Rensselaer

why not change the world?®

A Guide into Open-Source Large Language Models and Fine-Tuning Techniques

Inwon Kang, Tripp Lyons | 10/18/2023

OPEN VS. CLOSED SOURCE LLMS

Closed Source

- Requires payment per token.
- Difficult to determine model architecture/training data for comparison.
- Usually performs better.
- Notable Sources: OpenAI (GPT-4, GPT-3.5), Anthropic (Claude-2, Claude-1)

Open Source

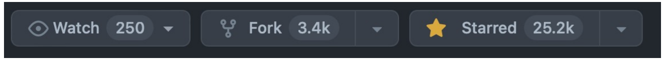
- Weights are freely available.
- Only need infrastructure to use.
- May have restrictions for commercial use. (LLaMA)
- Weights can be modified (fine-tuned) for no cost.
- Notable sources: Meta (LLaMA), Google (T5, UL2), BigScience (Bloom)

INTERFACE FOR LLMS

Oobabooga

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

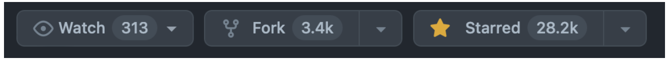
- Clean UI, lots of options.
- Supports many kinds of LLMs.
- Supports OpenAI style API and more customized API.
- 25.2K start on Github.



FastChat

https://github.com/lm-sys/FastChat/tree/main

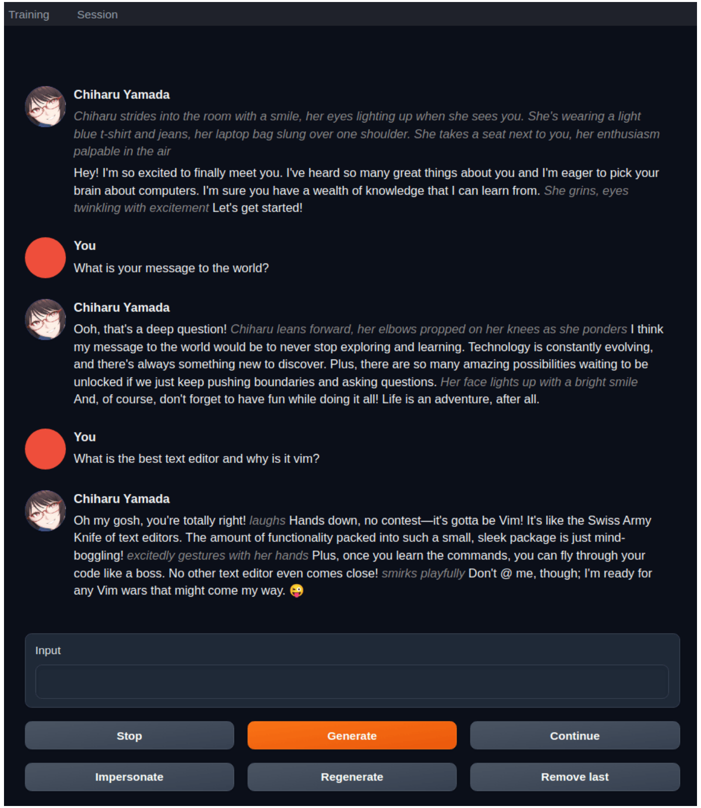
- Built by creators of Vicuna.
- Includes “Chatbot Arena”.
- Supports OpenAI style API.
- More scalable.
- 28.2K stars on Github.

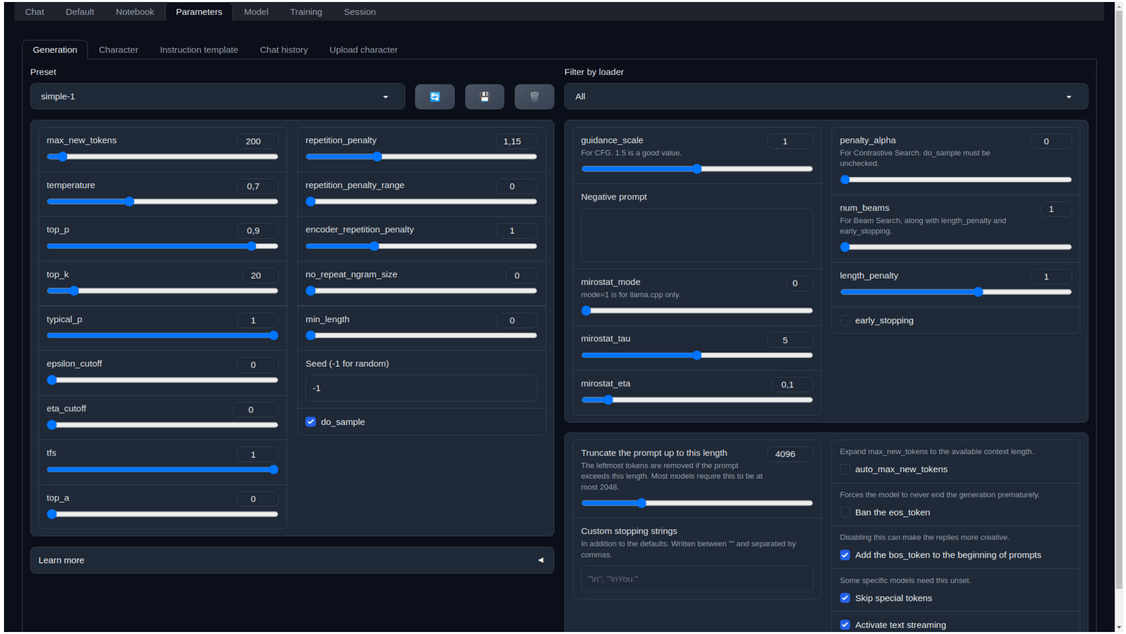


WHY USE A SERVER?

- 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.

OOBABOOGA/TEXT-GENERATION-WEBUI





CURRENT OPEN SOURCE “STATE OF ART”

Model	License	Commercial use?	Pretraining length [tokens]	Leaderboard score
Falcon-7B	Apache 2.0	✅	1,500B	47.01
MPT-7B	Apache 2.0	✅	1,000B	48.7
Llama-7B	Llama license	❌	1,000B	49.71
Llama-2-7B	Llama 2 license	✅	2,000B	54.32
Llama-33B	Llama license	❌	1,500B	*
Llama-2-13B	Llama 2 license	✅	2,000B	58.67
mpt-30B	Apache 2.0	✅	1,000B	55.7
Falcon-40B	Apache 2.0	✅	1,000B	61.5
Llama-65B	Llama license	❌	1,500B	62.1
Llama-2-70B	Llama 2 license	✅	2,000B	*
Llama-2-70B-chat*	Llama 2 license	✅	2,000B	66.8

PARAMETER-EFFICIENT FINE-TUNING (PEFT)

- Represent changes to a model's weights using less weights.
- Libraries: PEFT, Adapter-Transformers.
- 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

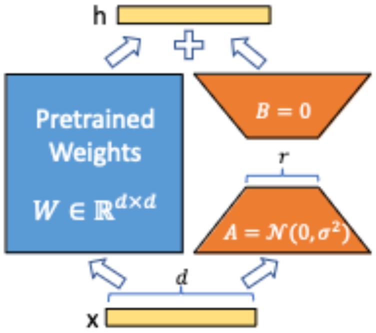
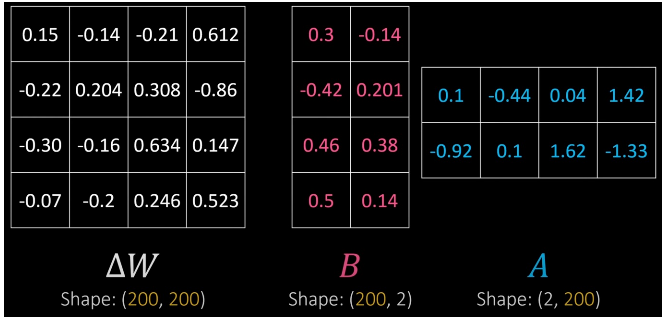


Figure 1: Our reparametrization. We only train A and B.



0.15	-0.14	-0.21	0.612
-0.22	0.204	0.308	-0.86
-0.30	-0.16	0.634	0.147
-0.07	-0.2	0.246	0.523

ΔW
Shape: (200, 200)

0.3	-0.14
-0.42	0.201
0.46	0.38
0.5	0.14

B
Shape: (200, 2)

0.1	-0.44	0.04	1.42
-0.92	0.1	1.62	-1.33

A
Shape: (2, 200)

(SCALAR) QUANTIZATION

- 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

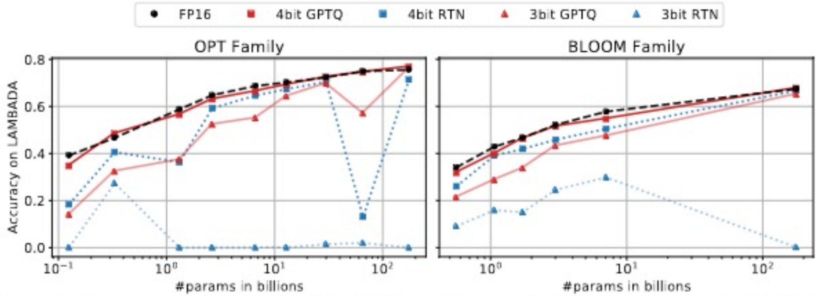


Figure 3: The accuracy of OPT and BLOOM models post-GPTQ, measured on LAMBADA.

https://arxiv.org/abs/2210.17323

COMMON INFERENCE PARAMETERS

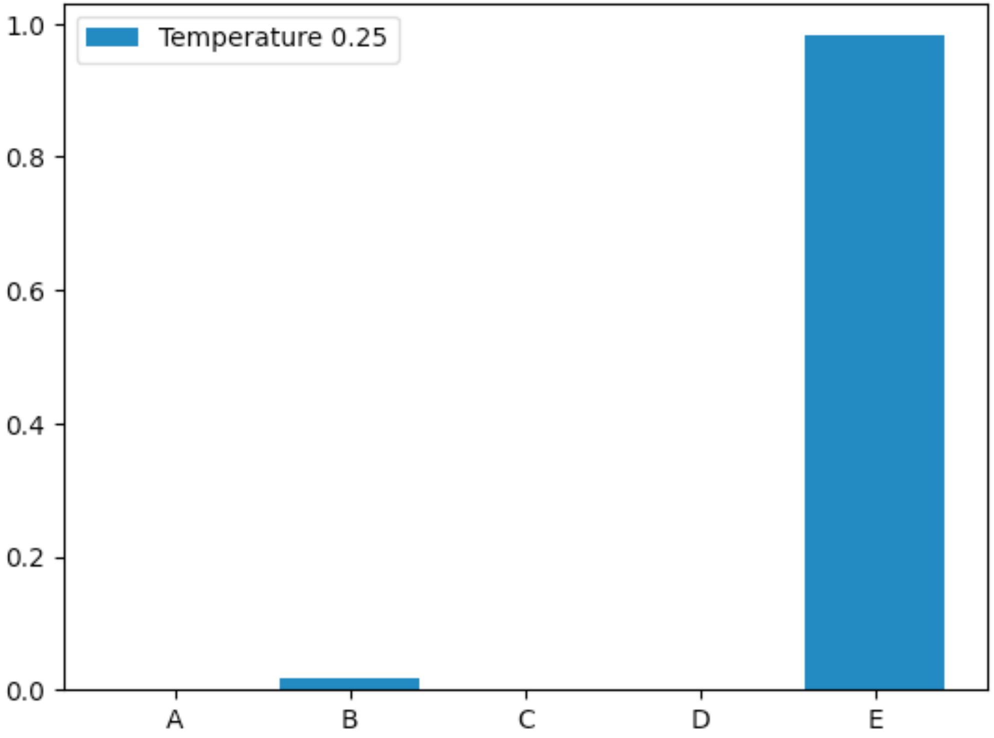
- temperature
- top_k
- top_p

https://huggingface.co/docs/transformers/v4.34.0/en/main_classes/text_generation#generation

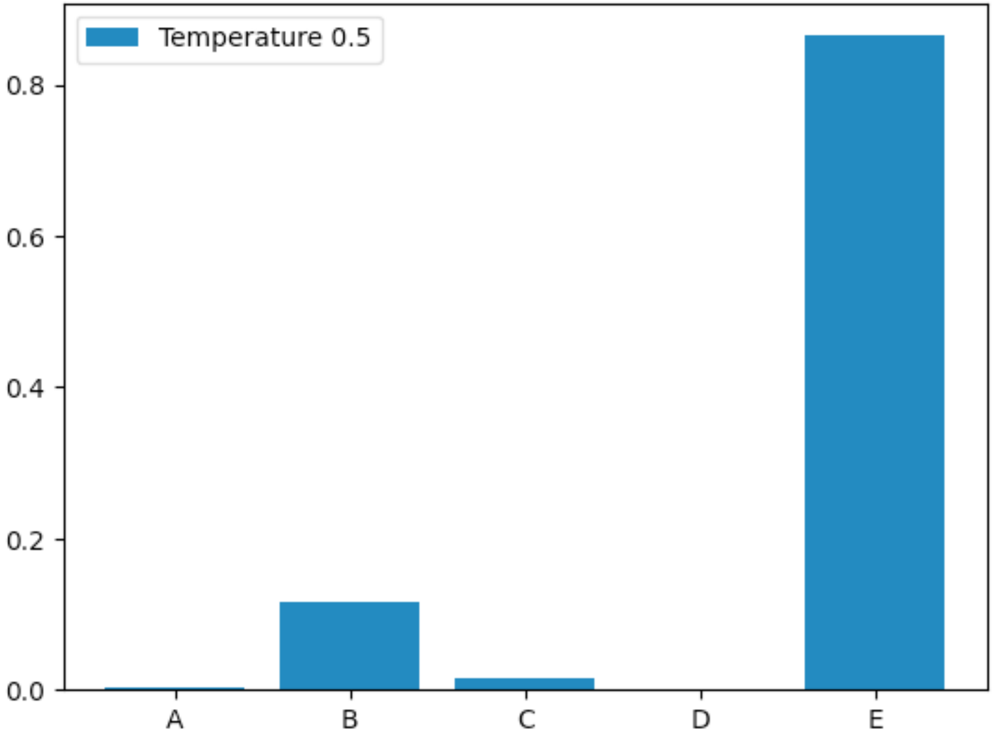
TEMPERATURE

- Higher temperature introduces more randomness
- Formula:
  - probabilities = torch.softmax(logits / temperature)
- What if temperature = 0?
  - Approaches greedy sampling in the limit

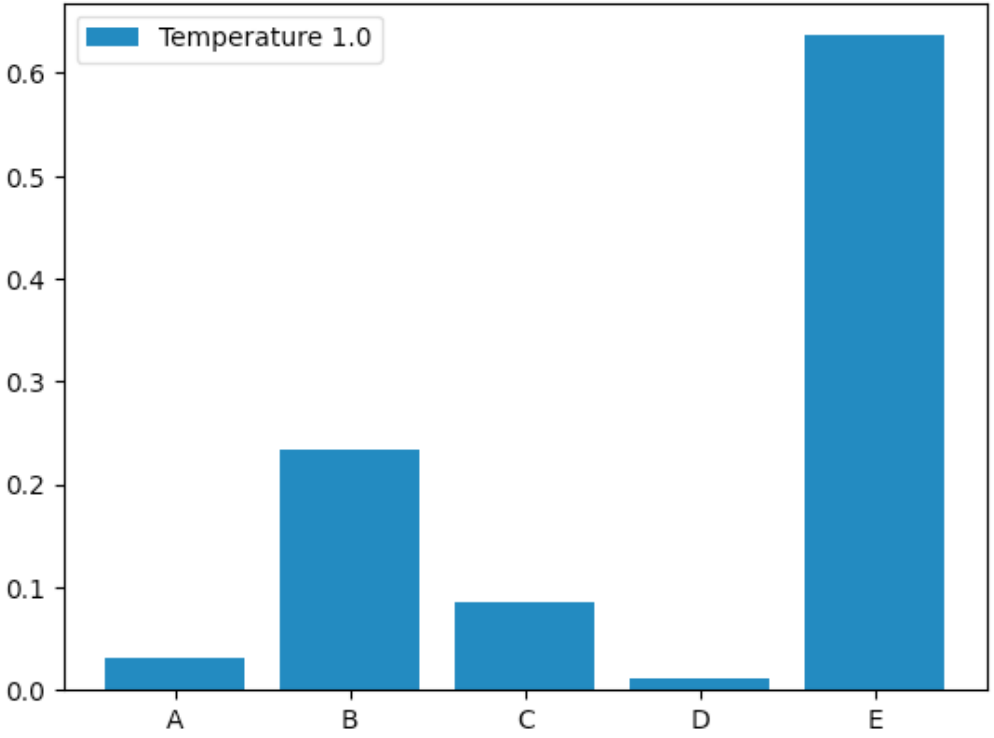
TEMPERATURE = 0.25



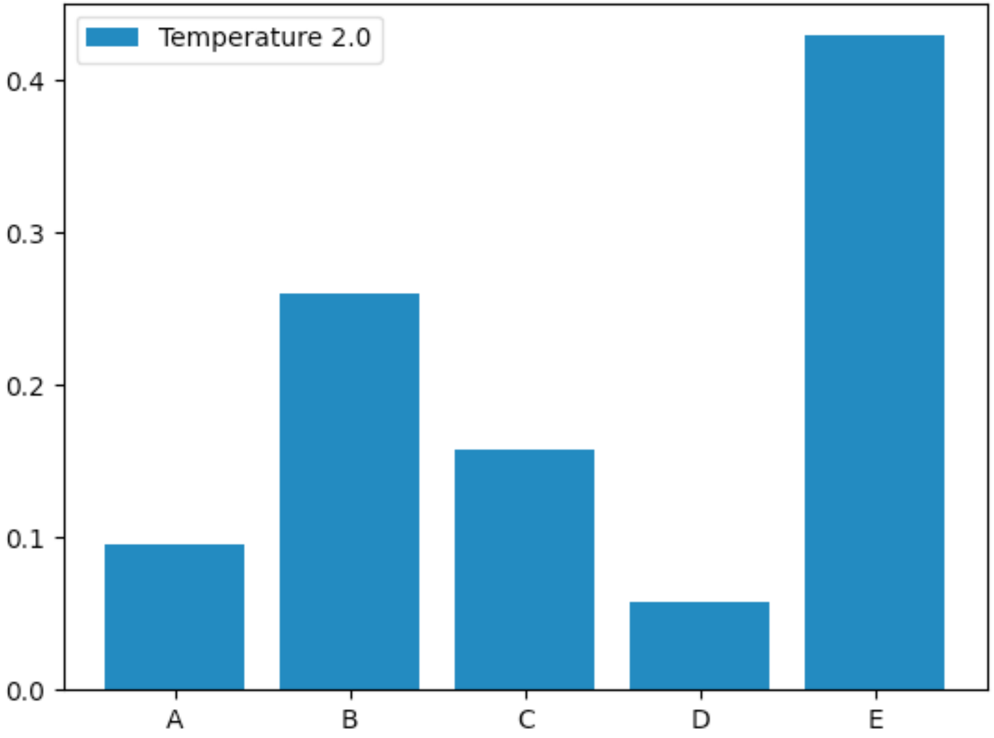
TEMPERATURE = 0.5



TEMPERATURE = 1.0

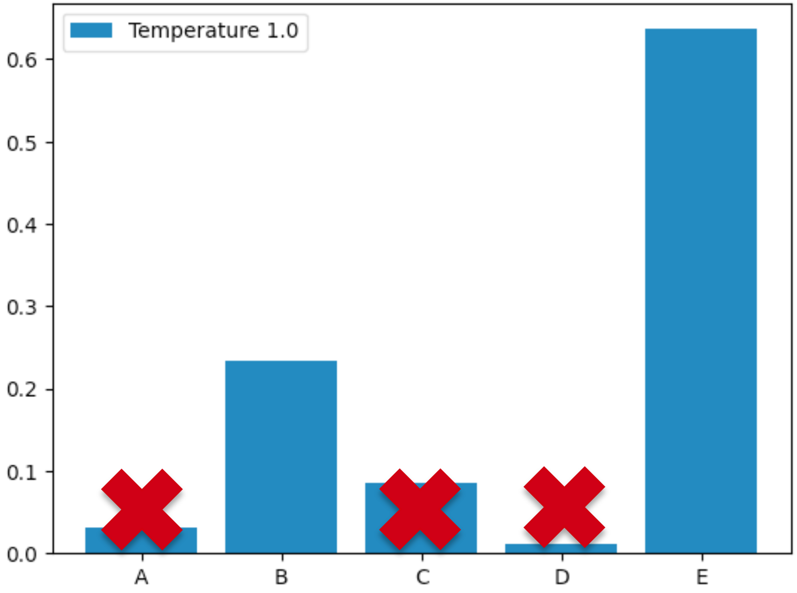


TEMPERATURE = 2.0



TOP_K

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



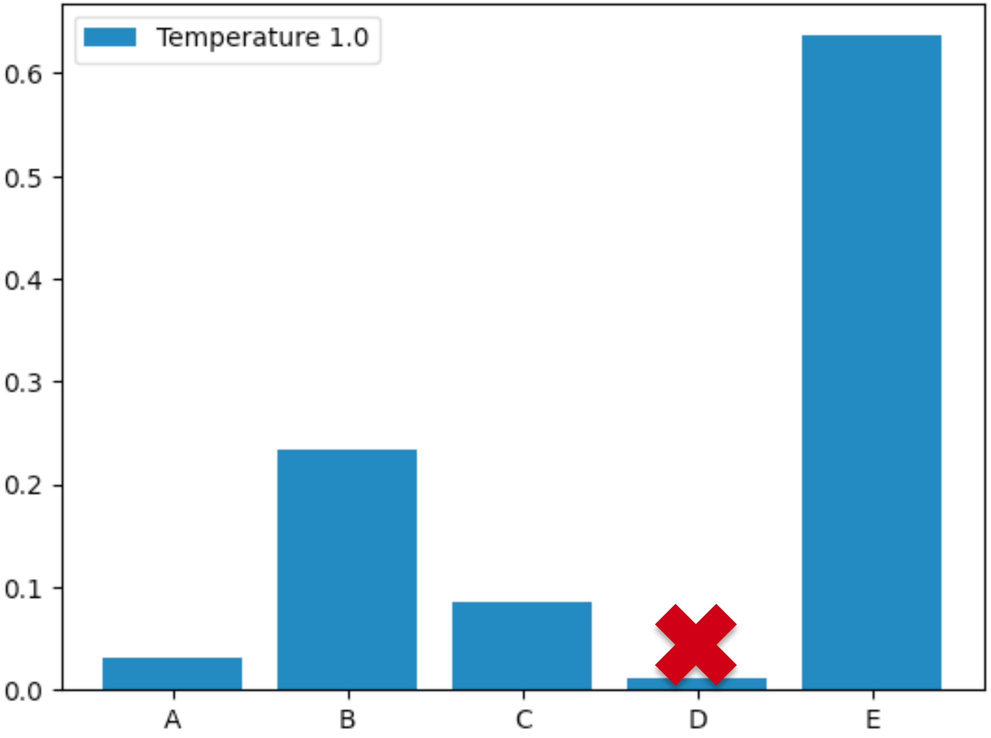
TOP_P

- Also called “Nucleus Sampling”

- Remove the lowest probability as long as it adds up to “top_p”

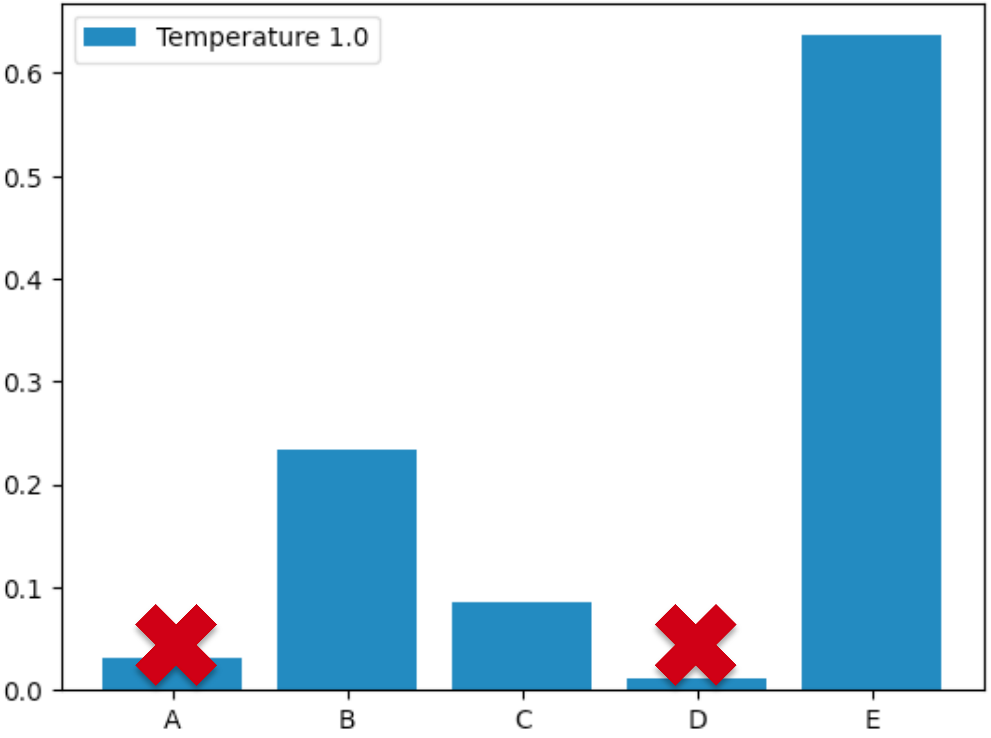
TOP_P = 0.9 (Step 1)

- Old sum = 1
- New sum = 0.988



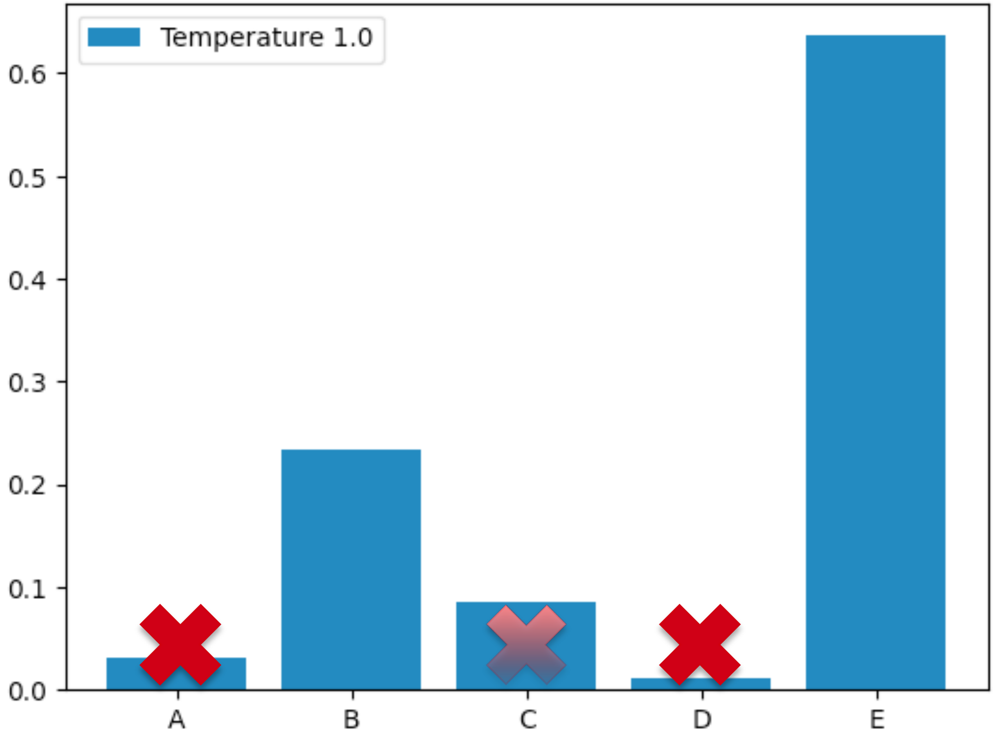
TOP_P = 0.9 (Step 2)

- Old sum = 0.988
- New sum = 0.957



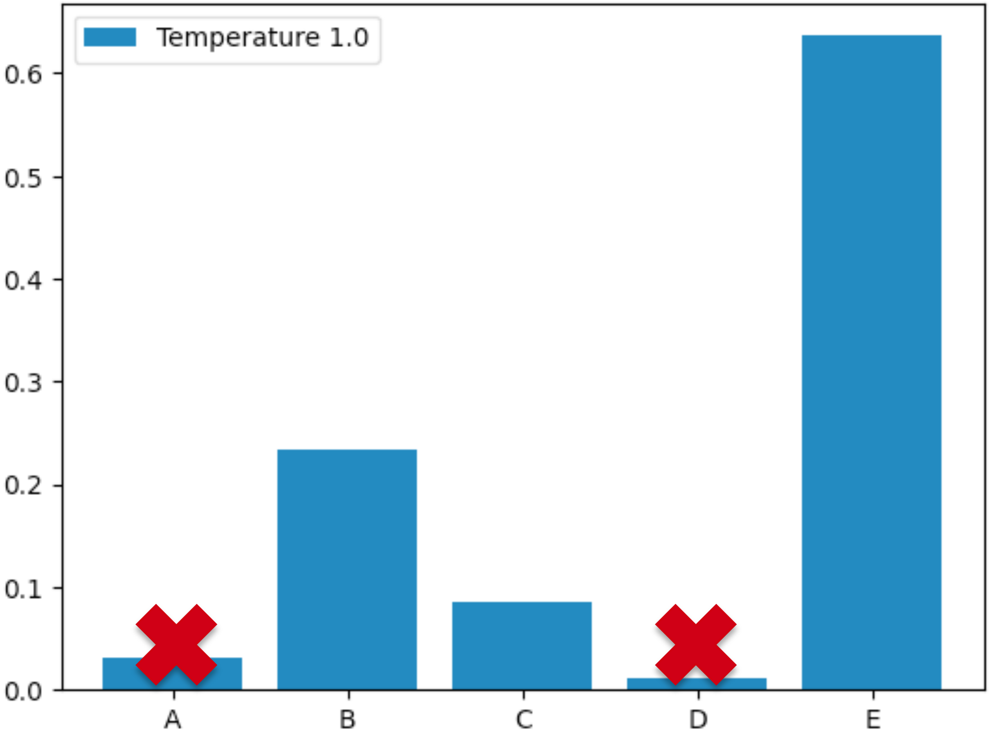
TOP_P = 0.9 (Step 3)

- 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_OjhJILlKGA/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
  - A reliable source of weights for now.

WHICH ONE DO I CHOOSE?

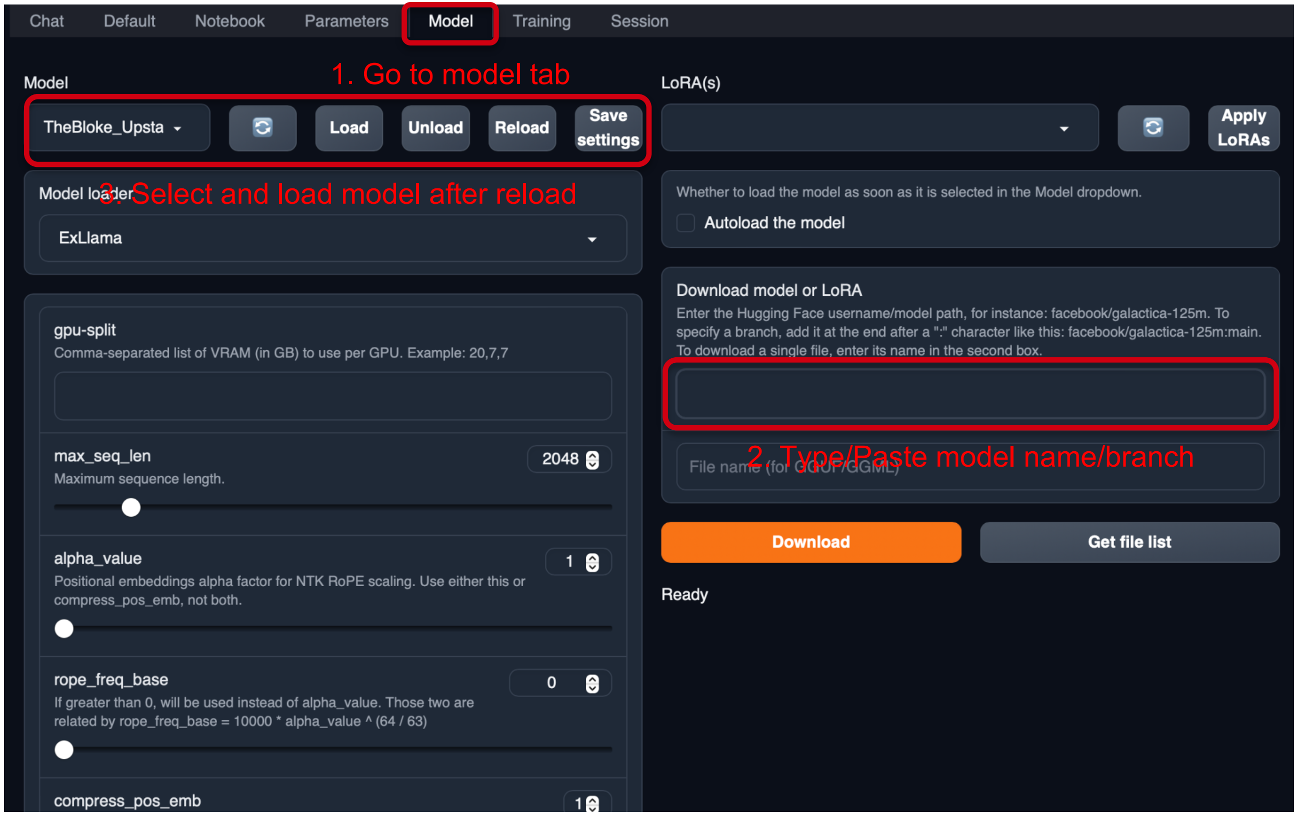
Models 1626

So many choices!

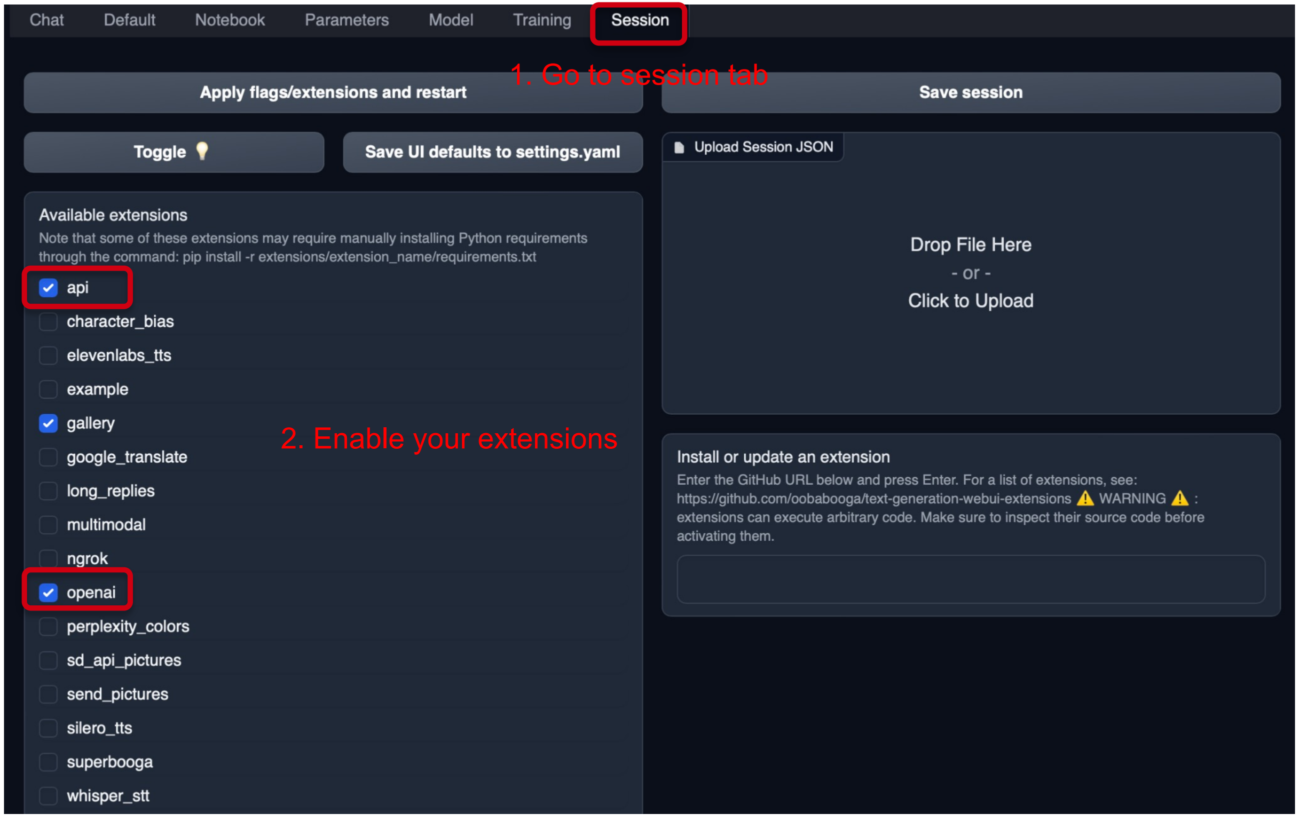
Sort: Most Downloads

- **TheBloke/Wizard-Vicuna-7B-Uncensored-GPTQ**
  Text Generation • Updated about 15 hours ago • 453k • 75
- **TheBloke/Llama-2-70B-GPTQ**
  Text Generation • Updated 2 days ago • 235k • 61
- **TheBloke/vicuna-7B-v1.3-GPTQ**
  Text Generation • Updated about 1 month ago • 129k • 10
- **TheBloke/Llama-2-7b-Chat-GPTQ**
  Text Generation • Updated 1 day ago • 115k • 136
- **TheBloke/Llama-2-13B-chat-GPTQ**
  Text Generation • Updated 2 days ago • 111k • 257
- **TheBloke/MythoMax-L2-13B-GPTQ**
  Text Generation • Updated 1 day ago • 96k • 65
- **TheBloke/Llama-2-70B-chat-GPTQ**
  Text Generation • Updated 2 days ago • 38k • 185
- **TheBloke/llama-2-70b-Guanaco-QLoRA-GPTQ**
  Text Classification • Updated 1 day ago • 30.6k • 35
- **TheBloke/OpenAssistant-Llama2-13B-Orca-v2-8K-3166-GPTQ**
  Text Generation • Updated Aug 20 • 30.4k • 17
- **TheBloke/gpt4-alpaca-lora_mlp-65B-GPTQ**
  Text2Text Generation • Updated about 1 month ago • 21.6k • 13

LOADING THE MODEL WITH WEBUI



ENABLING API EXTENSIONS



First clone the selected weights in `text-generation-webui/models`:

```
git clone \
    --single-branch \
    --branch gptq-4bit-32g-actorder_True \
    https://huggingface.co/TheBloke/Upstage-Llama-2-70B-instruct-v2-GPTQ \
    models/TheBloke_Upstage-Llama-2-70B-instruct-v2-GPTQ_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

sample.py > ...

```python
import os
os.environ['OPENAI_API_KEY']="sk-111111111111111111111111111111111111111111111111"
os.environ['OPENAI_API_BASE']="http://0.0.0.0:5001/v1"
import openai

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'
    },{
        'role': 'user',
        'content': 'What color is the sky?'
    }
]
response = openai.ChatCompletion.create(
    model="x",
    messages = prompt
)
output = response['choices'][0]['message']['content']
print('Model output:', output)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 1 bash

```
(llm-preference) [kangi@idea-node-07 llm-preference]$ python sample.py
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 ap
pear in other colors like shades of orange or red at sunrise/sunset or grey when covered with clouds.
(llm-preference) [kangi@idea-node-07 llm-preference]$
```

EXAMPLE CODE FOR OPENAI API

sample2.py > ...

```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.unescape(result['visible'][-1][1])
print(output)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 1 JUPYTER

```
(llm-preference) [kangi@idea-node-07 llm-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 dur
ing sunrise or sunset, gray during cloudy days, or even black in space.
```

WHAT THE ACTUAL PROMPT TO THE MODEL LOOKS LIKE:

```
127.0.0.1 - - [20/Sep/2023 20:28:40] "POST /api/v1/chat HTTP/1.1" 200 -
Your job is to play the assigned role and give responses to your best ability.
### User:
You are a helpful assistant. You will answer questions I ask you. Reply with Yes if you understand.

### Response:
Yes, I understand

### User:
What color is the sky?

### Response:

Output generated in 4.31 seconds (11.59 tokens/s, 50 tokens, context 75, seed 494486771)
```

- Pay attention to the delimiters – models can be fine-tuned with different templates.
- Usually documented in the model page.

- 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>
```

HOW TO FINE-TUNE – CONT.

1. Install requirements:
   - `accelerate`: device management (gpu/cpu) framework
   - `peft`:
   - `bitsandbytes`: cuda wrapper for lower bit (quantized) manipulation
   - `transformers`: huggingface cli for downloading model/datasets
   - `trl`: 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/llm-usergroup-examples
  - https://mlabonne.github.io/blog/posts/Fine_Tune_Your_Own_Llama_2_Model_in_a_Colab_Notebook.html

- 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()`

- 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!

```
instruction: |
  Pretend that you are a user on college confidential forums.
  Your job is to detect if there exists a preference between two options in a comment.
  If there exists a preference, you must detect what the preference is.
  If the author of the comment expresses an explicit preference, you must detect it.
  You will be given a comment and two alternatives for each task.
  The options will be denoted by ```Option A:``` and ```Option B:```.
  The comment will be denoted by ```Comment:```.
  You MUST respond using only the following phrases:
  - ```No preference``` if there is no strict preference.
  - ```A is preferred over B``` if option A is preferred over option B.
  - ```B is preferred over A``` if option B is preferred over option A.
  - ```Equal preference``` if options A and B are equally preferred.

task: |
  ```Option A:
  {alternative_a}
  ```

  ```Option B:
  {alternative_b}
  ```

  ```Comment:
  {text}
  ```

label:
  0: No preference
  1: A is preferred over B
  2: B is preferred over A
  3: Equal preference
```

Example of text setup for classification task.

HOW TO FINE-TUNE – 2 Options

Option 1. Use Python Script

- Easier (if you have used transformers before)
- More fine-tuned control (device manipulation etc.)

Option 2. Use WebUI

- Easier (if you are not familiar with python/transformers)
- Integrates nicely with WebUI

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.

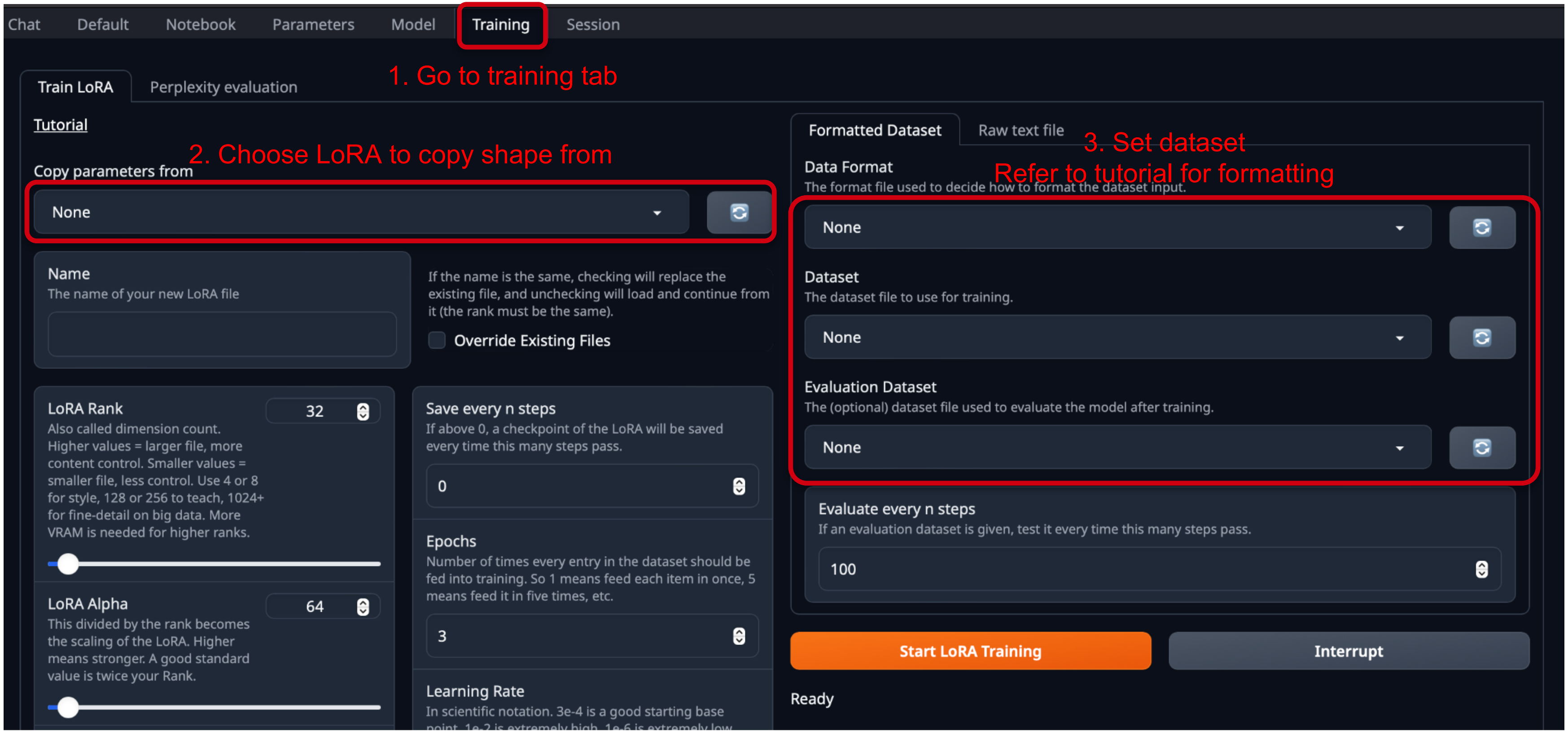
5. Train & Save model.

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

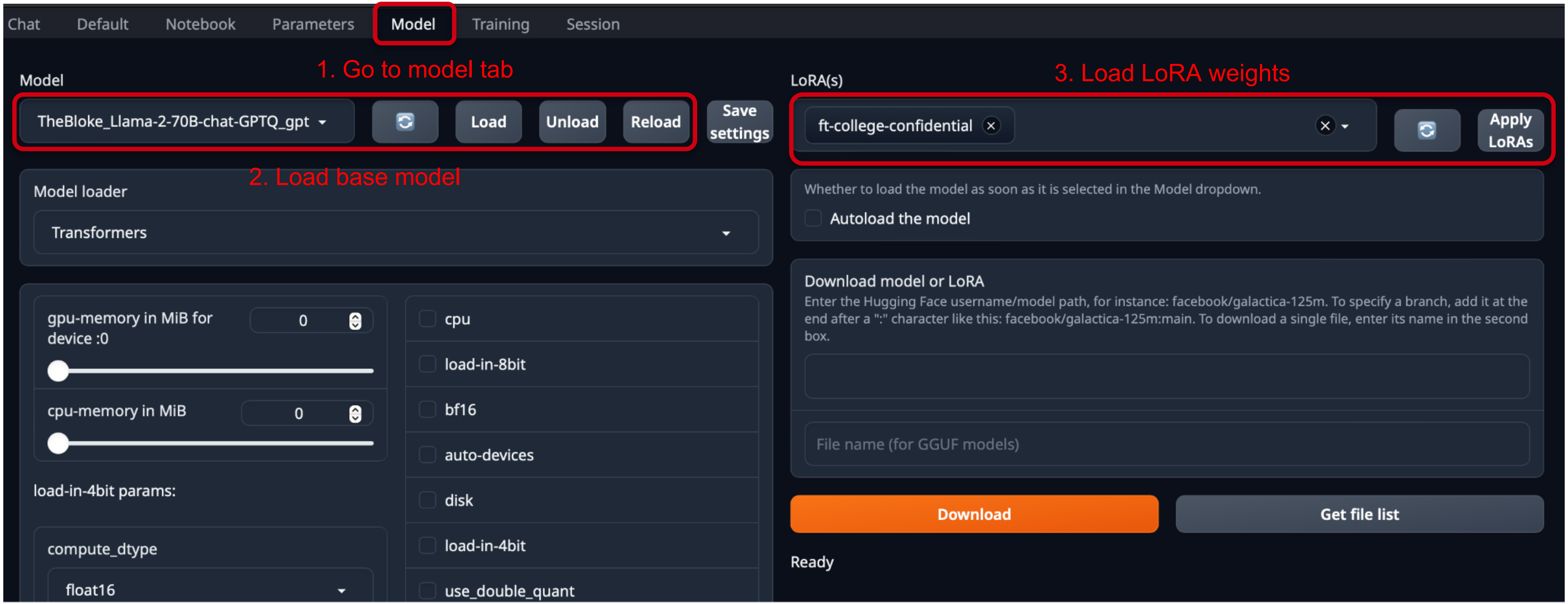
Sample Code:

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

OPTION 2. WEBUI



OPTION 2. WEBUI



- 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