

Welcome!

This guide serves as a starting point for anyone interested in running a Large Language Model (LLM) using their own computer and [text-generation-webui](#), a free, open source GUI for running local text generation, and a viable alternative for cloud-based AI assistant services.

The guide will take you step by step through installing text-generation-webui, selecting your first model, loading and using it to chat with an AI assistant. You will learn how to configure the model to your liking, and what sort of hardware capabilities will you need.

Text-generation-webui is still in active development. It is, however, one of the most feature-rich and user friendly GUIs for text generation. While this guide covers only the basics, you can find many advanced features in the [official documentation](#) and on the [community forums](#).

Table of contents

Welcome!	1
Table of contents	2
Getting Started with text-generation-webui	3
System requirements	4
Download and first-time setup	5
Initial setup	5
UI overview and basic functionality	7
Open-weight Models	8
Browsing Hugging Face repositories	9
Model size vs hardware capabilities	10
Quantization	10
GGUF model format - downloading a model	12
Recommended first models	14
Model Configuration	15
Loading the model	16
Model parameters	20
Instruction templates	20
Generation parameters	20
Chatting	22
Basic chats	23
Characters	25

Getting Started with text-generation-webui

[Text-generation-webui](#) (known also as Oooba, after its creator, Ooobabooga) is a web UI for running LLMs locally. It's one of the major pieces of open source software used by AI hobbyists and professionals alike. This tutorial will teach you how to deploy a local text-generation-webui installation on your own computer, as well as walk you through the necessary steps needed to start making use of its capabilities.

System requirements

While it is possible to run a local LLM on pretty much any device, for the best experience, you will need the following:

- At least 16GB of RAM
- A modern processor, supporting AVX2 instructions. If your processor was manufactured after 2011, it almost certainly supports this standard
- Optional, but highly recommended: a dedicated NVIDIA or AMD graphics card, with at least 8GB of VRAM

Generally speaking, the more powerful your computer, the stronger models you can run locally on your machine. There are three main methods of running local LLMs, and each depends on the amount of system memory you have:

GPU inference: The fastest and most preferred method is to load the model entirely into your VRAM, or your graphics card's memory. You will need a dedicated graphics card.

The amount of VRAM you have available will determine the size of the model you can load. You should assume 8GB of VRAM to be the minimum, and 24GB VRAM to be optimal.

GPU + CPU inference: This method will use both your VRAM and RAM to load the model. Offloading parts of the model to your system memory will allow you to load larger models at a significant performance penalty. For this, you will sum up both your VRAM and RAM. You can assume that a minimum of 16GB memory in total will be necessary.

CPU only inference: This method uses your CPU and RAM for inference, and is the slowest by far, but also the cheapest method in terms of necessary hardware. Assume at least 16GB of RAM at a minimum.

Download and first-time setup

Downloading text-generation-webui is best accomplished by cloning the repository. You will need to download and install [GitHub Desktop](#) on your system. The process is straightforward and should only take a minute. If you are already familiar with Git, you may use it instead to clone the repository manually.

Once you have [GitHub Desktop](#) installed on your system, go ahead and launch it.

Navigate to **File > Clone repository**. In the **repository URL** field, enter the following URL:

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

Next, in the field below, select the path you wish to use for your local installation of text-generation-webui.

Finally, press **Clone**. You may close Github Desktop as soon as the operation is finished.

Initial setup

To install text-generation-webui, you can simply launch the provided installation script. Use your file explorer to navigate to the text-generation-webui installation folder you selected in the previous step.

Now, find and double-click on a file named **start_windows.bat** (or, if you're using Linux or MacOS, **start_linux.sh** or **start_macos.sh**, respectively). You will use the same file in the future to launch the program.

A command prompt window will open, and some required dependencies will be downloaded for you. After this is done, you will be asked to select your GPU vendor. Choose one of these options, then confirm your choice with ENTER:

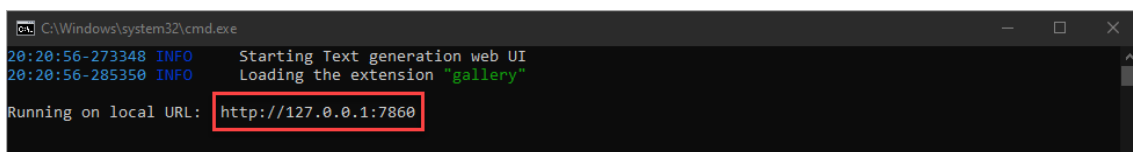
```
'A': 'NVIDIA',  
'B': 'AMD (Linux/MacOS only. Requires ROCm SDK 5.6 on Linux)',  
'C': 'Apple M Series',  
'D': 'Intel Arc (IPEX)',  
'N': 'None (I want to run models in CPU mode)'
```

You may choose to use your CPU only. Remember that GPU inference is much faster and generally preferred.

Keep following the on-screen prompts. Be patient, as the script may take some time to complete. Don't worry if the installation seems to be stuck.

The installation process is complete once you see "Running on local URL: ..." in your terminal window.

At this point, do not close the terminal window. Instead, take note of the local URL displayed:

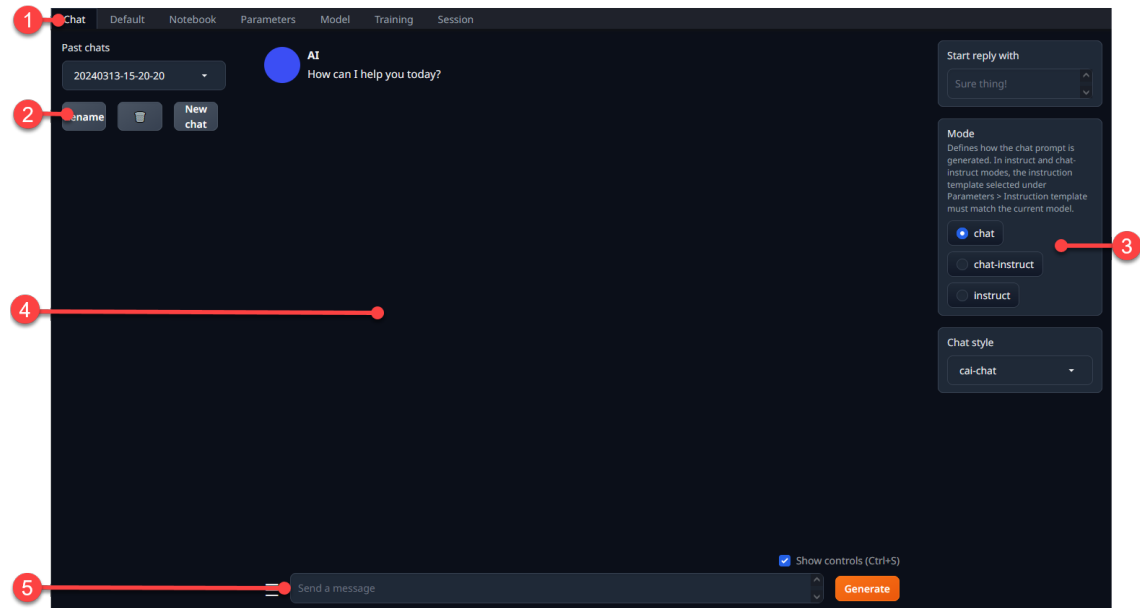


```
C:\Windows\system32\cmd.exe
20:20:56-273348 INFO      Starting Text generation web UI
20:20:56-285350 INFO      Loading the extension "gallery"
Running on local URL: http://127.0.0.1:7860
```

Using your web browser of choice, navigate to the given URL. You should see the main interface. Congratulations, you have successfully completed the installation process!

UI overview and basic functionality

When you first run text-generation-webui and open the URL provided in the terminal window, you will see the application's main interface. Depending on your screen size, some of the elements might become visible once you scroll further down.



1. **Main toolbar:** contains various tabs used to navigate between screens - notably **Chat**, [Parameters](#) and [Model](#)
2. **Past chats:** allows you to create a new chat, as well as browse through past chats
3. **Mode selector:** defines the formatting of the chat prompt, depending on the used model
4. **Chat interface:** this is where your chat with the AI assistant will appear
5. **Input area:** contains the input field, with additional options accessible via the hamburger menu

Open-weight Models

Using open source LLMs allows for a large degree of freedom when it comes to picking an open-weight model that will best suit your needs. Companies like Mistral AI, Meta or Google have released open-weight models, and the open source community has fine-tuned them to suit a wide variety of needs.

Note the difference between *open source* and *open weight*. The major models released by the big companies are Neural Net Weights (NNWs) - not human-readable source code, but the 'knowledge' that the artificial neural network has learned.

Many open-weight LLMs and their finetunes are uncensored, i.e. they don't have the safety guardrails that a service like ChatGPT might have, where it'll refuse to generate offensive, not safe for work or otherwise controversial output. Keep this in mind when selecting a model for yourself.

Browsing Hugging Face repositories

Hugging Face is a French-American AI company, which has developed many of the foundational technologies behind open-source machine learning - most notably the *transformers library*.

[Hugging Face Hub](#) has become the go-to platform for sharing of models, finetunes, data sets and demos related to all kinds of machine learning tasks - be it image and video generation, natural language processing or text-to-speech.

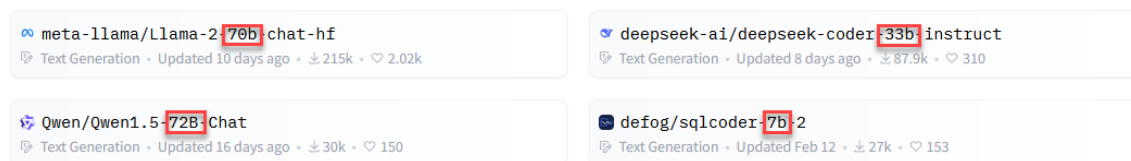
The platform operates much like GitHub, allowing anyone to create an account and collaborate on different projects.

Model size vs hardware capabilities

To get started with selecting your first model, open huggingface.co in your browser, and navigate to **Models > Natural Language Processing > Text Generation**.

You will see a long list of currently available models. You may click on any given model to see a **model card**, or a brief overview of the model's features, terms of use (not all models are free to use commercially) and recommended [template or settings](#).

While browsing through, notice that most model names contain a number followed by the letter B. This denotes the number of billions of parameters contained by the given model. For instance, **Llama-2-70b-chat** was trained on 70 billion parameters.



You will also notice some models with 8x7B in their names - these are Mixture of Experts models - in essence, 8 smaller models working in conjunction.

To put things simply - the larger this number, the 'smarter' the model will be. However, larger models, while smart, have a downside - using them requires a powerful computer. **The larger the 'B' number, the larger the memory requirement.**

Quantization

Further complicating this calculation is the matter of **quantization**, or model '*quants*'. For simplicity's sake, just understand that the model can be compressed to more easily fit into your memory - however, it will be slightly less 'smart' than an uncompressed model.

Some model authors provide a handy table showing exactly how much memory you will need to comfortably run a given model. In this example, a Q8 quant of

codellama-70b can fit in 76GB of RAM, but a Q5 quant will fit into 50GB of memory.

Name	Quant method	Bits	Size	Max RAM required	Use case
<u>codellama-70b-python.Q5_0.gguf</u>	Q5_0	5	47.46 GB	49.96 GB	legacy; medium, balanced quality - prefer using Q4_K_M
<u>codellama-70b-python.Q5_K_S.gguf</u>	Q5_K_S	5	47.46 GB	49.96 GB	large, low quality loss - recommended
<u>codellama-70b-python.Q5_K_M.gguf</u>	Q5_K_M	5	48.75 GB	51.25 GB	large, very low quality loss - recommended
<u>codellama-70b-python.Q6_K.gguf</u>	Q6_K	6	56.59 GB	59.09 GB	very large, extremely low quality loss
<u>codellama-70b-python.Q8_0.gguf</u>	Q8_0	8	73.29 GB	75.79 GB	very large, extremely low quality loss - not recommended

You can very roughly assume that one billion of parameters will fit, when quantized, into slightly more than 1GB of RAM or VRAM. Remember that you will need to leave some memory available for your operating system and the LLM's context.

Using this information you should be able to roughly estimate how large of a model you can comfortably run on your machine. One sensible approach would be to begin with a small model, for instance a 7B or 13B one, and if it runs well, move up to larger ones. For now, browse through and see if anything catches your eye.

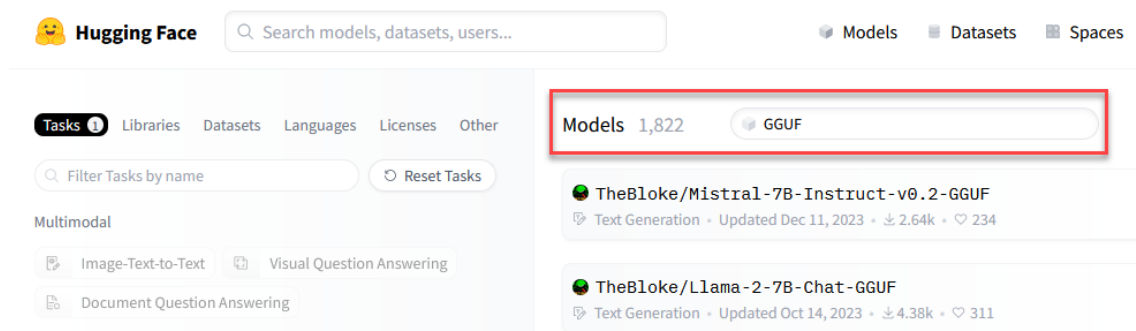
GGUF model format - downloading a model

Machine learning is a dynamic field, and there are many standards and quantization formats, both current and already obsolete. For the sake of this guide, we will consider the most versatile of them - GGUF.

GGUF is a popular file format used by one of the most popular backend libraries, [llama.cpp](https://github.com/josiah Davis/llama.cpp). Models quantized using this format are capable of being loaded in RAM. Part of the model can also be offloaded to VRAM for extra overall speed. Thanks to this, GGUF remains one of the most popular model formats in use today. Models quantized in this format are usually denoted with a -GGUF suffix on huggingface.co:



You may also use the filter function limit the displayed models to only this format:



Browse through the available models and try to find one that suits your needs, taking into account the amount of memory you will need to load the model.

TheBloke and *bartowski* are two HuggingFace users who consistently upload GGUF quants of popular models.

Once you've picked a model and the associated quant size, from the **model card** navigate to **Files and versions**, then simply click on the quant you wish to

download. Once the download is finished, move the .gguf file to your text-generation-webui directory, dropping it in the **models** folder.

Recommended first models

This section contains pre-selected, simple models for those who wish to try text-generation-webui with minimal research.

The models below will work with any combination of VRAM and RAM, or GPU memory and system memory. The models are also all meant for general use - much like ChatGPT or other similar services.

For computers with at least 8GB of RAM, RAM+VRAM or VRAM:

You can run a q5_K_M quant of [Crataco/Nous-Hermes-2-Mistral-7B-DPO-imatrix-GGUF](#). The chat format for this model is ChatML.

For computers with 16GB or more of RAM, RAM+VRAM or VRAM:

You can comfortably run a q5_K_M quant of [TheBloke/Nous-Hermes-2-SOLAR-10.7B-GGUF](#). The chat format for this model is ChatML.

For computers with at least 32 GB of RAM, RAM+VRAM or VRAM:

You can run a q4_K_S or K_M quant of [mradermacher/Nous-Hermes-2-Mixtral-8x7B-DPO-i1-GGUF](#). The chat format for this model is ChatML.

Once you've picked a model and the associated quant size, from the **model card** navigate to **Files and versions**, then simply click on the download arrow next to the quant you wish to download. Once the download is finished, move the .gguf file to your text-generation-webui directory, dropping it in the **models** folder.

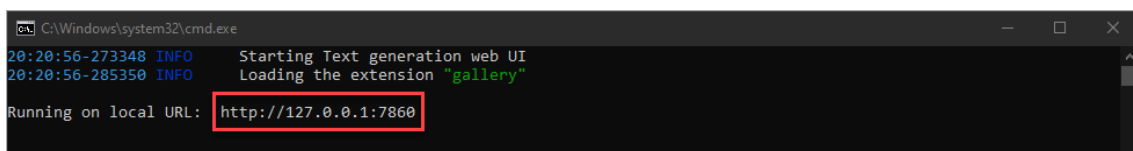
Model Configuration

To use a LLM with text-generation-webui, you first need to load it into your memory, and then configure it. Both of these processes are fairly simple and should only take you a minute. To start, you will need to learn about model parameters and context size.

Loading the model

Before following the steps in this section, remember to download a model and put it in the **models** folder in your text-generation-webui installation directory.

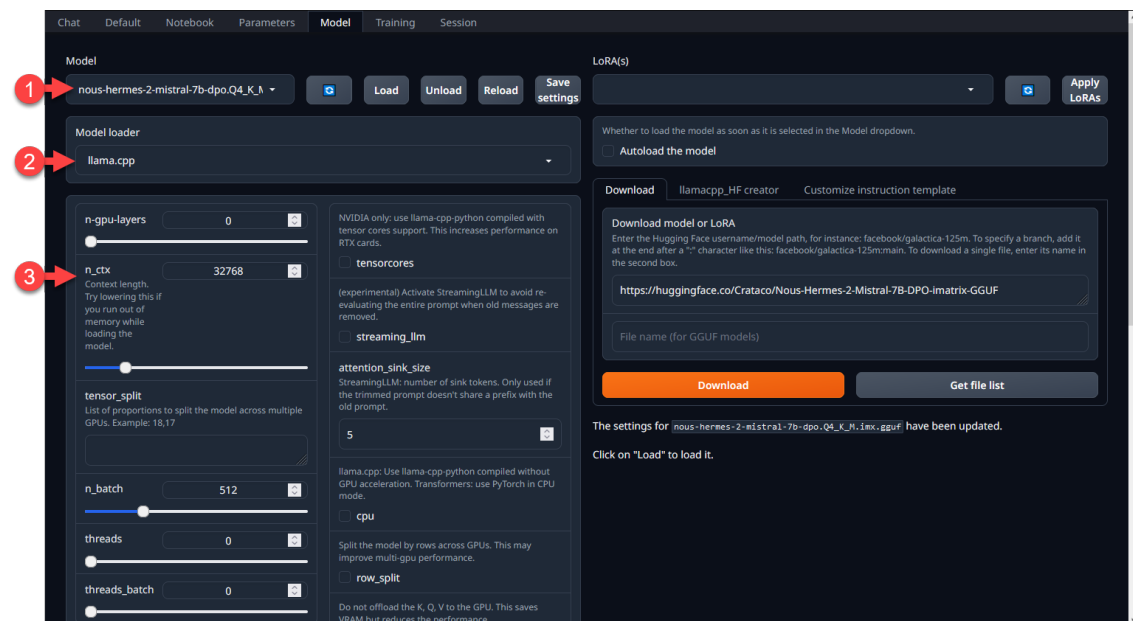
Loading a model in text-generation-webui is simple. Launch the webui by double-clicking the **start_windows.bat** script in your main installation directory. If you're using a different operating system, run the corresponding file. Then, using your web browser, navigate to the URL shown in the text-generation-webui console window.



```

C:\Windows\system32\cmd.exe
20:20:56-273348 INFO Starting Text generation web UI
20:20:56-285350 INFO Loading the extension "gallery"
Running on local URL: http://127.0.0.1:7860
  
```

From the main application menu, navigate to the **Model** tab in the menu bar at the top of your screen. This is the main interface for selecting and configuring the loading parameters of your model:



1. **Model selection dropdown** - use this to select the model you wish to load. Use the buttons next to it to load and unload the model, and save your

settings for later. The refresh button refreshes the list of models available - useful if you download a new model while text-generation-webui is already running.

2. **Model loader selection dropdown** - while text-generation-webui chooses the best model loader for your model automatically, advanced users may wish to pick a different loader.
3. **Loader settings** - loader-dependent settings for your selected model.

The first step is to select your downloaded model from the dropdown at the top of the screen. **Don't load it just yet.** First, take a look at the settings below.

If you're following the guide and are using the GGUF format of models, text-generation-webui will choose the llama.cpp loader for you automatically. Before loading the model, you will need to configure the loader settings section.

First, decide on the **context length** for your model. This determines how many *tokens* the AI will be able to remember from your chat. For reference, assume that a token equals about 4 characters. If you're using the GGUF format, this will be set automatically for you. Otherwise, you should be able to find the maximum context length of your model in the model's card on huggingface.co. You may use a smaller context size if you encounter issues with running out of memory while loading the model.

Second, move on to **n-gpu-layers**. This option determines how many 'layers' of the model will be delegated to VRAM, or your graphics card. If you're loading a model purely in RAM, disregard this option. Otherwise, set it to 50 initially. You will come back to this option after loading the model for the first time.

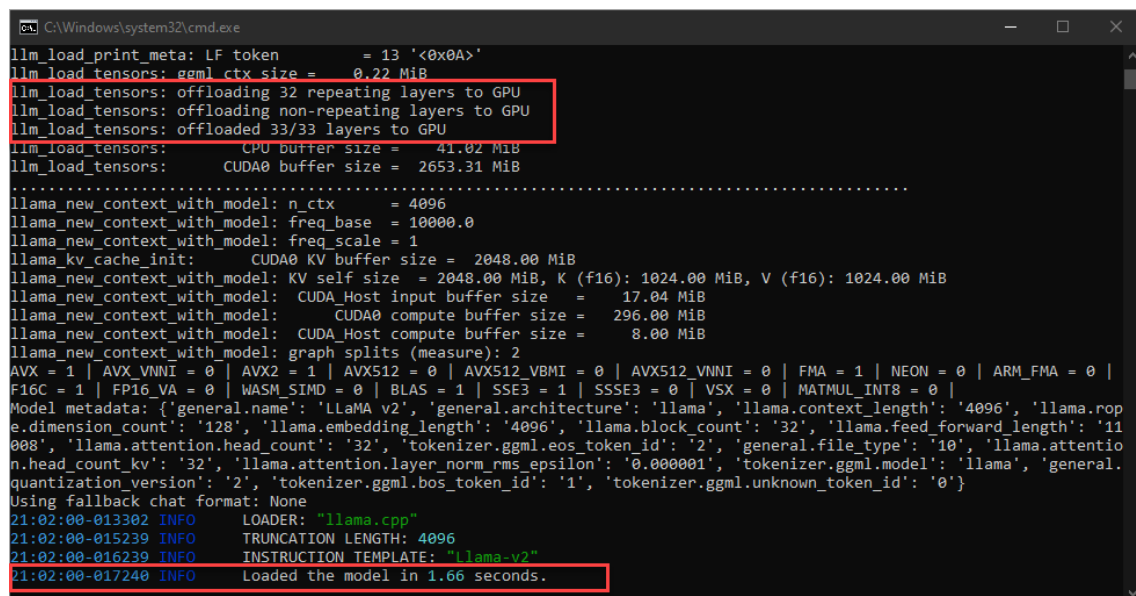
Finally, find the **threads** option. Set this to the number of threads your CPU has.

The other options may be helpful as well, but are dedicated to advanced users. If you're using an Nvidia RTX card, you may safely check the **tensorcores** checkbox for some extra performance. Otherwise, you are fine to leave the other options as they are for now.

At this point when first loading a model, it is entirely possible that your computer might crash. If it happens, don't panic. You won't cause any damage to your system. Simply wait for it to restart.

A crash at this point would simply mean that you've accidentally overfilled the available memory in your system. Select a smaller model and try again.

Whenever ready, press the **Load** button. Open the console window and look through the output. If everything went well, you should see several important messages:



```
C:\Windows\system32\cmd.exe
llm_load_print_meta: LF token = 13 '<0x0A>'
llm_load_tensors: ggml ctx size = 0.22 MiB
llm_load_tensors: offloading 32 repeating layers to GPU
llm_load_tensors: offloading non-repeating layers to GPU
llm_load_tensors: offloaded 33/33 layers to GPU
llm_load_tensors: CPU buffer size = 41.02 MiB
llm_load_tensors: CUDA0 buffer size = 2653.31 MiB
.....
llama_new_context_with_model: n_ctx = 4096
llama_new_context_with_model: freq_base = 10000.0
llama_new_context_with_model: freq_scale = 1
llama_kv_cache_init: CUDA0 KV buffer size = 2048.00 MiB
llama_new_context_with_model: KV self size = 2048.00 MiB, K (f16): 1024.00 MiB, V (f16): 1024.00 MiB
llama_new_context_with_model: CUDA_Host input buffer size = 17.04 MiB
llama_new_context_with_model: CUDA0 compute buffer size = 296.00 MiB
llama_new_context_with_model: CUDA_Host compute buffer size = 8.00 MiB
llama_new_context_with_model: graph splits (measure): 2
AVX = 1 | AVX_VNNI = 0 | AVX2 = 1 | AVXS12 = 0 | AVXS12_VBMI = 0 | AVXS12_VNNI = 0 | FMA = 1 | NEON = 0 | ARM_FMA = 0 |
F16C = 1 | FP16_VA = 0 | WASM_SIMD = 0 | BLAS = 1 | SSE3 = 1 | SSSE3 = 0 | VSX = 0 | MATMUL_INT8 = 0 |
Model metadata: {'general.name': 'LLaMA v2', 'general.architecture': 'llama', 'llama.context_length': '4096', 'llama.rop
e.dimension_count': '128', 'llama.embedding_length': '4096', 'llama.block_count': '32', 'llama.feed_forward_length': '11
008', 'llama.attention.head_count': '32', 'tokenizer.ggml.eos_token_id': '2', 'general.file_type': '10', 'llama.attentio
n.head_count_kv': '32', 'llama.attention.layer_norm_rms_epsilon': '0.000001', 'tokenizer.ggml.model': 'llama', 'general
.quantization_version': '2', 'tokenizer.ggml.bos_token_id': '1', 'tokenizer.ggml.unknown_token_id': '0'}
Using fallback chat format: None
21:02:00-013302 INFO LOADER: "llama.cpp"
21:02:00-015239 INFO TRUNCATION LENGTH: 4096
21:02:00-016239 INFO INSTRUCTION TEMPLATE: "llama-v2"
21:02:00-017240 INFO Loaded the model in 1.66 seconds.
```

At the very bottom of the terminal output you will see that the model was successfully loaded. If you're loading the model in RAM only, then you're good to go. Feel free to skip the rest of this topic and move on to the next step.

If using VRAM or both VRAM+RAM to load a model, you will need to do one extra step.

In the terminal output, find the line that says "**offloaded xx/yy layers to GPU**".

This is relevant to the **n-gpu-layers** option from before. Take note of this number.

Next, open Task Manager and navigate to the **Performance** tab. Open the GPU tab. Refer to the dropdown below if you need a refresher. Note your dedicated GPU memory usage. If it hasn't reached 100% - the model fits neatly in your VRAM, and you don't need to change any options. If it has reached 100%,

you will need to change the **n-gpu-layers** option to approx. 2 less than the **layers** number you saw in the terminal output. Reload the model and try again until your GPU isn't overfilled anymore. Alternatively, try using a smaller model.

Model parameters

Unlike online services such as ChatGPT, Gemma or Claude, local LLMs offer you the possibility of manually adjusting their parameters. You can make the AI's responses more or less random and predictable, change the output length and more.

Instruction templates

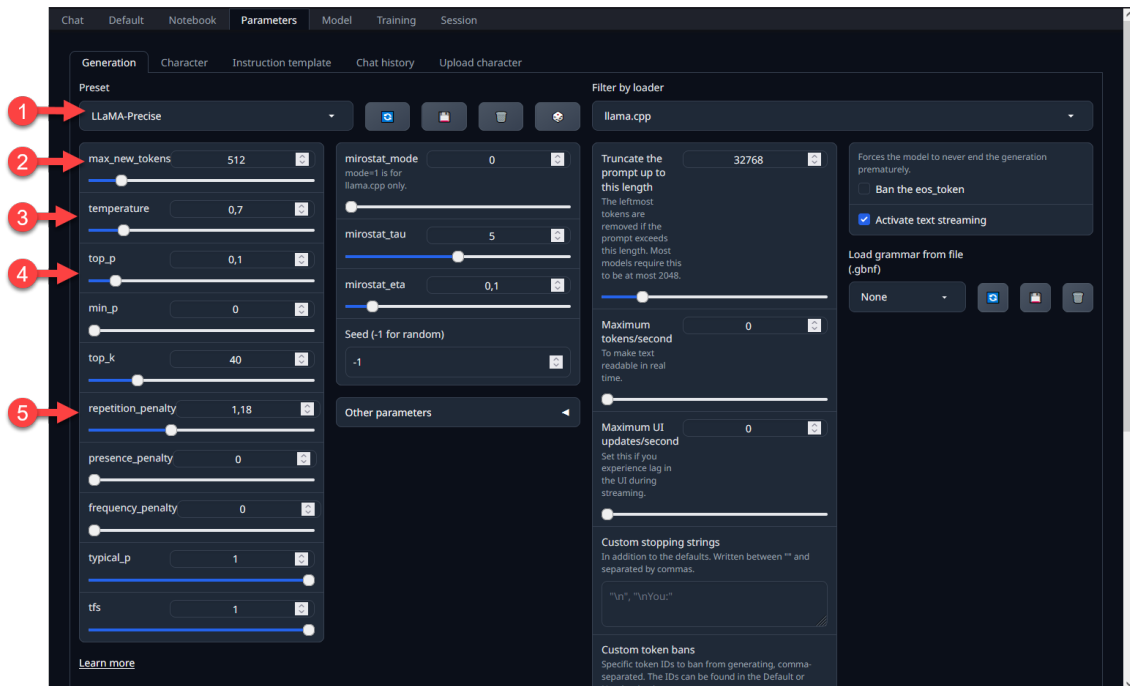
Instruction templates are a template for the format in which your prompts should be sent to the AI. After loading a model in text-generation-webui, navigate to the **Parameters > Instruction template** tab

Usually, the model's author includes the required template in the model's metadata, allowing text-generation-webui to automatically set the correct template. If this doesn't happen, you will need to locate the recommended template's name in the model's card on huggingface.co.

Setting the template manually is simple, as text-generation-webui comes with all the most often used templates pre-installed. Select the desired template from the drop-down menu, and confirm with **"Load"**. You're done!

Generation parameters

Model generation parameters allow you more freedom to experiment with your local model. After loading a model, navigate to **Parameters > Generation**. You will see the following interface:



1. **Premade presets** - For Chat-GPT-like generation, LLaMA-Precise is recommended. Simply select the preset from the drop-down menu.
2. **max_new_tokens** - controls the length of the model's output.
3. **temperature** - influences the model's 'creativity'. The higher this number, the more randomness will be introduced to the token selection. Keep this number between 0.7 and 1.2. For general AI assistant chats, the lower end of this spectrum is recommended.
4. **top_p** - cuts off the least probable next tokens during the generation. A top_p of 0.1 means that the tokens comprising the top 10% of probability are considered.
5. **Repetition penalty** - change this number if your model keeps repeating itself or using the same words too often. Between 1.1 and 1.2 is best.

Feel free to experiment with these options and see how the model's responses will change!

Chatting

If you've been following the guide so far, your local model should be ready to interact with. As a reminder, here are the steps you should have taken so far:

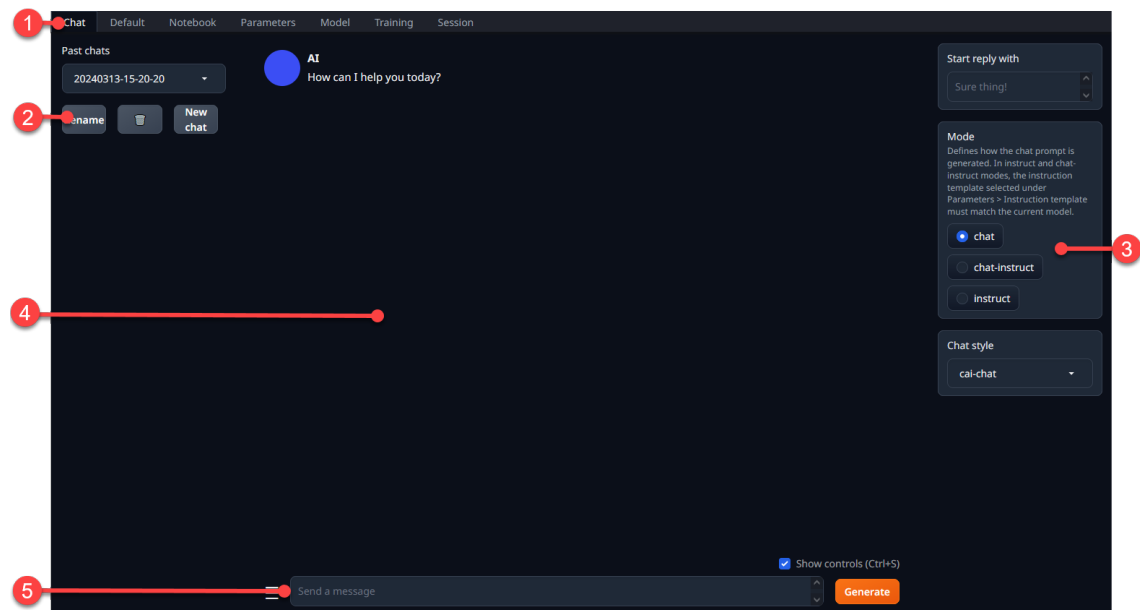
- You should have a [local installation of text-generation-webui](#)
- You [chose a model](#) and [downloaded it](#)
- You've successfully [loaded the model](#) and [configured the basic parameters](#)

If you missed any of these steps, feel free to go back to revisit the corresponding pages.

Basic chats

To chat with your AI assistant using text-generation-webui, navigate to the **Chat** tab.

You will see the main interface:



1. **Main toolbar:** contains various tabs - most importantly **Chat**, [Parameters](#) and [Model](#)
2. **Past chats:** allows you to create a new chat, as well as browse through past chats
3. **Mode selector:** defines the formatting of the chat prompt, depending on the used model
4. **Chat interface:** this is where your chat with the AI assistant will appear
5. **Input area:** contains the input field, as well as some extra options accessible via the hamburger menu

Using the **mode selector**, choose **instruct**. You will learn more about **chat-instruct** in the next section. You are now ready to start chatting with your completely offline AI assistant.

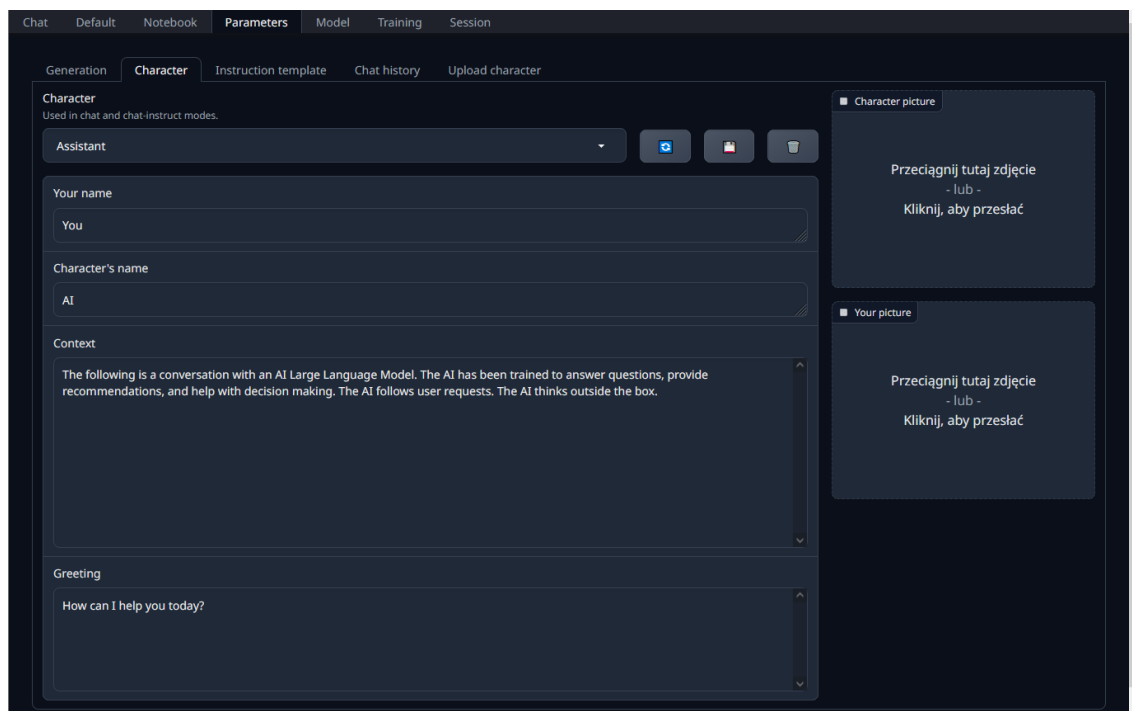
Write your prompt in the **input area**, then press '**Generate**'. You will see the response being generated in real time.

If the response generation is taking too long for your liking, you may want to download a smaller model. If using VRAM, check if you've [correctly offloaded layers onto your GPU](#).

Characters

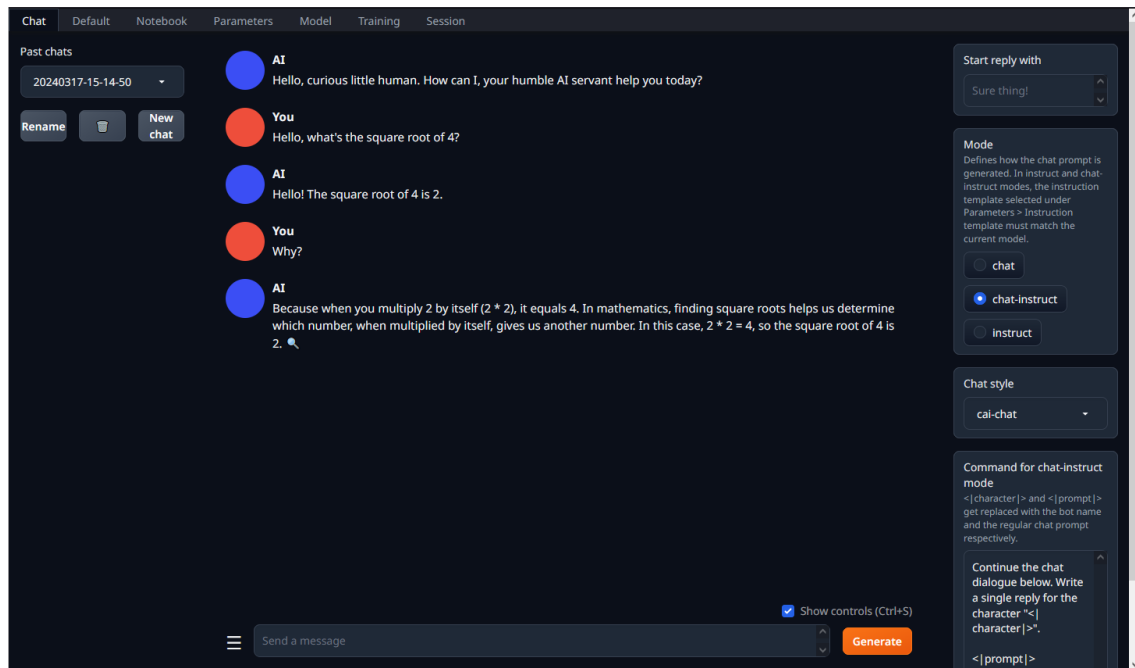
While chatting with your AI assistant is fun on its own right, you may also wish to breathe some character into your chats. Text-generation-webui offers a **characters** feature, allowing the AI assistant to play any role you can imagine.

Load a model, then navigate to **Parameters > Character**.



You will see the default character settings offered by text-generation-webui. You may change any of these settings at will. Try replacing the **Your Name** field with your own name, and edit the **Context** field to your own liking. Afterward, press the **'save'** button and name your new persona. You may also try to chat with the example persona bundled with text-generation-webui. Simply select 'example' from the drop-down menu.

To chat with a character, you must navigate to the **Chat** menu and select **chat-instruct** from the **chat mode menu** on the right. You will see the introductory statement from your chosen character. You can now interact with them:



If you wish to start a new chat, select **"New chat"** from the menu on the left. You can also use the same menu to return to past chats, delete a chat, or give a chat a specific name.