## OPEN VS. CLOSED SOURCE LLMS

<table>
<thead>
<tr>
<th>Closed Source</th>
<th>Open Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Requires payment per token.</td>
<td>- Weights are freely available.</td>
</tr>
<tr>
<td>- Difficult to determine model architecture/training data for comparison.</td>
<td>- Only need infrastructure to use.</td>
</tr>
<tr>
<td>- Usually performs better.</td>
<td>- May have restrictions for commercial use. (LLaMA)</td>
</tr>
<tr>
<td>- Notable Sources: OpenAI (GPT-4, GPT-3.5), Anthropic (Claude-2, Claude-1)</td>
<td>- Weights can be modified (fine-tuned) for no cost.</td>
</tr>
<tr>
<td></td>
<td>- Notable sources: Meta (LLaMA), Google (T5, UL2), BigScience (Bloom)</td>
</tr>
</tbody>
</table>
A Guide into Open-Source Large Language Models and Fine-Tuning Techniques

INTERFACE FOR LLMS

Oobabooga
- Clean UI, lots of options.
- Supports many kinds of LLMs.
- Supports OpenAI style API and more customized API.
- 25.2K start on Github.

https://github.com/oobabooga/text-generation-webui

FastChat
- Built by creators of Vicuna.
- Includes “Chatbot Arena”.
- Supports OpenAI style API.
- More scalable.
- 28.2K stars on Github.

https://github.com/lm-sys/FastChat/tree/main
WHY USE A SERVER?

- **Scalable for multi-user case**
  - If multiple users need to query the model, it will not be scalable.
  - Model will take space on GPU even for inference.

- **Easy to switch out to OpenAI**
  - If using the OpenAI API, the main code stays the same, only the endpoint needs to be switched out.
A Guide into Open-Source Large Language Models and Fine-Tuning Techniques
### CURRENT OPEN SOURCE “STATE OF ART”

<table>
<thead>
<tr>
<th>Model</th>
<th>License</th>
<th>Commercial use?</th>
<th>Pretraining length [tokens]</th>
<th>Leaderboard score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Falcon-7B</td>
<td>Apache 2.0</td>
<td>✓</td>
<td>1,500B</td>
<td>47.01</td>
</tr>
<tr>
<td>MPT-7B</td>
<td>Apache 2.0</td>
<td>✓</td>
<td>1,000B</td>
<td>48.7</td>
</tr>
<tr>
<td>Llama-7B</td>
<td>Llama license</td>
<td>✗</td>
<td>1,000B</td>
<td>49.71</td>
</tr>
<tr>
<td>Llama-2.7B</td>
<td>Llama 2 license</td>
<td>✓</td>
<td>2,000B</td>
<td>54.32</td>
</tr>
<tr>
<td>Llama-33B</td>
<td>Llama license</td>
<td>✗</td>
<td>1,500B</td>
<td>*</td>
</tr>
<tr>
<td>Llama-2-13B</td>
<td>Llama 2 license</td>
<td>✓</td>
<td>2,000B</td>
<td>58.67</td>
</tr>
<tr>
<td>mpt-30B</td>
<td>Apache 2.0</td>
<td>✓</td>
<td>1,000B</td>
<td>55.7</td>
</tr>
<tr>
<td>Falcon-40B</td>
<td>Apache 2.0</td>
<td>✓</td>
<td>1,000B</td>
<td>61.5</td>
</tr>
<tr>
<td>Llama-65B</td>
<td>Llama license</td>
<td>✗</td>
<td>1,500B</td>
<td>62.1</td>
</tr>
<tr>
<td>Llama-2-70B</td>
<td>Llama 2 license</td>
<td>✓</td>
<td>2,000B</td>
<td>*</td>
</tr>
<tr>
<td>Llama-2-70B-chat*</td>
<td>Llama license</td>
<td>✓</td>
<td>2,000B</td>
<td>66.8</td>
</tr>
</tbody>
</table>
Represent changes to a model’s weights using less weights.


Uses less memory during training.

Uses less disk space to store the changes.

Most common method is LoRA (Low-Rank Adaptation):
- Uses two small matrices to generate a large but low-rank matrix that is added to the weights.
Use smaller data types than float32 (int8, int4, or even 1 bit).

Libraries: bitsandbytes, AutoGPTQ, ExLlama.

Pros:
- Less memory during inference.
- Less memory during training (using QLoRA)

Cons:
- Lower model accuracy

Figure 3: The accuracy of OPT and BLOOM models post-GPTQ, measured on LAMBADA.

https://arxiv.org/abs/2210.17323
COMMON INFEREN CE PARAMETERS

- temperature
- top_k
- top_p

https://huggingface.co/docs/transformers/v4.34.0/en/main_classes/text_generation#generation
Higher temperature introduces more randomness

Formula:
- $\text{probabilities} = \text{torch.softmax}(\text{logits} / \text{temperature})$

What if temperature $= 0$?
- Approaches greedy sampling in the limit
TEMPERATURE = 0.25

A Guide into Open-Source Large Language Models and Fine-Tuning Techniques
TEMPERATURE = 0.5
TEMPERATURE = 2.0
TOP_K

- Only allow a certain number of the highest probability tokens to be sampled
- “certain number” = “top_k”
- Example: top_k = 2
TOP_P

- Also called “Nucleus Sampling”
- Remove the lowest probability as long as it adds up to “top_p”
TOP_P = 0.9 (Step 1)

- Old sum = 1
- New sum = 0.988
TOP\_P = 0.9 (Step 2)

- Old sum = 0.988
- New sum = 0.957
TOP_P = 0.9 (Step 3)

- Old sum = 0.957
- New sum = 0.870
  - We can't go past 0.9
TOP_P = 0.9 (Step 4)

- Old sum = 0.870
- New sum = 0.957
SO HOW DO I GET THE WEIGHTS?

- **Official request form for LLaMa 2:**
  - [https://docs.google.com/forms/d/e/1FAIpQLSfqNECQnMkycAp2jP4Z9TFX0cGR4uf7b_fBxjY_OjhJILIKGA/viewform](https://docs.google.com/forms/d/e/1FAIpQLSfqNECQnMkycAp2jP4Z9TFX0cGR4uf7b_fBxjY_OjhJILIKGA/viewform)
  - Signing up through this form will give you access to the official LLaMa2 repository on huggingface as well.

- **Unofficial weights (including 3rd party fine-tuned):**
  - [https://huggingface.co/TheBloke](https://huggingface.co/TheBloke)
  - A reliable source of weights for now.
### WHICH ONE DO I CHOOSE?

So many choices!

<table>
<thead>
<tr>
<th>Models</th>
<th>Downloads</th>
<th>Likes</th>
</tr>
</thead>
<tbody>
<tr>
<td>TheBloke/Wizard-Vicuna-7B-Uncensored-GPTQ</td>
<td>453k</td>
<td>75</td>
</tr>
<tr>
<td>TheBloke/vicuna-7B-v1.3-GPTQ</td>
<td>129k</td>
<td>10</td>
</tr>
<tr>
<td>TheBloke/Llama-2-13B-chat-GPTQ</td>
<td>111k</td>
<td>257</td>
</tr>
<tr>
<td>TheBloke/Llama-2-70B-chat-GPTQ</td>
<td>38k</td>
<td>185</td>
</tr>
<tr>
<td>TheBloke/OpenAssistant-Llama2-13B-Orca-v2-8K-3166-GPTQ</td>
<td>30.4k</td>
<td>17</td>
</tr>
<tr>
<td>TheBloke/Llama-2-70B-GPTQ</td>
<td>235k</td>
<td>61</td>
</tr>
<tr>
<td>TheBloke/Llama-2-7b-Chat-GPTQ</td>
<td>115k</td>
<td>136</td>
</tr>
<tr>
<td>TheBloke/MythoMax-L2-13B-GPTQ</td>
<td>96k</td>
<td>65</td>
</tr>
<tr>
<td>TheBloke/llama-2-70b-Guanaco-QLoRA-GPTQ</td>
<td>30.6k</td>
<td>35</td>
</tr>
<tr>
<td>TheBloke/gpt4-alpaca-lora_mlp-65B-GPTQ</td>
<td>21.6k</td>
<td>13</td>
</tr>
</tbody>
</table>
LOADING THE MODEL WITH WEBUI

1. Go to model tab
2. Type/Paste model name/branch
3. Select and load model after reload
ENABLING API EXTENSIONS

1. Go to session tab

2. Enable your extensions
First clone the selected weights in **text-generation-webui/models**:
```
git clone --single-branch --branch gptq-4bit-32g-actorder_True
```

Then start the server with desired options in **text-generation-webui**:
```
python server.py --model TheBloke_Upstage-Llama-2-70B-instruct-v2-GPTQ_gptq-4bit-32g-actorder_True --loader exllama --api --extensions openai
```
EXAMPLE CODE FOR OPENAI API

```python
import os
os.environ['OPENAI_API_KEY'] = "sk-1111111111111111111111111111111111111111111111111111111111111"

prompt = [
    {'role': 'user',
    'content': 'You are a helpful assistant. You will answer questions I ask you. Reply with Yes if you understand.'
    },
    {'role': 'assistant',
    'content': 'Yes, I understand'}
    ]

response = openai.ChatCompletion.create(
    model="g",
    messages = prompt
)
output = response['choices'][0]['message']['content']
print('Model output:', output)
```

Model output: The sky appears blue due to the scattering of sunlight by the atmosphere. However, during different times of day and in various weather conditions, it can appear in other colors like shades of orange or red at sunrise/sunset or grey when covered with clouds.
EXAMPLE CODE FOR OPENAI API

```python
import os
import requests
from chat_api import DEFAULT_CHAT_PARAMS
import html

instruction = 'Your job is to play the assigned role and give responses to your best ability.'

chat_history = [
    'You are a helpful assistant. You will answer questions I ask you. Reply with Yes if you understand.
    'Yes, I understand'
]

params = dict(
    **DEFAULT_CHAT_PARAMS,
    user_input = 'What color is the sky?','
    history = dict(
        internal = chat_history,
        visible = chat_history,
    ),
    context_instruct = instruction,
)

response = requests.post('http://localhost:5000/api/v1/chat', json=params)
result = response.json()['results'][0]['history']
output = html.escape(result['visible'][-1][1])
print(output)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER

(lm-preference) [kang@idea-node-07 lm-preference]$ python sample2.py
The sky appears blue due to the scattering of sunlight by the atmosphere. However, it can also appear different colors at various times such as orange during sunrise or sunset, gray during cloudy days, or even black in space.
Pay attention to the delimiters – models can be fine-tuned with different templates.

Usually documented in the model page.
HOW TO FINE-TUNE

- The task of the model is still text completion.
- Each row in the training dataset is a piece of text that describes the input and desired output for that input:

Example:

<s>[INST] You will create a table with macronutrients, micronutrients and kcal of the following foods: 100g oatmeal, banana 100g, apple 100g, pasta ragù 100g [/INST] Here is a table showing the macronutrients, micronutrients, and kcal of 100g of oatmeal, banana, apple, and pasta ragù: Food Carbohydrates (g) Fat (g) Protein (g) Fiber (g) Potassium (mg) Vitamin C (mg) kcal Oatmeal 66.3 6.9 16.9 8.0 — — 389 Banana 23.0 0.3 1.3 2.6 358 8.7 89 Apple 14.0 0.2 0.3 2.4 107 4.6 52 Pasta Ragù 53.0 12.0 12.0 2.0 — — 300 Note: The above table is an estimation and may vary slightly depending on the specific </s><s>[INST] Thanks, that's awesome! [/INST] No prob, bud! </s>
1. **Install requirements:**
   - **accelerate**: device management (gpu/cpu) framework
   - **peft**: cuda wrapper for lower bit (quantized) manipulation
   - **bitsandbytes**: huggingface cli for downloading model/datasets
   - **transformers**: for training reinforcement learning models on transformers

2. **Load desired model.**
   - Any of the models from TheBloke can be loaded through transformers

3. **(optional but probably reasonable) Apply LoRA to the model instead of training the full model using peft.**
   - Or download the model with the LoRA included.

Same as fine-tuning any other transformers model!

- **Useful References:**
  - [https://github.com/inwonakng/lm-usergroup-examples](https://github.com/inwonakng/lm-usergroup-examples)
  - [https://mlabonne.github.io/blog/posts/Fine_Tune_Your_Own_Llama_2_Model_in_a_Colab_Notebook.html](https://mlabonne.github.io/blog/posts/Fine_Tune_Your_Own_Llama_2_Model_in_a_Colab_Notebook.html)
SOME NOTES

- LLaMa2 models need to have their sequence length and gradient options set explicitly.
  - After loading model:
  - `model = exllama_set_max_input_length(model, 8192)`
  - `model.enable_input_require_grads()`
  - This will probably be patched soon, needed for now.

- Instead of quantizing your own version, most models on TheBloke provide already quantized weights. Consider using those instead.
  - Can be set by `revision` field of `AutoModelForCausalLM.pretrained()`
PREPARING DATA

- LLMs understand text.
- The input should be wrapped into a pure text format.
  - Can also handle markdown delimiters or other text-based formats like JSON.
- The template headings can be chosen arbitrarily
  - But make sure they are distinct!

Example of text setup for classification task.
### HOW TO FINE-TUNE – 2 Options

<table>
<thead>
<tr>
<th>Option 1. Use Python Script</th>
<th>Option 2. Use WebUI</th>
</tr>
</thead>
<tbody>
<tr>
<td>▪ Easier (if you have used transformers before)</td>
<td>▪ Easier (if you are not familiar with python/transformers)</td>
</tr>
<tr>
<td>▪ More fine-tuned control (device manipulation etc.)</td>
<td>▪ Integrates nicely with WebUI</td>
</tr>
</tbody>
</table>
OPTION 1. Python Script

1. Load the model using `transformers`.

2. (Optional) Quantize using `bitsandbytes`.
   1. Not necessary if the model is already quantized.

3. Parse dataset into a prompt format.

4. Configure training parameters.
   - `save_dir`, `lr`, `optimizer`, `wd`, etc.

5. Train & Save model.

The saved folder can be used by text-generation-webui once it is moved under `text-generation-webui/loras`

Sample Code:

https://github.com/inwonakng/llm-usergroup-examples/blob/main/fine-tuning/huggingface.py
OPTION 2. WEBUI

1. Go to training tab

2. Choose LoRA to copy shape from

3. Set dataset

Refer to tutorial for formatting
OPTION 2. WEBUI

1. Go to model tab
2. Load base model
3. Load LoRA weights
Training larger models on a single GPU may be very time consuming.

If you have access to a cluster with multiple nodes, consider using ray[train] for distributed training.

https://docs.ray.io/en/latest/train/examples/deepspeed/gptj_deepspeed_fine_tuning.html#gptj-deepspeed-finetune

Same as before, progress can be observed using Tensorboard or Weights and Biases.
QUESTION / COMMENTS?

- Inwon Kang: kangi@rpi.edu
- Tripp Lyons: lyonsd2@rpi.edu