

基础模型

核心档案：Qwen1.5-1.8B-Chat

- **开发商：**阿里巴巴（阿里云）
- **品牌：**通义千问 (Tongyi Qianwen)，缩写为 **Qwen**。
- **模型系列：**Qwen1.5 (这是 2024 年初发布的第二代重大更新)
- **模型大小：**1.8B (18 亿参数)
- **模型类型：**Chat (聊天与指令遵循模型)

拆解模型名称：Qwen1.5 - 1.8B - Chat

这个名字里的每一部分都有特定含义，理解它们至关重要：

1. "Qwen1.5" (通义千问 1.5)

这代表它是“通义千问”系列的 **1.5 版本**。

- **Qwen 1.0 (第一代)：**2023 年发布，是阿里的第一版开源模型。
- **Qwen 1.5 (第二代)：**2024 年初发布，**是一次重大的架构升级**。它在很多方面进行了改进，使其性能更强、推理（运行）速度更快，并且与 LLaMA 等主流开源模型（如 Llama 2）在架构上更加兼容。
- **它是一个家族：**Qwen1.5 系列包含多个尺寸，比如 **0.5B、1.8B、4B、7B、14B、32B 和 72B**。你选择的是这个家族中一个非常轻巧的成员。

2. "1.8B" (18 亿参数)

"B" 代表 "Billion" (十亿)，所以 **1.8B** 意味着这个模型由**大约 18 亿个参数 (Parameters) **组成。

- **参数是什么？**你可以把参数理解为模型大脑中的“神经元连接的强度”。模型的知识、推理能力、语言风格……所有的一切都存储在这些数字里。
- **为什么 1.8B 很重要？**
 - **它决定了模型的大小和“智能”基座。** 18 亿在当今动辄 700 亿 (70B) 甚至 1 万亿 (1T) 参数的模型中，属于**“小型模型”** (Small Language Model, SLM)。
 - **它非常适合微调学习！** 它的“小”是它最大的优点：
 1. **显存友好：**原始模型 (16-bit) 大约需要 3.6GB 显存，你使用 4-bit 量化 (QLoRA) 后，模型本体只需要 ~0.9GB 显存！
 2. **训练可行：**正因为它小，我们才能在单张 A10 (24GB 显存) 上，加载模型、加载数据、并**为它创建优化器** (这才是最占显存的部分，你 nvidia-smi 里的 13GB 大部分是它)。
 3. **速度快：**微调 18 亿参数，显然比微调 700 亿参数快得多。

3. "Chat" (聊天模型)

这是理解你的任务**最关键的一个词**。

在开源社区，模型通常分两种发布：

1. **Base (基础模型)：**

- 这是一个“原始”模型。它读了几万亿字的文本，只学会了一件事：“**预测下一个词**”。
- 你给它 “今天天气很”，它会续写 “好，阳光明媚”。
- 但你问它 “今天天气怎么样？”，它可能会续写 “明天天气怎么样？后天呢？” 。它**不理解“回答问题”**这个概念，它只会续写。

2. Chat (聊天模型):

- 这就是你用的模型。阿里巴巴的工程师们已经对 **Base 模型** 进行过了第一轮微调。
- 他们使用了海量的“指令-回答”数据（就像你现在用的 **alpaca_gpt4_zh**），通过**监督式微调 (SFT)** 和**人类反馈强化学习 (RLHF)**，教会了模型如何**“作为助手来回答问题”**。
- 它知道该如何遵循指令、如何组织语言来回答问题，而不是机械地续写。

这对你的任务意味着：你不是从 0 开始教 Qwen 聊天。你是在一个**已经很会聊天的“优等生”的基础上，再用 **alpaca_gpt4_zh** 这套“精品教材”对它进行“风格强化”**，让它聊天的风格更像 GPT-4 生成的中文回答。

💡 它的主要特点

1. **中英双语的“优等生”** **Qwen** 系列从一开始就是中英双语并重训练的，它在中文上的表现通常优于那些以英文为主再“补习”中文的模型（比如 Llama）。
2. **高效的现代架构** **Qwen1.5** 使用了当前最先进的技术，例如：
 - **GQA (Grouped-Query Attention)**：一种注意力机制的变体，能大幅降低推理（聊天）时对显存的占用，并提升速度。
 - **32K 上下文窗口**：它能一次性“阅读”和理解非常长的文本（约 32,000 个 token）。
3. **完全开放，生态成熟** 它使用 **Apache 2.0** 许可证，允许商用。并且它被 **transformers** (Hugging Face 核心库)、**vLLM** (推理加速库) 以及你正在用的 **LLaMA-Factory** 完美支持。

alpaca_gpt4_zh数据集介绍

核心档案：alpaca_gpt4_zh

- **全名：** Alpaca GPT-4 (Chinese version)
- **类型：** 指令微调 (Instruction Fine-Tuning) 数据集
- **来源：** 斯坦福大学 **Alpaca** 项目的社区改进版。
- **语言：** 简体中文
- **质量：** 高 (由 GPT-4 生成或优化)

﴿ 拆解数据集：Alpaca + GPT-4 + zh

1. "Alpaca" (羊驼)：指令微调的开创者

首先，你必须了解什么是 **Alpaca** (羊驼)。

- **起源：** 2023 年 3 月，斯坦福大学的研究者们做了一个轰动性的实验。他们想知道：“我们能用很少的钱，复刻出 ChatGPT (GPT-3.5) 的神奇能力吗？”
- **方法：**
 1. 他们拿了 LLaMA-7B (Meta 的开源基础模型) 作为“底座”。

2. 他们让当时最强的 GPT-3.5 (text-davinci-003) 生成了 **52,000 条** 英语的“指令-回答”数据对。
 3. 他们用这 5.2 万条数据对 LLaMA-7B 进行了微调。
- **结果：**奇迹发生了。只花了不到 600 美元，这个 7B 的小模型在很多任务上的表现**几乎能和 GPT-3.5 媲美**。
 - **结论：****Alpaca** 项目证明了，**基础模型 (Base Model) 的质量 + 高质量指令数据 = 强大的聊天模型**。它开启了全球“指令微调”的浪潮。

2. "zh" (中文): 本地化

Alpaca 原始数据集是全英文的。在你选用的版本中，**zh** 代表**中文 (Chinese)**。

开源社区（主要是 **llama-factory** 项目的作者和其他贡献者）将这 5.2 万条英文指令**翻译成了高质量的简体中文**。

- **英文示例：**
 - **instruction:** "Write a three-sentence poem about Shanghai."
- **中文示例：**
 - **instruction:** "写一个关于上海的三句诗。"

3. "GPT-4" (质量保证): 更高质量的回答

这是这个数据集的**点睛之笔**。

原始的 **Alpaca** 数据集是用 GPT-3.5 生成的 **output** (回答)。但在你用的这个版本里，社区做了进一步优化：

- **回答由 GPT-4 生成或精炼。**
- **为什么这很重要？** GPT-4 在知识深度、推理能力和遵循复杂指令的能力上远超 GPT-3.5。这意味着你用来“喂”给 Qwen 模型的“标准答案”是**更高质量的**。
- 你不是在教你的模型模仿 GPT-3.5，你是在**教它模仿 GPT-4** 的中文回答风格。

数据集结构

这个数据集通常是一个 **.json** 文件 (在你的 **LLaMA-Factory/data/** 目录中，就叫 **alpaca_gpt4_zh.json**)。它由一个巨大的列表构成，列表中的每一项都是一个 JSON 对象，格式如下：

```
[  
  {  
    "instruction": "描述一下什么是人工智能。",  
    "input": "",  
    "output": "人工智能 (Artificial Intelligence, 简称AI) 是指由人制造出来的机器所表现出来的智能。它试图理解智能的实质，并生产出一种新的能以人类智能相似的方式做出反应的智能机器。该领域的研究包括机器人、语言识别、图像识别、自然语言处理和专家系统等。"  
  },  
  {  
    "instruction": "列出三种常见的水果，并说明它们各自的特点。",  
    "input": "",  
    "output": "1. **苹果**：特点是口感脆甜，富含维生素和纤维素，有多种颜色和品种，如红富士、青苹果等。\\n2. **香蕉**：特点是口感软糯，味道香甜，富含钾元素，易于消化，方便携带和食用。\\n3. **橙子**：特点是果肉多汁，酸甜可口，富含维生素C，外皮有特殊香气，有助于开胃。"  
  },  
]
```

```
{  
    "instruction": "根据以下描述，写一个角色简介。",  
    "input": "角色：一个活了三百年的吸血鬼，但他害怕黑暗。",  
    "output": "**姓名**：阿卢卡多·冯·海尔辛 (Alucard von Helsing) \n**年龄**：外观 25  
岁，实际 300 岁\n**种族**：吸血鬼\n**特点**：与传统的吸血鬼不同，阿卢卡多对黑暗有着病态  
的恐惧（暗夜恐惧症）。他总是随身携带大量的蜡烛和提灯，只在黄昏或黎明时分活动。他白天睡在阳  
光无法直射的阁楼里，而不是阴暗的地下室。这种奇特的习性使他既被人类排斥，也被同族视为怪  
胎。"  
}  
]
```

- **instruction**: 核心的**指令**，即你希望模型完成的任务。
- **input**: **可选的上下文**。如果指令本身足够清晰（如“描述什么是 AI”），这里就是空的。如果指令需要补
充信息（如“根据以下描述...”），这里就会提供补充信息。
- **output**: **标准答案**。这就是你希望模型学习并模仿的高质量回答。

⌚ 总结

你选择的 **alpaca_gpt4_zh** 是一个**经典的、高质量的中文指令数据集**。

你的微调任务，就是强迫 **Qwen-1.8B-Chat** 模型去**“背诵”和“模仿”这数万条由 GPT-4 撰写的中文回答。通过这个过程，它会学会如何更准确、更丰富、更像一个高级 AI 助手那样来响应你的各种中文指令**。

微调的基本原理

你的任务分为两个层面：“**你要求它做什么**”(任务目标) 和 “**它在技术上如何实现**”(微调原理)。

1. 你的具体微调任务是什么？

你正在执行一个**监督式微调 (Supervised Fine-Tuning, SFT)** 任务，更通俗地讲，也常被称为**指令微调 (Instruction Fine-Tuning)**。

- **目标**：让一个已经会聊天的基础模型 (**Qwen1.5-1.8B-Chat**)，学习并模仿一个高质量数据集 (**alpaca_gpt4_zh**) 的回答风格和知识。
- **打个比方**：
 - 你的**基础模型** (**Qwen-1.8B-Chat**) 像一个已经掌握了大量知识、会说中文的“优等生”。
 - 你的**数据集** (**alpaca_gpt4_zh**) 像一本“标准答案辅导书”，里面的例题和答案都是由更强的“宗师” (GPT-4) 编写的。
 - 你的**微调任务**，就是强迫这个“优等生”放弃自己的一些回答习惯，转而去模仿“标准答案辅导书”里的风格和措辞。

训练过程如下：

1. **提问**：程序从 **alpaca_gpt4_zh** 数据集中取出一个“指令”，例如：“**instruction**”: “**描述一下什么是人工智能。**”
2. **回答**：程序让你的 Qwen 模型尝试回答这个问题。
3. **对比**：程序拿出数据集中的“标准答案”(**“output”**: “**人工智能 (Artificial Intelligence...)** ”)。

4. **学习 (反向传播):** 程序计算出 Qwen 模型的回答与“标准答案”之间的“差距”(学术上称为**损失 Loss**，这就是你监控中看到的 **Loss** 值)。
5. **调整:** 程序根据这个“差距”，**微小地调整**模型的参数，告诉它：“下次再遇到类似问题，你应该回答得更像这个‘标准答案’。”

你设置了 **Epochs: 3.0**，意味着程序会把这本“辅导书”里的所有题目带着模型完整学习 3 遍，以确保它“学得更牢”。

2. 所用微调的细节和原理是什么？

这是最关键的部分。你**没有**在执行传统的“全参数微调”，而是使用了一种更高效、更先进的技术，它的名字叫**QLoRA**。

你在 Web UI 上的设置 **Finetuning method: lora + Quantization: 4-bit**，这两个选项组合起来就是 QLoRA。

为什么需要 QLoRA？

一个 1.8B (18 亿) 参数的模型，如果进行“全参数微调”，需要天文数字的显存。

- 模型参数本身 ($1.8B * 16\text{-bit}$) 约 3.6GB。
- 但训练时，还需要存储**梯度 (Gradients)**和**优化器状态 (Optimizer States)** (如 AdamW)。
- 优化器状态通常是参数量的 12 到 16 倍！这意味着你可能需要 **$3.6\text{GB} * 16 = 57.6\text{GB}$** 以上的显存。
- 你的 A10 只有 24GB 显存，**完全不可能**装下。

QLoRA 就是用来解决这个问题的。它由两个核心技术组成：

A. "Q" = 4-bit Quantization (4位量化)

- **原理:** 这是一种极致的内存压缩技术。
- **细节:** 它没有以完整的 16-bit 精度 (一个数字占 16 位) 加载 1.8B 的模型参数，而是将每个参数“压缩”到**仅仅 4-bit**。
- **效果:**
 1. **显存极大节省:** 1.8B 的基础模型在显存中的占用，从约 3.6GB 骤降到 1GB 以下。
 2. **模型冻结:** 在整个训练过程中，这 1.8B 个 4-bit 参数被**完全冻结 (Frozen)**，它们不会被更新。这节省了存储梯度的海量空间。

B. "LoRA" = Low-Rank Adaptation (低秩适配)

- **原理:** 既然 18 亿个原始参数都被冻结了，我们还怎么训练模型呢？LoRA 的思想是：“**我不改动你原来的知识，我在旁边加挂‘补丁’。**”
- **细节:**
 1. LoRA 会在模型的核心层 (主要是注意力层) 旁边，注入**两个非常小的、新加的“适配器”矩阵 (Adapter Matrices)**。
 2. 在训练时，那 18 亿个 4-bit 参数被冻结，**只有这些新加的、小小的“适配器”参数是可训练的。**
 3. 这些“适配器”的参数量非常小，可能只有几百万个，仅占原始模型参数量的 0.1% 左右。
- **效果:**

1. **显存达标：** 因为你只需要为这区区几百万个“适配器”参数计算梯度和存储优化器状态，所以显存占用大幅下降。你 `nvidia-smi` 中看到的 13GB 显存，大部分就是这些适配器参数及其优化器占用的。
2. **高效微调：** 你成功地在 24GB 显存内，实现了对 18 亿参数模型的“间接”微调。
3. **结果分离：** 训练结束后，你保存的“模型”(在 `train_2025...` 目录里)，**不是一个新的 18 亿参数模型**，而仅仅是那些被训练好的“适配器”文件（通常只有几十 MB）。

所以，

你的具体任务是**监督式微调 (SFT)**。

你所用的原理是 **QLoRA**：

1. 你把 18 亿参数的 Qwen 模型**量化 (Quantize)** 成 4-bit 塞进显存并**冻结**。
2. 你**只训练外挂的、轻量级的 LoRA 适配器**。
3. 通过训练这个“小补丁”，来达到“撬动”整个大模型回答风格的目的。

这是一个在显存、效率和效果之间取得了完美平衡的方案。