Nikolay Vorontsov

LLMs and GenAI for NLP, 2024
Report on the Exercises in Labs 1 – 6
GitHub repository: **nicksnlp**

This is what I have done:

Week4

Fine-tuning a model with LLMs, PEFT, LoRA

supervised_finetuning.ipynb

I've created accounts on HuggingFace, Weights&Biases, I will use my regualar Google account. I use access tokens in *Secrets* on Colab.

I am running notebook on Colab Pay-As-You-Go, T4 GPU.

- ☑ Loaded dataset, here is an example:

```
Below is an instruction that describes a task. Write a response that
appropriately completes the request.

### Instruction:
Give three tips for staying healthy

### Response:
1. Eat a balanced and nutritious diet: Make sure your meals are inclusive
of a variety of fruits and vegetables, lean protein, whole grains, and
healthy fats. This helps to provide your body with the essential nutrients
to function at its best and can help prevent chronic diseases.

2. Engage in regular physical activity: Exercise is crucial for
maintaining strong bones, muscles, and cardiovascular health. Aim for at
least 150 minutes of moderate aerobic exercise or 75 minutes of vigorous
exercise each week.

3. Get enough sleep: Getting enough quality sleep is crucial for physical
and mental well-being. It helps to regulate mood, improve cognitive
function, and supports healthy growth and immune function. Aim for 7-9
hours of sleep each night.
```

- ☑ Created config object
- ☑ Downloaded the model
- ☑ Downloaded tokeniser

Applied for an Academic account at **W&B**. Now I can hopefully visualise and save models easier.

I have tried to set-up training parametres, but keep getting errors for different arguments (*max_seq_length*, *dataset_text_field*, *packing*) of their incompatibility with SFTTrainer:

For example:

```
TypeError: SFTTrainer.__init__() got an unexpected keyword argument
'max_seq_length'
```

Solution:

I have changed `tokenizer` into `proccessing_class` in `SDTTrainer`.

Commenting out `max_seq_length=None` etc. from the `SFTTTrainer` arguments seems also to work.

For now I will follow the pre-existed setup, with no truncation, padding, max_length. But there is an option to add the following into the code:

```
# Define the maximum sequence length (optional)
max_length = 512  # Set a reasonable length for your model

# Function to process the dataset by tokenizing and padding/truncating
def tokenize_function(batch):
    # Tokenize the 'text' field
    return tokenizer(batch['text'], padding="max_length", truncation=True,
max_length=max_length)

# Apply the function to the entire dataset
dataset = dataset.map(tokenize_function, batched=True)
```

Since I have got this warning:

```
/usr/local/lib/python3.10/dist-packages/trl/trainer/sft_trainer.py:300:
UserWarning: You passed a processing_class with `padding_side` not equal
to `right` to the SFTTrainer. This might lead to some unexpected behaviour
due to overflow issues when training a model in half-precision. You might
consider adding `processing_class.padding_side = 'right'` to your code.
  warnings.warn(
```

I have added the following line into the code, before defining the `trainer`:

```
tokenizer.padding_side = "right"
```

However, I am not completely sure now if `right` was the correct option for padding, or whether I could get away with no padding.

Here they used the `right`:

- Fine-Tuning Mistral

While for generation the `left` padding side is suggested:

- Generation with LLMs

A note:
If I was to run training several times, I should consider adding specific names (`name="small_run_1K"`) for training runs for better management in W&B into `wand.init(...)`, as well as:

```
training_arguments = TrainingArguments(
    output_dir="./results",
    run_name="unique_run_name",  # Add a custom name here
...
```

- [x] Send `trainer.train()` to run...
  Estimated time needed for training (1 epoch): ~ 8 hours

  UPDATE: Unfortunately, I have been cut off from colab, after it was almost done.
  [561/625 7:03:31 < 48:29, 0.02 it/s, Epoch 0.90/1]

  Verdict: Running this notebook with the available resources, without saving checkpoints outside of Colab was not a good idea... The data is lost and the time too...

  I will change the subset into 1K to check the pipeline, and retrain, I will call the model *shrimp*

- [ ] Evaluate training results and loss with W&B
- [ ] Save the model (Where!? Yes, in Colab environment...)
- [ ] Load the base model When loading `base_model` I have set up `device_map = "auto"`, and implemented quantisation, by adding: `load_in_4bit=True'` and `quantization_config={"bits": 4, "per_channel": True}` into parameters.
- [ ] Merge the `base_model` and `new_model` and push into HuggingFace
- [x] Created a model card for this model:
  https://huggingface.co/nicksnlp/shrimp/blob/main/README.md

Possible alternative: Saving checkpoints and other data to W&B or to Google Drive, when training larger models, reloading from a checkpoint to resume training. Preferably do the training somewhere outside of Colab.

Selecting another base model and dataset

I wanted to try using T5 Google Model and a labelled QA dataset to adapt it for detecting hallucinations in texts. While training this model I will save the checkpoints into W&B and/or to Google Drive.

Utilising DPO instead of supervised fine-tuning

Fine_tune_a_Mistral_7b_model_with_DPO.ipynb

Since this is a bonus exercise, I will hopefully do it later on...

References:

1. https://chatgpt.com/share/677021c2-0128-800b-957b-511b29768fd4
2. https://chatgpt.com/share/67708817-1a4c-800b-a17a-c99bcdcbd05d
3. https://chatgpt.com/share/67709535-13c0-800b-b07a-2446d28e701a