使用 LLaMA-Factory 微调 llama3 模型

https://github.com/hiyouga/LLaMA-Factory

LLaMA-Factory 是一个用于大型语言模型（LLM）微调的工具，它旨在简化大型语言模型的微调过程，使得用户可以快速地对模型进行训练和优化，以提高模型在特定任务上的性能。

这个工具支持多种预训练的大型语言模型，例如 LLaMA、LLaVA、Mistral、Mixtral-MoE、Qwen、Yi、Gemma、Baichuan、ChatGLM 和 Phi 等。

LLaMA-Factory 的特点包括：

1. **支持多种微调方法**：它集成了连续预训练、有监督微调(SFT)、偏好对齐（RLHF）等多种微调方法。
2. **高效的微调技术**：与 ChatGLM 官方的 P-Tuning 微调相比，LLaMA Factory 的 LoRA 微调提供了显著的加速比，并且在特定任务上取得了更高的性能分数。
3. **易用性**：LLaMA-Factory 提供了高层次的抽象接口，使得开发者可以开箱即用，快速上手操作。
4. **WebUI 支持**：借鉴 Stable Diffusion WebUI，该项目提供了基于 gradio 的网页版工作台，方便初学者可以迅速上手操作。
5. **模型导出和推理**：支持模型的导出和推理，包括动态合并 LoRA 模型进行推理。
6. **API Server**：支持启动 API Server，使得训练好的模型可以通过网络接口被远程访问和调用。
7. **评测 benchmark**：提供了主流评测 benchmark 支持，如 mmlu、cmmlu、ceval 等，用于评估模型的泛化能力。

LLaMA-Factory 旨在降低大型语言模型微调的门槛，使得更多的研究者和开发者能够利用这些强大的模型来解决具体的实际问题。

配置环境

/mnt/workspace路径下执行

```
git clone https://github.com/hiyouga/LLaMA-Factory.git
conda create -n llama_factory python=3.10
conda activate llama_factory
cd LLaMA-Factory
pip install -e .[metrics]
```

下载llama3模型文件

/mnt/workspace路径下执行

```
mkdir models
cd models
```

/mnt/workspace/models路径下执行

```
pip install modelscope
git clone https://www.modelscope.cn/LLM-Research/Meta-Llama-3-8B-Instruct.git
```

使用 SHA-256 算法检查文件（可选）

```
shasum -a 256 model-00001-of-00004.safetensors

shasum -a 256 model-00002-of-00004.safetensors

shasum -a 256 model-00003-of-00004.safetensors

shasum -a 256 model-00004-of-00004.safetensors
```

项目实战1：llama3中文增强大模型(lora微调)

使用预训练模型+LoRA适配器的方式, 可以在不重新训练整个模型的情况下, 快速且高效地将模型适应到新的任务或领域, 如将英文模型适应到中文对话。这是一种常见且实用的微调方法。

训练

修改文件 examples/lora_single_gpu/llama3_lora_sft.yaml

```yaml
### model
model_name_or_path: /mnt/workspace/models/Meta-Llama-3-8B-Instruct

### method
stage: sft
do_train: true
finetuning_type: lora
lora_target: q_proj,v_proj

### dataset
dataset: alpaca_gpt4_zh
template: llama3
cutoff_len: 1024
max_samples: 1000
overwrite_cache: true
preprocessing_num_workers: 16

### output
output_dir: /mnt/workspace/models/llama3-lora-zh
logging_steps: 100
save_steps: 500
plot_loss: true
overwrite_output_dir: true

### train
per_device_train_batch_size: 1
gradient_accumulation_steps: 8
learning_rate: 0.0001
num_train_epochs: 1.0
lr_scheduler_type: cosine
warmup_steps: 0.1
fp16: true

### eval
val_size: 0.1
per_device_eval_batch_size: 1
evaluation_strategy: steps
```

```
eval_steps: 500
```


解释这个yaml文件中的主要参数:

model

- model_name_or_path: 指定预训练模型的路径或名称, 这里使用的是 /mnt/workspace/models/Meta-Llama-3-8B-Instruct 路径下的 Meta-Llama 3.8B 模型作为基础模型。

method

- stage: 表示训练阶段, 这里是 sft(Supervised Fine-Tuning)。
- do_train: 表示要进行训练。
- finetuning_type: 微调方法, 这里使用 LoRA(Low-Rank Adaptation)。
- lora_target: LoRA 作用的对象, 这里是 attention 层的 query 和 value 映射矩阵。

dataset

- dataset: 训练数据集名称,这里用的是 alpaca_gpt4_zh 数据集。 （注意：数据集可以选择多个 alpaca_zh,alpaca_gpt4_zh,oaast_sft_zh）
- template: 数据集的格式模板,这里是 llama3。
- cutoff_len: 输入序列的最大长度, 超过会截断, 这里设为1024。
- max_samples: 从数据集中取的最大样本数, 这里取前1000个样本。
- preprocessing_num_workers: 数据预处理的进程数, 这里用16个进程并行处理。

output

- output_dir: 训练日志和模型的保存路径。
- logging_steps: 每隔多少步记录一次日志, 这里每100步记录一次。
- save_steps: 每隔多少步保存一次模型, 这里每500步保存一次。
- plot_loss: 是否绘制loss曲线图。

train

- per_device_train_batch_size: 每个设备上的训练批大小, 这里设为1。
- gradient_accumulation_steps: 梯度累积的步数, 这里累积8步才更新一次模型参数。
- learning_rate: 学习率, 这里设为0.0001。
- num_train_epochs: 训练的epoch数, 这里训练1个epoch。**(还可以指定max_steps 3000 : 训练3000步。注意：真正项目训练增大总步数；注意：num_train_epochs和max_steps之中选择一个)**
- lr_scheduler_type: 学习率调度器类型, 这里用cosine, 即先warmup再降温。
- warmup_steps: 用初始学习率warmup的步数, 这里取总步数的10%。
- fp16: 是否使用fp16混合精度训练, 以加速训练并减少显存占用。

eval

- val_size: 从训练集中划分出的验证集比例, 这里取10%作为验证集。
- per_device_eval_batch_size: 验证时每个设备的批大小,这里为1。
- evaluation_strategy: 验证策略, 这里根据steps数来验证。
- eval_steps: 每隔多少步验证一次, 这里每500步验证一次。

以上就是这个yaml配置文件的主要参数解释。这些参数设置了一个使用LoRA对Meta-Llama-3-8B-Instruct模型在alpaca_gpt4_zh数据集上进行微调的训练过程。

克隆数据集并修改dataset_info.json

在/mnt/workspace/LLaMA-Factory/data路径下使用下面的命令克隆alpaca_gpt4_zh数据集

```
git clone https://www.modelscope.cn/datasets/llamafactory/alpaca_gpt4_zh.git
```

然后，修改dataset_info.json文件

```
  "alpaca_gpt4_zh": {
    "file_name": "alpaca_gpt4_data_zh.json"
  },
```

/mnt/workspace/LLaMA-Factory路径下执行

```
llamafactory-cli train examples/lora_single_gpu/llama3_lora_sft.yaml
```

推理

修改examples/inference/llama3_lora_sft.yaml

```
model_name_or_path: /mnt/workspace/models/Meta-Llama-3-8B-Instruct
adapter_name_or_path: /mnt/workspace/models/llama3-lora-zh
template: llama3
finetuning_type: lora
```

- `--model_name_or_path /mnt/workspace/models/Meta-Llama-3-8B-Instruct`: 这个参数指定了预训练语言模型的名称或路径。在这个例子中, 使用的是位于`/mnt/workspace/models/Meta-Llama-3-8B-Instruct`的Meta的Llama-3-8B-Instruct模型。这个模型将作为基础模型,在其上进行微调。
- `--adapter_name_or_path /mnt/workspace/models/llama3-lora-zh`: 这个参数指定了用于微调的适配器(Adapter)的名称或路径。在这个例子中, 使用的是位于`/mnt/workspace/models/llama3-lora-zh`的适配器。这个适配器是通过LoRA技术在基础模型上微调得到的。
- `--template llama3`: 这个参数指定了对话模板的名称。模板定义了对话的格式和风格。在这个例子中, 使用的是名为`llama3`的模板。
- `--finetuning_type lora`: 这个参数指定了微调的类型。在这个例子中, 使用的是LoRA技术进行微调。

/mnt/workspace/LLaMA-Factory路径下执行

```
llamafactory-cli chat examples/inference/llama3_lora_sft.yaml
```

评估

MMLU(Massive Multitask Language Understanding)是一个大规模的多任务语言理解基准测试, 用于评估大型语言模型在各个领域的语言理解能力。

特点:

1. 涵盖57个学科领域, 包括人文、社科、STEM等, 任务类型多样, 难度不一。
2. 全部由人类专家精心挑选的题目组成, 类似于标准化考试的问答题。
3. 所有题目都是选择题形式, 有4个选项, 其中1个为正确答案。较少依赖预先的知识积累。
4. 共有约15K个题目, 规模较大, 全面考察模型的语言理解和推理能力。

MMLU的设计目标是构建一个全面、具有挑战性、免于偏见的基准测试, 用于衡量当前大型语言模型的真实水平。通过多角度、多领域的考察, 可以揭示模型在不同任务上的优劣势。

评估方式:

- 对每个学科领域的题目, 分别计算模型的正确率。
- 计算所有学科的平均正确率, 作为模型的整体表现。
- 将模型的表现与人类专家的正确率进行比较。

MMLU上的结果已经成为衡量大模型实力的重要指标之一。

修改文件examples/lora_single_gpu/llama3_lora_eval.yaml

```
### model
model_name_or_path: /mnt/workspace/models/Meta-Llama-3-8B-Instruct
adapter_name_or_path: /mnt/workspace/models/llama3-lora-zh

### method
finetuning_type: lora

### dataset
task: mmlu
split: test
template: fewshot
lang: en
n_shot: 5

### output
save_dir: saves/llama3-8b/lora/eval_mmlu

### eval
batch_size: 1
```

参数解释:

- `model_name_or_path /mnt/workspace/models/Meta-Llama-3-8B-Instruct`: 这个参数指定了预训练语言模型的名称或路径。在这个例子中,使用的是位于`/mnt/workspace/models/Meta-Llama-3-8B-Instruct`的Meta的Llama-3-8B-Instruct模型。
- `adapter_name_or_path /mnt/workspace/models/llama3-lora-zh`: 这个参数指定了用于微调的适配器(Adapter)的名称或路径。在这个例子中, 使用的是位于`/mnt/workspace/models/llama3-lora-zh`的适配器。
- `template fewshot`: 这个参数指定了评估模板的名称。 使用的是名为`fewshot`的模板, 表示这是一个少样本(few-shot)学习的评估。
- `finetuning_type lora`: 这个参数指定了微调的类型。使用的是LoRA技术进行微调。
- `task mmlu`: 这个参数指定了评估任务的名称。使用的是MMLU(Massive Multitask Language Understanding)任务, 这是一个大规模多任务语言理解评估基准。
- `split test`: 这个参数指定了数据集的分割。 使用的是测试集(test split)。
- `lang en`: 这个参数指定了评估任务的语言。 使用的是英文(en)。
- `n_shot 5`: 这个参数指定了少样本学习中使用的样本数量。 使用了5个样本。
- `batch_size 1`: 这个参数指定了评估时的批量大小, 即每次并行处理的样本数量。

综合起来, 这个命令的意思是: 使用`llamafactory-cli`工具, 加载Llama-3-8B-Instruct预训练模型和llama3-lora-zh适配器, 然后在MMLU任务的英文测试集上, 使用5个样本的少样本学习设置, 以批量大小1进行评估。

/mnt/workspace/LLaMA-Factory路径下执行

```
llamafactory-cli eval examples/lora_single_gpu/llama3_lora_eval.yaml
```

CEVAL任务是一个专门为评估中文语言模型而设计的基准测试, 它包含了一系列的中文语言理解和生成任务, 如文本分类、命名实体识别、问答、摘要生成等。

通过在CEVAL上的评估, 我们可以全面地了解一个中文语言模型的性能和能力。

修改文件examples/lora_single_gpu/llama3_lora_eval.yaml

```
### model
model_name_or_path: /mnt/workspace/models/Meta-Llama-3-8B-Instruct
adapter_name_or_path: /mnt/workspace/models/llama3-lora-zh

### method
finetuning_type: lora

### dataset
task: ceval
split: test
template: fewshot
lang: zh
n_shot: 5

### output
save_dir: saves/llama3-8b/lora/eval_ceval

### eval
batch_size: 1
```

/mnt/workspace/LLaMA-Factory路径下执行

```
llamafactory-cli eval examples/lora_single_gpu/llama3_lora_eval.yaml
```

CMMLU任务是一个专门为评估中文语言模型在多任务设置下的语言理解能力而设计的基准测试。

它包含了多个不同类型的任务, 如文本分类、情感分析、语义相似度判断等。

通过在CMMLU上的评估, 我们可以了解一个中文语言模型在处理不同类型的语言理解任务时的通用能力和适应性。

修改文件examples/lora_single_gpu/llama3_lora_eval.yaml

```
### model
model_name_or_path: /mnt/workspace/models/Meta-Llama-3-8B-Instruct
adapter_name_or_path: /mnt/workspace/models/llama3-lora-zh

### method
finetuning_type: lora

### dataset
task: cmmlu
split: test
template: fewshot
lang: zh
n_shot: 5

### output
save_dir: saves/llama3-8b/lora/eval_cmmul
```

```
### eval
batch_size: 1
```

/mnt/workspace/LLaMA-Factory路径下执行

```
llamafactory-cli eval examples/lora_single_gpu/llama3_lora_eval.yaml
```

LORA合并（可选）

修改文件examples/merge_lora/llama3_lora_sft.yaml

```
### Note: DO NOT use quantized model or quantization_bit when merging lora
adapters

### model
model_name_or_path: /mnt/workspace/models/Meta-Llama-3-8B-Instruct
adapter_name_or_path: /mnt/workspace/models/llama3-lora-zh
template: llama3
finetuning_type: lora

### export
export_dir: /mnt/workspace/models/llama3-lora-zh-full
export_size: 5
export_device: cuda
export_legacy_format: false
```

- `model_name_or_path /mnt/workspace/models/Meta-Llama-3-8B-Instruct`: 这个参数指定了预训练语言模型的名称或路径。在这个例子中, 使用的是位于`/mnt/workspace/models/Meta-Llama-3-8B-Instruct`的Meta的Llama-3-8B-Instruct模型。
- `adapter_name_or_path /mnt/workspace/models/llama3-lora-zh`: 这个参数指定了用于微调的适配器(Adapter)的名称或路径。在这个例子中, 使用的是位于`/mnt/workspace/models/llama3-lora-zh`的适配器。
- `template llama3`: 这个参数指定了导出模板的名称。 使用的是llama3模板。
- `finetuning_type lora`: 这个参数指定了微调的类型。 使用的是LoRA技术进行微调。
- `export_dir /mnt/workspace/models/llama3-lora-zh-full`: 这个参数指定了导出模型的保存路径。 导出的模型将保存在`/mnt/workspace/models/llama3-lora-zh-full`目录下。
- `export_size 5`: 这个参数指定了导出模型的大小。 导出的模型将分成多个部分，每个部分在5GB以内。
- `export_device cuda`: 这个参数指定了导出模型时使用的设备。 使用的是CUDA设备, 即GPU。
- `export_legacy_format false`: 这个参数指定了是否使用旧版的导出格式。 设置为false, 表示使用新版格式。

/mnt/workspace/LLaMA-Factory路径下执行

```
llamafactory-cli export examples/merge_lora/llama3_lora_sft.yaml
```

合并后的推理

修改examples/inference/llama3.yaml

```
model_name_or_path: /mnt/workspace/models/llama3-lora-zh-full
template: llama3
```

```
llamafactory-cli chat examples/inference/llama3.yaml
```

项目实战2：llama3医疗问答大模型(lora微调)

数据集准备

数据集来源：https://modelscope.cn/datasets/xiaofengalg/Chinese-medical-dialogue/summary

/mnt/workspace/LLaMA-Factory/data路径下执行

```
git clone https://www.modelscope.cn/datasets/xiaofengalg/Chinese-medical-dialogue.git
```

修改/mnt/workspace/LLaMA-Factory/data路径下的文件dataset_info.json

```
  "alpaca_medical_zh": {
    "file_name": "chinese_medical_dialogue.json"
  },
```

训练

/mnt/workspace/LLaMA-Factory路径下执行

```
pip install bitsandbytes
```

修改文件 examples/lora_single_gpu/llama3_lora_sft.yaml

```
### model
model_name_or_path: /mnt/workspace/models/Meta-Llama-3-8B-Instruct

### method
stage: sft
do_train: true
finetuning_type: lora
lora_target: q_proj,v_proj

### dataset
dataset: alpaca_medical_zh
template: llama3
cutoff_len: 1024
max_samples: 1000
overwrite_cache: true
preprocessing_num_workers: 16

### output
output_dir: /mnt/workspace/models/llama3-lora-medical
logging_steps: 100
save_steps: 100
```

```
plot_loss: true
overwrite_output_dir: true

### train
per_device_train_batch_size: 1
gradient_accumulation_steps: 8
learning_rate: 0.0001
num_train_epochs: 1.0
lr_scheduler_type: cosine
warmup_steps: 0.1
fp16: true

### eval
val_size: 0.1
per_device_eval_batch_size: 1
evaluation_strategy: steps
eval_steps: 500
```

解释这个yaml文件中的主要参数:

model

- model_name_or_path: 指定预训练模型的路径或名称, 这里使用的是 /mnt/workspace/models/Meta-Llama-3-8B-Instruct 路径下的 Meta-Llama 3.8B 模型作为基础模型。

method

- stage: 表示训练阶段, 这里是 sft(Supervised Fine-Tuning)。
- do_train: 表示要进行训练。
- finetuning_type: 微调方法, 这里使用 LoRA(Low-Rank Adaptation)。
- lora_target: LoRA 作用的对象, 这里是 attention 层的 query 和 value 映射矩阵。

dataset

- dataset: 训练数据集名称,这里用的是 alpaca_medical_zh 数据集。
- template: 数据集的格式模板,这里是 llama3。
- cutoff_len: 输入序列的最大长度, 超过会截断, 这里设为1024。
- max_samples: 从数据集中取的最大样本数, 这里取前1000个样本。
- preprocessing_num_workers: 数据预处理的进程数, 这里用16个进程并行处理。

output

- output_dir: 训练日志和模型的保存路径。
- logging_steps: 每隔多少步记录一次日志, 这里每100步记录一次。
- save_steps: 每隔多少步保存一次模型, 这里每500步保存一次。
- plot_loss: 是否绘制loss曲线图。

train

- per_device_train_batch_size: 每个设备上的训练批大小, 这里设为1。
- gradient_accumulation_steps: 梯度累积的步数, 这里累积8步才更新一次模型参数。
- learning_rate: 学习率, 这里设为0.0001。
- num_train_epochs: 训练的epoch数, 这里训练1个epoch。**(还可以指定max_steps 3000 : 训练3000步。注意：真正项目训练增大总步数；注意：num_train_epochs和max_steps之中选择一个)**
- lr_scheduler_type: 学习率调度器类型, 这里用cosine, 即先warmup再降温。
- warmup_steps: 用初始学习率warmup的步数, 这里取总步数的10%。

- fp16: 是否使用fp16混合精度训练, 以加速训练并减少显存占用。

eval

- val_size: 从训练集中划分出的验证集比例, 这里取10%作为验证集。
- per_device_eval_batch_size: 验证时每个设备的批大小,这里为1。
- evaluation_strategy: 验证策略, 这里根据steps数来验证。
- eval_steps: 每隔多少步验证一次, 这里每500步验证一次。

以上就是这个yaml配置文件的主要参数解释。

/mnt/workspace/LLaMA-Factory路径下执行

```
llamafactory-cli train examples/lora_single_gpu/llama3_lora_sft.yaml
```

推理

修改examples/inference/llama3_lora_sft.yaml

```
model_name_or_path: /mnt/workspace/models/Meta-Llama-3-8B-Instruct
adapter_name_or_path: /mnt/workspace/models/llama3-lora-medical
template: llama3
finetuning_type: lora
```

- `model_name_or_path /mnt/workspace/models/Meta-Llama-3-8B-Instruct`: 这个参数指定了预训练语言模型的名称或路径。在这个例子中,使用的是位于`/mnt/workspace/models/Meta-Llama-3-8B-Instruct`的Meta的Llama-3-8B-Instruct模型。这个模型将作为基础模型,在其上进行微调。
- `adapter_name_or_path /mnt/workspace/models/llama3-lora-medical`: 这个参数指定了用于微调的适配器(Adapter)的名称或路径。在这个例子中,使用的是位于`/mnt/workspace/models/llama3-lora-medical`的适配器。这个适配器是通过LoRA技术在基础模型上微调得到的。
- `template llama3`: 这个参数指定了对话模板的名称。模板定义了对话的格式和风格。在这个例子中, 使用的是名为`llama3`的模板。
- `finetuning_type lora`: 这个参数指定了微调的类型。在这个例子中, 使用的是LoRA技术进行微调。

/mnt/workspace/LLaMA-Factory路径下执行

```
llamafactory-cli chat examples/inference/llama3_lora_sft.yaml
```

项目实战3：llama3医疗问答大模型(qlora微调)

数据集准备

数据集来源：https://modelscope.cn/datasets/xiaofengalg/Chinese-medical-dialogue/summary

```
git clone https://www.modelscope.cn/datasets/xiaofengalg/Chinese-medical-dialogue.git
```

修改/mnt/workspace/LLaMA-Factory/data路径下的文件dataset_info.json

```
  "alpaca_medical_zh": {
    "file_name": "chinese_medical_dialogue.json"
  },
```

训练

/mnt/workspace/LLaMA-Factory路径下执行

```
pip install bitsandbytes
```

修改文件 examples/qlora_single_gpu/llama3_lora_sft_bitsandbytes.yaml

```
### model
model_name_or_path: /mnt/workspace/models/Meta-Llama-3-8B-Instruct
quantization_bit: 4

### method
stage: sft
do_train: true
finetuning_type: lora
lora_target: q_proj,v_proj

### dataset
dataset: alpaca_medical_zh
template: llama3
cutoff_len: 1024
max_samples: 1000
overwrite_cache: true
preprocessing_num_workers: 16

### output
output_dir: /mnt/workspace/models/llama3-qlora-medical
logging_steps: 10
save_steps: 100
plot_loss: true
overwrite_output_dir: true
```

```yaml
### train
per_device_train_batch_size: 1
gradient_accumulation_steps: 8
learning_rate: 0.0001
num_train_epochs: 3.0
lr_scheduler_type: cosine
warmup_steps: 0.1
fp16: true

### eval
val_size: 0.1
per_device_eval_batch_size: 1
evaluation_strategy: steps
eval_steps: 500
```

解释这个yaml文件中的主要参数:

model

- model_name_or_path: 指定预训练模型的路径或名称, 这里使用的是 /mnt/workspace/models/Meta-Llama-3-8B-Instruct 路径下的 Meta-Llama 3.8B 模型作为基础模型。
- quantization_bit: 4 设置量化位数为4bit,用于减小模型体积

method

- stage: 表示训练阶段, 这里是 sft(Supervised Fine-Tuning)。
- do_train: 表示要进行训练。
- finetuning_type: 微调方法, 这里使用 LoRA(Low-Rank Adaptation)。
- lora_target: LoRA 作用的对象, 这里是 attention 层的 query 和 value 映射矩阵。

dataset

- dataset: 训练数据集名称,这里用的是 alpaca_medical_zh 数据集。
- template: 数据集的格式模板,这里是 llama3。
- cutoff_len: 输入序列的最大长度, 超过会截断, 这里设为1024。
- max_samples: 从数据集中取的最大样本数, 这里取前1000个样本。
- preprocessing_num_workers: 数据预处理的进程数, 这里用16个进程并行处理。

output

- output_dir: 训练日志和模型的保存路径。
- logging_steps: 每隔多少步记录一次日志, 这里每100步记录一次。
- save_steps: 每隔多少步保存一次模型, 这里每100步保存一次。
- plot_loss: 是否绘制loss曲线图。

train

- per_device_train_batch_size: 每个设备上的训练批大小, 这里设为1。
- gradient_accumulation_steps: 梯度累积的步数, 这里累积8步才更新一次模型参数。
- learning_rate: 学习率, 这里设为0.0001。
- num_train_epochs: 训练的epoch数, 这里训练3个epoch。**(还可以指定max_steps 3000 : 训练3000步。注意：真正项目训练增大总步数；注意：num_train_epochs和max_steps之中选择一个)**
- lr_scheduler_type: 学习率调度器类型, 这里用cosine, 即先warmup再降温。
- warmup_steps: 用初始学习率warmup的步数, 这里取总步数的10%。
- fp16: 是否使用fp16混合精度训练, 以加速训练并减少显存占用。

eval

- val_size: 从训练集中划分出的验证集比例, 这里取10%作为验证集。
- per_device_eval_batch_size: 验证时每个设备的批大小,这里为1。
- evaluation_strategy: 验证策略, 这里根据steps数来验证。
- eval_steps: 每隔多少步验证一次, 这里每500步验证一次。

以上就是这个yaml配置文件的主要参数解释。

/mnt/workspace/LLaMA-Factory路径下执行

```
llamafactory-cli train
examples/qlora_single_gpu/llama3_lora_sft_bitsandbytes.yaml
```

推理

修改examples/inference/llama3_lora_sft.yaml

```
model_name_or_path: /mnt/workspace/models/Meta-Llama-3-8B-Instruct
adapter_name_or_path: /mnt/workspace/models/llama3-qlora-medical
template: llama3
finetuning_type: lora
```

- `model_name_or_path /mnt/workspace/models/Meta-Llama-3-8B-Instruct`: 这个参数指定了预训练语言模型的名称或路径。在这个例子中,使用的是位于`/mnt/workspace/models/Meta-Llama-3-8B-Instruct`的Meta的Llama-3-8B-Instruct模型。这个模型将作为基础模型,在其上进行微调。
- `adapter_name_or_path /mnt/workspace/models/llama3-qlora-medical`: 这个参数指定了用于微调的适配器(Adapter)的名称或路径。在这个例子中,使用的是位于`/mnt/workspace/models/llama3-qlora-medical`的适配器。这个适配器是通过QLoRA技术在基础模型上微调得到的。
- `template llama3`: 这个参数指定了对话模板的名称。模板定义了对话的格式和风格。在这个例子中, 使用的是名为`llama3`的模板。
- `finetuning_type lora`: 这个参数指定了微调的类型。在这个例子中, 使用的是LoRA技术进行微调。

/mnt/workspace/LLaMA-Factory路径下执行

```
llamafactory-cli chat examples/inference/llama3_lora_sft.yaml
```