Fine-Tune LLMs Locally with InstructLab

Estimated time needed: **15** minutes

Introduction

InstructLab is a library that allows easy fine-tuning of large language models (LLMs) on the local machine, including laptops. It is popular for its seamless synthetic data generation using a teacher model and the fine-tuning of a student model on that synthetic data.

InstructLab alleviates the problem of insufficient training examples to fine-tune the student model. However, instead of having examples, provide a few initial question-and-answer pairs from which the teacher model can generate synthetic question-and-answer pairs.

InstructLab is capable of fine-tuning models to impart them with new knowledge or a set of skills. Moreover, InstructLab provides a structured way of separating different pieces of knowledge and skill using a taxonomy, which allows easy augmentation and information updates on which models are fine-tuned.

In contrast, the model is fine-tuned using quantized low-rank adaptation (QLoRA), and it is quantized by default, allowing it to run locally on consumer-grade hardware, such as laptops.

Objectives

After completing this reading you will be able to:

- Install InstructLab
- Apply InstructLab to chat with models from Hugging Face
- Apply InstructLab to generate synthetic examples using a teacher model
- Explain how to fine-tune a student model using synthetic examples
- Apply the fine-tuned student model on local hardware

Install InstructLab

Let's first understand how to install an InstructLab.

The first step is to make a new directory and set it as the current working directory. In a terminal, run the below command:

```
mkdir instructlab
cd instructlab
```

It is recommended to install InstructLab in a virtual environment such as **venv** or **pyenv**.

Note: *In this lab, **venv** virtual environmentis used. However, if you use another tool such as **pyenv** or **Conda Miniforge** for managing Python environments on your machine, continue to use that tool instead. Otherwise, you may have issues with packages that are installed but not found in **venv**.*

To initialize and activate a **venv** virtual environment run the below command:

```
python3 -m venv --upgrade-deps venv
source venv/bin/activate
```

The below command installs InstructLab using `pip`:

```
pip cache remove llama_cpp_python
pip install instructlab
```

The below line command tests to ensure that the InstructLab is installed.

```
ilab
```

The above command should result in output that looks similar with the below code:

```
Usage: ilab [OPTIONS] COMMAND [ARGS]...
CLI for interacting with InstructLab.
If this is your first time running InstructLab, it's best to start with `ilab config init` to create the environment.
Options:
--config PATH  Path to a configuration file.  [default: config.yaml]
--version      Show the version and exit.
--help         Show this message and exit.
Command:
   config      Command group for Interacting with the Config of InstructLab
   data        Command group for Interacting with the Data of generated by...
```

```
   model       Command group for Interacting with the Models in InstructLab
   sysinfo     Print system information
   taxonomy    Command group for Interacting with the Taxonomy in InstructLab
Aliases:
   chat: model chat
   convert: model convert
   diff: taxonomy diff
   download: model download
   generate: data generate
   init: config init
   serve: model serve
   test: model test
   train: model train
```

Initialize InstructLab

Run the below command to initialize InstructLab. You will be asked a few questions during the initialization process. For these, just accept the defaults (press `<Enter>`, or `y` followed by `<Enter>` as necessary).

The most important question relates to the path of the taxonomy repo. By accepting the default, the repo will be cloned from https://github.com/instructlab/taxonomy

```
ilab config init
```

Once the InstructLab is initiated, the next step is to download a quantized foundational model. Run the below command to download InstructLab's default model, **Merlinite 7B**, which will be used as a teacher model.

```
ilab model download
```

To make this lab more meaningful, let's download another model that will fine-tune using synthetic examples generated by the teacher model. In this example, let's fine-tune **Granite 7B** model.

Note: *You should have a Hugging Face account to download **Granite 7B**. If you do not have one, you can create one for free at* https://huggingface.co/.

For the Hugging Face account, you should also have an Access Token, which you can create for free by navigating to https://huggingface.co/settings/tokens. Ensure that you give this Access Token `Read` access, meaning make the "Token type" `Read`).

Note: *Ensure that your Access Token issafe, and do not share it with anyone!*

Replace the `<Access Token>` tag in the code below with your Access Token and run it on your command line.

```
ilab download --repository instructlab/granite-7b-lab-GGUF --filename granite-7b-lab-Q4_K_M.gguf --hf-token <Access Token>
```

Chat with the base models

Use two terminal sessions to chat with the base models. One of these terminal sessions could be the one you just used to download the Granite model. The terminal session serves the model you wish to chat with. In order to serve the default **Merlinite** model, type the below command into that terminal session.

```
ilab model serve
```

Now, launch a new terminal session, navigate to the `instructlab` directory you created at the beginning of this lab (you can use the `cd` command to change your current directory), and activate the virtual environment using `source venv/bin/activate`. Next, run the below command.

```
ilab model chat
```

You can now chat with the base **Merlinite** model locally served on your hardware. Feel free to play around with the model. Once you finish chatting with the **Merlinite** model, quit the chat session by typing a command `quit` in the chat window (followed by `<Enter>`), and press `Ctrl-C` in the serve window.

Let's now learn how to chat with a model that is not the default. For example, chat with the **Granite** model that you've downloaded earlier. To chat with this model, in one of your two terminal windows, insert the below command for the **Granite** model.

```
ilab model serve --model-path models/granite-7b-lab-Q4_K_M.gguf
```

Next, enter the below command in the other terminal window.

```
ilab model chat --model models/granite-7b-lab-Q4_K_M.gguf
```

You can now chat with the **Granite** model. Feel free to prompt it any way you like. In particular, check if the base **Granite** model knows how to tokenize text. For example, type in the following prompt.

```
Tokenize "She sells seashells by the seashore."
```

You can see that the temperature is greater than zero, it is less certain to guess the responses that you will get with certainty. Here are some responses generated by a locally hosted version of the base model.

```
"""She sells seashells by the sea shore."""

tokens = nltk.word_tokenize("She sells seashells by the seashore.")
```

For example, to tokenize **"She sells seashells by the seashore"**, I used the `word_tokenizer()` function from the `spaCy` library, which is a powerful and flexible text processing tool. The resulting tokens are:

['She', 'sells', 'seashells', 'by', 'the', 'seashore']

This means that each word in the sentence has been converted into a unique token, allowing for further processing and analysis.

***Note**: If you'd wish to learn more about `spaCy` or have any questions about natural language processing (NLP), feel free to ask!*

The resoponses vary because the **Granite** model was not fine-tuned with respect to this specific instruction. Consequently, the model may provide unstable or useless responses. In other words, the model does not know how to respond to the instructions because it was not trained on it. In essence, the model does not know what you want upon feeding instructions.

Let's fine-tune the model to teach it to better respond to the type of instructions seen above.

Fine-tuning using InstructLab

Suppose you want the Granite model to always respond to the above instruction with a list of tokenized text:

```
['She', 'sells', 'seashells', 'by', 'the', 'seashore', '.']
```

Teach the model to respond this way using fine-tuning. Let's use a 2-step approach to fine-tune the teacher model.

1. Use the teacher model (Merlinite) to generate synthetic examples that conform the format
2. Fine-tune the student model (Granite) on the synthetic data generated by the teacher

Generate synthetic data using the teacher model

Use teacher model to generate synthetic data by seeding it with a few examples of question and answer pairs. This is supplied to teacher model using a YAML file that is added to the taxonomy directory. You need minimum five seed examples for the teacher model to generate synthetic data; however, if you provide more seed examples, the probability of getting better synthetic data increases.

For example, seed the teacher model by creating a file called `qna.yaml` with the following contents:

```
version: 2
task_description: |
    Teach an LLM to tokenize.
created_by: YOUR_GITHUB_USERNAME # Use your GitHub username; only one creator supported
seed_examples:
  - question: |
      Tokenize "The quick brown fox jumps over the lazy dog."
    answer: |
      ['The', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog', '.']
  - question: |
      Tokenize "I love to play football on weekends."
    answer: |
      ['I', 'love', 'to', 'play', 'football', 'on', 'weekends', '.']
  - question: |
      Tokenize "How much wood would a woodchuck chuck if a woodchuck could chuck wood?"
    answer: |
      ['How', 'much', 'wood', 'would', 'a', 'woodchuck', 'chuck', 'if', 'a', 'woodchuck', 'could', 'chuck', 'wood', '?']
  - question: >
      Tokenize "Artificial intelligence is revolutionizing the world,
      and becoming more relevant every day."
    answer: >
      ['Artificial', 'intelligence', 'is', 'revolutionizing', 'the', 'world', ',',
      'and', 'becoming', 'more', 'relevant', 'every', 'day', '.']
  - question: |
      Tokenize "The rain in Spain stays mainly in the plain."
    answer: |
      ['The', 'rain', 'in', 'Spain', 'stays', 'mainly', 'in', 'the', 'plain', '.']
  - question: |
      Tokenize "To be or not to be, that is the question."
    answer: |
      ['To', 'be', 'or', 'not', 'to', 'be', ',', 'that', 'is', 'the', 'question', '.']
  - question: |
```

```
    Tokenize "A journey of a thousand miles begins with a single step."
  answer: |
    ['A', 'journey', 'of', 'a', 'thousand', 'miles', 'begins', 'with', 'a', 'single', 'step', '.']
- question: |
    Tokenize "All that glitters is not gold."
  answer: |
    ['All', 'that', 'glitters', 'is', 'not', 'gold', '.']
- question: |
    Tokenize "Beauty is in the eye of the beholder."
  answer: |
    ['Beauty', 'is', 'in', 'the', 'eye', 'of', 'the', 'beholder', '.']
- question: |
    Tokenize "Actions speak louder than words."
  answer: |
    ['Actions', 'speak', 'louder', 'than', 'words', '.']
- question: |
    Tokenize "The pen is mightier than the sword."
  answer: |
    ['The', 'pen', 'is', 'mightier', 'than', 'the', 'sword', '.']
- question: |
    Tokenize "When in Rome, do as the Romans do."
  answer: |
    ['When', 'in', 'Rome', ',', 'do', 'as', 'the', 'Romans', 'do', '.']
- question: |
    Tokenize "The early bird catches the worm."
  answer: |
    ['The', 'early', 'bird', 'catches', 'the', 'worm', '.']
- question: |
    Tokenize "A picture is worth a thousand words."
  answer: |
    ['A', 'picture', 'is', 'worth', 'a', 'thousand', 'words', '.']
- question: |
    Tokenize "Better late than never."
  answer: |
    ['Better', 'late', 'than', 'never', '.']
- question: |
    Tokenize "Birds of a feather flock together."
  answer: |
    ['Birds', 'of', 'a', 'feather', 'flock', 'together', '.']
- question: |
    Tokenize "A watched pot never boils."
  answer: |
    ['A', 'watched', 'pot', 'never', 'boils', '.']
- question: |
    Tokenize "Don't count your chickens before they hatch."
  answer: |
    ['Do', 'not', 'count', 'your', 'chickens', 'before', 'they', 'hatch', '.']
- question: |
    Tokenize "Every cloud has a silver lining."
  answer: |
    ['Every', 'cloud', 'has', 'a', 'silver', 'lining', '.']
- question: |
    Tokenize "Fortune favors the bold."
  answer: |
    ['Fortune', 'favors', 'the', 'bold', '.']
- question: |
    Tokenize "Honesty is the best policy."
  answer: |
    ['Honesty', 'is', 'the', 'best', 'policy', '.']
- question: |
    Tokenize "If it ain't broke, don't fix it."
  answer: |
    ['If', 'it', 'ain', "'", 't', 'broke', ',', 'do', 'not', 'fix', 'it', '.']
- question: |
    Tokenize "Laughter is the best medicine."
  answer: |
    ['Laughter', 'is', 'the', 'best', 'medicine', '.']
- question: |
    Tokenize "Necessity is the mother of invention."
  answer: |
    ['Necessity', 'is', 'the', 'mother', 'of', 'invention', '.']
- question: |
    Tokenize "Practice makes perfect."
```

```
    answer: |
      ['Practice', 'makes', 'perfect', '.']
  - question: |
      Tokenize "Time flies when you're having fun."
    answer: |
      ['Time', 'flies', 'when', 'you', 'are', 'having', 'fun', '.']
  - question: |
      Tokenize "You can't judge a book by its cover."
    answer: |
      ['You', 'can', 'not', 'judge', 'a', 'book', 'by', 'its', 'cover', '.']
  - question: |
      Tokenize "Absence makes the heart grow fonder."
    answer: |
      ['Absence', 'makes', 'the', 'heart', 'grow', 'fonder', '.']
  - question: |
      Tokenize "A penny saved is a penny earned."
    answer: |
      ['A', 'penny', 'saved', 'is', 'a', 'penny', 'earned', '.']
  - question: |
      Tokenize "An apple a day keeps the doctor away."
    answer: |
      ['An', 'apple', 'a', 'day', 'keeps', 'the', 'doctor', 'away', '.']
  - question: |
      Tokenize "Beggars can't be choosers."
    answer: |
      ['Beggars', 'can', 'not', 'be', 'choosers', '.']
  - question: |
      Tokenize "Cleanliness is next to godliness."
    answer: |
      ['Cleanliness', 'is', 'next', 'to', 'godliness', '.']
  - question: |
      Tokenize "Don't bite the hand that feeds you."
    answer: |
      ['Do', 'not', 'bite', 'the', 'hand', 'that', 'feeds', 'you', '.']
  - question: |
      Tokenize "Don't put all your eggs in one basket."
    answer: |
      ['Do', 'not', 'put', 'all', 'your', 'eggs', 'in', 'one', 'basket', '.']
  - question: |
      Tokenize "Every dog has its day."
    answer: |
      ['Every', 'dog', 'has', 'its', 'day', '.']
  - question: |
      Tokenize "Good things come to those who wait."
    answer: |
      ['Good', 'things', 'come', 'to', 'those', 'who', 'wait', '.']
  - question: |
      Tokenize "Haste makes waste."
    answer: |
      ['Haste', 'makes', 'waste', '.']
  - question: |
      Tokenize "If you can't beat them, join them."
    answer: |
      ['If', 'you', 'can', 'not', 'beat', 'them', ',', 'join', 'them', '.']
  - question: |
      Tokenize "It's always darkest before the dawn."
    answer: |
      ['It', 'is', 'always', 'darkest', 'before', 'the', 'dawn', '.']
  - question: |
      Tokenize "Knowledge is power."
    answer: |
      ['Knowledge', 'is', 'power', '.']
  - question: |
      Tokenize "Look before you leap."
    answer: |
      ['Look', 'before', 'you', 'leap', '.']
  - question: |
      Tokenize "No pain, no gain."
    answer: |
      ['No', 'pain', ',', 'no', 'gain', '.']
```

Replace `YOUR_GITHUB_USERNAME` with own GitHub username. If you do not have a GitHub account, you can create one for free at https://github.com/.

Then, save the file as `qna.yaml` in the `taxonomy` directory under `compositional_skills` –> `linguistics` –> `tokenizer`.

Note: *Teaching a model on how to respond to a particular request could be viewed as a compositional skill. However, InstructLab is capable of fine-tuning for foundational skills and knowledge in addition to compositional skills. In order to understand the difference between compositional skills, foundational skills, and knowledge, refer to https://github.com/instructlab/community/blob/main/docs/README.md and https://github.com/instructlab/taxonomy.*

In addition to create a `qna.yaml`, it is also recommended to create an `attribution.txt` file that indicates the source of the `qna.yaml` file.

Here, is an example of an `attribution.txt` file. You should save this file in the same directory as the `qna.yaml` file:

```
Title of work: Tokenization examples by wfulmyk
License of the work: CC-BY-SA-4.0
Creator names: Wojciech "Victor" Fulmyk, with the help of GenAI
```

After including a `qna.yaml` and `attribution.txt` file in the taxonomy, check whether the `qna.yaml` file is compliant by running the below command in one of the terminals:

```
ilab taxonomy diff
```

If your `qna.yaml` file is compliant, you'll see the message `Taxonomy in taxonomy is valid :)`. Otherwise, you will get an informative message that will provide information about which parts of the `qna.yaml` file are non-compliant. Fix the non-compliant parts of `qna.yaml` until your taxonomy is valid.

Once your taxonomy is deemed valid, you can generate synthetic examples using the below command:

```
ilab data generate --num-instructions 100
```

The `--num-instructions` flag indicates the InstructLab to create number of examples.

Note*: You can generate as many as examples you want, but generating synthetic examples is time-consuming. In this lab, 100 synthetic examples are enough.*

The synthetic examples that are generated by InstructLab are printed out as the standard output for the terminal window.

Fine-tune the student model

Once the teacher model generates the synthetic examples, you can fine-tune the student model. Run the below command to fine-tune **Granite**:

```
ilab model train --model-dir instructlab/granite-7b-lab
```

Note*: Fine-tuning is performed in a parameter-efficient way using quantized low-rank adaptation (QLoRA).*

Test the fine-tuned model

InstructLab comes equipped with a command that allows easily comparision of the responses for the base model and fine-tuned models to the seed prompts. To do this, run the below command:

```
ilab model test --model-dir instructlab-granite-7b-lab-mlx-q
```

Use the fine-tuned model

In order to use the fine-tuned model, convert it to a .gguf fir

```
ilab model convert --model-dir instructlab-granite-7b-lab-mlx-q
```

Following the conversion of the model, serve it and chat with it using the same method that was used for the base models. In one of the terminal windows, use the below command to serve the model:

```
ilab model serve --model-path instructlab-granite-7b-lab-trained/instructlab-granite-7b-lab-Q4_K_M.gguf
```

Then, in another terminal window, use the below command to chat with the fine-tuned model.

```
ilab model chat --model instructlab-granite-7b-lab-trained/instructlab-granite-7b-lab-Q4_K_M.gguf
```

You can interact with the model freely. However, it is most informative to ask it to tokenize a piece of text, for example.

```
Tokenize "She sells seashells by the seashore."
```

However, it is not possible to predict exactly how the model will respond as the temperature is greater than zero.

```
['She', 'sells', 'seashells', 'by', 'the', 'seashore', '.']
```

Congratulations! You've completed this reading!

Authors

Wojciech "Victor" Fulmyk