VIP Fall 2024

Fine-tuning LLMs for Materials

Tutorial III

In the first tutorial, we explored how to convert crystal structures into text representations. In the second, we set up an environment for `torchtune`. In this tutorial, we'll build on that by creating a custom dataset for `torchtune`, using the Cartesian representation from Tutorial I.

1. Setting Up

In the attache zip file, you will find the following file structure:

```
configs/
    - llama2-7b.yaml
    - llama2-7b-inference.yaml
data/mp_20/
    - raw_train/
    - raw_val/
    - raw_test/
llm4materials/
    - datasets/
            - torchtune_dataset.py
    - encoders
            - cartesian.py
    - inference
            - infer.py
outputs/
setup.py
```

Intuitively, the `configs/` folder holds our `yaml/` config files for finetuning an LLM, the `data/` folder contains the raw data that we need, the `outputs/` folder is where we save training output files (as specified in the `yaml` file).

The `setup.py` script allows us to install a package named `llm4materials`. In this way, we will be able to import the custom dataset implemented at `llm4materials/datasets` directly.

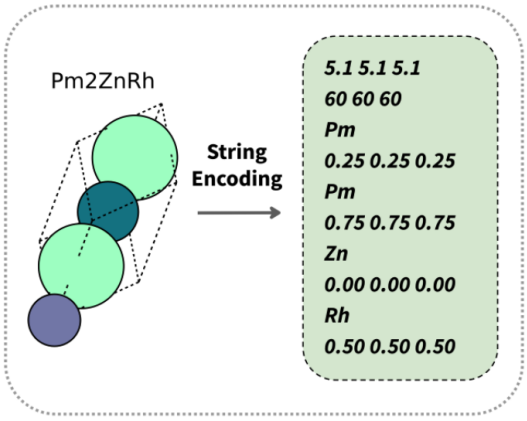
First, let's

1. Connect to ICE
2. Request a GPU
3. Upload the zip file to ICE and unzip it
4. Load `anaconda3` and activate your `torchtune` environment
5. With your environment activated, install `ase` if it is not available: `pip install ase`
6. In the same directory as `setup.py`, install `llm4materials` : `pip install -e .`

Now, you will be able to import definitions from `llm4materials`.

Creating Custom Dataset

Take a look at the file `llm4materials/encoders/cartesian.py`. It contains a partial implementation of the Cartesian encoder for crystal structures. Specifically, the Cartesian encoder has an `encode` method that takes in `ase.atoms.Atoms` objects and returns their corresponding text representations, similar to what we did in Tutorial I (see Figure below).



> TO-DO: complete the Cartesian encoder class.

Next, take a look at the file `llm4materials/datasets/torchtune_dataset.py`. It contains a class named `TextCompletionDataset` and a creator function `text_completion_dataset`.

In the `TextCompletionDataset` class, there is a class method named `_load_data(self, source)`, which gathers all the cif files in the given `source` path and returns a list of `ase.atoms.Atoms` objects.

> TO-DO: complete the `TextCompletionDataset` class.

The `_prepare_sample(self, sample)` method has been completed for you. In essence, we are transforming a single `Atoms` object (`sample`) into textual representation prepended with a prompt header. An example of the full text prompt is:

```
Below is a description of a bulk material.
Generate a description of the lengths and angles of the lattice vectors
and then the element type and coordinates for each atom within the lattice:
5.1 5.1 5.1
60 60 60
Pm
0.25 0.25 0.25
Pm
0.75 0.75 0.75
Zn
0.00 0.00 0.00
Rh
0.50 0.50 0.50
```

The above text is then tokenized by the LLM's tokenizer to serve as input to the LLM. A duplicate copy of the tokenized text is created to serve as the labels/targets.

Training

Once you have completed the above classes, you are ready to fine-tune the LLM on crystal structures! Go to `configs/llama2-7b.yaml` and change the necessary fields (e.g., paths to your downloaded weights). To fine-tune, just do

```
tune run lora_finetune_single_device \
--config configs/llama2-7b.yaml
```

Inference

If you have successfully trained your model for enough epochs, you will now be able to generate new string representations of structures. To do so, take a look at the inference `yaml` config file: `configs/llama2-7b-inference.yaml`. Make sure that the `checkpoint_files` are pointing to the trained checkpoints. For example, if you trained your model for 10 epochs, you should set it to the following:

```
checkpoint_files: [
    hf_model_0001_9.pt,
    hf_model_0002_9.pt,
  ]
```

The `pt` files can be found in the output folder you set in your training `yaml` config file. To generate, do

```
tune run llm4materials/inference/infer.py \
--config configs/llama2-7b-inference.yaml
```