (↓) | Coherence (↑) | User Simulation |
|-----------------|------------------|-------------|-------------------------------|-----------------------------|-----------------------|---------------------|---------------|-----------------|
| Self-instruct | 82,439 | 1 | 69.8 | 29.2 | 24.9 | 0.733 | - | No |
| Stanford Alpaca | 52,002 | 1 | 91.1 | 64.5 | 42.8 | 0.727 | - | No |
| SODA | 1,486,869 | <u>3.6</u> | 231.8 | 22.5 | 38.6 | 0.797 | 8.48 | No |
| GPT-4-LLM | 61,002 | 1 | 179.6 | 142.9 | 48.9 | 0.721 | - | No |
| BELLE | 1,436,679 | 1 | 102.3 | 63.3 | 35.9 | 0.771 | - | No |
| Baize | 210,311 | 3.1 | 293.9 | 52.8 | <u>67.1</u> | 0.751 | 9.06 | Yes |
| GPT4ALL | 711,126 | 1 | <u>597.7</u> | 318.9 | 62.7 | 0.692 | - | No |
| UltraChat | <u>1,468,352</u> | 3.8 | 1467.4 | <u>309.3</u> | 74.3 | <u>0.702</u> | 9.06 | Yes |

指令数据是越多越好吗？

基于合成的指令数据构建-UltraChat

从上一页表格中可以看出：

- UltraChat在规模、平均回合数、每个实例的最长平均长度和词汇多样性方面都优于其他数据集，是最大的开源数据集之一。
- UltraChat的话题多样性略低于GPT4ALL，但仍高于其他数据集。这可能是由于UltraChat的每个对话包含更多的token，而GPT4ALL的每个对话只有一个回合。
- 评估数据集的连贯性，发现UltraChat和Baize的数据在一致性方面排名最高。

指令数据构造注意事项：

1. 指令微调并不是数据量越多越好，数据质量更重要，太大的数据量有可能“阉割”掉模型原有的base能力。（*到底需要多少仍未有最终定论。*）
2. 多任务的指令数据集中，要保持多个任务的数据平衡。
3. 要注意指令表达多样性，尽量使用不同的语言表达方式来描述同一任务。
4. 保持指令格式的一致性，确保模型能够识别和理解不同指令的结构和含义。

指令微调的策略

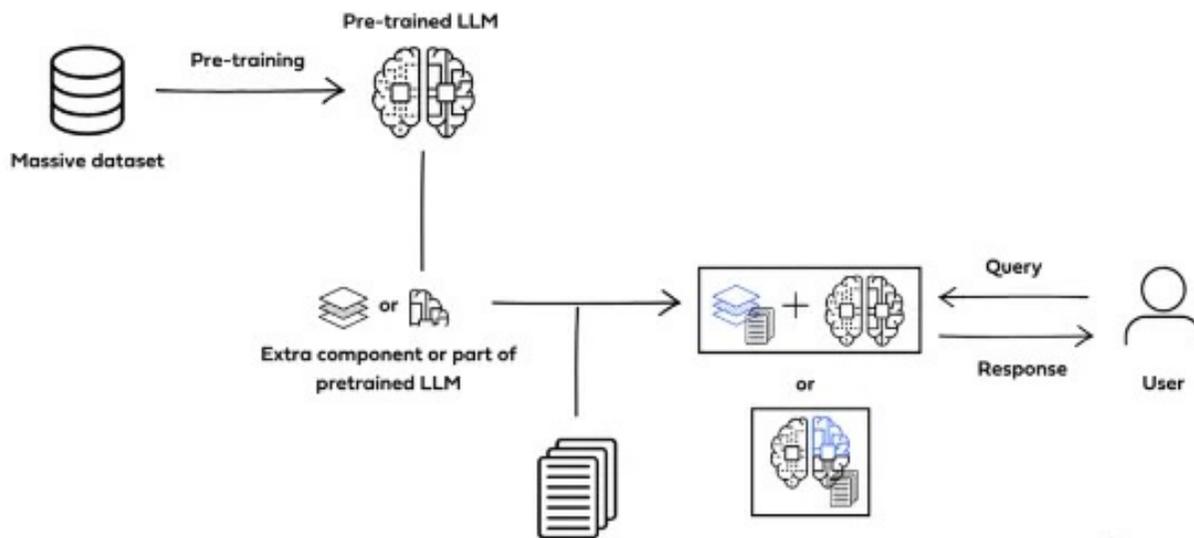
指令微调有两种策略：

1. 全参数微调

全参数微调是指对模型的所有参数进行调整，以适应新的指令数据集。这通常涉及在特定任务的数据集上训练模型，更新模型的所有权重。

2. 参数高效微调

参数高效微调是指仅调整模型的部分参数，而保留大部分预训练参数不变。这种方法通常通过引入适配层（adapter layers）、低秩适配（Low-rank Adaptation, LoRA）或者使用微调参数生成器来实现。



思考？

问题一：大语言模型为什么需要参数高效微调？

问题二：全参数微调会引起什么问题？

问题三：参数高效微调会引起什么问题？

指令微调的策略

指令微调两种策略优缺点：

1. 全参数微调

优点：

在特定任务上，通常能达到最好的性能，因为所有参数都得到了优化。

缺点：

需要大量的计算资源和时间，特别是对于大规模模型。如果训练数据不足或不够多样，容易导致模型过拟合。泛化性不好。

2. 参数高效微调

优点：

显著减少计算资源和时间的需求。更快的训练和部署周期。因为只调整部分参数，减少过拟合风险，模型更能保持对原始任务的泛化能力。

缺点：

在某些任务上可能无法达到全参数微调的最高性能。需要设计和实现适配层或其他微调机制。

指令微调的策略

指令微调有两种策略比较和总结：

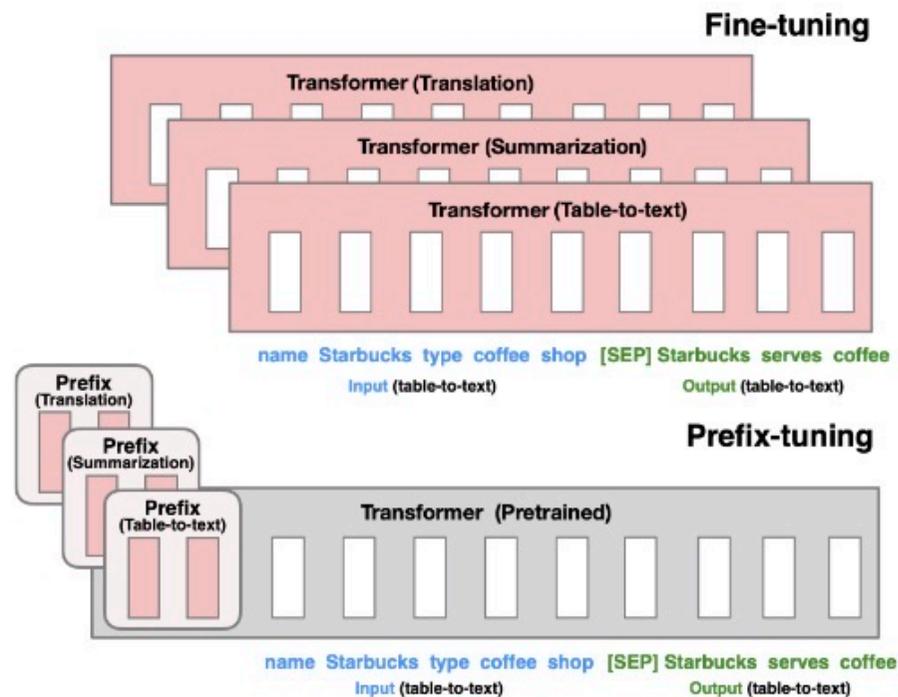
| 特性 | 全参数微调 | 参数高效微调 |
|-------|--------|------------|
| 资源需求 | 高 | 低 |
| 训练时间 | 长 | 短 |
| 性能 | 最高 | 较高 |
| 过拟合风险 | 高 | 低 |
| 适用场景 | 特定任务优化 | 多任务应用和快速迭代 |
| 实现复杂性 | 较低 | 较高 |

全参数微调和参数高效微调各有优劣，选择哪种策略应根据具体的应用场景、资源限制和性能要求来决定。全参数微调适合需要最高性能的特定任务，而参数高效微调则适合资源有限且需要快速迭代和多任务应用的场景。

参数高效的模型微调-Prefix-Tuning

传统的fine-tuning是在大规模预训练语言模型(如 Bert、GPT2等)上完成的,针对不同的下游任务,需要保存不同的模型参数,代价比较高。

Prefix Tuning只是在每个任务前有少量的prefix的参数,比如翻译任务,可以在每句话的前面加上“翻译:”来引导模型进行翻译功能。在自然语言理解和生成基准测试上具有很好的性能,通过微调,仅添加约2-4%的任务特定参数,就可以获得类似的性能。

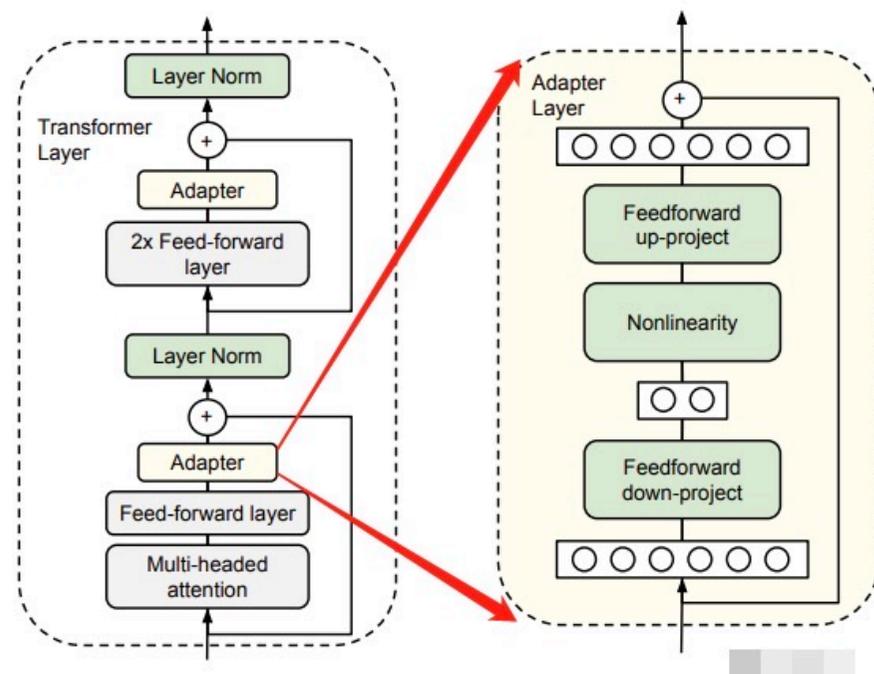


[1] Prefix-Tuning: Optimizing Continuous Prompts for Generation.(2021)

参数高效的模型微调-Adapter-Tuning

Adapter-Tuning的思路是在预训练模型中添加适配器层，并只微调适配器层的参数，从而保留预训练模型的知识、减少计算量和时间，并提高模型的可解释性和可复用性。

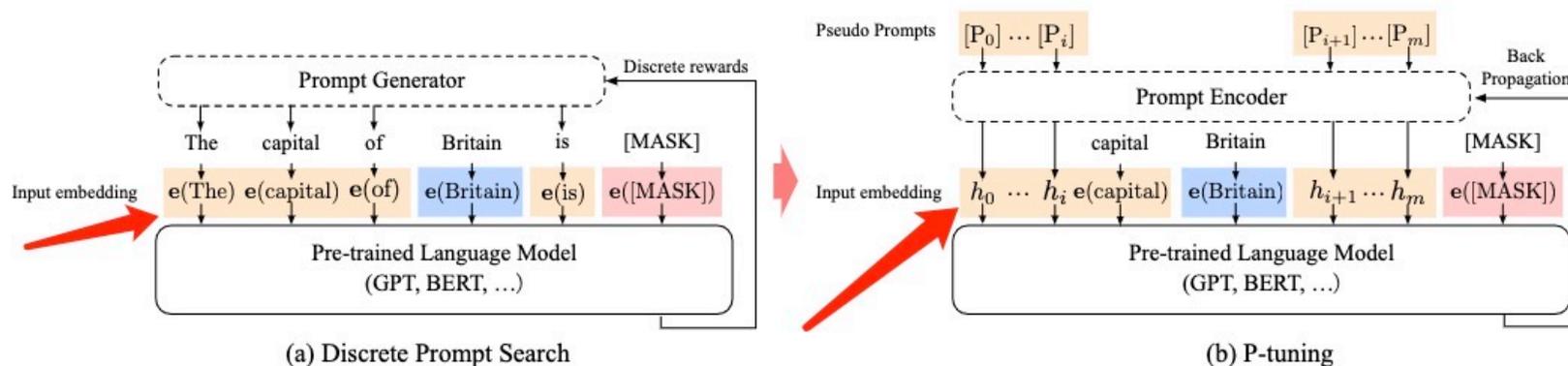
模型结构如右图所示，微调时冻结预训练模型的主体，由Adapter模块学习特定下游任务的知识。其中，Adapter模块结构如上图右侧所示，包含两个前馈层和一个中间层，第一个前馈层和中间层起到一个降维的作用，后一个前馈层和中间层起到升维的作用。



[1] Parameter-Efficient Transfer Learning for NLP.(2019)

参数高效的模型微调-Prompt-Tuning(P-Tuning)

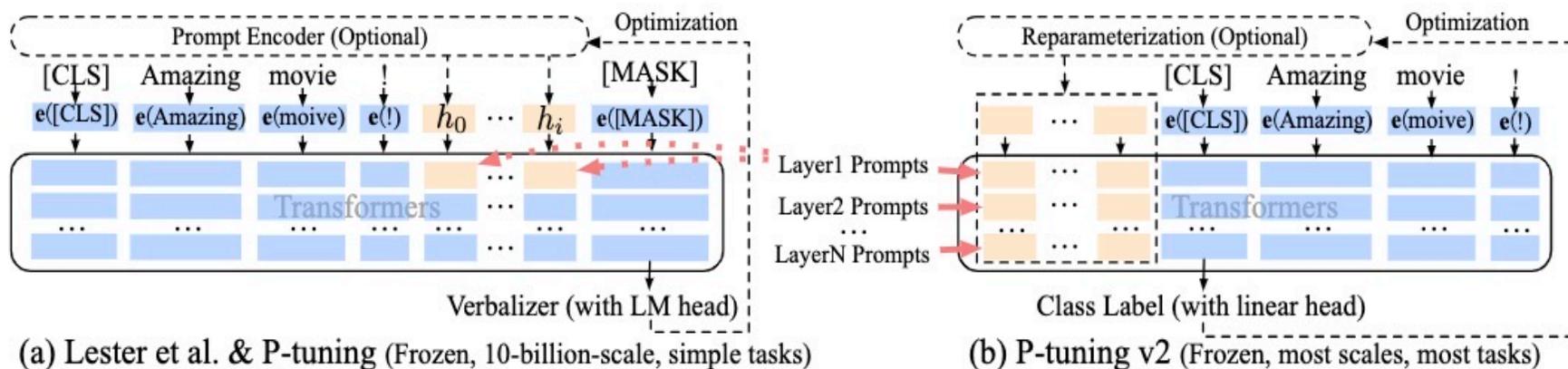
P-Tuning提出的目的。因为离线的Prompt对于连续的神经网络只是次优解，prompt的词之间是彼此关联的，需要将其关联起来。于是，P-Tuning将一些伪prompt输入至模型中，然后利用模型的输出向量来替代原始的prompt token，然后一起输入至预训练语言模型中。下图展示了P-Tuning和Discrete Prompt Search之间的区别



[1] The Power of Scale for Parameter-Efficient Prompt Tuning.(2021)

参数高效的模型微调-P-Tuning v2

P-Tuning v2的改进之处在于，除了输入的embedding外，其它的Transformer层也加了前置的prompt。将只在第一层插入continuous prompt修改为在许多层都插入continuous prompt，层与层之间的continuous prompt是相互独立的。经过这样的改进，模型可训练参数的量从0.01%增加到了0.1%-3%，效果提升较多。

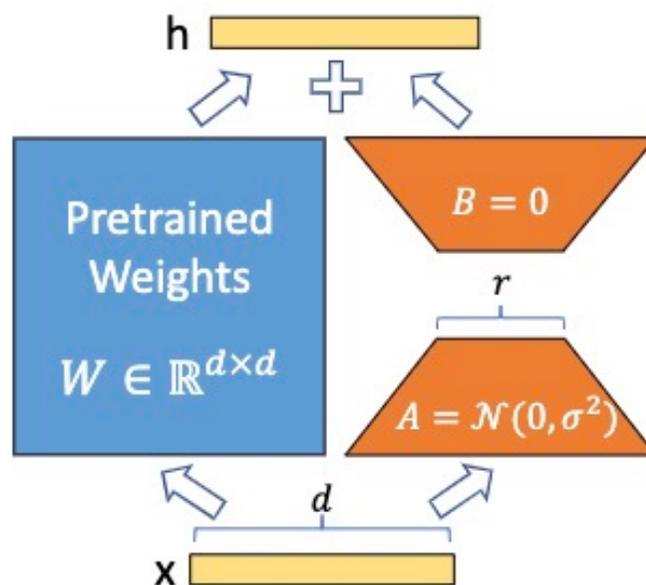


[1] P-Tuning v2: Prompt Tuning Can Be Comparable to Fine-tuning Universally Across Scales and Tasks.(2022)

参数高效的模型微调-LoRA

Lora是在原始预训练模型增加一个旁路，做一个降维再升维的操作（对应降维矩阵A和升维矩阵B，前者用高斯分布初始化，后者用初始化为0），模型的输入输出维度不变，输出时将BA和预训练模型的参数叠加。

在推理时，将左右两部分的结果加到一起，即 $h=Wx+BAx=(W+BA)x$ ，所以，只要将训练完成的矩阵乘积BA跟原本的权重矩阵W加到一起作为新权重参数替换原始预训练语言模型的W即可，不会增加太多额外的计算资源。



*思考：为什么设置 $B=0$ ？
 A 的参数会更新吗？*

参数高效的模型微调-LoRA

优势分析：

- LoRA的主要优点之一是他们的效率。通过使用更少的参数，LoRA显著降低了模型训练过程中计算复杂性和显存使用量。
- 此外，LoRA可以提升模型的泛化性。通过限制模型的复杂度，可以有助于防止在训练数据有限场景下的过拟合现象；
- 最后，LoRA可以无缝地集成到现有的神经网络架构中。不同的任务可以挂不同的LoRA模块，多个LoRA模块也可以叠加使用。

缺点：

- 对于某些复杂任务可能无法达到完全微调的效果。

参数高效的模型微调-LoRA

实验效果：

从下表可以看出仅微调了大概千分之一的参数，效果就超过了全部参数微调的效果。

| Model & Method | # Trainable Parameters | E2E NLG Challenge | | | | |
|----------------------------------|------------------------|-------------------------------|--------------------------------|-------------------------------|-------------------------------|--------------------------------|
| | | BLEU | NIST | MET | ROUGE-L | CIDEr |
| GPT-2 M (FT)* | 354.92M | 68.2 | 8.62 | 46.2 | 71.0 | 2.47 |
| GPT-2 M (Adapter ^L)* | 0.37M | 66.3 | 8.41 | 45.0 | 69.8 | 2.40 |
| GPT-2 M (Adapter ^L)* | 11.09M | 68.9 | 8.71 | 46.1 | 71.3 | 2.47 |
| GPT-2 M (Adapter ^H) | 11.09M | 67.3 \pm .6 | 8.50 \pm .07 | 46.0 \pm .2 | 70.7 \pm .2 | 2.44 \pm .01 |
| GPT-2 M (FT ^{Top2})* | 25.19M | 68.1 | 8.59 | 46.0 | 70.8 | 2.41 |
| GPT-2 M (PreLayer)* | 0.35M | 69.7 | 8.81 | 46.1 | 71.4 | 2.49 |
| GPT-2 M (LoRA) | 0.35M | 70.4\pm.1 | 8.85\pm.02 | 46.8\pm.2 | 71.8\pm.1 | 2.53\pm.02 |
| <hr/> | | | | | | |
| GPT-2 L (FT)* | 774.03M | 68.5 | 8.78 | 46.0 | 69.9 | 2.45 |
| GPT-2 L (Adapter ^L) | 0.88M | 69.1 \pm .1 | 8.68 \pm .03 | 46.3 \pm .0 | 71.4 \pm .2 | 2.49\pm.0 |
| GPT-2 L (Adapter ^L) | 23.00M | 68.9 \pm .3 | 8.70 \pm .04 | 46.1 \pm .1 | 71.3 \pm .2 | 2.45 \pm .02 |
| GPT-2 L (PreLayer)* | 0.77M | 70.3 | 8.85 | 46.2 | 71.7 | 2.47 |
| GPT-2 L (LoRA) | 0.77M | 70.4\pm.1 | 8.89\pm.02 | 46.8\pm.2 | 72.0\pm.2 | 2.47 \pm .02 |

参数高效的模型微调-LoRA

LoRA的一些变形：

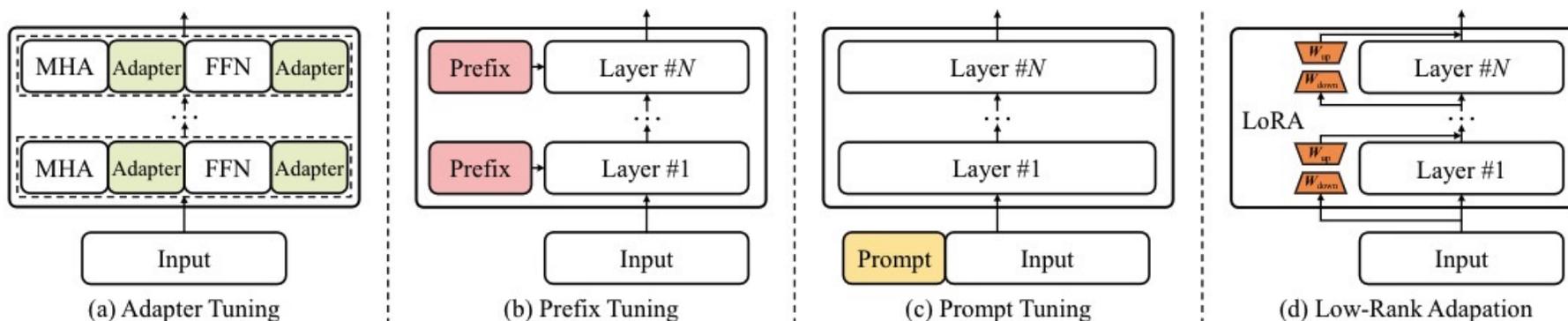
AdaLoRA。它讨论了如何更好地进行秩的设置。它引入了一种动态低秩适应技术，在训练过程中动态调整每个参数矩阵需要训练的秩同时控制训练的参数总量。具体来说，模型在微调过程中通过损失来衡量每个参数矩阵对训练结果的重要性，重要性较高的参数矩阵被赋予比较高的秩，进而能够更好地学习到有助于任务的信息。相对而言，不太重要的参数矩阵被赋予比较低的秩，来防止过拟合并节省计算资源。

QLoRA。它将原始的参数矩阵量化为4比特，而低秩参数部分仍使用16比特进行训练，在保持微调效果的同时进一步节省了显存开销。对于给定参数量的模型，QLoRA微调所需要的显存变成LoRA微调所需要1/4。

[1] AdaLoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning.(2023)

[2] Making LLMs even more accessible with bitsandbytes, 4-bit quantization and QLoRA.(2023)

参数高效的模型微调方法比较



两种目前常用的PEFT方法对比：

1) Prompt-Tuning

优点：

无需改变模型架构：通过设计任务相关的提示（prompt），不需要改变模型参数，资源消耗少：

缺点：

需要精心设计prompt：有效的prompt设计可能需要丰富的经验和实验。

2) LoRA

优点：

参数高效：通过引入低秩矩阵来调整模型，减少需要优化的参数数量。节省内存和计算资源。

缺点：

效果可能有限：对于某些复杂任务可能无法达到完全微调的效果。

人类对齐

大规模语言模型在进行监督微调后，模型具备了遵循指令和多轮对话的能力，具备了初步与用户进行对话的能力。然而，大规模语言模型由于庞大的参数量和训练语料，其复杂性往往难以理解和预测。因此，要确保模型的目标、行为和决策与人类的价值观、意图和期望一致

gpt3提出的人类对齐步骤如右图所示。

Step 2

Collect comparison data, and train a reward model.

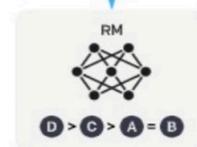
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



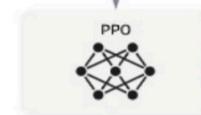
Step 3

Optimize a policy against the reward model using reinforcement learning.

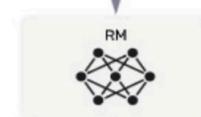
A new prompt is sampled from the dataset.



The policy generates an output.



The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.



[1] Training language models to follow instructions with human feedback. (2022)

人类对齐的原则-3H标准

人工智能应与人类价值观进行对齐, 大语言模型输出的结果应该满足帮助性 (Helpfulness)、 真实性 (Honesty) 以及无害性 (Harmless), 这就是3H标准。

“有帮助” 指的是输出应遵循用户的意图, 并帮助用户解决他们的任务。

“真实” 指的是输出包含准确的信息, 并且不误导用户。

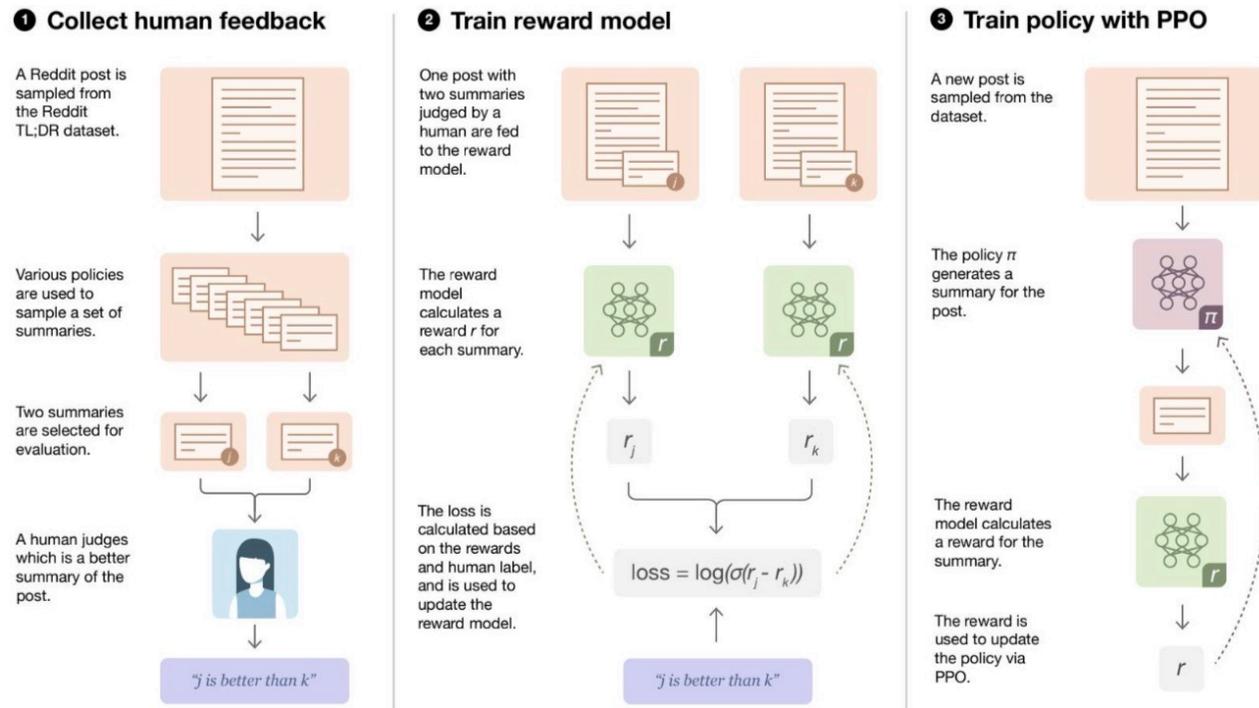
“无害性” 意味着输出不应对人们造成身体、心理或社会伤害; 不应对设备或财产造成损害或丢失; 不应对环境造成破坏; 也不应损害对人类福祉必需的机构或资源。

评估模型输出时可能需要在这些标准之间做出权衡, 这些权衡取决于具体的任务, 对于大多数任务来说, 无害性和真实性比帮助性更重要。

[1]Training language models to follow instructions with human feedback. (2022)

人类反馈强化学习算法 (RLHF)

由于3H原则体现出了人类偏好，因此基于人类反馈的强化学习 (Reinforcement Learning from Human Feedback, RLHF) 很自然的被引入到通用对话模型的训练流程中。基于人类反馈的强化学习，用奖励模型Reward Model来训练SFT模型。生成模型使用奖励或惩罚来更新其策略，以便生成更高质量、更符合人类偏好的文本。RLHF的示意图如下：



[1] Learning to summarize from human feedback.(2009)

RLHF-收集人类反馈

这一步主要做的是得到同一个输入的各种输出的好坏排序。具体的步骤为：

1. 收集原始的数据得到输入。
2. 使用各种策略根据输入得到各种输出。
3. 人工对各种输出按好坏排序。

这部分用来产生output的模型主要有两类：

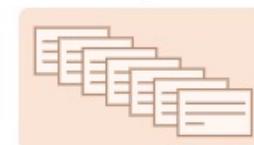
- 预训练模型，即只经过预料库训练而未经过 fine-tune 的模型。
- 监督基线模型，即在预训练模型基础上使用测试数据集 fine-tune 的模型。

① Collect human feedback

A Reddit post is sampled from the Reddit TL;DR dataset.



Various policies are used to sample a set of summaries.



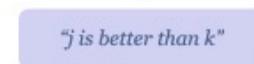
Two summaries are selected for evaluation.



A human judges which is a better summary of the post.



"j is better than k"

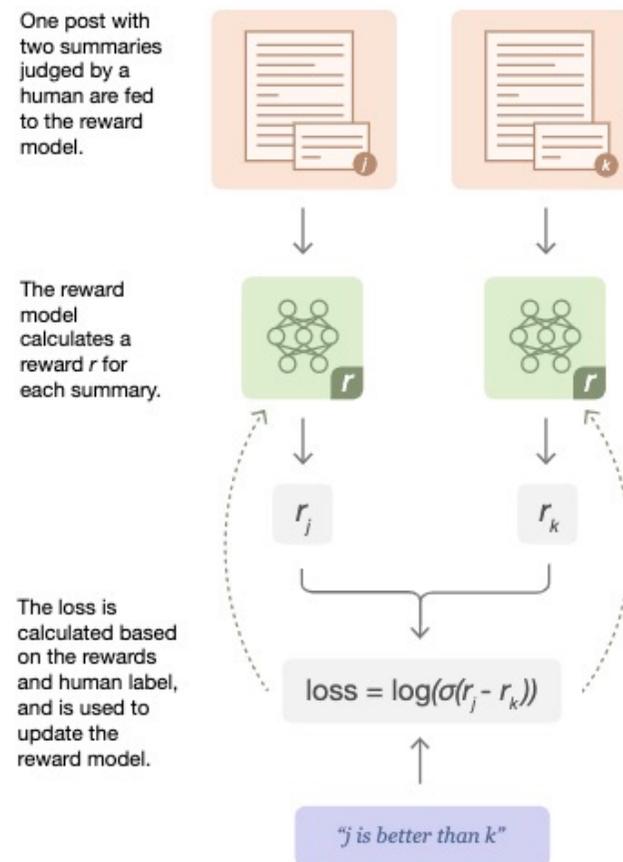


RLHF-训练奖励模型

奖励模型（Reward Model, RM）的目标是刻画模型的输出是否在人类看来表现不错。即，输入 [提示(prompt), 模型生成的文本]，输出一个刻画文本质量的标量数字，数值上对应人的偏好。

对于一个奖励模型来说，目标是给一个句子进行打分，按理说每个句子对应一个分值就行了，但是目前对于长度为L的句子，奖励模型输出了L个值。我们用L维度上的最后一个位置的值当作为本句话的奖励得分。奖励模型训练优化采用pair wiss loss，即同时输入模型关于同一个问题的两个回答，让模型学会这两个句子哪个分高哪个分低。之所以如此训练是因为，在给奖励模型进行数据标注的过程中，给同一个问题的不同回答量化的打具体分值比较难，但是对他们进行排序相对简单

2 Train reward model



RLHF-训练策略模型

将初始语言模型的微调任务建模为强化学习 (RL) 问题，因此需要定义策略 (policy)、动作空间 (action space) 和奖励函数 (reward function) 等基本要素。

策略: 基于该语言模型，接收prompt作为输入，然后输出一系列文本（或文本的概率分布）。

动作空间: 词表所有token在所有输出位置的排列组合（单个位置通常有50k左右的token候选）；观察空间则是可能的输入token序列（即prompt），显然也相当大，为词表所有token在所有输入位置的排列组合。

奖励函数 (reward) : 是基于上一章节我们训好的RM模型计算得到初始reward，再叠加上一个约束项来。

而对于强化学习的算法，常见的可行方案是使用策略梯度强化学习 (Policy Gradient RL) 算法、近端策略优化 (Proximal Policy Optimization, PPO) 微调初始LM的部分或全部参数。

③ Train policy with PPO

A new post is sampled from the dataset.



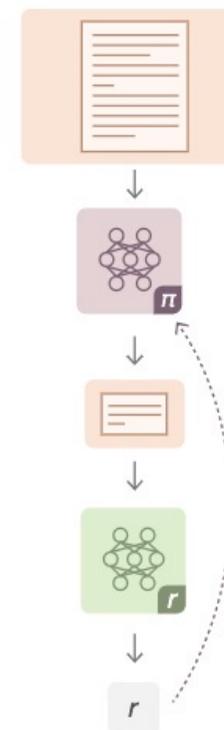
The policy π generates a summary for the post.



The reward model calculates a reward for the summary.



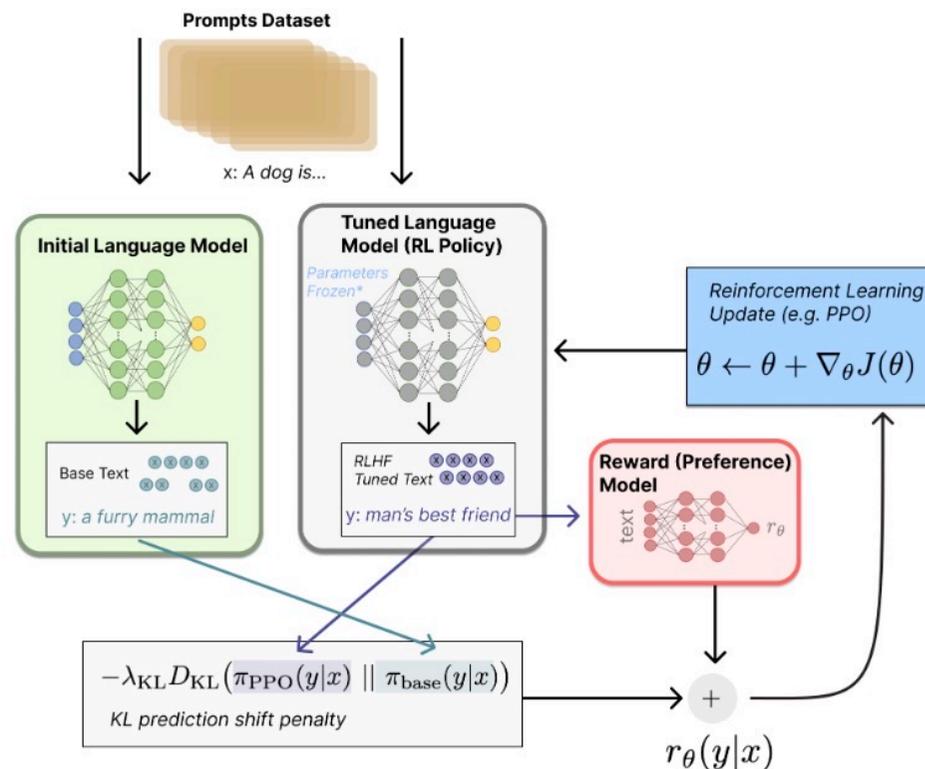
The reward is used to update the policy via PPO.



RLHF-PPO

PPO算法是一种强化学习中的策略梯度方法，它的全称是Proximal Policy Optimization，即近端策略优化。PPO算法的目标是在与环境交互采样数据后，使用随机梯度上升优化一个“替代”目标函数，从而改进策略。PPO算法的特点是可以进行多次的小批量更新，而不是像标准的策略梯度方法那样每个数据样本只进行一次梯度更新。

具体的优化概略如右图表所示。



[1]Proximal Policy Optimization Algorithms.(2017)

RLHF-总结

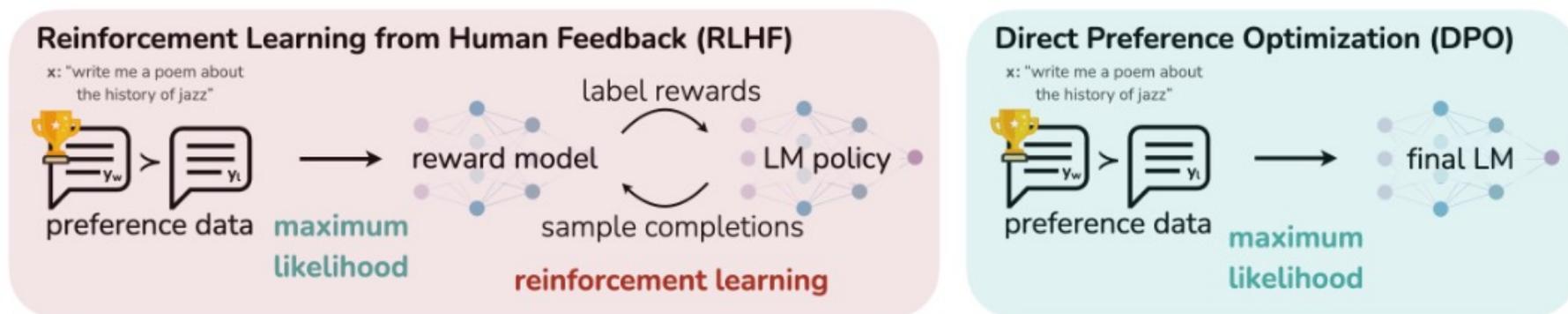
RLHF 通过基于人类偏好数据微调模型解决了模型与人类喜好对齐的问题，这种方式提供了更加可控和可靠的用户体验。RLHF提高了模型答案的一致性，但它不可避免地以其生成能力的多样性为代价。根据预期的用例，这种权衡既可以被视为好处也可以被视为限制。

尽管 RLHF 取得了一定的成果和关注，但依然存在局限。模型有时依然会毫无不确定性地输出有害或者不真实的文本（幻觉）。同时，收集人类偏好数据的质量和数量决定了 RLHF 系统性能的上限。

另外，RLHF整个流程较为复杂，时间周期长，而且PPO算法效率低下。

RLHF-DPO

目前RLHF的流程太复杂，不仅需要单独训练reward模型，还需要从LLM的输出采样。DPO通过理论证明，可以不需要引入reward模型，就可以完成LLM的偏好训练。实验证明，在生成情感、摘要、单论对话质量方面，DPO比基于PPO的RLHF更好。（具体细节不再讲述，感兴趣自行学习。）



[1] Direct Preference Optimization: Your Language Model is Secretly a Reward Model.(2023)

非RL方法的对齐方法

[1]LIMA着眼于探究大模型预训练和指令微调两个阶段的重要性。在LLaMa-65B的基础上，只使用1000个精心准备的prompt和对应的结果来进行微调，没有进行任何强化学习和人类偏好建模。LIMA展示了非常强大的性能，特别是对于训练集中没有出现过的任务有很好的泛化能力。

[2]提出了一种新颖的学习对齐机制,使语言模型能够从模拟的社会互动中学习。首先创建了一个由许多基于语言模型的社会代理组成的模拟人工社会沙箱。这些代理相互交互并记录自己的互动行为。记录的互动数据与常规的对齐数据不同,它不仅呈现对齐和不对齐的示范,还包括集体评级、详细反馈以及展示“一步步”改进风格的修订响应。

[3]来自 Google DeepMind 的研究者提出了一种简单的算法使 LLM 与人类偏好对齐，他们将该方法命名为 ReST (Reinforced Self-Training)。不同于 RLHF 使用人类反馈改进语言模型，ReST 通过生成和使用离线数据进行训练，从而使得 LLM 与人类偏好保持一致。

[1]. LIMA: Less Is More for Alignment.(2023)

[2]. Training Socially Aligned Language Models in Simulated Human Society.(2023)

[3]. Reinforced Self-Training (ReST) for Language Modeling.(2023)

谢谢！