NTU ADL 2023 Fall - HW3

written by Chih Han, Yang. B10902069

LLM Tuning

Description

1. Initially, I trained on 1000 examples from the dataset using a batch size of 4. I experimented with the entire 10000 examples in the training set, still with a batch size of 4. However, severe overfitting occurred, leading to a substantial increase in perplexity. After several attempts, I found that around 250 optimization steps were sufficient to surpass the baseline.
2. I fine-tuned my model in the following steps:
    - **Defining Hyperparameters**: The specifics of model hyperparameters are elaborated in the subsequent section.
    - **Utilizing `transformer.Seq2SeqTrainer`**: Employing `transformer.Seq2SeqTrainer` to perform the fine-tuning. We should define a class to save the peft model checkpoints.
    - **Configuring Model and Tokenizer**: We should define the model and tokenizer's configuration. Given the utilization of Llama in this project, it's crucial to include `eos_token` and `bos_token` to indicate the start and end of the sequence. We should also define the Lora Configuration to allow training solely on the adaptor rather than the entire model.
    - **Resizing Tokenizer and Model Embedding**: Adapting the tokenizer and model embedding sizes becomes necessary to ensure compatibility with the special tokens within the tokenizer.
    - **Data Preprocessing**: This stage is composed by several preprocessing tasks, including input tokenization, attention-mask configuration, and label tokenization. Notably, the incorporation of instruction-tuning techniques involves giving the data with specific prompts. These prompts aid the model in generating more robust results. The prompt to the instruction is: `f"你是人工智慧助理，以下是用戶和人工智能助理之間的對話。你要對用戶的問題提供有用、安全、詳細和禮貌的回答。USER: {instruction} ASSISTANT:"`

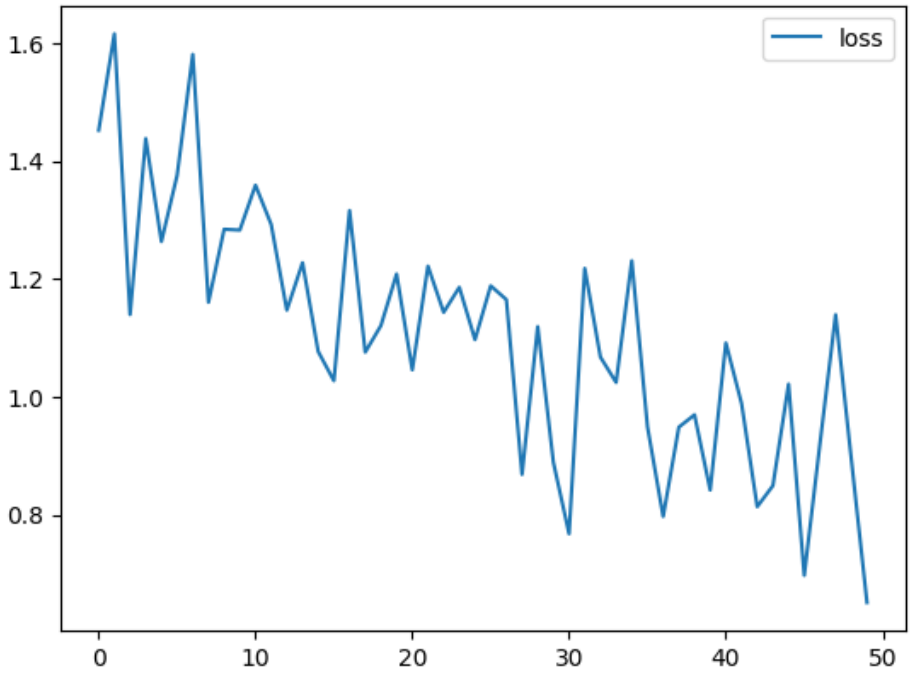
Additionally, concatenate the output label with the input, enabling the model to learn from the instruction-output pair. This process closely resembles few-shot learning with a single example.

    - **Deploy Trainer**: Deploy the Trainer to do the rest of the work on fine-tuning.
3. The hyper-parameters are the following:
    - `max_source_len=256`: The maximum length of the source input to the model. It's not set larger due to the lack of memory of my CUDA device.
    - `max_target_len=128`: The maximum length of the target output to the model. It's not set larger due to the lack of memory of my CUDA device.
    - `lora_r=64`: The dimension, aka the virtual "rank", of the virtual transforming matrix.
    - `lora_alpha=16`: This is the scaling factor for the weight matrix.
    - `optim="paged_adamw_32bit"`: The paged_adamw_32bit optimizer is a variant of AdamW that is designed to be more efficient on 32-bit GPUs.
    - `per_device_train_batch_size=1`, `gradient_accumulation_steps=4`: The batch size is 4 by multiplying the two. Gradient accumulation is used here.
    - `max_steps=250`: The number of optimization steps.
    - `learning_rate=1e-4`: The initial learning rate.
    - `lr_scheduler_type="linear"`: The learning rate scheduler is linear.

Performance

The overall performance is satisfactory, surpassing the baseline with a final mean perplexity score of **3.9474233360290527**.

- To prevent overfitting and ensure better generalization on the private dataset, I limited the training steps.
- The loss graph illustrates fluctuations, largely due to the dataset's diversity. However, the general trend shows a consistent decrease in loss, indicating progressive learning.



LLM Inference Strategies

Zero-Shot

For zero-shot inference, the model should make predictions or inferences without any specific examples on the target task. As you can see in the following code section, only the instruction and a few prompts are given. Note that the term instruction should be referred to as input.

```
f"你是人工智慧助理，以下是用戶和人工智能助理之間的對話。 \
你要對用戶的問題提供有用、安全、詳細和禮貌的回答。 \
USER: {instruction} ASSISTANT:"
```

Few-Shot

For few-shot inference, the model is provided with a small number of examples (contextual information) related to the task it needs to perform, then make predictions on the target task. As you can see in the following code section, the model is given a few examples labeled by “範例”, which is example in chinese. Note that the term instruction should be referred to as input.

```
f"你是人工智慧助理，以下是用戶和人工智能助理之間的對話。 \
你要對用戶的問題提供有用、安全、詳細和禮貌的回答。 \
USER: {instruction} \n範例：[{';'.join(kwargs['example'])}] ASSISTANT:"
```

I provided the model with three examples per inference, raising some considerations. Initially, a higher number of examples could enhance the model's comprehension of the task, potentially leading to more robust outputs. Conversely, an excessive quantity of examples might introduce noise, particularly if the model's scale is insufficient to manage such information overload.

Comparison

The zero-shot and few-shot inference is conducted on a model without QLoRA fine-tuning. Here's the result (mean-perplexity on public test data):

- zero-shot: 5.164038824558258
- few-shot: 5.65949741268158
- LoRA: **3.9474233360290527**

The LoRA fine-tuned model showcased superior performance compared to the other two inference strategies. Interestingly, the few-shot approach performed worse than zero-shot, which I attribute to noise interference just mentioned.

Bonus: Other methods

I experimented with another pretrained model—the Chinese Llama—which also has 7 billion parameters like the Taiwan Llama. I kept all hyperparameters the same.

After fine-tuning using the LoRA process, the final mean perplexity on the public test data was **8.538008956909179**. This result was notably worse than the **3.9474233360290527** achieved by the fine-tuned Taiwan-Llama.

While this difference is intriguing, further insights require a detailed examination of both models and their respective pre-training methods.

Useful Links

- Majorly inspired by: Link
- Homework Description: Link

- LLaMA2:
  - Huggingface API Documentation
  - Official Website
  - Source Code of Huggingface API
- peft:
  - Module Description
  - Source code of prepare_model_for_kbit_training
  - Huggingface Example
  - Huggingface API Documentation
- LoRA (Low Rank Adaptation):
  - LoRA Description
  - LoRA Paper
  - QLoRA Description
  - Source code of LoRA Model generate()
  - Source code of LoRA Model
- perplexity:
  - Perplexity Explanatory Video
  - Perplexity Description
- Trainer:
  - Huggingface Description
  - Source code of Trainer
  - Huggingface API Documentation
  - Loss history of Trainer_
- Other Tools:
  - torch.Tensor.contiguous
  - torch.mean
  - Datasets
  - GenerationConfig
  - Gradient Checkpointing
  - HfArgumentParser
- Chinese-LLaMa:
  - Model Github Repo
  - Model Description