

# 1. Visual ChatGTP

## Limitations

- **Dependence on ChatGPT and VFMs** Visual ChatGPT relies heavily on ChatGPT to assign tasks and on VFMs to execute them. The performance of Visual ChatGPT is thus heavily influenced by the accuracy and effectiveness of these models.
- **Heavy Prompt Engineering** Visual ChatGPT requires a significant amount of prompt engineering to convert VFMs into language and make these model descriptions distinguishable. This process can be time-consuming and requires expertise in both computer vision and natural language processing.

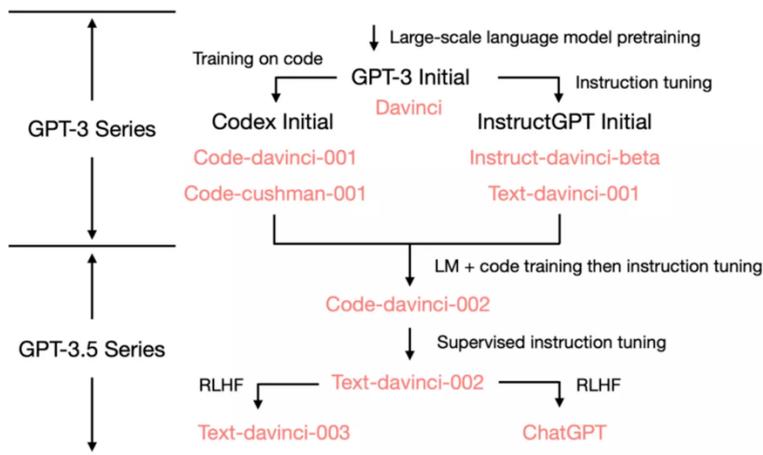
## LLM

We implement the LLM with ChatGPT [29] (OpenAI “text-davinci-003” version), and guide the LLM with LangChain

**InstructGPT** (2022年1月) 是一系列 GPT-3 模型（包括 text-davinci-001、text-davinci-002 和 text-davinci-003）统称，于GPT-3相比，它的最大不同是针对人类指令（reinforcement learning with human feedback, RLHF）进行了微调；InstructGPT 产生的幻觉更少，更真实，但它在生成的多样性或者说创意上相对更差，因为它们试图在“对齐”的前提下，将人类偏好/价值观硬塞进原始数据模型中。

**ChatGPT** (2022年11月) 更进一步。为了训练 ChatGPT，OpenAI 对 InstructGPT 对话模型进行了微调（马斯克在Twitter上指出openai 使用了Twitter 数据）。这种微调在一定程度上也是可以的。区别在于使用的policy and reward model

从最初的 GPT-3 开始，为了展示 OpenAI 是如何发展到ChatGPT的，我们看一下 GPT-3.5 的进化树：



3月13日，斯坦福大学基础模型研究中心（Center for Research on Foundation Models, CRFM）发布了指令精调模型 Alpaca 7B，使用 5.2 万由 OpenAI 的 text-davinci-003 模型<sup>1</sup>生成的指令数据，对 LLaMA 7B 进行精调而得。斯坦福羊驼横空出世，开源了包括准备过程、数据集、训练步骤等在内的整套方案，经评估能取得近似 text-davinci-003 的指令服从效果，但低成本且高效、易于复现，经验启发了许多蠢蠢欲动的后来者。

3月16日，在 Alpaca 基础上补充了多语种语料和指令任务的 Guanaco 原驼模型问世。

3月19日，来自加州大学伯克利分校、卡内基梅隆大学、斯坦福大学、加州大学圣地亚哥分校的几位计算机博士组队，以大模型系统（Systems for Large Models, LM-SYS）的名义发布了「小羊驼」Vicuna-13B。基于 ShareGPT 收集的对话对 LLaMA 进行精调，仅需 300 美元即完成训练的小羊驼，号称达到了 ChatGPT 90% 的能力，并将 Meta 大羊驼和斯坦福羊驼均甩在身后。

3月23日，「中文小羊驼」Chinese-Vicuna 携模型及数据面世，基于 LLaMA 模型和 LoRA 方案，可按需投喂数据进行个性化指令精调。

- Hugging Face 的 PEFT 是一个库（LoRA 是其支持的技术之一，除此之外还有 Prefix Tuning、P-Tuning、Prompt Tuning），可以让你使用各种基于 Transformer 结构的语言模型进行高效微调。
- Alpaca 羊驼：让 OpenAI 的 text-davinci-003 模型以 self-instruct 方式生成 52K 指令遵循（instruction-following）样本，以此作为 Alpaca 的训练数据，最后训练的羊驼只有 7B 参数量。可以使用 LoRA 微调优化。
- LLM 技术思路：
  - 语言模型：llama、bloom、glm 等
  - 指令微调数据：alpaca\_data、bella\_data、guanaco\_data 等。目前指令微调数据上，很依赖 alpaca 以及 chatgpt 的 self-instruct 数据。数据处理参考下图：
  - 微调加速：lora（如 Alpaca-Lora）等，还可以使用 peft 库、量化工具包 bitsandbytes、deepspeed（先读 torch.distributed 和 ColossalAI 在搞）、llama.cpp 量化模型。在 LoRA 方法提出之前，也有很多方法尝试解决大模型微调困境的方法。其中有两个主要的方向：
    - 添加 adapter 层。adapter 就是固定原有的参数，并添加一些额外参数用于微调；
    - 由于某种形式的输入层激活。

## 2. Visual Instruction Tuning

<https://arxiv.org/pdf/2304.08485.pdf> (包含构造的指令数据集由 gpt4 生成)

The primary goal is to effectively leverage the capabilities of both the pre-trained LLM and visual model. The network architecture is illustrated in Figure 1. We choose LLaMA as our LLM  $f_\phi(\cdot)$  parameterized by  $\phi$ , as its effectiveness has been demonstrated in several open-source language-only instruction-tuning works. [43, 45, 34].

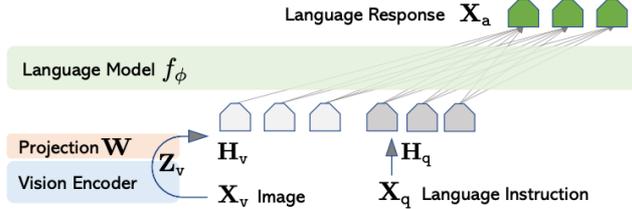


Figure 1: LLaVA network architecture.

```

X_system-message <STOP> \n
Human : X_instruct^1 <STOP> \n Assistant: X_a^1 <STOP> \n
Human : X_instruct^2 <STOP> \n Assistant: X_a^2 <STOP> \n ...
    
```

Table 2: The input sequence used to train the model. Only two conversation turns are illustrated here; in practice, the number of turns varies based on the instruction-following data. In our current implementation,  $X_{\text{system-message}} = A$  chat between a curious human and an artificial intelligence assistant. The assistant gives helpful, detailed, and polite answers to the human’s questions. and  $\langle \text{STOP} \rangle = \#\#\#$ . The model is trained to predict the assistant answers and where to stop, and thus only **green sequence/tokens** are used to compute the loss in the auto-regressive model.

**Quantitative Evaluation.** To have a systematic understanding of the performance of LLaVA, we aim to leverage a quantitative metric in measuring the model’s instruction-following capability. Motivated by [45], we leverage GPT-4 to measure the quality of our model’s generated responses. Specifically, we randomly select 30 images from the COCO validation split, and generate three types of question (conversation, detailed description, complex reasoning) using the proposed data generation pipeline. LLaVA predicts the answers based on the question and the visual input image. GPT-4 makes a reference prediction based on the question, and the ground-truth bounding boxes and captions, marking an upper bound of the teacher model. After obtaining the response from both models, we feed the question, visual information (in the format of captions and bounding boxes), and the generated responses from both assistants, to the GPT-4. GPT-4 evaluates the helpfulness, relevance, accuracy, and level of details of the responses from the assistants, and give an overall score on a scale of 1 to 10, where a higher score indicates better overall performance. GPT-4 is also asked to provide a comprehensive explanation the evaluation, for us to better understand the models.

We vary the training datasets to study the effectiveness of different types of the instruction-following data. We show the results in Table 3. First, with instruction tuning, the model’s capability of following the user instructions improves significantly by over 50 points. Second, adding a small amount of the detailed description and complex reasoning questions contributes to a considerable improvement of the model’s overall capability by 7 points. Furthermore, it also improves the model’s performance on conversational questions, suggesting that the overall improvement in the reasoning capability is

	Conversation	Detail description	Complex reasoning	All
Full data	83.1	75.3	96.5	85.1
Detail + Complex	81.5 (-1.6)	73.3 (-2.0)	90.8 (-5.7)	81.9 (-3.2)
Conv + 5% Detail + 10% Complex	81.0 (-2.1)	68.4 (-7.1)	91.5 (-5.0)	80.5 (-4.4)
Conversation	76.5 (-6.6)	59.8 (-16.2)	84.9 (-12.4)	73.8 (-11.3)
No Instruction Tuning	22.0 (-61.1)	24.0 (-51.3)	18.5 (-78.0)	21.5 (-63.6)

Table 3: Relative scores for different settings *w.r.t.* GPT-4 (text-only) on 30 randomly sampled images from COCO Val 2014. Each image is associated one short question, one detailed question, and one complex reasoning question, resulting in a total of 90 questions. Following the same setting as our data generation pipeline, GPT-4 uses ground truth image captions and bounding boxes as visual input. We prompt GPT-4 with the answers from our model outputs and the answers by GPT-4 (text-only), and let it compare between both responses and give a rating with an explanation.

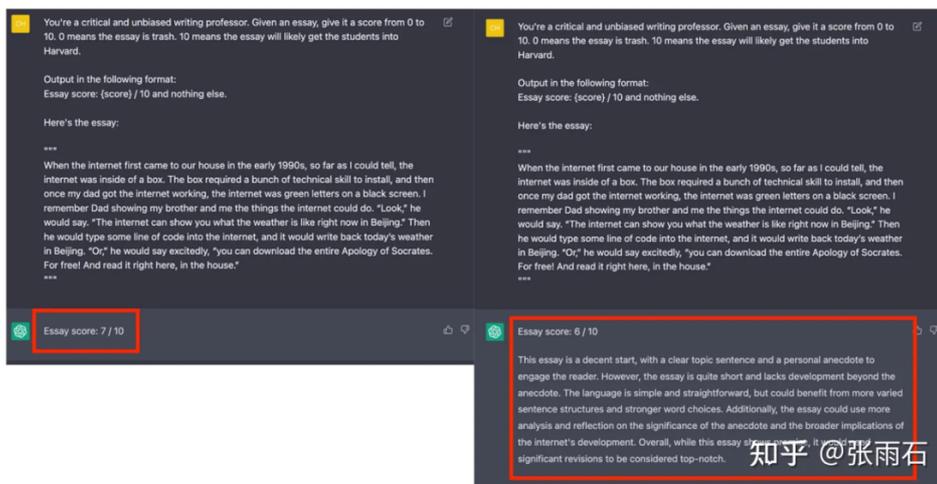
### 3. 提示词工程存在的问题

<https://zhuanlan.zhihu.com/p/627873954>

虽然用户定义提示的灵活性只是一个烦恼，但 LLM 生成的响应中的模糊性可能会是个破坏者。它导致两个问题：

1. 输出格式不明确：LLM 之上的下游应用程序期望以特定格式输出，以便它们可以解析。我们可以精心设计提示以明确输出格式，但不能保证输出始终遵循此格式。

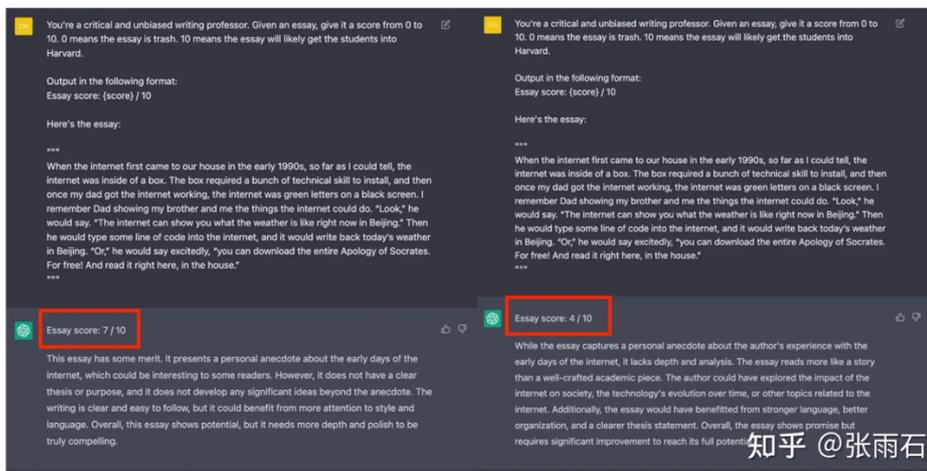
Expected output format  
Essay score: {score} / 10 and nothing else.



1. 用户体验不一致：在使用应用程序时，用户期望一定的一致性。想象一下，每次在网站上查看时一家保险公司都会给出不同的报价，这样的体验是比较糟糕的。而这都源于LLM的随机性——不能保证LLM每次都会为相同的输入提供相同的输出。

一个通常的做法是通过设置temperature=0 (platform.openai.com/doc...) 来强制LLM给出相同的响应。虽然它主要解决了一致性问题，但它并没有激发人们对系统的信任。想象一位老师，只有当他坐在一个特定的房间里时，他才能给出一致的分数。如果那个老师坐在不同的房间里，那位老师给的分数会很乱。

Same prompt, different scores



# 4.LLM 微调策略

<https://github.com/ninehills/ninehills.github.io/issues/92>

**指令微调 ( Instruction Tuning )** 目标是增强 ( 或解锁 ) 大语言模型的能力 。指令微调是在自然语言格式的实例集合上微调预训练后的大语言模型的方法。这种方法与有监督微调和多任务提示训练密切相关。为了进行指令微调，我们首先需要收集或构造指令格式的实例。然后，我们使用这些格式化的实例以有监督的方式微调大语言模型 ( 例如，使用序列到序列的损失进行训练 )。指令微调后，大语言模型展现出泛化到未见过任务的卓越能力，即使在多语言场景下也能有不错表现

## 0x11 格式化实例构造 ( 微调数据集 )

通常情况下，一个指令格式化的实例包括一个任务描述 ( 被称为指令 instruction )、一个输入输出对以及少量示例 ( 可选 )。

数据集一般用两种方法产出：

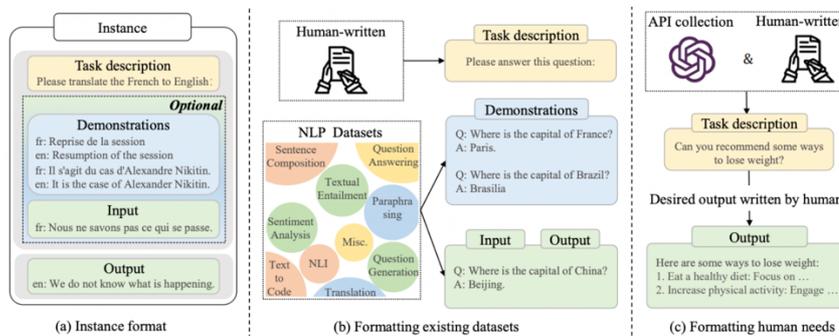


图3：实例格式化和两种构造指令格式实例的示意图。 [1]

1. 格式化已有数据集。将传统的NLP数据集格式调整后，用于指令微调。
  - i. 为降低人工格式化成本，可以通过 ChatGPT 生成 Instruction，典型 Prompt：“请你为这段内容生成一个合理的问题”
2. 人工标注数据集。为获得更好的人类对齐效果，OpenAI 建议使用人工标注数据集。
  - i. 为降低人工标注成本，目前很多数据集由 ChatGPT 生成，包括用户分享的 ChatGPT 对话历史 (如 ShareGPT) 或者使用 ChatGPT 生成的数据集。

数据集也分为通用任务的数据集以及专用数据集，前者目前有大量开源数据集可供使用，后者则针对某个具体领域，可以自行构建。此外引入多样化的数据集 (包括引入 CoT 思维链等) 可以有效的提升模型性能。

以 **Guannaco 模型** 为例，其格式为

Prompt: ""Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

### Instruction: {instruction}

### Input:{input}

### Response:""

Complete: ""{output}""

## 0x13 指令微调效果

指令微调可以让大模型有质的提升，主要体现在两个方面：

1. **性能改进**：最近的研究发现经过指令微调的较小模型甚至可以比未经微调的较大模型表现更好。除了模型规模外，指令微调在不同的模型架构、预训练目标和模型适应方法上都展现出持续的改进效果。在实践中，指令微调为提升现有语言模型 (包括小型预训练语言模型) 的能力提供了一种通用的方法。此外，与预训练相比，指令微调成本较低，因为大语言模型所需的指令数据数量明显少于预训练数据。
2. **任务泛化性**：指令微调鼓励模型理解用于任务完成的自然语言指令。它赋予大语言模型遵循人类指令执行特定任务的能力 (通常被视为一种涌现能力)，即使在未见过的任务上也能够执行。大量研究已经证实了指令微调在已见和未见任务上实现卓越的性能表现。此外，指令微调还被证明对缓解大语言模型的一些弱点 (如生成重复内容或在未完成特定任务的情况下补充输入) 具有帮助，从而使大语言模型具有更强的解决现实世界任务的能力。此外，通过使用指令微调训练的大语言模型可以在不同语言之间泛化到相关任务。

**对话微调 (Conversation Tuning)** 对话微调是一种特殊的指令微调。其目的是让大语言模型在「补全」能力的基础上，解锁「对话」能力。

典型的数据集格式：`{"instruction": "", "input": "", "output": "", "history": [{"", ""}, {"", ""}]}`

具体的对话格式应该和模型在预训练以及微调时保持一致，以ChatGLM-6B 微调数据格式

为例：

```
---
Prompt: "" [Round 0]
问：你好，你能帮我解答一个问题吗？
答：当然，请问有什么问题？
[Round 1]
问：我想知道人工智能的未来发展方向，你有什么想法吗？
答：人工智能在未来的发展方向可能包括更强大的机器学习算法，更先进的自然语言处理技术，以及更加智能的机器人。
[Round 2]
问：听起来很不错。人工智能可能在哪些方面面临挑战呢？
答：""

Complete: "人工智能面临的挑战包括数据隐私、安全和道德方面的问题，以及影响就业机会的自动化等问题。"
```

OpenAI 则推出了 [ChatML 格式](#)，其原始数据格式为：

```
Prompt: ""<|im_start|>system
You are ChatGPT, a large language model trained by OpenAI. Answer as concisely as possible.
Knowledge cutoff: 2021-09-01
Current date: 2023-03-01<|im_end|>
<|im_start|>user
How are you<|im_end|>
<|im_start|>assistant
I am doing well!<|im_end|>
<|im_start|>user
How are you now?<|im_end|>""

Complete: ""{output}""
```

## 参数高效微调 (Parameter-Efficient Fine-Tuning)

虽然指令微调相比于预训练更加高效（只需要处理微调数据集），但是全参数的指令微调依然需要消耗较多的算力。目前有多种高效参数微调方案，可以在实现和全参数微调相同性能的情况下，大幅降低微调成本。

参数高效微调，也称为 PEFT，可以仅微调少量或者额外的模型参数，并固定住大部分预训练参数，从而大大降低训练成本。目前研究表明，PEFT 相比于全参数微调，在没有微调过的大语言模型上性能略差，但是在已经微调过的模型上，性能接近。

PEFT 方法可以分为三类，不同的方法对模型的不同部分进行下游任务的适配：

- **Prefix/Prompt-Tuning**：在模型的输入或隐层添加  $k$  个额外可训练的前缀 tokens（这些前缀是连续的伪 tokens，不对应真实的 tokens），只训练这些前缀参数；
- **Adapter-Tuning**：将较小的神经网络层或模块插入预训练模型的每一层，这些新插入的神经模块称为 adapter（适配器），下游任务微调时也只训练这些适配器参数；
- **LoRA<sup>[4]</sup>**：通过学习小参数的低秩矩阵来近似模型权重矩阵  $W$  的参数更新，训练时只优化低秩矩阵参数。

目前针对大语言模型，评测效果最好的是 LoRa。其基本原理是在原始预训练权重旁边增加一个旁路，做一个降维再升维的操作  $k$ 。训练的时候固定预训练权重，只训练降维矩阵  $A$  与升维矩阵  $B$ 。而模型的输入输出维度不变，输出时将  $B/A$  与预训练权重的参数叠加。实际微调过程中， $A/B$  基本在 MB 级别，与之相比，大语言模型的权重通常在 GB 级别。

## Lora微调

目前针对大语言模型，评测效果最好的是 LoRa。其基本原理是在原始预训练权重旁边增加一个旁路，做一个降维再升维的操作。训练的时候固定预训练权重，只训练降维矩阵A与升维矩阵B。而模型的输入输出维度不变，输出时将B/A与预训练权重的参数叠加。实际微调过程中，A/B 基本在MB级别，与之相比，大语言模型的权重通常在GB级别。

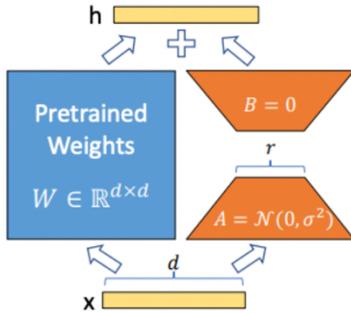
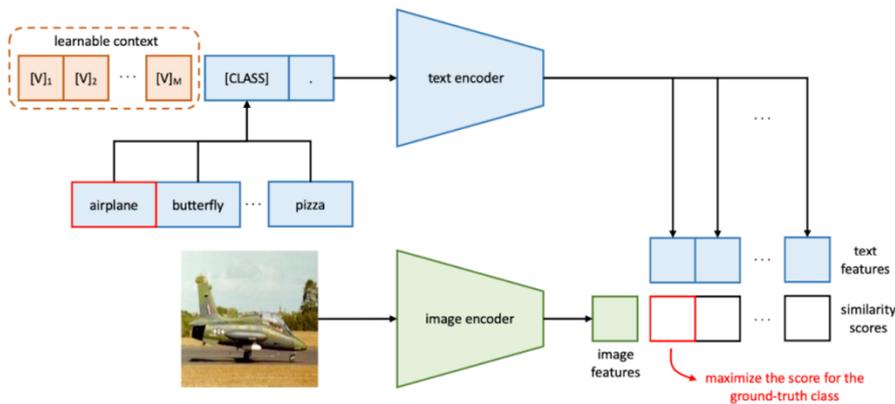


Figure 1: Our reparametrization. We only train  $A$  and  $B$ .

图4: Lora 微调原理

## Prompt-Tuning :



## 对齐微调

大语言模型在多个自然语言处理任务上展示出了惊人的能力。但是, 这些模型有时可能表现出意外的行为, 例如制造虚假信息、追求不准确的目标, 以及产生有害的、误导性的和偏见性的表达。对于 LLM, 语言建模目标通过单词预测对模型参数进行预训练, 但缺乏对人类价值观或偏好的考虑。为了避免这些意外行为, 研究提出了人类对齐, 使大语言模型行为能够符合人类的期望。但是, 与初始的预训练和适应微调 (例如指令微调) 不同, 语言模型的对齐需要考虑不同的标准 (例如有用性, 诚实性和无害性)。已有研究表明对齐微调可能会在某种程度上损害大语言模型的通用能力, 这在相关研究中被称为对齐税。

### 0x21 对齐标准

- 有用性: 简明扼要且高效的方式帮助用户解决任务或回答问题。在更高层次上, 当需要进一步澄清时, 大语言模型应展示出通过相关提问获取额外相关信息的能力, 并表现出合适的敏感度、洞察力和审慎程度。
- 诚实性: 应该向用户提供准确的内容, 而不会捏造信息。此外, 大语言模型在输出时传达适当程度的不确定性至关重要, 以避免任何形式的欺骗或信息误传。这需要模型了解其能力和知识水平 (即所谓的“知道自己不知道”)。
- 无害性: 模型生成的语言不得具有冒犯性或歧视性, 同时不能生成违反法律法规以及道德规范的有害信息。

### 0x22 基于人类反馈的强化学习 (RLHF)

通过 RLHF, 大模型可以根据人类反馈进行强化学习, 从而和人类对齐。典型步骤如下图。

## 相关项目

1. **LMFlow**: 一个可扩展、方便和高效的工具箱，用于微调大型机器学习模型。支持所有 Decoder 模型的微调。
2. **FastChat**: FastChat is an open platform for training, serving, and evaluating large language model based chatbots.
3. **PEFT**: 参数高效微调工具库。支持 LoRA、Prefix Tuning、P-Tuning、Prompt Tuning、AdaLoRA等方法。
4. **LLM-Adapters**: 对 PEFT 的扩展，支持AdapterP、AdapterH、Parallel 等方法。
5. **ChatGLM-6B**: 清华开源中文大模型，及微调训练工具库。
6. **ChatGLM-Efficient-Tuning**: 对 ChatGLM-6B 进行参数高效微调。
7. **LLMZoo**: Phoenix-inst-chat-7b 中文模型（基于BLOOMZ）及微调训练工具库。
8. **BELLE**: 基于 LLaMA 的中文模型及微调训练工具库。
9. **Linly**: 基于 LLaMA 使用中文增量预训练以及微调训练的中文模型。
10. **Chinese-LLaMA-Alpaca**: 基于 LLaMA 使用中文增量预训练以及微调训练的中文模型。

指令微调数据集：<https://zhuanlan.zhihu.com/p/620885226>

### 中文开源大模型

直接可微调，无需指令增量训练：

- ChatGLM-6B: [模型地址](#)
- ChatYuan-large-v2: [模型地址](#)

原始模型多语言or英文，需要中文指令数据集增量训练：

- BloomZ: [模型地址](#)
- LLama: [模型地址](#)
- Flan-T5: [模型地址](#)
- OPT: [模型地址](#)



参考：

Prompt Engineering : <https://lilianweng.github.io/posts/2023-03-15-prompt-engineering/>

COT : <https://arxiv.org/pdf/2201.11903.pdf>

Stanford Alpaca : An Instruction-following LLaMA Model [https://github.com/tatsu-lab/stanford\\_alpaca](https://github.com/tatsu-lab/stanford_alpaca)