

微调实战-1: Embedding 微调

第1步: 创建虚拟环境

```
conda create -n swift_env python=3.12 -y
conda init bash && source /root/.bashrc
conda activate swift_env
conda install ipykernel
ipython kernel install --user --name=swift_env
```

第2步: 安装 ms-swift

```
pip install git+https://github.com/modelscope/ms-swift.git
```

第3步: 安装其他依赖

```
pip install deepspeed liger-kernel
pip install scikit-learn
pip install -U sentence-transformers

# 建议科学上网后, 再执行下面的命令 📌
pip install flash-attn --no-build-isolation
```

第4步: 下载训练数据集

```
# 设置下面命令, 无需科学上网
export HF_ENDPOINT=https://hf-mirror.com

pip install --upgrade huggingface_hub

huggingface-cli download --repo-type dataset --resume-download microsoft/ms_marco --local-dir
/root/autodl-tmp/dataset/stsb
```

数据格式:

```
{
  "sentence1": "A plane is taking off.",
  "sentence2": "An air plane is taking off.",
  "score": 1.0
}
```

第5步: 开始 embedding 模型训练

方法-1:全参数训练

```
# 每个节点使用的GPU数量
nproc_per_node=1
NPROC_PER_NODE=$nproc_per_node \
swift sft \
  # 预训练模型路径
  --model /root/autodl-tmp/models/Qwen3-Embedding-0.6B \
  # 任务类型为embedding训练
  --task_type embedding \
  # 模型架构类型
  --model_type qwen3_emb \
  # 全参数微调
  --train_type full \
  # 训练数据集
  --dataset /root/autodl-tmp/dataset/stsb:positive \
  # 将5%的训练数据用作验证集
  --split_dataset_ratio 0.05 \
  # 按训练步数进行评估
  --eval_strategy steps \
  # 指定模型输出目录
  --output_dir output \
  # 每200步保存一次模型检查点
  --eval_steps 200 \
  # 训练轮次
  --num_train_epochs 5 \
  # 每200步保存一次模型检查点
  --save_steps 200 \
  # 每个GPU的训练批次大小
  --per_device_train_batch_size 8 \
  # 每个GPU的评估批次大小
  --per_device_eval_batch_size 4 \
  # 梯度累积步数
  --gradient_accumulation_steps 2 \
  # 学习率
  --learning_rate 6e-6 \
  # 损失函数类型, 需要label字段 (相似度分数)
  --loss_type cosine_similarity \
  # 指定标签字段名称
  --label_names labels \
  # 是否丢弃最后一个不完整的批次, 保证批次大小一致
  --data_loader_drop_last true
  # 多卡分布式
  --deepspeed zero3
```

```
{'loss': 0.00765588, 'grad_norm': 0.44140625, 'learning_rate': 0.0, 'memory(GiB)': 6.86, 'train_speed(iter/s)': 1.78775, 'epoch': 4.96, 'global_step/max_steps': '520/525', 'percentage': '99.05%', 'elapsed_time': '4m 50s', 'remaining_time': '2s'}
{'loss': 0.00655698, 'grad_norm': 0.29492188, 'learning_rate': 0.0, 'memory(GiB)': 6.86, 'train_speed(iter/s)': 1.786265, 'epoch': 5.0, 'global_step/max_steps': '525/525', 'percentage': '100.00%', 'elapsed_time': '4m 53s', 'remaining_time': '0s'}
Train: 100% | 525/525 [04:53<00:00, 1.44it/s]
[INFO:swift] Successfully registered '/root/miniconda3/envs/swift_venv/lib/python3.12/site-packages/swift/llm/dataset/data/dataset_info.json'.
{'eval_loss': 0.00913714, 'eval_pearson_cosine': 0.39544213, 'eval_pearson_euclidean': 0.43161228, 'eval_pearson_manhattan': 0.42718083, 'eval_pearson_dot_product': 0.39544207, 'eval_spearman_cosine': 0.40579028, 'eval_spearman_euclidean': 0.40560324, 'eval_spearman_manhattan': 0.40579028, 'eval_spearman_dot_product': 0.40579028, 'eval_runtime': 9.3256, 'eval_samples_per_second': 9.436, 'eval_steps_per_second': 2.359, 'epoch': 5.0, 'global_step/max_steps': '525/525', 'percentage': '100.00%', 'elapsed_time': '5m 2s', 'remaining_time': '0s'}
Val: 100% | 22/22 [00:01<00:00, 11.74it/s]
[INFO:swift] Saving model checkpoint to /root/autodl-tmp/Qwen3-Embedding/output/v0-20250617-141804/checkpoint-525
{'train_runtime': 307.2229, 'train_samples_per_second': 27.244, 'train_steps_per_second': 1.709, 'train_loss': 0.00984347, 'epoch': 5.0, 'global_step/max_steps': '525/525', 'percentage': '100.00%', 'elapsed_time': '5m 7s', 'remaining_time': '0s'}
Train: 100% | 525/525 [05:07<00:00, 1.71it/s]
[INFO:swift] last_model_checkpoint: /root/autodl-tmp/Qwen3-Embedding/output/v0-20250617-141804/checkpoint-525
[INFO:swift] best_model_checkpoint: /root/autodl-tmp/Qwen3-Embedding/output/v0-20250617-141804/checkpoint-525
[INFO:swift] images_dir: /root/autodl-tmp/Qwen3-Embedding/output/v0-20250617-141804/images
[INFO:swift] End time of running main: 2025-06-17 14:23:26.690644
(swift_venv) root@autodl-container-da414aae78-98e2c2c1:~/autodl-tmp/Qwen3-Embedding#
```

✦ 损失函数类型对比:

MS-SWIFT 框架支持多种损失函数, 用于训练 Embedding。选择合适的损失函数对模型性能至关重要, 不同的损失函数适用于不同的数据格式和训练目标。

1. InfoNCE 损失

```
--loss_type infonce
```

原理:

- 对比学习损失函数, 最大化正样本对相似度, 最小化负样本对相似度
- 使用批内对比学习策略, 将同批次内其他样本作为负样本

数据格式:

```
# 不含负样本
{"query": "sentence1", "response": "sentence1-positive"}

# 含多个负样本
{"query": "sentence1", "response": "sentence1-positive", "rejected_response": ["sentence1-negative1", "sentence1-negative2"]}
```

环境变量配置:

```
export INFONCE_TEMPERATURE=0.01 # 温度参数, 控制相似度分布锐利度
export INFONCE_USE_BATCH=True # 使用批内负样本
export INFONCE_HARD_NEGATIVES=False # 困难负样本数量标准化
export INFONCE_MASK_FAKE_NEGATIVE=False # 屏蔽假负样本
```

2. 余弦相似度损失

```
--loss_type cosine_similarity
```

原理:

- 直接优化预测相似度与真实相似度标签的差异
- 使用 MSE 损失计算 $||\text{input_label} - \text{cosine_sim}(u,v)||_2$

数据格式:

```
{ "query": "sentence1", "response": "sentence2", "label": 0.8 }
```

3. 对比学习损失

```
--loss_type contrastive
```

原理:

- 经典的对比学习损失, 正样本拉近, 负样本推远
- 需要设置 margin 参数

数据格式:

```
{ "query": "sentence1", "response": "sentence2", "label": 1 } # 1表示正样本, 0表示负样本
```

4. 在线对比学习损失

```
--loss_type online_contrastive
```

原理:

- 对比学习的改进版本, 选择困难正样本和困难负样本
- 通常比标准对比学习效果更好

数据格式:

```
{ "query": "sentence1", "response": "sentence2", "label": 1 } # 1表示正样本, 0表示负样本
```

5. 训练数据类型对应的损失函数

数据类型	推荐损失函数	原因
只有正样本对	<code>infonce</code>	利用批内负样本, 训练信号强
正负样本对	<code>contrastive</code> 或 <code>online_contrastive</code>	直接利用标注的负样本
连续相似度分数	<code>cosine_similarity</code>	直接优化相似度预测

方法-2: LoRA微调

```
# (1) LoRA 训练
nproc_per_node=1
NPROC_PER_NODE=$nproc_per_node \
swift sft \
    --model /root/autodl-tmp/models/Qwen3-Embedding-0.6B \
```

```
--task_type embedding \
--model_type qwen3_emb \
--train_type lora \
--lora_rank 16 \
--lora_alpha 32 \
--lora_dropout 0.1 \
--dataset sentence-transformers/stsb:positive \
--split_dataset_ratio 0.05 \
--eval_strategy steps \
--output_dir output \
--eval_steps 200 \
--num_train_epochs 5 \
--save_steps 200 \
--per_device_train_batch_size 8 \
--per_device_eval_batch_size 4 \
--gradient_accumulation_steps 2 \
--learning_rate 6e-6 \
--loss_type cosine_similarity \
--label_names labels \
--dataloader_drop_last true
```

(2) 合并 LoRA 权重

```
swift export --adapters /root/autodl-tmp/Qwen3-Embedding/output/v1-20250619-131635/checkpoint-525 --
merge_lora true --output_dir /root/autodl-tmp/Qwen3-Embedding/output/merged_model --safe_serialization
true --exist_ok true
```

```
[INFO:swift] Start time of running main: 2025-06-17 14:51:29.329702
[INFO:swift] swift.__version__: 3.6.0.dev0
[INFO:swift] merge_device_map: None
[INFO:swift] Loading the model using model_dir: /root/autodl-tmp/models/Qwen3-Embedding-0.6B
[INFO:swift] model_kwargs: {'device_map': 'cuda:0'}
[2025-06-17 14:51:30,894] [INFO] [real_accelerator.py:254:get_accelerator] Setting ds_accelerator to cuda (auto detect)
[2025-06-17 14:51:32,266] [INFO] [logging.py:107:log_dist] [Rank -1] [TorchCheckpointEngine] Initialized with serialization = False
[INFO:swift] default_system: None
[INFO:swift] max_length: 2048
[INFO:swift] response_prefix: ''
[INFO:swift] agent_template: react_en
[INFO:swift] Merge LoRA...
[INFO:swift] Saving merged weights...
[INFO:swift] Successfully merged LoRA and saved in /root/autodl-tmp/Qwen3-Embedding/output/merged_model.
[INFO:swift] End time of running main: 2025-06-17 14:51:34.805782
(swift_venv) root@autodl-container-da414aae78-98e2c2c1:~/autodl-tmp/Qwen3-Embedding#
```

(3) 测试合并后的模型

1 修改脚本: examples/qwen3_embedding_transformers.py 中模型加载路径:

```
model_path = "/root/autodl-tmp/Qwen3-Embedding/output/merged_model"
```

2 执行命令

```
python examples/qwen3_embedding_transformers.py
```

```
(swift_venv) root@autodl-container-da414aae78-98e2c2c1:~/autodl-tmp/Qwen3-Embedding# python examples/qwen3_embedding_transformers.py
query outputs:
tensor([[[-0.0296, -0.0708, -0.0015, ..., 0.0666, 0.0197, 0.0002],
         [-0.0249, -0.0555, -0.0026, ..., -0.0049, -0.0028, -0.0042]],
        device='cuda:0', dtype=torch.float16, grad_fn=<DivBackward0>)]
doc outputs:
tensor([[[-0.0030, -0.0471, 0.0038, ..., 0.0720, 0.0311, -0.0100],
         [-0.0258, -0.0739, -0.0027, ..., 0.0163, -0.0275, 0.0188]],
        device='cuda:0', dtype=torch.float16, grad_fn=<DivBackward0>)]
相似度得分: [[75.3125, 21.5], [17.46875, 84.0625]]
(swift_venv) root@autodl-container-da414aae78-98e2c2c1:~/autodl-tmp/Qwen3-Embedding#
```

微调实战-2 : ReRanker 微调

请参考原创项目: Qwen3-Reranker-SFT

AI进阶学习

 B站课堂: <https://space.bilibili.com/3546748815411393/pugv>

 官方网站: <https://www.tgltommy.com/> (国内访问需科学上网)

如果你正在关注大模型 Agent、强化学习后训练、RLHF、DPO、GRPO、RLVR 等前沿方向, 欢迎关注我最新上线的精品课程:
[《大模型 Agent 强化学习实战: 从后训练、奖励设计到工业级对齐系统》](#)

这门课程围绕当前大模型 Agent 强化学习的核心技术路线展开, 不只是介绍零散算法和论文概念, 而是希望帮助学习者建立一套完整的技术地图: 从基础原理、论文脉络、关键算法, 到开源项目源码解析与工业级系统实践, 逐步理解 Agent RL 为什么重要、如何演进, 以及在真实应用中如何落地。

课程配套专属飞书知识库《Agent RL 学习宝典》, 包含学习路线、论文资料、项目说明、源码阅读辅助资料和后续持续更新内容, 方便长期查阅和跟进前沿技术。

Agent RL 变化很快, 但越是变化快, 越需要一套系统的学习框架。如果你希望系统进入 Agent RL 方向, 而不是停留在碎片化了解阶段, 这门课会是一个很好的起点。

课程官网: <https://www.tgltommy.com/p/agent-rl> (国内访问需科学上网)

B站课堂: <https://www.bilibili.com/cheese/play/ss842375604>

更多课程详情, 扫描以下二维码咨询课程助理:

新课首发

大模型 Agent
强化学习实战:
| 从后训练、奖励设计到工业级对齐系统

唐国梁Tommy

TGL AI
Tommy

官方微信课程助理



B站/公众号/YouTube: 唐国梁Tommy

**关注微信公众号
获取更多学习资源**



官方网站: TGLTommy.com(需科学上网)