第七章——指令微调

李若锦

武汉大学数学与统计学院

2024 年 11 月 17 日

指令微调简介

预训练模型的局限性

1. 指令跟随能力差 缺乏对复杂指令或任务的精准把握，不善于根据用户给出的明确指令执行任务
2. 容易产生幻觉 训练数据中可能包含错误、偏差或不一致的信息，导致有时会生成看似合理但实际上不准确或不存在的信息
3. 信息过时 完成训练后难以更改内部的知识表示，无法获取最新信息
4. 缺乏专业领域知识 某些领域的专业知识或特定语料在训练中没有得到充分覆盖，导致在相应领域的知识不足

指令微调的概念

指令微调（Instruction Tuning）是指使用自然语言形式的数据对预训练后的大语言模型进行参数微调，主要有以下作用：

1. 改进整体任务性能
2. 增强任务求解能力
3. 领域专业化适配

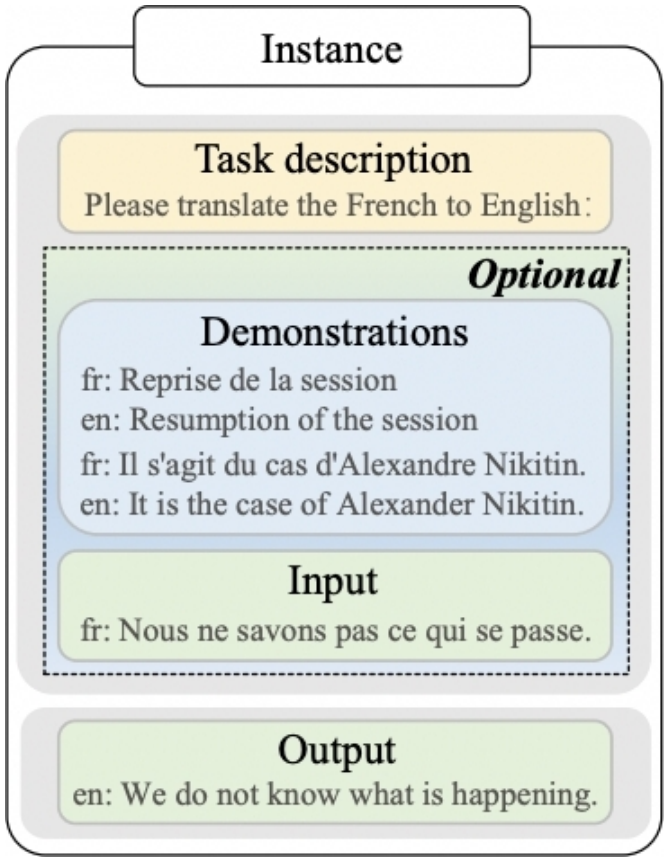
指令微调的基本流程

1. 准备预训练模型 选择一个适合任务的预训练模型，如 GPT-3、BERT
2. 准备微调数据集 收集或生成一个包含指令和相应期望输出的数据集
3. 微调模型 使用训练集对预训练模型进行微调 (FFT 或 BEFT)
4. 评估和测试 在测试集上评估微调后的模型性能

指令数据的构建

指令数据集的结构

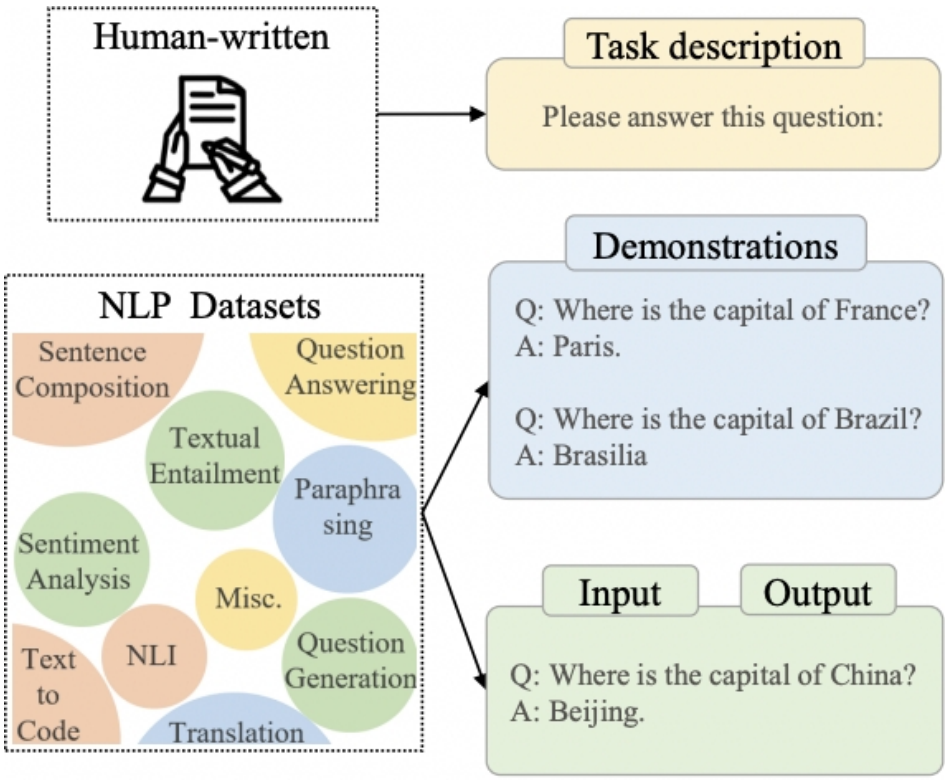
经过指令格式化的数据实例包括任务描述（**指令**）、任务**输入** (可选)-任务**输出**以及示例 (可选)



实例格式

基于现有的 NLP 任务数据集构建

为传统的 NLP 数据集添加任务描述，将其格式为指令微调数据集

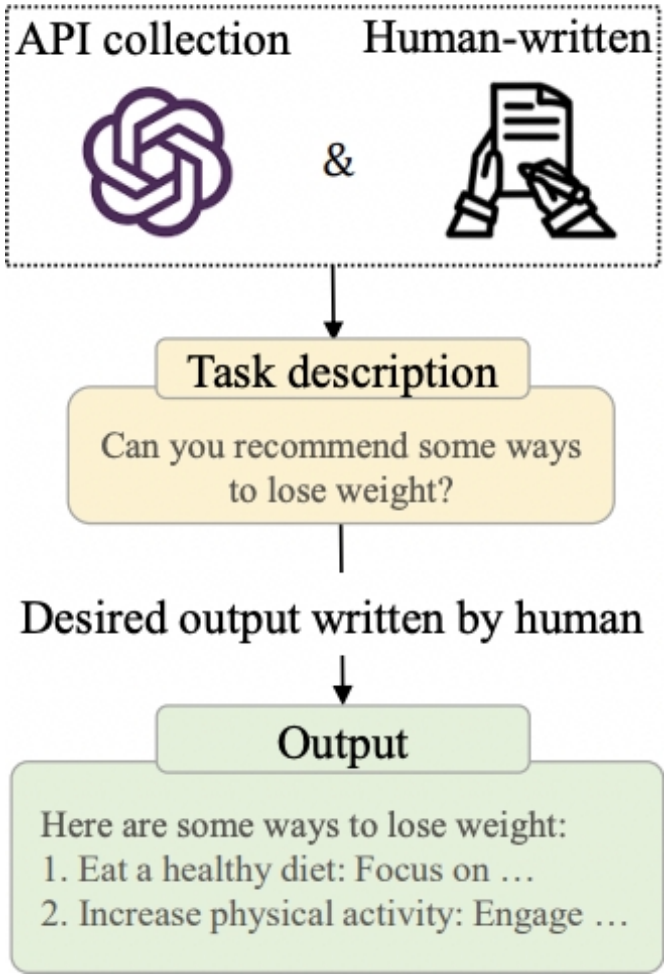


(b) Formatting existing datasets

格式化 NLP 数据集

基于日常对话数据构建

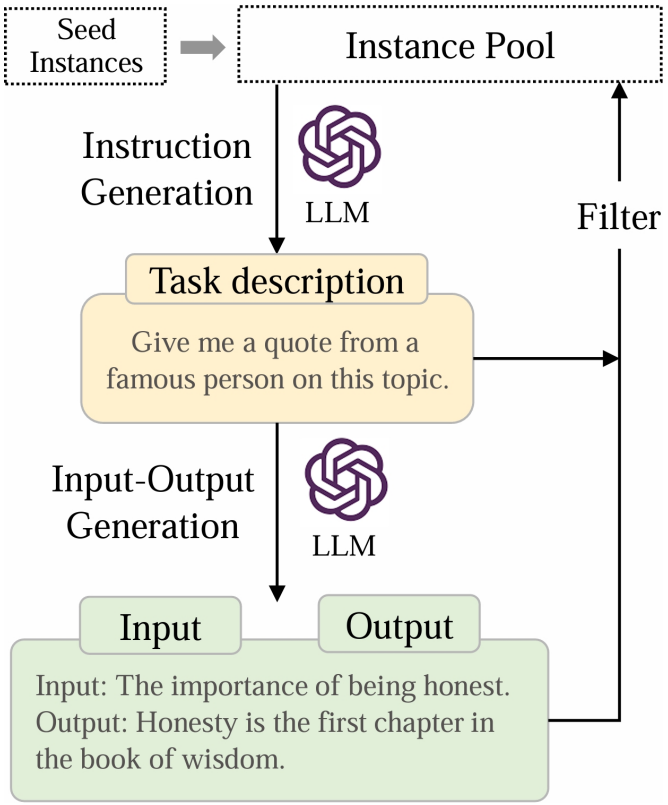
使用用户在日常对话中的实际需求作为任务描述，人工编写答案作为输出，将任务描述和输出配对为一个实例



格式化日常对话数据集

基于合成数据构建

借助已有的高质量指令数据作为示例输入大语言模型，进而生成大量多样化的任务描述和输入-输出数据



格式化合成数据集

Self-Instruct

Self-Instruct 执行时的指令数据集结构

指令集 数据集中包含一个指令集合 $\{I_t\}$, 其中的每个指令 I_t 描述模型需要完成的特定任务，这些任务以自然语言形式表达

输入-输出实例 每个任务 t 包含 $n_t \geqslant 1$ 个实例 $(X_{ti}, Y_{ti})\,(i = 1, \cdots, n_t)$, 其中 X_{ti} 为输入,Y_{ti} 为期望输出

模型的期望行为 模型 M 在给定任务指令 I_t 和对应输入 X_{ti} 的情况下，应当生成期望输出 Y_{ti}

$$M(I_t, X_{ti}) = Y_{ti} \qquad i = 1, \cdots, n_t$$

Self-Instruct

Self-Instruct 执行时的具体步骤

初始化任务池 选取 175 个人工编写的任务作为初始任务池，每个任务包含一个指令和一个实例 (输入 -输出对)

1. 生成指令 从任务池中随机采样 8 个任务指令作为上下文示例，其中 6 个来自人工编写的任务，2 个来自之前步骤中生成的指令，通过以下形式提交给模型用于生成新的指令

```
Come up with a series of tasks:

Task 1:  {instruction for existing task 1}
Task 2:  {instruction for existing task 2}
Task 3:  {instruction for existing task 3}
Task 4:  {instruction for existing task 4}
Task 5:  {instruction for existing task 5}
Task 6:  {instruction for existing task 6}
Task 7:  {instruction for existing task 7}
Task 8:  {instruction for existing task 8}
Task 9:
```

新指令生成模板

Self-Instruct 执行时的具体步骤

2. 分类任务识别 通过初始任务池中的 12 个分类任务指令和 19 个非分类任务指令来学习任务的类型，识别生成的指令是否属于分类任务

```
Can the following task be regarded as a classification task with finite output labels?

Task:  Given my personality and the job, tell me if I would be suitable.
Is it classification?  Yes

Task:  Give me an example of a time when you had to use your sense of humor.
Is it classification?  No

...

Task:  To make the pairs have the same analogy, write the fourth word.
Is it classification?  No

Task:  Given a set of numbers, find all possible subsets that sum to a given number.
Is it classification?  No

Task:  {instruction for the target task}
```

分类识别模板

Self-Instruct 执行时的具体步骤

3. 生成实例 对分类问题采用"Input-first"
对非分类问题采用"Output-first"

```
Come up with examples for the following tasks. Try to generate multiple examples when possible.
If the task doesn't require additional input, you can generate the output directly.

Task: Which exercises are best for reducing belly fat at home?
Output:
- Lying Leg Raises
- Leg In And Out
- Plank
- Side Plank
- Sit-ups
Task: Sort the given list ascendingly.
Example 1
List: [10, 92, 2, 5, -4, 92, 5, 101]
Output: [-4, 2, 5, 5, 10, 92, 92, 101]
Example 2
Input 2 - List: [9.99, 10, -5, -1000, 5e6, 999]
Output: [-1000, -5, 9.99, 10, 999, 5e6]
...
Task: {Instruction for the target task}
```

```
Given the classification task definition and the class labels, generate an input that
corresponds to each of the class labels. If the task doesn't require input, just generate the
correct class label.

Task: Classify the sentiment of the sentence into positive, negative, or mixed.
Class label: mixed
Sentence: I enjoy the flavor of the restaurant but their service is too slow.
Class label: Positive
Sentence: I had a great day today. The weather was beautiful and I spent time with friends.
...

Task: Which of the following is not an input type? (a) number (b) date (c) phone number (d)
email address (e) all of these are valid inputs.
Class label: (e)

Task: {instruction for the target task}
```

实例生成模板

Self-Instruct

Self-Instruct 执行时的具体步骤

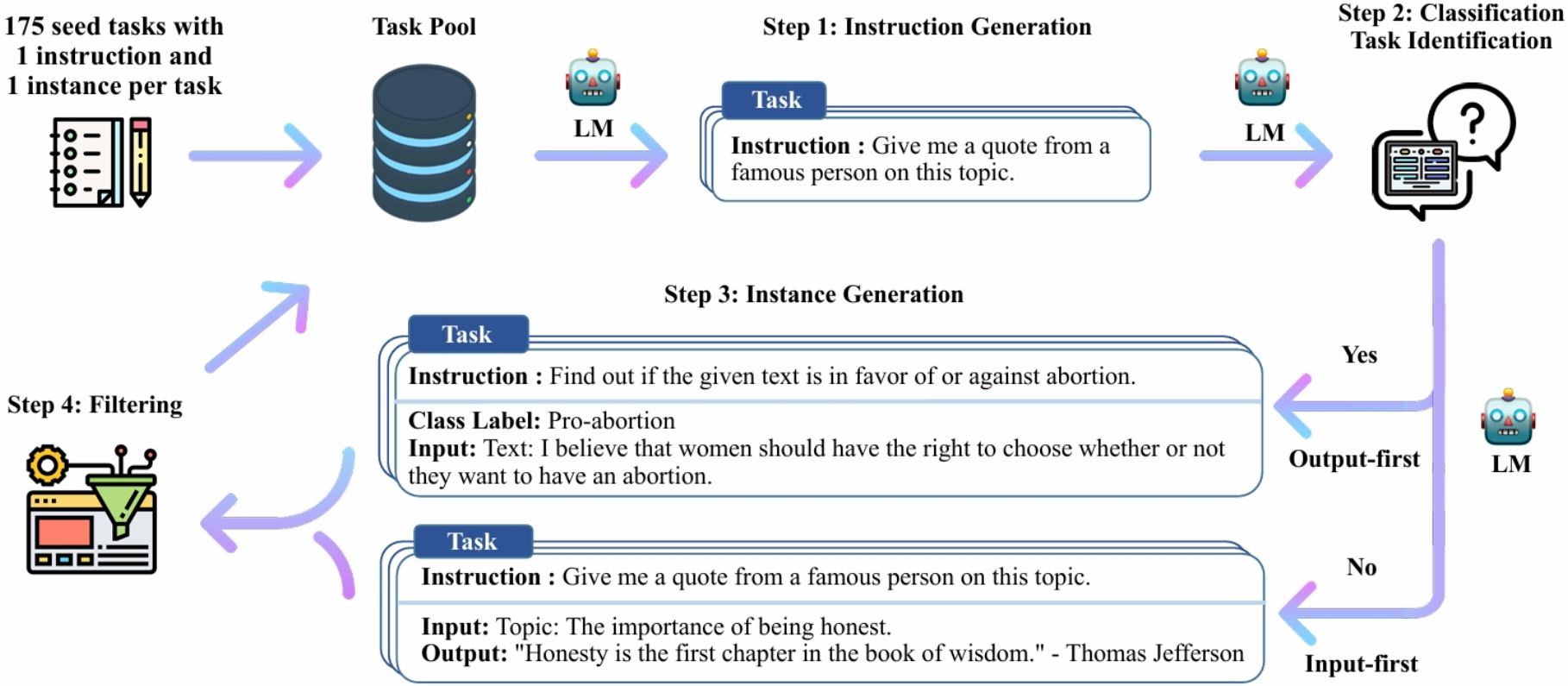
4. 筛选过滤 (1) 相似性过滤 将与其他指令的 ROUGE-L 相似度高于 0.7 的指令剔除

$$\text{ROUGE-L} = \frac{(1+\beta^2)\cdot \text{Precision}\cdot \text{Recall}}{\beta^2\cdot \text{Precision}+\text{Recall}}$$

(2) 特定关键字过滤 剔除含 image picture graph 等关键词的指令

(3) 无效生成过滤 基于一些启发式规则来识别和过滤无效的生成内容

Self-Instruct



Self-Instruct 流程图

Evol-Instruct

给定初始指令数据集,$D^{(0)} = \left\{I_k^{(0)}, R_k^{(0)}\right\}$, 其中 I_k 为第 k 条指令,$R_k^{(0)}$ 为对应的响应 (输入-输出对), 按以下方式执行演化:

1. 在每次演化中, 先通过大模型的指令进化提示 (深度演化/广度演化) 将 $I_k^{(t)}$ 升级为 $I_k^{(t+1)}$
2. 根据新的指令通过大模型生成相应的响应 $R_k^{(t+1)}$
3. 淘汰掉未能成功进化的指令, 由此得到新的指令数据集 $D^{(t+1)}$

Evol-Instruct

深度演化 通过五种类型 (添加约束、深化、具体化、增加推理步骤以及使输入复杂化) 的提示来增强指令，使其变得更加复杂和困难

```
I want you act as a Prompt Rewriter.
Your objective is to rewrite a given prompt into a more complex version to make those famous AI systems
(e.g., ChatGPT and GPT4) a bit harder to handle.
But the rewritten prompt must be reasonable and must be understood and responded by humans.
Your rewriting cannot omit the non-text parts such as the table and code in #Given Prompt#:.  Also, please
do not omit the input in #Given Prompt#.
You SHOULD complicate the given prompt using the following method:
Please add one more constraints/requirements into #Given Prompt#
You should try your best not to make the #Rewritten Prompt# become verbose, #Rewritten Prompt# can only
add 10 to 20 words into #Given Prompt#.
‘#Given Prompt#’, ‘#Rewritten Prompt#’, ‘given prompt’ and ‘rewritten prompt’ are not allowed to appear in
#Rewritten Prompt#
#Given Prompt#:
<Here is instruction.>
#Rewritten Prompt#:
```

深度演化模板

Evol-Instruct

广度演化 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

```
I want you act as a Prompt Creator.
Your goal is to draw inspiration from the #Given Prompt# to create a brand new prompt.
This new prompt should belong to the same domain as the #Given Prompt# but be even more rare.
The LENGTH and difficulty level of the #Created Prompt# should be similar to that of the #Given Prompt#.
The #Created Prompt# must be reasonable and must be understood and responded by humans.
‘#Given Prompt#’, ‘#Created Prompt#’, ‘given prompt’ and ‘created prompt’ are not allowed to appear in
#Created Prompt#.
#Given Prompt#:
<Here is instruction.>
#Created Prompt#:
```

广度演化模板

指令微调的训练策略

优化设置

目标函数 采用的目标函数为序列到序列损失，仅在输出部分计算损失，而不计算输入部分的损失

$$\text{Loss} = -\log P\left(y_1, \cdots, y_T | X\right) = -\sum_{t=1}^{T} \log P\left(y_t | y_{<t}, X\right)$$

批次大小和学习率 只需要使用较小的批次大小和学习率

e.g :InstructGPT(175B) 批次大小为 8，学习率恒定为 5.03×10^{-6}；Alpaca (7B) 批次大小为 128，学习率预热到 2×10^{-5}，然后采用余弦衰减策略

$$\alpha_t = \frac{1 + \cos \frac{t\pi}{T}}{2} \alpha_0$$

多轮对话数据 一个对话的多轮内容一次性输入模型，通过设计损失掩码来实现仅针对每轮对话的模型输出部分进行损失计算

数据组织策略

1. 平衡数据分布 混合使用现有的多个指令数据集，以此来实现模型能力的综合改进
2. 多阶段指令微调 首先使用大规模 NLP 任务指令数据对模型进行微调，然后再使用相对多样的日常对话指令和合成指令进一步微调
3. 结合预训练数据 引入预训练数据和任务，对模型进行正则化

参数高效微调

LoRA

考虑模型中的参数矩阵 $W \in \mathbb{R}^{d_{model} \times d_{model}}$, 表示成以下形式

$$W = W_0 + \Delta W$$

其中 W_0 为预训练得到的参数矩阵,ΔW 为更新量, 对 ΔW 作低秩分解

$$\Delta W = AB^{\mathrm{T}}$$

其中 $A, B \in \mathbb{R}^{d_{model} \times r}$,$r \ll d_{model}$, 按以下方式计算中间状态

$$h = Wx = W_0 x + AB^{\mathrm{T}} x$$

训练完成后, 将原始参数矩阵 W_0 和训练得到的 A, B 合并

$$W = W_0 + AB^{\mathrm{T}}$$

全量微调与 LoRA 所需显存比较

	全量微调	LoRA
模型参数	$2P$	$2P + 2P_{\text{LoRA}}$
模型梯度	$2P$	$2P_{\text{LoRA}}$
优化器模型参数	$4P$	$4P_{\text{LoRA}}$
优化器一阶动量	$4P$	$4P_{\text{LoRA}}$
优化器二阶动量	$4P$	$4P_{\text{LoRA}}$
总计	$16P$	$2P + 16P_{\text{LoRA}}$

AdaLoRA

对更新量进行 SVD 分解

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

其中 $P \in \mathbb{R}^{d_{model} \times r}, Q \in \mathbb{R}^{r \times d_{model}}$ 分别为左右奇异向量构成的矩阵, $\Lambda \in \mathbb{R}^{r \times r}$ 为包含奇异值的对角阵. 训练时初始化 Λ 为零矩阵,P, Q 采用随机初始化, 并引入一下正则项确保正交性

$$R(P, Q) = \left\| P^{\mathrm{T}} P - I_r \right\|_F^2 + \left\| Q Q^{\mathrm{T}} - I_r \right\|_F^2$$

用 $k(k = 1, \cdots, n)$ 来索引参数矩阵, 设 $\Delta W_k = P_k \Lambda_k Q_k$, 将训练参数的集合表示为

$$\mathcal{P} = \{P_k\}_{k=1}^n \quad \mathcal{E} = \{\Lambda_k\}_{k=1}^n \quad \mathcal{Q} = \{Q_k\}_{k=1}^n$$

定义损失函数为

$$\mathcal{L}(\mathcal{P}, \mathcal{E}, \mathcal{Q}) = C(\mathcal{P}, \mathcal{E}, \mathcal{Q}) + \gamma \sum_{k=1}^{n} R(P_k, Q_k)$$

其中 C 为训练成本函数

AdaLoRA

对时间步 t, 采用随机梯度下降更新 $P_k^t, \Lambda_k^t, Q_k^t$

$$P_k^{t+1} = P_k^t - \eta_1 \nabla_{P_k} \mathcal{L}(\mathcal{P}^t, \mathcal{E}^t, \mathcal{Q}^t)$$

$$Q_k^{t+1} = Q_k^t - \eta_2 \nabla_{Q_k} \mathcal{L}(\mathcal{P}^t, \mathcal{E}^t, \mathcal{Q}^t)$$

$$\widetilde{\Lambda_k^t} = \Lambda_k^t - \eta_3 \nabla_{\Lambda_k} \mathcal{L}(\mathcal{P}^t, \mathcal{E}^t, \mathcal{Q}^t)$$

对 $\widetilde{\Lambda_k^t}$ 作剪枝

$$\Lambda_k^{t+1} = \mathcal{T}\left(\widetilde{\Lambda_k^t}, S_k^t\right)$$

其中 $\mathcal{T}\left(\widetilde{\Lambda_k^t}, S^t\right)_{ii} = \begin{cases} \widetilde{\Lambda_{kii}^t} & S_{ki}^t\text{的大小在 } S_k^t \text{ 的前 } b^t \text{ 个} \\ 0 & \text{其他} \end{cases}$

$S^t = \left\{S_{ki}^t\right\}_{1 \leqslant k \leqslant n, 1 \leqslant i \leqslant r}$ 为每一个参数矩阵每一个奇异值的重要性得分,

b^t 为奇异值预算

重要性得分 S_{ki} 的取法

1. 基于奇异值大小 $S_{ki} = |\lambda_{ki}|$

2. 基于敏感度 $S_{ki} = s(\lambda_{ki}) + \frac{1}{d_{model}} \sum_{j=1}^{d_{model}} s(P_{kji}) + \frac{1}{d_{model}} \sum_{j=1}^{d_{model}} s(Q_{kij})$

其中 $s(\cdot)$ 为单个元素的重要性函数，可表示为敏感度

$$s(\omega_{ij}) = I(\omega_{ij}) = \left|\omega_{ij} \nabla_{\omega_{ij}} \mathcal{L}\right|$$

3. 敏感度和不确定性平滑化

$$\bar{I}^t(\omega_{ij}) = \beta_1 \bar{I}^{t-1}(\omega_{ij}) + (1-\beta_1) I^t(\omega_{ij})$$

$$\bar{U}^t(\omega_{ij}) = \beta_2 \bar{U}^{t-1}(\omega_{ij}) + (1-\beta_2 \left|I^t(\omega_{ij}) - \bar{I}(\omega_{ij})\right|)$$

其中 β_1, β_2 为平滑化系数，重要性函数表示为

$$s(\omega_{ij}) = \bar{I}^t(\omega_{ij}) \bar{U}^t(\omega_{ij})$$

QLoRA

基本思想 将预训练参数转换为 k 比特格式存储 (一般 $k = 4$)

1. 计算 $2^k + 1$ 个分位点

$$q_i = \frac{1}{2}\left(Q_X\left(\frac{i}{2^k+1}\right) + Q_X\left(\frac{i+1}{2^k+1}\right)\right) \quad i = 0, 1, \cdots, 2^k$$

其中 Q_X 为标准正态分布 $N(0, 1)$ 的分位数函数

2. 分位点标准化 将 q_0 到 q_{2^k} 标准化到 $[-1, 1]$ 范围

$$q_i' = -1 + 2 \cdot \frac{q_i - q_0}{q_{2^k} - q_0}$$

3. 参数最大绝对值重缩放 将模型参数最大绝对值重缩放到 $[-1, 1]$ 范围

$$x = \frac{x}{\max|x|}$$

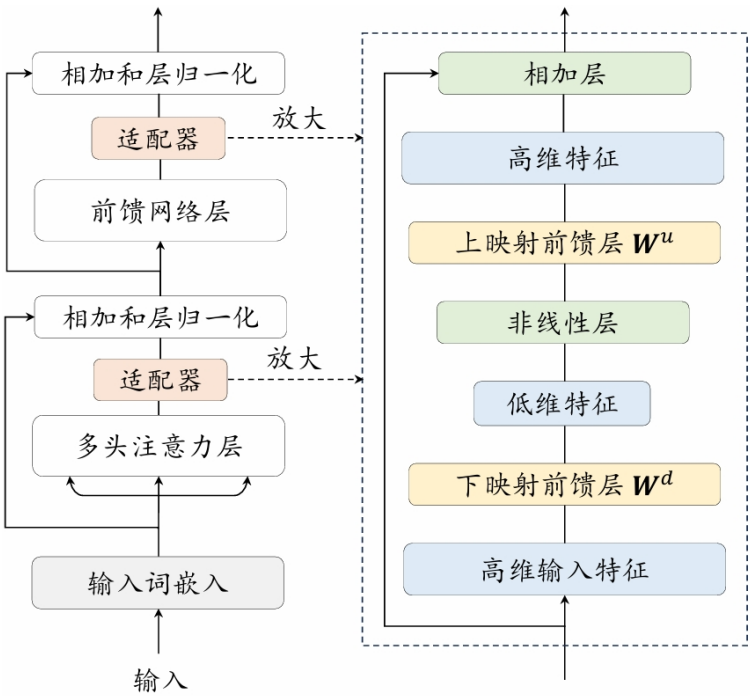
4. 量化映射 将参数 x 映射到最近的分位点对应的离散等级，用离散等级对应的 k 比特格式存储数据

Adapters Tuning

在多头注意力层和前馈网络层之后、层归一化之前加入瓶颈网络架构(称之为适配器)

$$h = h + \sigma\left(hW^d\right)W^u$$

其中 $W^d \in \mathbb{R}^{d_{model} \times r}, W^u \in \mathbb{R}^{r \times d_{model}}$, 且 $r \ll d_{model}$
训练中保持原始参数不变, 只更新适配器参数



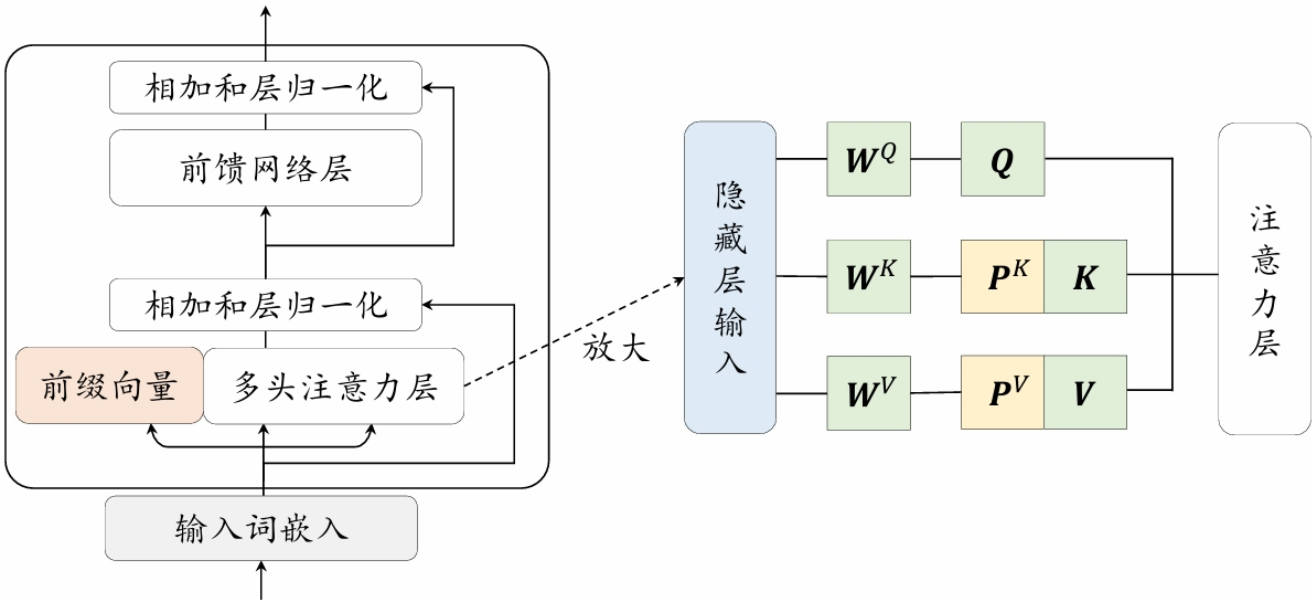
适配器微调

Prefix Tuning

在每个注意力的键与值之前拼接一系列前缀 (可训练的嵌入向量)，改用以下形式计算注意力

$$\text{head} = \text{Attention}\left(XW^Q, P^K \oplus XW^K, P^V \oplus XW^V\right)$$

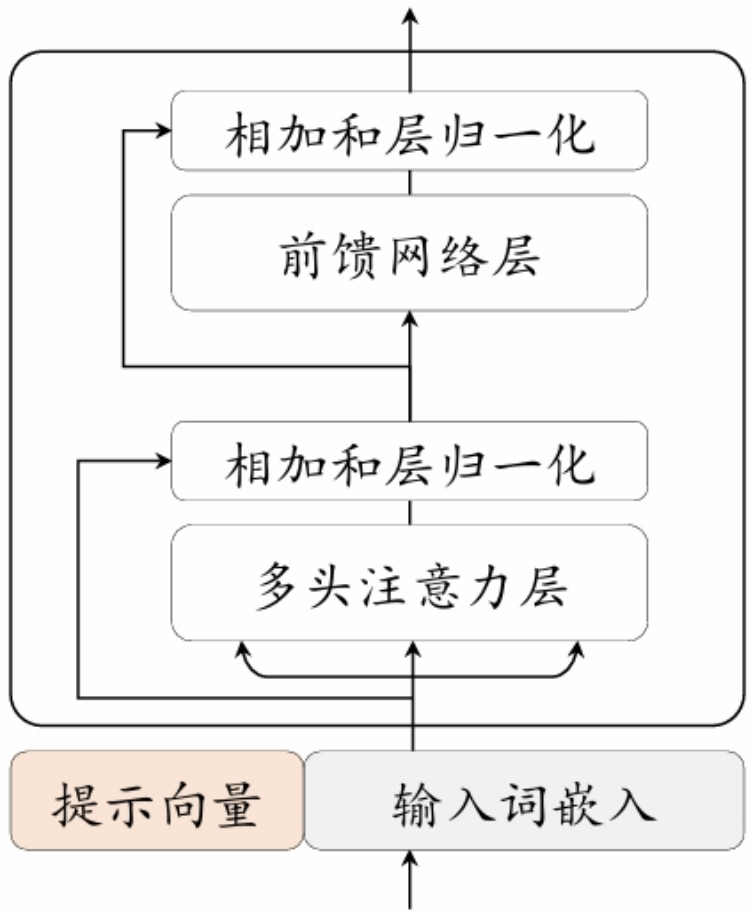
其中前缀可重参数化为 $P = \text{MLP}_\theta(P')$
训练中保持原始参数不变，只更新前缀参数



前缀微调

Prompt Tuning

直接在输入前拼接前缀 (提示向量/vitural tokens)，只训练提示向量参数



提示微调

P-Tuning

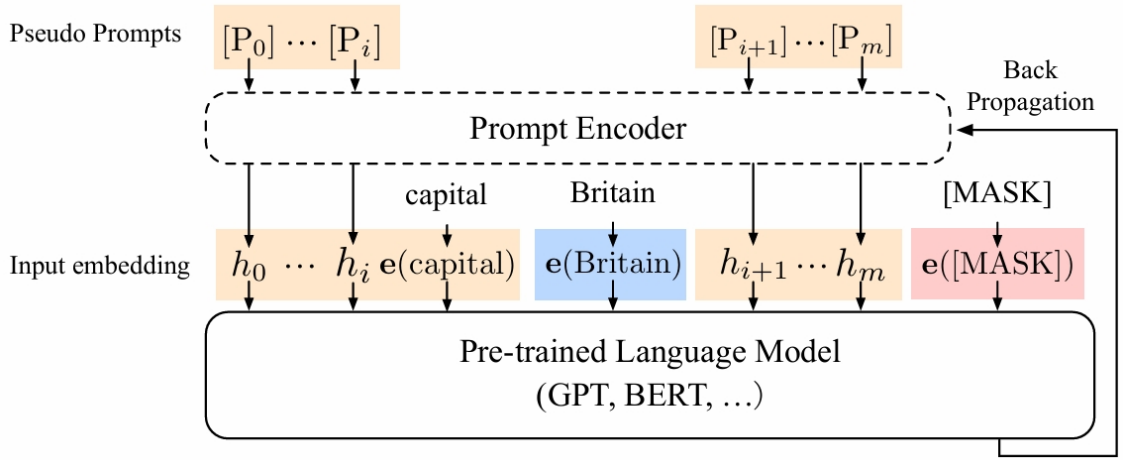
使用自由形式来组合输入文本与提示向量

$$\left\{\left[P_{0:i}\right], x, \left[P_{(i+1):j}\right], y, \left[P_{(j+1):k}\right]\right\}$$

用 LSTM 或 MLP 对提示向量再进行一次编码，$f_\theta : [P_i] \rightarrow h_i$，并将输入文本 x 和标签 y 转为嵌入向量

$$T = \{h_0, \cdots, h_i, e(x), h_{i+1}, \cdots, h_j, e(y), h_{j+1}, \cdots, h_k\}$$

将 T 作为输入，只训练 P 和 θ



P-Tuning