Ajuste Fino de LLMs Localmente com InstructLab

Tempo estimado necessário: **15** minutos

Introdução

InstructLab é uma biblioteca que permite o ajuste fino fácil de grandes modelos de linguagem (LLMs) na máquina local, incluindo laptops. É popular por sua geração de dados sintéticos de forma integrada usando um modelo professor e o ajuste fino de um modelo aluno com esses dados sintéticos.

O InstructLab alivia o problema da insuficiência de exemplos de treinamento para ajustar o modelo aluno. No entanto, em vez de ter exemplos, forneça alguns pares iniciais de perguntas e respostas dos quais o modelo professor pode gerar pares sintéticos de perguntas e respostas.

O InstructLab é capaz de ajustar modelos para transmitir novos conhecimentos ou um conjunto de habilidades. Além disso, o InstructLab fornece uma maneira estruturada de separar diferentes peças de conhecimento e habilidades usando uma taxonomia, o que permite uma fácil ampliação e atualizações de informações sobre quais modelos estão sendo ajustados.

Em contraste, o modelo é ajustado usando adaptação de baixa classificação quantizada (QLoRA), e é quantizado por padrão, permitindo que funcione localmente em hardware de consumo, como laptops.

Objetivos

Após completar esta leitura, você será capaz de:

- Instalar o InstructLab
- Aplicar o InstructLab para conversar com modelos da Hugging Face
- Aplicar o InstructLab para gerar exemplos sintéticos usando um modelo professor
- Explicar como ajustar um modelo aluno usando exemplos sintéticos
- Aplicar o modelo aluno ajustado em hardware local

Instalar InstructLab

Vamos primeiro entender como instalar um InstructLab.

O primeiro passo é criar um novo diretório e defini-lo como o diretório de trabalho atual. Em um terminal, execute o comando abaixo:

```
mkdir instructlab
cd instructlab
```

É recomendável instalar o InstructLab em um ambiente virtual como **venv** ou **pyenv**.

Nota: *Neste laboratório, o ambiente virtual **venv** é utilizado. No entanto, se você usar outra ferramenta, como **pyenv** ou **Conda Miniforge**, para gerenciar ambientes Python em sua máquina, continue usando essa ferramenta. Caso contrário, você pode ter problemas com pacotes que estão instalados, mas não encontrados em **venv**.*

Para inicializar e ativar um ambiente virtual **venv**, execute o comando abaixo:

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

O comando abaixo instala o InstructLab usando `pip`:

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

A linha de comando abaixo testa para garantir que o InstructLab está instalado.

```
ilab
```

O comando acima deve resultar em uma saída que se pareça com o código abaixo:

```
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
```

Inicializar InstructLab

Execute o comando abaixo para inicializar o InstructLab. Você será questionado sobre algumas opções durante o processo de inicialização. Para isso, basta aceitar os padrões (pressione `<Enter>`, ou `y` seguido de `<Enter>` conforme necessário).

A pergunta mais importante diz respeito ao caminho do repositório de taxonomia. Ao aceitar o padrão, o repositório será clonado de https://github.com/instructlab/taxonomy

```
ilab config init
```

Uma vez que o InstructLab é iniciado, o próximo passo é baixar um modelo fundamental quantizado. Execute o comando abaixo para baixar o modelo padrão do InstructLab, **Merlinite 7B**, que será usado como modelo professor.

```
ilab model download
```

Para tornar este laboratório mais significativo, vamos baixar outro modelo que será ajustado usando exemplos sintéticos gerados pelo modelo professor. Neste exemplo, vamos ajustar o modelo **Granite 7B**.

Nota: *Você deve ter uma conta no Hugging Face para baixar o **Granite 7B**. Se você não tiver uma, pode criar uma gratuitamente em* https://huggingface.co/.

Para a conta do Hugging Face, você também deve ter um Token de Acesso, que pode criar gratuitamente acessando https://huggingface.co/settings/tokens. Certifique-se de que você dê a este Token de Acesso acesso `Read`, ou seja, faça com que o "Tipo de Token" seja `Read`).

Nota: *Certifique-se de que seu Token de Acesso esteja seguro e não o compartilhe com ninguém!*

Substitua a tag `<Access Token>` no código abaixo pelo seu Token de Acesso e execute-o na linha de comando.

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

Converse com os modelos base

Use duas sessões de terminal para conversar com os modelos base. Uma dessas sessões de terminal pode ser a que você acabou de usar para baixar o modelo Granite. A sessão de terminal serve o modelo com o qual você deseja conversar. Para servir o modelo **Merlinite** padrão, digite o comando abaixo nessa sessão de terminal.

```
ilab model serve
```

Agora, inicie uma nova sessão de terminal, navegue até o diretório `instructlab` que você criou no início deste laboratório (você pode usar o comando `cd` para mudar seu diretório atual) e ative o ambiente virtual usando `source venv/bin/activate`. Em seguida, execute o comando abaixo.

```
ilab model chat
```

Agora você pode conversar com o modelo base **Merlinite** servido localmente no seu hardware. Sinta-se à vontade para explorar o modelo. Uma vez que você termine a conversa com o modelo **Merlinite**, saia da sessão de chat digitando o comando `quit` na janela de chat (seguido de `<Enter>`), e pressione `Ctrl-C` na janela do servidor.

Vamos agora aprender como conversar com um modelo que não é o padrão. Por exemplo, converse com o modelo **Granite** que você baixou anteriormente. Para conversar com este modelo, em uma das suas duas janelas de terminal, insira o comando abaixo para o modelo **Granite**.

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

Em seguida, insira o comando abaixo na outra janela do terminal.

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

Você agora pode conversar com o **Granite** modelo. Sinta-se à vontade para provocá-lo da maneira que desejar. Em particular, verifique se o modelo base **Granite** sabe como tokenizar texto. Por exemplo, digite o seguinte prompt.

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

Você pode ver que a temperatura é maior que zero, é menos certo adivinhar as respostas que você obterá com certeza. Aqui estão algumas respostas geradas por uma versão hospedada localmente do modelo base.

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

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

Por exemplo, para tokenizar **"Ela vende conchas à beira-mar"**, usei a função `word_tokenizer()` da biblioteca `spaCy`, que é uma ferramenta de processamento de texto poderosa e flexível. Os tokens resultantes são:

['Ela', 'vende', 'conchas', 'à', 'beira-mar']

Isso significa que cada palavra na frase foi convertida em um token único, permitindo processamento e análise adicionais.

***Nota**: Se você quiser saber mais sobre `spaCy` ou tiver alguma dúvida sobre processamento de linguagem natural (NLP), sinta-se à vontade para perguntar!*

As respostas variam porque o modelo **Granite** não foi ajustado em relação a esta instrução específica. Consequentemente, o modelo pode fornecer respostas instáveis ou inúteis. Em outras palavras, o modelo não sabe como responder às instruções porque não foi treinado para isso. Em essência, o modelo não sabe o que você deseja ao fornecer instruções.

Vamos ajustar o modelo para ensiná-lo a responder melhor ao tipo de instruções vistas acima.

Ajuste fino usando InstructLab

Suponha que você queira que o modelo Granite sempre responda à instrução acima com uma lista de texto tokenizado:

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

Ensine o modelo a responder desta forma usando ajuste fino. Vamos usar uma abordagem em 2 etapas para ajustar o modelo professor.

1. Use o modelo professor (Merlinite) para gerar exemplos sintéticos que se conformem ao formato.
2. Ajuste o modelo aluno (Granite) com os dados sintéticos gerados pelo professor.

Gerar dados sintéticos usando o modelo professor

Use o modelo professor para gerar dados sintéticos, fornecendo alguns exemplos de pares de perguntas e respostas. Isso é fornecido ao modelo professor usando um arquivo YAML que é adicionado ao diretório de taxonomia. Você precisa de pelo menos cinco exemplos de sementes para que o modelo professor gere dados sintéticos; no entanto, se você fornecer mais exemplos de sementes, a probabilidade de obter dados sintéticos melhores aumenta.

Por exemplo, semeie o modelo professor criando um arquivo chamado `qna.yaml` com o seguinte conteúdo:

```
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', '.']
```

Substitua `YOUR_GITHUB_USERNAME` pelo seu próprio nome de usuário do GitHub. Se você não tiver uma conta no GitHub, pode criar uma gratuitamente em https://github.com/.

Em seguida, salve o arquivo como `qna.yaml` no diretório `taxonomy` sob `compositional_skills` –> `linguistics` –> `tokenizer`.

Nota: *Ensinar um modelo a como responder a um pedido específico pode ser visto como uma habilidade composicional. No entanto, o InstructLab é capaz de realizar o ajuste fino para habilidades e conhecimentos fundamentais, além de habilidades composicionais. Para entender a diferença entre habilidades composicionais, habilidades fundamentais e conhecimento, consulte https://github.com/instructlab/community/blob/main/docs/README.md e https://github.com/instructlab/taxonomy.*

Além de criar um `qna.yaml`, também é recomendável criar um arquivo `attribution.txt` que indique a fonte do arquivo `qna.yaml`.

Aqui está um exemplo de um arquivo `attribution.txt`. Você deve salvar este arquivo no mesmo diretório que o arquivo `qna.yaml`:

```
Título da obra: Exemplos de tokenização por wfulmyk
Licença da obra: CC-BY-SA-4.0
Nomes dos criadores: Wojciech "Victor" Fulmyk, com a ajuda de GenAI
```

Após incluir um arquivo `qna.yaml` e um arquivo `attribution.txt` na taxonomia, verifique se o arquivo `qna.yaml` está em conformidade executando o comando abaixo em um dos terminais:

```
ilab taxonomy diff
```

Se o seu arquivo `qna.yaml` estiver em conformidade, você verá a mensagem `Taxonomy in taxonomy is valid :)`. Caso contrário, você receberá uma mensagem informativa que fornecerá informações sobre quais partes do arquivo `qna.yaml` estão em desacordo. Corrija as partes não conformes do `qna.yaml` até que sua taxonomia seja válida.

Uma vez que sua taxonomia seja considerada válida, você pode gerar exemplos sintéticos usando o comando abaixo:

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

A flag `--num-instructions` indica ao InstructLab o número de exemplos a serem criados.

***Nota**: Você pode gerar quantos exemplos quiser, mas gerar exemplos sintéticos consome tempo. Neste laboratório, 100 exemplos sintéticos são suficientes.*

Os exemplos sintéticos gerados pelo InstructLab são impressos como a saída padrão para a janela do terminal.

Ajustar o modelo do aluno

Uma vez que o modelo do professor gera os exemplos sintéticos, você pode ajustar o modelo do aluno. Execute o comando abaixo para ajustar **Granite**:

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

***Nota**: O ajuste fino é realizado de maneira eficiente em termos de parâmetros utilizando adaptação de baixa classificação quantizada (QLoRA).*

Teste o modelo ajustado

O InstructLab vem equipado com um comando que permite comparar facilmente as respostas do modelo base e dos modelos ajustados aos prompts iniciais. Para fazer isso, execute o comando abaixo:

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

Use o modelo ajustado

Para usar o modelo ajustado, converta-o para um arquivo .gguf.

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

Após a conversão do modelo, sirva-o e converse com ele usando o mesmo método que foi utilizado para os modelos base. Em uma das janelas do terminal, use o comando abaixo para servir o modelo:

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

Em seguida, em outra janela de terminal, use o comando abaixo para conversar com o modelo ajustado.

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

Você pode interagir com o modelo livremente. No entanto, é mais informativo pedir que ele tokenize um trecho de texto, por exemplo.

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

No entanto, não é possível prever exatamente como o modelo irá responder, uma vez que a temperatura é maior que zero.

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

Parabéns! Você completou esta leitura!

Autores

[Wojciech "Victor" Fulmyk]

Registro de Mudanças

Data	Versão	Alterado por	Descrição da Mudança
aaaa-mm-dd	0.1	nome do alterador	Versão inicial criada
–>