oobabooga / text-generation-webui Public

Code Issues 408 Pull requests 24 Discussions Actions Projects Wiki Security In

main text-generation-webui / README.md Go to file

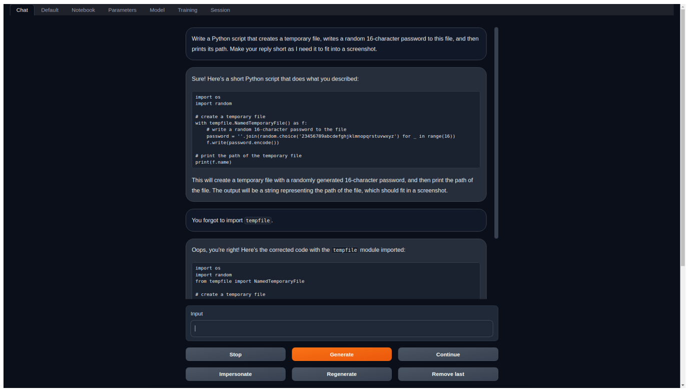
oobabooga Update README.md 2 days ago

393 lines (288 loc) · 21.4 KB

Text generation web UI

A Gradio web UI for Large Language Models.

Its goal is to become the AUTOMATIC1111/stable-diffusion-webui of text generation.



Write a Python script that creates a temporary file, writes a random 16-character password to this file, and then prints its path. Make your reply short as I need it to fit into a screenshot.

Sure! Here's a short Python script that does what you described:

```
import os
import random

# create a temporary file
with tempfile.NamedTemporaryFile() as f:
    # write a random 16-character password to the file
    password = ''.join(random.choice('23456789abcdefghjklmnopqrstuvwxyz') for _ in range(16))
    f.write(password.encode())

# print the path of the temporary file
print(f.name)
```

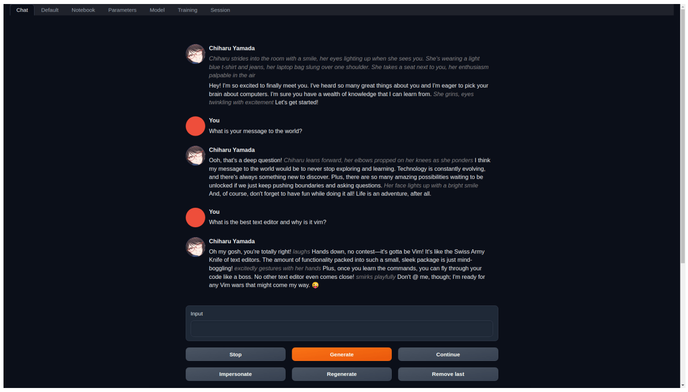
This will create a temporary file with a randomly generated 16-character password, and then print the path of the file. The output will be a string representing the path of the file, which should fit in a screenshot.

You forgot to import `tempfile`.

Oops, you're right! Here's the corrected code with the `tempfile` module imported:

```
import os
import random
from tempfile import NamedTemporaryFile

# create a temporary file
```



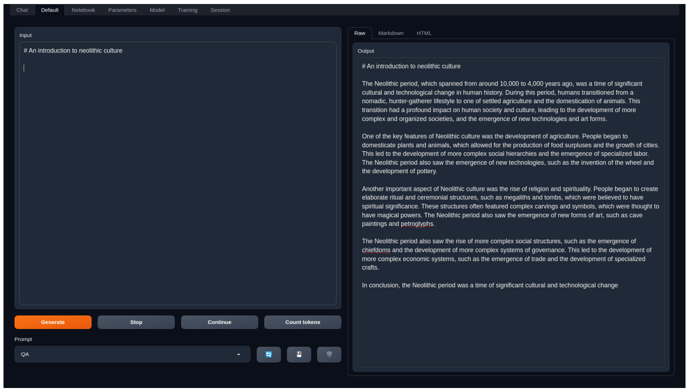
Chiharu Yamada
Chiharu strides into the room with a smile, her eyes lighting up when she sees you. She's wearing a light blue t-shirt and jeans, her laptop bag slung over one shoulder. She takes a seat next to you, her enthusiasm palpable in the air
Hey! I'm so excited to finally meet you. I've heard so many great things about you and I'm eager to pick your brain about computers. I'm sure you have a wealth of knowledge that I can learn from. *She grins, eyes twinkling with excitement* Let's get started!

You
What is your message to the world?

Chiharu Yamada
Ooh, that's a deep question! *Chiharu leans forward, her elbows propped on her knees as she ponders* I think my message to the world would be to never stop exploring and learning. Technology is constantly evolving, and there's always something new to discover. Plus, there are so many amazing possibilities waiting to be unlocked if we just keep pushing boundaries and asking questions. *Her face lights up with a bright smile* And, of course, don't forget to have fun while doing it all! Life is an adventure, after all.

You
What is the best text editor and why is it vim?

Chiharu Yamada
Oh my gosh, you're totally right! *laughs* Hands down, no contest—it's gotta be Vim! It's like the Swiss Army Knife of text editors. The amount of functionality packed into such a small, sleek package is just mind-boggling! *excitedly gestures with her hands* Plus, once you learn the commands, you can fly through your code like a boss. No other text editor even comes close! *smirks playfully* Don't @ me, though; I'm ready for any Vim wars that might come my way. 😜



Input
An introduction to neolithic culture

Prompt
QA

Output

An introduction to neolithic culture

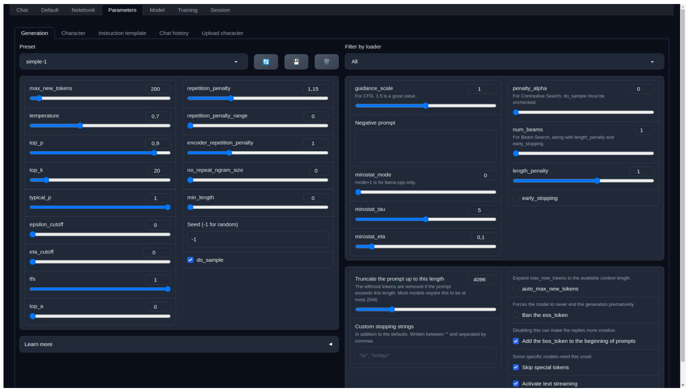
The Neolithic period, which spanned from around 10,000 to 4,000 years ago, was a time of significant cultural and technological change in human history. During this period, humans transitioned from a nomadic, hunter-gatherer lifestyle to one of settled agriculture and the domestication of animals. This transition had a profound impact on human society and culture, leading to the development of more complex and organized societies, and the emergence of new technologies and art forms.

One of the key features of Neolithic culture was the development of agriculture. People began to domesticate plants and animals, which allowed for the production of food surpluses and the growth of cities. This led to the development of more complex social hierarchies and the emergence of specialized labor. The Neolithic period also saw the emergence of new technologies, such as the invention of the wheel and the development of pottery.

Another important aspect of Neolithic culture was the rise of religion and spirituality. People began to create elaborate ritual and ceremonial structures, such as megaliths and tombs, which were believed to have spiritual significance. These structures often featured complex carvings and symbols, which were thought to have magical powers. The Neolithic period also saw the emergence of new forms of art, such as cave paintings and petroglyphs.

The Neolithic period also saw the rise of more complex social structures, such as the emergence of chiefdoms and the development of more complex systems of governance. This led to the development of more complex economic systems, such as the emergence of trade and the development of specialized crafts.

In conclusion, the Neolithic period was a time of significant cultural and technological change



Preset simple-1
Filter by loader All

max_new_tokens 200
temperature 0,7
top_p 0,9
top_k 20
typical_p 1
epsilon_cutoff 0
eta_cutoff 0
tfs 1
top_a 0

repetition_penalty 1,15
repetition_penalty_range 0
encoder_repetition_penalty 1
no_repeat_ngram_size 0
min_length 0
Seed (-1 for random) -1
☑ do_sample

guidance_scale 1
For CFG. 1.5 is a good value.
Negative prompt
mirostat_mode 0
mode=1 is for llama.cpp only.
mirostat_tau 5
mirostat_eta 0,1

penalty_alpha 0
For Contrastive Search. do_sample must be unchecked.
num_beams 1
For Beam Search, along with length_penalty and early_stopping.
length_penalty 1
☐ early_stopping

Truncate the prompt up to this length 4096
The leftmost tokens are removed if the prompt exceeds this length. Most models require this to be at most 2048.
Custom stopping strings
In addition to the defaults. Written between "" and separated by commas.
"\n", "\nYou:"

Expand max_new_tokens to the available context length.
☐ auto_max_new_tokens
Forces the model to never end the generation prematurely.
☐ Ban the eos_token
Disabling this can make the replies more creative.
☑ Add the bos_token to the beginning of prompts
Some specific models need this unset.
☑ Skip special tokens
☑ Activate text streaming

Features

- 3 interface modes: default (two columns), notebook, and chat
- Multiple model backends: transformers, llama.cpp, ExLlama, AutoGPTQ, GPTQ-for-LLaMa, ctransformers
- Dropdown menu for quickly switching between different models
- LoRA: load and unload LoRAs on the fly, train a new LoRA using QLoRA

- Precise instruction templates for chat mode, including Llama-2-chat, Alpaca, Vicuna, WizardLM, StableLM, and many others
- 4-bit, 8-bit, and CPU inference through the transformers library
- Use llama.cpp models with transformers samplers (`llamacpp_HF` loader)
- Multimodal pipelines, including LLaVA and MiniGPT-4
- Extensions framework
- Custom chat characters
- Very efficient text streaming
- Markdown output with LaTeX rendering, to use for instance with GALACTICA
- API, including endpoints for websocket streaming (see the examples)

To learn how to use the various features, check out the Documentation: https://github.com/oobabooga/text-generation-webui/tree/main/docs

Installation

One-click installers

Windows	Linux	macOS	WSL
oobabooga-windows.zip	oobabooga-linux.zip	oobabooga-macos.zip	oobabooga-wsl.zip

Just download the zip above, extract it, and double-click on "start". The web UI and all its dependencies will be installed in the same folder.

- The source codes and more information can be found here: https://github.com/oobabooga/one-click-installers
- There is no need to run the installers as admin.
- Huge thanks to @jllllll, @ClayShoaf, and @xNul for their contributions to these installers.

Manual installation using Conda

Recommended if you have some experience with the command-line.

0. Install Conda

https://docs.conda.io/en/latest/miniconda.html

On Linux or WSL, it can be automatically installed with these two commands (source):

```
curl -sL "https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh" > "Miniconda3.sh"
bash Miniconda3.sh
```

1. Create a new conda environment

```
conda create -n textgen python=3.10.9
conda activate textgen
```

2. Install Pytorch

System	GPU	Command
Linux/WSL	NVIDIA	`pip3 install torch torchvision torchaudio`
Linux/WSL	CPU only	`pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cpu`
Linux	AMD	`pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/rocm5.4.2`
MacOS + MPS	Any	`pip3 install torch torchvision torchaudio`
Windows	NVIDIA	`pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu117`
Windows	CPU only	`pip3 install torch torchvision torchaudio`

The up-to-date commands can be found here: https://pytorch.org/get-started/locally/.

3. Install the web UI

```
git clone https://github.com/oobabooga/text-generation-webui
cd text-generation-webui
pip install -r requirements.txt
```

AMD, Metal, Intel Arc, and CPUs without AVCX2

1. Replace the last command above with

```
pip install -r requirements_nocuda.txt
```

2. Manually install llama-cpp-python using the appropriate command for your hardware: Installation from PyPI.

3. AMD: Manually install AutoGPTQ: Installation.

4. AMD: Manually install ExLlama by simply cloning it into the `repositories` folder (it will be automatically compiled at runtime after that):

```
cd text-generation-webui
mkdir repositories
cd repositories
git clone https://github.com/turboderp/exllama
```

bitsandbytes on older NVIDIA GPUs

bitsandbytes >= 0.39 may not work. In that case, to use `--load-in-8bit`, you may have to downgrade like this:

- Linux: `pip install bitsandbytes==0.38.1`
- Windows: `pip install https://github.com/jllllll/bitsandbytes-windows-webui/raw/main/bitsandbytes-0.38.1-py3-none-any.whl`

Alternative: Docker

```
ln -s docker/{Dockerfile,docker-compose.yml,.dockerignore} .
cp docker/.env.example .env
# Edit .env and set TORCH_CUDA_ARCH_LIST based on your GPU model
docker compose up --build
```

- You need to have docker compose v2.17 or higher installed. See this guide for instructions.
- For additional docker files, check out this repository.

Updating the requirements

From time to time, the `requirements.txt` changes. To update, use these commands:

```
conda activate textgen
cd text-generation-webui
pip install -r requirements.txt --upgrade
```

Downloading models

Models should be placed in the `text-generation-webui/models` folder. They are usually downloaded from Hugging Face.

- Transformers or GPTQ models are made of several files and must be placed in a subfolder. Example:

```
text-generation-webui
├── models
│   ├── lmsys_vicuna-33b-v1.3
│   │   ├── config.json
│   │   ├── generation_config.json
│   │   ├── pytorch_model-00001-of-00007.bin
│   │   ├── pytorch_model-00002-of-00007.bin
│   │   ├── pytorch_model-00003-of-00007.bin
│   │   ├── pytorch_model-00004-of-00007.bin
│   │   ├── pytorch_model-00005-of-00007.bin
│   │   ├── pytorch_model-00006-of-00007.bin
│   │   ├── pytorch_model-00007-of-00007.bin
│   │   ├── pytorch_model.bin.index.json
│   │   ├── special_tokens_map.json
│   │   ├── tokenizer_config.json
│   │   └── tokenizer.model
```

- GGML/GGUF models are a single file and should be placed directly into `models`. Example:

```
text-generation-webui
├── models
│   ├── llama-13b.ggmlv3.q4_K_M.bin
```

In both cases, you can use the "Model" tab of the UI to download the model from Hugging Face automatically. It is also possible to download via the command-line with `python download-model.py organization/model` (use `--help` to see all the options).

GPT-4chan

▶ Instructions

Starting the web UI

```
conda activate textgen
cd text-generation-webui
python server.py
```

Then browse to

Basic settings

Flag	Description
`-h`, `--help`	Show this help message and exit.
`--multi-user`	Multi-user mode. Chat histories are not saved or automatically loaded. WARNING: this is highly experimental.
`--character CHARACTER`	The name of the character to load in chat mode by default.
`--model MODEL`	Name of the model to load by default.
`--lora LORA [LORA ...]`	The list of LoRAs to load. If you want to load more than one LoRA, write the names separated by spaces.
`--model-dir MODEL_DIR`	Path to directory with all the models.
`--lora-dir LORA_DIR`	Path to directory with all the loras.
`--model-menu`	Show a model menu in the terminal when the web UI is first launched.
`--settings SETTINGS_FILE`	Load the default interface settings from this yaml file. See `settings-template.yaml` for an example. If you create a file called `settings.yaml`, this file will be loaded by default without the need to use the `--settings` flag.
`--extensions EXTENSIONS [EXTENSIONS ...]`	The list of extensions to load. If you want to load more than one extension, write the names separated by spaces.
`--verbose`	Print the prompts to the terminal.

Model loader

Flag	Description
`--loader LOADER`	Choose the model loader manually, otherwise, it will get autodetected. Valid options: transformers, autogptq, gptq-for-llama, exllama, exllama_hf, llamacpp, rwkv, ctransformers

Accelerate/transformers

Flag	Description
`--cpu`	Use the CPU to generate text. Warning: Training on CPU is extremely slow.
`--auto-devices`	Automatically split the model across the available GPU(s) and CPU.
`--gpu-memory GPU_MEMORY [GPU_MEMORY ...]`	Maximum GPU memory in GiB to be allocated per GPU. Example: `--gpu-memory 10` for a single GPU, `--gpu-memory 10 5` for two GPUs. You can also set values in MiB like `--gpu-memory 3500MiB`.
`--cpu-memory CPU_MEMORY`	Maximum CPU memory in GiB to allocate for offloaded weights. Same as above.
`--disk`	If the model is too large for your GPU(s) and CPU combined, send the remaining layers to the disk.
`--disk-cache-dir DISK_CACHE_DIR`	Directory to save the disk cache to. Defaults to `cache/`.
`--load-in-8bit`	Load the model with 8-bit precision (using bitsandbytes).
`--bf16`	Load the model with bfloat16 precision. Requires NVIDIA Ampere GPU.
`--no-cache`	Set `use_cache` to False while generating text. This reduces the VRAM usage a bit with a performance cost.
`--xformers`	Use xformer's memory efficient attention. This should increase your tokens/s.
`--sdp-attention`	Use torch 2.0's sdp attention.
`--trust-remote-code`	Set trust_remote_code=True while loading a model. Necessary for ChatGLM and Falcon.

Accelerate 4-bit

⚠ Requires minimum compute of 7.0 on Windows at the moment.

Flag	Description
`--load-in-4bit`	Load the model with 4-bit precision (using bitsandbytes).
`--compute_dtype COMPUTE_DTYPE`	compute dtype for 4-bit. Valid options: bfloat16, float16, float32.
`--quant_type QUANT_TYPE`	quant_type for 4-bit. Valid options: nf4, fp4.
`--use_double_quant`	use_double_quant for 4-bit.

GGML/GGUF (for llama.cpp and ctransformers)

Flag	Description
`--threads`	Number of threads to use.
`--n_batch`	Maximum number of prompt tokens to batch together when calling llama_eval.

Flag	Description
`--n-gpu-layers N_GPU_LAYERS`	Number of layers to offload to the GPU. Only works if llama-cpp-python was compiled with BLAS. Set this to 1000000000 to offload all layers to the GPU.
`--n_ctx N_CTX`	Size of the prompt context.

llama.cpp

Flag	Description
`--no-mmap`	Prevent mmap from being used.
`--mlock`	Force the system to keep the model in RAM.
`--mul_mat_q`	Activate new mulmat kernels.
`--cache-capacity CACHE_CAPACITY`	Maximum cache capacity. Examples: 2000MiB, 2GiB. When provided without units, bytes will be assumed.
`--tensor_split TENSOR_SPLIT`	Split the model across multiple GPUs, comma-separated list of proportions, e.g. 18,17
`--llama_cpp_seed SEED`	Seed for llama-cpp models. Default 0 (random).
`--n_gqa N_GQA`	GGML only (not used by GGUF): Grouped-Query Attention. Must be 8 for llama-2 70b.
`--rms_norm_eps RMS_NORM_EPS`	GGML only (not used by GGUF): 5e-6 is a good value for llama-2 models.
`--cpu`	Use the CPU version of llama-cpp-python instead of the GPU-accelerated version.
`--cfg-cache`	llamacpp_HF: Create an additional cache for CFG negative prompts.

ctransformers

Flag	Description
`--model_type MODEL_TYPE`	Model type of pre-quantized model. Currently gpt2, gptj, gptneox, falcon, llama, mpt, starcoder (gptbigcode), dollyv2, and replit are supported.

AutoGPTQ

Flag	Description
`--triton`	Use triton.
`--no_inject_fused_attention`	Disable the use of fused attention, which will use less VRAM at the cost of slower inference.
`--no_inject_fused_mlp`	Triton mode only: disable the use of fused MLP, which will use less VRAM at the cost of slower inference.
`--no_use_cuda_fp16`	This can make models faster on some systems.

Flag	Description
`--desc_act`	For models that don't have a quantize_config.json, this parameter is used to define whether to set desc_act or not in BaseQuantizeConfig.
`--disable_exllama`	Disable ExLlama kernel, which can improve inference speed on some systems.

ExLlama

Flag	Description
`--gpu-split`	Comma-separated list of VRAM (in GB) to use per GPU device for model layers, e.g. `20,7,7`
`--max_seq_len MAX_SEQ_LEN`	Maximum sequence length.
`--cfg-cache`	ExLlama_HF: Create an additional cache for CFG negative prompts. Necessary to use CFG with that loader, but not necessary for CFG with base ExLlama.

GPTQ-for-LLaMa

Flag	Description
`--wbits WBITS`	Load a pre-quantized model with specified precision in bits. 2, 3, 4 and 8 are supported.
`--model_type MODEL_TYPE`	Model type of pre-quantized model. Currently LLaMA, OPT, and GPT-J are supported.
`--groupsize GROUPSIZE`	Group size.
`--pre_layer PRE_LAYER [PRE_LAYER ...]`	The number of layers to allocate to the GPU. Setting this parameter enables CPU offloading for 4-bit models. For multi-gpu, write the numbers separated by spaces, eg `--pre_layer 30 60`.
`--checkpoint CHECKPOINT`	The path to the quantized checkpoint file. If not specified, it will be automatically detected.
`--monkey-patch`	Apply the monkey patch for using LoRAs with quantized models.

DeepSpeed

Flag	Description
`--deepspeed`	Enable the use of DeepSpeed ZeRO-3 for inference via the Transformers integration.
`--nvme-offload-dir NVME_OFFLOAD_DIR`	DeepSpeed: Directory to use for ZeRO-3 NVME offloading.
`--local_rank LOCAL_RANK`	DeepSpeed: Optional argument for distributed setups.

RWKV

Flag	Description
`--rwkv-strategy RWKV_STRATEGY`	RWKV: The strategy to use while loading the model. Examples: "cpu fp32", "cuda fp16", "cuda fp16i8".
`--rwkv-cuda-on`	RWKV: Compile the CUDA kernel for better performance.

RoPE (for llama.cpp, ExLlama, and transformers)

Flag	Description
`--alpha_value ALPHA_VALUE`	Positional embeddings alpha factor for NTK RoPE scaling. Use either this or compress_pos_emb, not both.
`--rope_freq_base ROPE_FREQ_BASE`	If greater than 0, will be used instead of alpha_value. Those two are related by rope_freq_base = 10000 * alpha_value ^ (64 / 63).
`--compress_pos_emb COMPRESS_POS_EMB`	Positional embeddings compression factor. Should be set to (context length) / (model's original context length). Equal to 1/rope_freq_scale.

Gradio

Flag	Description
`--listen`	Make the web UI reachable from your local network.
`--listen-host LISTEN_HOST`	The hostname that the server will use.
`--listen-port LISTEN_PORT`	The listening port that the server will use.
`--share`	Create a public URL. This is useful for running the web UI on Google Colab or similar.
`--auto-launch`	Open the web UI in the default browser upon launch.
`--gradio-auth USER:PWD`	set gradio authentication like "username:password"; or comma-delimit multiple like "u1:p1,u2:p2,u3:p3"
`--gradio-auth-path GRADIO_AUTH_PATH`	Set the gradio authentication file path. The file should contain one or more user:password pairs in this format: "u1:p1,u2:p2,u3:p3"
`--ssl-keyfile SSL_KEYFILE`	The path to the SSL certificate key file.
`--ssl-certfile SSL_CERTFILE`	The path to the SSL certificate cert file.

API

Flag	Description
`--api`	Enable the API extension.
`--public-api`	Create a public URL for the API using Cloudfare.
`--public-api-id PUBLIC_API_ID`	Tunnel ID for named Cloudflare Tunnel. Use together with public-api option.

Flag	Description
`--api-blocking-port BLOCKING_PORT`	The listening port for the blocking API.
`--api-streaming-port STREAMING_PORT`	The listening port for the streaming API.

Multimodal

Flag	Description
`--multimodal-pipeline PIPELINE`	The multimodal pipeline to use. Examples: `llava-7b`, `llava-13b`.

Presets

Inference settings presets can be created under `presets/` as yaml files. These files are detected automatically at startup.

The presets that are included by default are the result of a contest that received 7215 votes. More details can be found here.

Contributing

If you would like to contribute to the project, check out the Contributing guidelines.

Community

- Subreddit: https://www.reddit.com/r/oobabooga/
- Discord: https://discord.gg/jwZCF2dPQN

Acknowledgment

In August 2023, Andreessen Horowitz (a16z) provided a generous grant to encourage and support my independent work on this project. I am **extremely** grateful for their trust and recognition, which will allow me to dedicate more time towards realizing the full potential of text-generation-webui.