

Parameter-Efficient Fine-Tuning(PEFT)

- 공개된 코드로 파인튜닝을 해본결과..
- 환각 현상
- 아직 2번 밖에 돌려보지 않아서 파인튜닝을 성공하지는 못한 상태
- Colab에서 하루 할당 GPU 시간이 정해져있음

```
Untitled1.ipynb
파일 수정 보기 삽입 런타임 도구 도움말

+ 코드 + 텍스트

# SFTTrainer를 사용하여 모델 학습 설정
trainer = SFTTrainer(
    model=model, # 학습할 모델
    tokenizer=tokenizer, # 토큰라이저
    train_dataset=dataset, # 학습 데이터셋
    eval_dataset=dataset,
    dataset_text_field="text", # 데이터셋에서 텍스트 필드의 이름
    max_seq_length=max_seq_length, # 최대 시퀀스 길이
    dataset_num_proc=2, # 데이터 처리에 사용할 프로세스 수
    packing=False, # 짧은 시퀀스에 대한 학습 속도를 5배 빠르게 할 수 있음
    args=TrainingArguments(
        per_device_train_batch_size=2, # 각 디바이스당 훈련 배치 크기
        gradient_accumulation_steps=4, # 그래디언트 누적 단계
        warmup_steps=5, # 워업 스텝 수
        num_train_epochs=3, # 훈련 에폭 수
        max_steps=100, # 최대 스텝 수
        do_eval=True,
        evaluation_strategy="steps",
        logging_steps=1, # logging 스텝 수
        learning_rate=2e-4, # 학습률
        fp16=not torch.cuda.is_bf16_supported(), # fp16 사용 여부, bf16이 지원되지 않는 경우에만 사용
        bf16=torch.cuda.is_bf16_supported(), # bf16 사용 여부, bf16이 지원되는 경우에만 사용
        optim="adamw_8bit", # 최적화 알고리즘
        weight_decay=0.01, # 가중치 감소
        lr_scheduler_type="cosine", # 학습률 스케줄러 유형
        seed=123, # 랜덤 시드
        output_dir="outputs", # 출력 디렉토리
    ),
)

/usr/local/lib/python3.11/dist-packages/transformers/training_args.py:1575: FutureWarning: `evaluation_strategy` is deprecated and will be removed in version 4.46 of 🤗 Transformers.
warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/training_args.py:1575: FutureWarning: `evaluation_strategy` is deprecated and will be removed in version 4.46 of 🤗 Transformers.
warnings.warn(

Applying chat template to train dataset (num_proc=2): 100% ██████████ 41/41 [00:01<00:00, 27.16 examples/s]

Tokenizing train dataset (num_proc=2): 100% ██████████ 41/41 [00:02<00:00, 21.18 examples/s]

Tokenizing train dataset (num_proc=2): 100% ██████████ 41/41 [00:00<00:00, 105.44 examples/s]

[ ] # 현재 메모리 상태를 보여주는 코드
gpu_stats = torch.cuda.get_device_properties(0) # GPU 속성 가져오기
start_gpu_memory = round(
    torch.cuda.max_memory_reserved() / 1024 / 1024 / 1024, 3
) # 시작 시 예약된 GPU 메모리 계산
max_memory = round(
    gpu_stats.total_memory / 1024 / 1024 / 1024, 3
) # GPU의 최대 메모리 계산
print(
    f"GPU = {gpu_stats.name}. Max memory = {max_memory} GB."
) # GPU 이름과 최대 메모리 출력
print(f"{start_gpu_memory} GB of memory reserved.") # 예약된 메모리 양 출력

GPU = Tesla T4. Max memory = 14.741 GB.
5.547 GB of memory reserved.

[ ] trainer_stats = trainer.train()

==((====))==  Unsloth - 2x faster free finetuning | Num GPUs = 1
  \ \  / |   Num examples = 41 | Num Epochs = 20
0^0/ \ \ / \   Batch size per device = 2 | Gradient Accumulation steps = 4
 \_____/      Total batch size = 8 | Total steps = 100
"-----"    Number of trainable parameters = 41,943,040
[ 35/100 06:16 < 12:22, 0.09 it/s, Epoch 5.76/20]

Step Training Loss Validation Loss
1 1.221700 6.921641
```

제목이 LLM 키우기인 이유

- 양질의 데이터 셋을 공급해야만 잘 대답함. (잘 이해하도록 문서를 정리하여 공급하는 것이 필요)
- RAG, 파인튜닝 모두 코드 베이스를 만들고, 코드를 개선해가는 과정이 필요
- GPU도 필요(Colab 환경 사용)
- 데이터 생산 -> 파인튜닝과 RAG 활용 방법을 꾸준히 develop 해야 좋은 답변을 기대할 수 있음
- 사내 지식을 활용한 LLM과 코드 리뷰 봇을 만들어서 꾸준히 develop 해보는것 어떨지 제안

