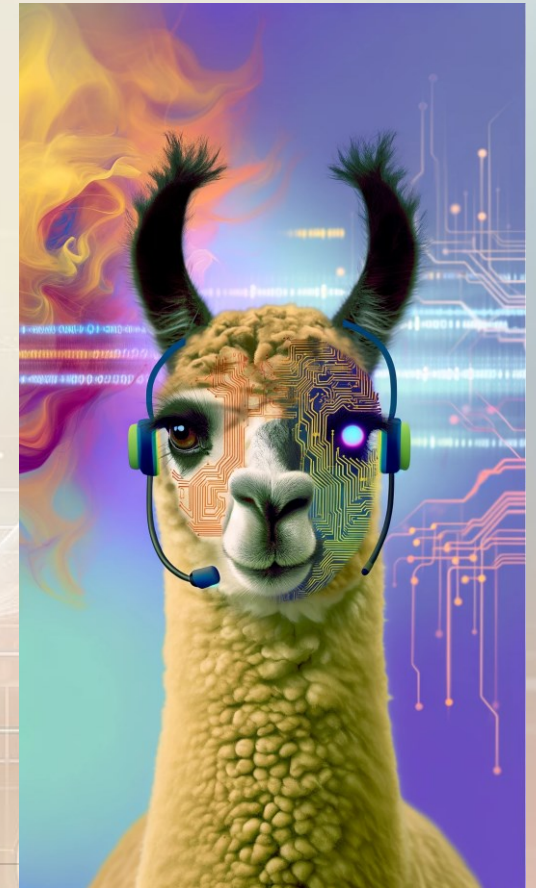


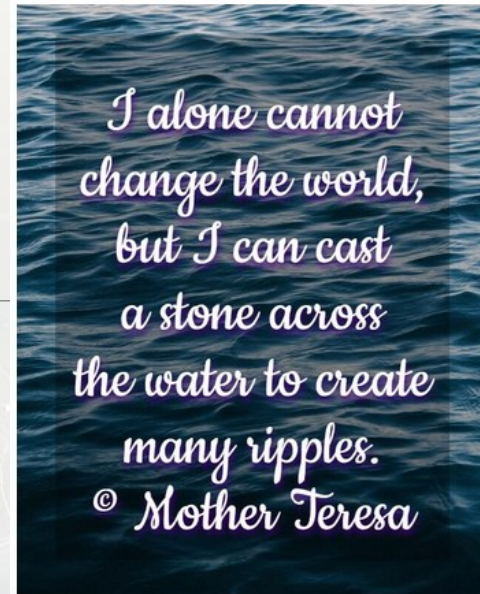
Run LLMs everywhere

DR. KARSTEN GAIER / NI SP GMBH / INFO@AI-SP.COM



LLMS for Inference

1. Coding/Programming
 - Completion
 - Translation (e.g. from one language to another)
 - Annotation
2. Information extraction / Summarizing content
 - Retrieval Augmented Generation (RAG)
 - Large Context
3. Questioning models with own data
 - LORA: e.g. <https://github.com/oobabooga/text-generation-webui>
4. Translation of text from one language into another one
5. Image Analysis



User: what is written on the image?

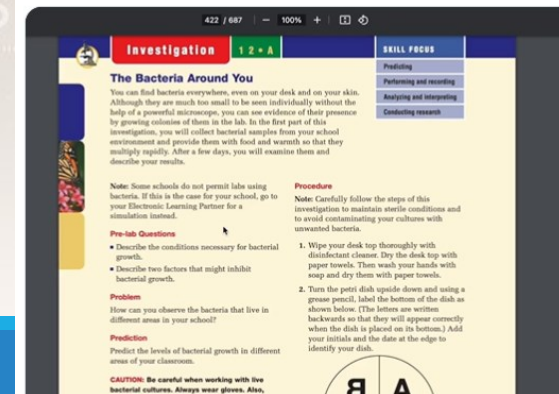
Llama: The image features a quote from Mother Teresa, which says: "I alone cannot change the world, but I can cast a stone across the water to create many ripples." This inspiring message is displayed on top of a beautiful blue ocean scene.

I fed an "entire" biology textbook into Gemini 1.5 Pro.

491,002 tokens.

I asked it 3 extremely specific questions, and it got each answer 100% correct.

1M token context windows are a gamechanger.



Why Local LLMs?

1. Privacy and Data Security

- When running LLMs locally, sensitive data does not need to be sent over the internet to a cloud service. This minimizes the risk of data breaches and ensures that confidential or proprietary information remains secure within the user's own infrastructure.

2. API Performance, Customization and Control

- Local deployment allows users to customize the LLM to their specific needs, including adjusting parameters or integrating with other local systems without the limitations or constraints that might be present in cloud-based platforms. Missing Service Levels.

3. Cost Savings

- While there is an initial investment in hardware and setup, running LLMs locally can be more cost-effective in the long term, especially for heavy users. There are no ongoing fees for API calls or data processing, which can significantly reduce operational costs.

4. Performance Optimization

- By running LLMs on local hardware, users can optimize performance based on their specific requirements. This includes leveraging high-performance computing resources to reduce latency and increase throughput, which is particularly important for real-time applications.

5. Compliance and Regulatory Requirements

- Certain industries are subject to strict regulatory requirements regarding data handling and processing. Running LLMs locally can make it easier to comply with these regulations by keeping data processing within a controlled environment, thus avoiding potential legal and regulatory complications associated with cloud services.

How Well do Local LLMs Perform?



CHATBOT Arena

Rank ▲	🤖 Model ▲	★ Arena Elo ▲	🇺🇸 95% CI ▲	🗳 Votes ▲	Organization ▲	License ▲	Knowledge Cutoff ▲
1	GPT-4-1106-preview	1254	+5/-5	38745	OpenAI	Proprietary	2023/4
2	GPT-4-0125-preview	1253	+10/-8	6308	OpenAI	Proprietary	2023/4
3	Bard (Gemini Pro)	1218	+8/-7	10313	Google	Proprietary	Online
4	GPT-4-0314	1191	+6/-6	20430	OpenAI	Proprietary	2021/9
5	GPT-4-0613	1164	+5/-6	32941	OpenAI	Proprietary	2021/9
6	Mistral Medium	1152	+5/-7	17847	Mistral	Proprietary	Unknown
7	Claude-1	1150	+7/-5	19017	Anthropic	Proprietary	Unknown
8	Qwen1.5-72B-Chat	1147	+8/-8	5204	Alibaba	Qianwen LICENSE	2024/2
9	Claude-2.0	1132	+6/-8	12753	Anthropic	Proprietary	Unknown
10	Gemini Pro (Dev API)	1122	+7/-7	9024	Google	Proprietary	2023/4
11	Claude-2.1	1120	+6/-4	27723	Anthropic	Proprietary	Unknown
12	Mixtral-8x7b-Instruct-v0.1	1120	+5/-6	18410	Mistral	Apache 2.0	2023/12
13	GPT-3.5-Turbo-0613	1118	+5/-5	36704	OpenAI	Proprietary	2021/9
14	Gemini Pro	1115	+9/-9	6958	Google	Proprietary	2023/4
15	Yi-34B-Chat	1111	+7/-8	7734	01 AI	Yi License	2023/6

Feb 24, <https://huggingface.co/spaces/lmsys/chatbot-arena-leaderboard>

CHATBOT Arena

Figure 3: Bootstrap of Elo Estimates (1000 Rounds of Random Sampling)

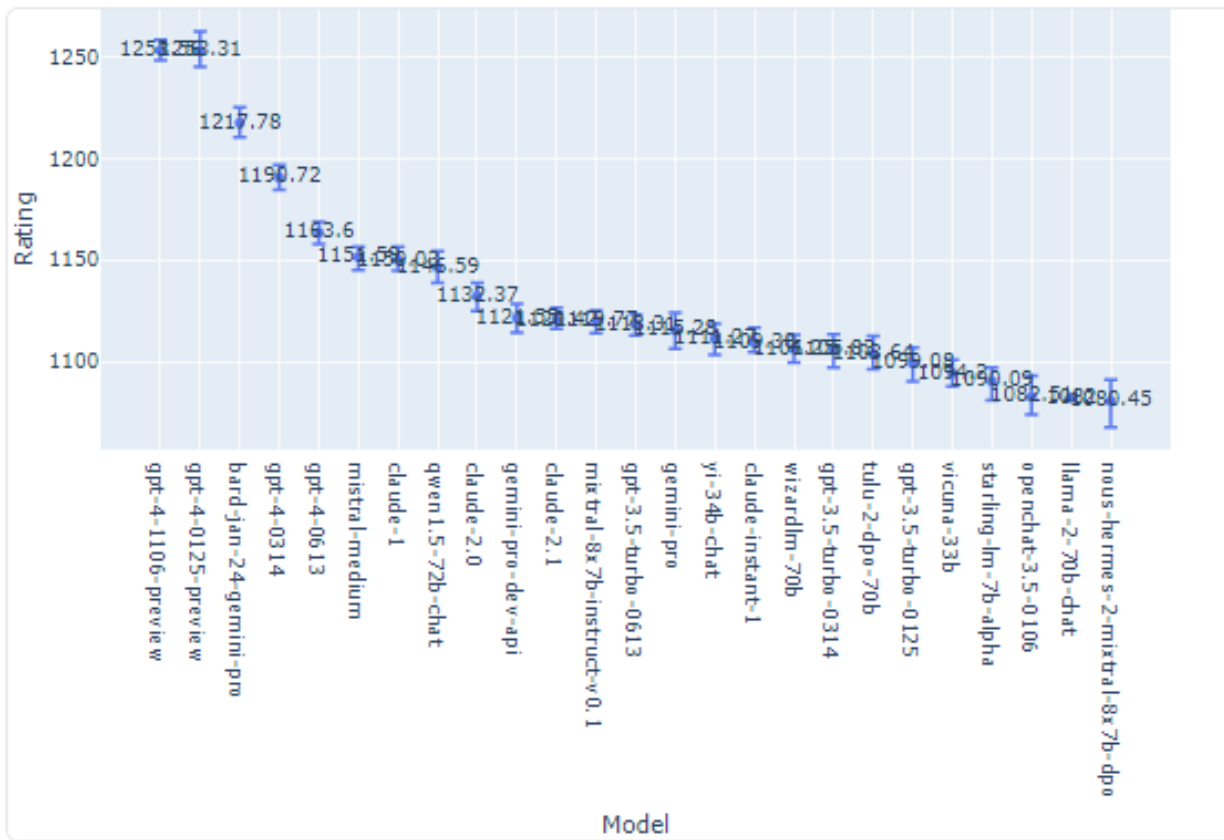
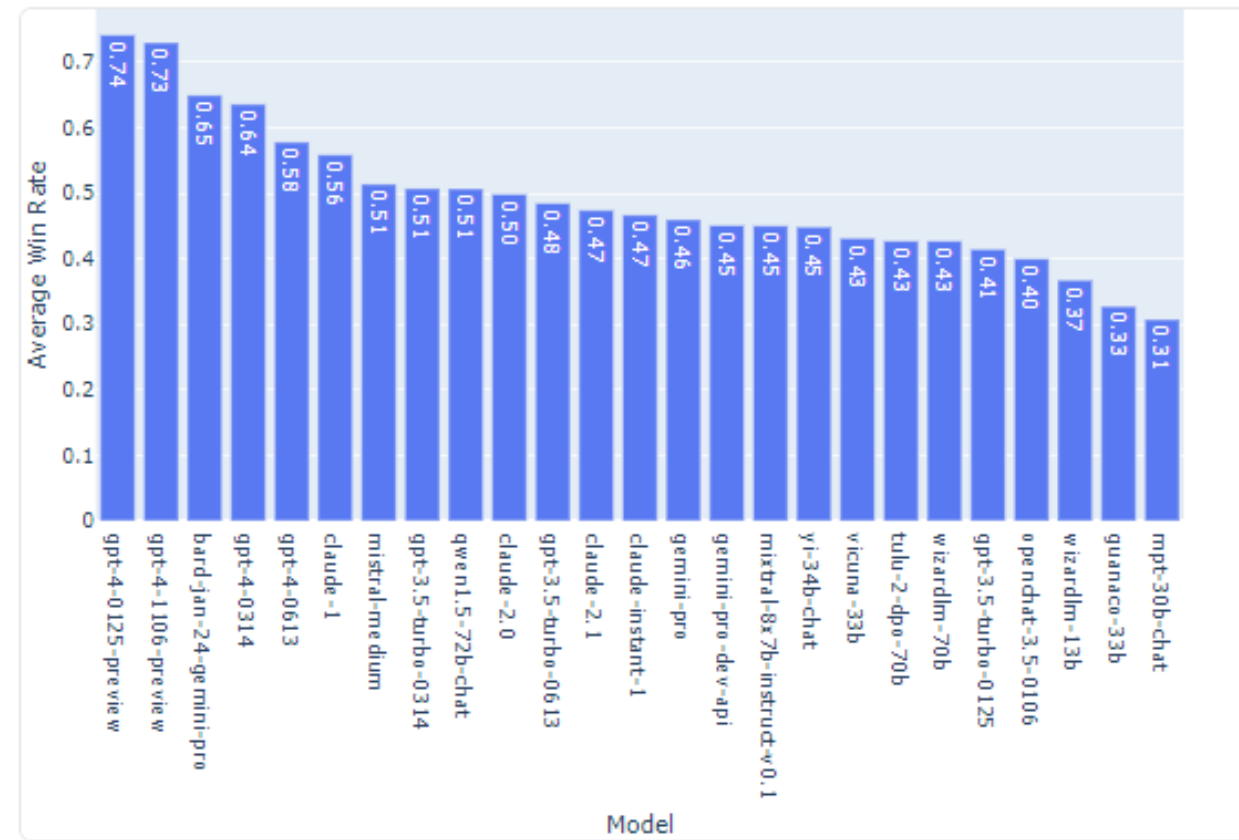


Figure 4: Average Win Rate Against All Other Models (Assuming Uniform Sampling and No Ties)



Coding LLMs

⚡ With EvalPlus ⚡

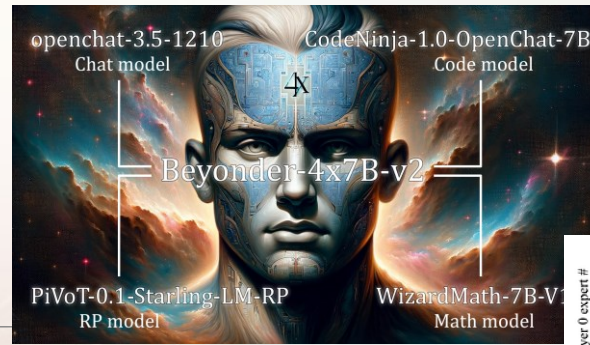
#	Model	pass@1
1	GPT-4-Turbo (Nov 2023) ⚡	⚡ 81.7
2	GPT-4 (May 2023) ⚡	⚡ 79.3
3	DeepSeek-Coder-33B-instruct ⚡	⚡ 75.0
4	WizardCoder-33B-V1.1 ⚡	⚡ 73.2
5	speechless-codellama-34B-v2.0 ⚡ ❤️	⚡ 71.3
6	GPT-3.5-Turbo (Nov 2023) ⚡	⚡ 70.7
7	MagiCoder-S-DS-6.7B ⚡ ❤️	⚡ 70.7
8	DeepSeek-Coder-6.7B-instruct ⚡	⚡ 70.1
9	code-millennials-34B ⚡	⚡ 70.1
10	DeepSeek-Coder-7B-instruct-v1.5 ⚡	⚡ 69.5
11	XwinCoder-34B ⚡	⚡ 67.7
12	Phind-CodeLlama-34B-v2	⚡ 67.1
13	OpenChat-3.5-7B-0106 ⚡ ❤️	⚡ 67.1
14	GPT-3.5 (May 2023) ⚡	⚡ 66.5
15	MagiCoder-S-CL-7B ⚡ ❤️	⚡ 66.5
16	CodeLlama-70B-Instruct ⚡	⚡ 65.2
17	WizardCoder-Python-34B-V1.0 ⚡	⚡ 64.6
18	speechless-coder-ds-6.7B ⚡ ❤️	⚡ 64.6

<https://evalplus.github.io/leaderboard.html>

Leaderboard Overview

1. Code <https://evalplus.github.io/leaderboard.html>
2. https://huggingface.co/spaces/PatronusAI/enterprise_scenarios_leaderboard
3. Reasoning <https://huggingface.co/spaces/NPHardEval/NPHardEval-leaderboard>
4. Emotional Intelligence benchmark <https://eqbench.com/>
5. Chatbot Arena <https://chat.lmsys.org/>
 1. <https://huggingface.co/spaces/lmsys/chatbot-arena-leaderboard>
6. https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard
 1. <https://huggingfaceh4-open-llm-leaderboard.hf.space/>
7. <https://crfm.stanford.edu/helm/lite/v1.0.0/#/leaderboard>

Choose Your LLM



General / Large Context

- yi-34b-200k-dare-megamerge-v8: 200k Context

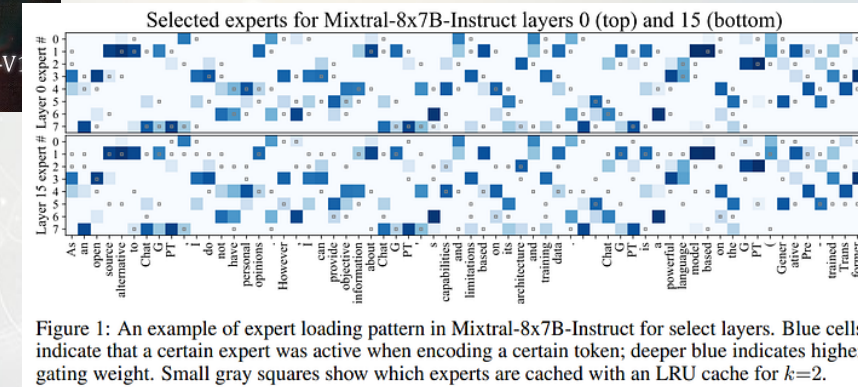
Mix of Experts – MoE

- beyonder-4x7b-v2.Q5_K_M.gguf
- Mixtral-8x7B-Instruct-v0.1

Great for coding

- wizardcoder-33b-v1.1 , Phind codellama 34b
- Mac User with local LLM: I really didn't mind it took it took 10 minutes, to be honest. I asked it to look over 50-thousands of lines of code and find a did. Instead of me doing that, I got a sandwich and just had the answer hand I'm perfectly fine with that ;)

Best for German: Sauerkraut



We just released our 70B German Model: SauerkrautLM-70b-v1


New Model

SauerkrautLM-v1

Available in
3B
7B
13B
70B

[www.huggingface.co/VAGOSolutions](https://huggingface.co/VAGOSolutions)
[www.vago-solutions.de](https://vago-solutions.de)

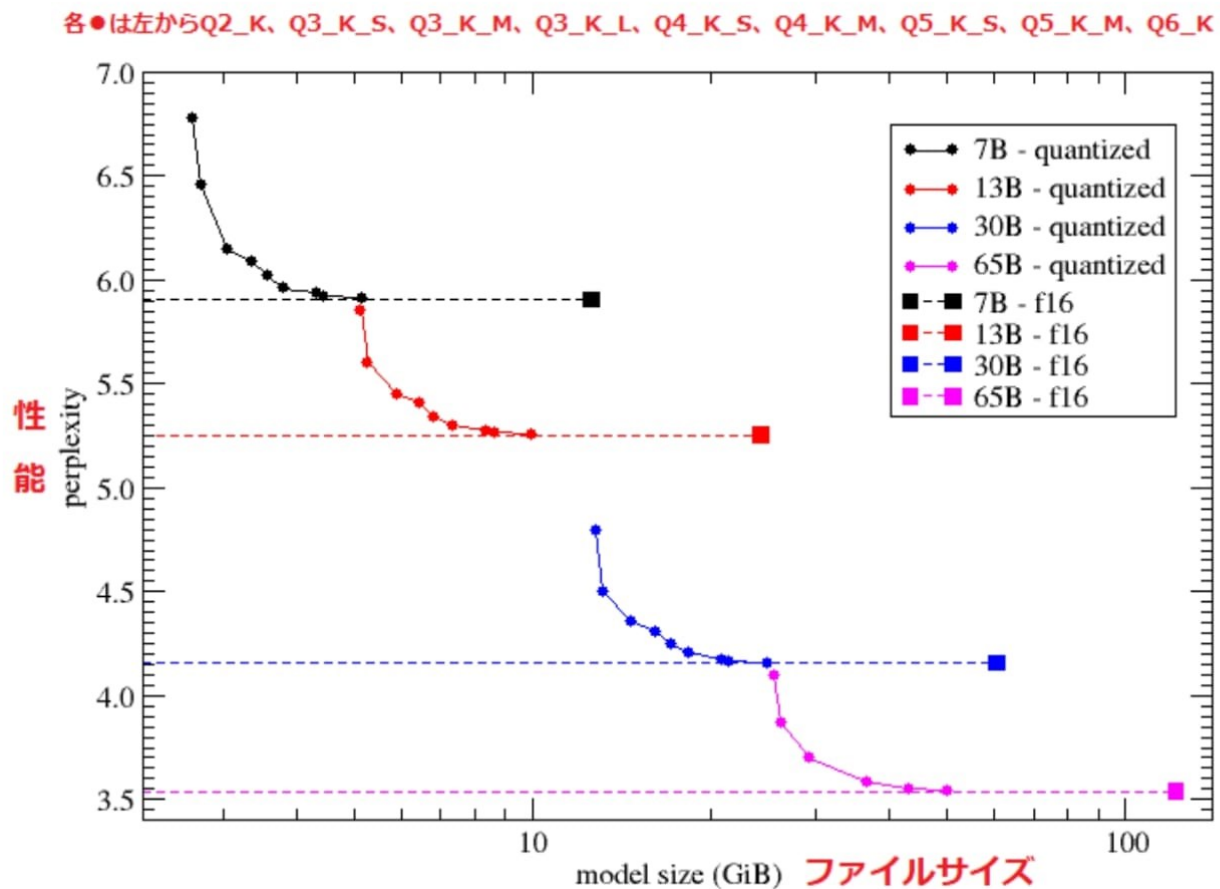
VAGO



A powerful 70b model for Germany! 🇩🇪 🍷 Our 70 billion parameter language model for the German language is now available for free use.

Download at: <https://huggingface.co/VAGOSolutions/SauerkrautLM-70b-v1>

Effect of Quantization



How to optimize Local LLM Inference Quality and Performance

Use larger models at lower quantizations

- E.g. prefer a 70B model with 2 - 4 bit over 13B with 8 bit

Memory bandwidth is key

- This is why GPUs are performing so well, e.g. AMD MI300X with 5.3 TB/sec memory bw, nVidia H200 with 4.8 TB/sec
- You can optimize a CPU only server with high memory bandwidth to achieve attractive inference speed

Local LLM Tools

	App Dev	UI	Doc Upload	Model Support	Hardware	Inference Speed	Ease of Setup	Open Source	GitHub Stars
Ollama	★ ★ ★			★ ★	Mac, Linux	★ ★ ★	★ ★ ★	Yes	35.8k
Transformers	★ ★			★ ★ ★	Any, Nvidia	★ ★	★ ★	Yes	120k
Langchain	★ ★			★ ★ ★	Any		★ ★	Yes	76.1k
llama.cpp	★ ★			★ ★	Any	★ ★ ★	★ ★	Yes	50.4k
GPT4All	★ ★ ★	★ ★ ★	★ ★ ★	★ ★	Any	★ ★	★ ★ ★	Yes	60.9k
LM Studio		★ ★ ★		★ ★	Any	★ ★	★ ★ ★	No	
Jan AI		★ ★ ★		★ ★	Any	★ ★	★ ★ ★	Yes	9k
llm				★ ★	Any	★ ★	★ ★ ★	Yes	2.3k
h2oGPT	★ ★ ★	★ ★ ★	★ ★ ★	★ ★	Any	★ ★	★	Yes	9.7k
localllm	★ ★ ★			★ ★	Any, GCP	★ ★	★ ★	Yes	112

Local LLM Tools - Summary

- If you are looking to develop an AI application, and you have a Mac or Linux machine, **Ollama** is great because it's very easy to set up, easy to work with, and fast.
- If you are looking to chat locally with documents, **GPT4All** is the best out of the box solution that is also easy to set up
- If you are looking for advanced control and insight into neural networks and machine learning, as well as the widest range of model support, you should try **transformers**
- In terms of speed, I think **Ollama** or **llama.cpp** are both very fast
- If you are looking to work with a CLI tool, **llm** is clean and easy to set up
- If you want to use Google Cloud, you should look into **localllm**
- For native support for roleplay and gaming (adding characters, persistent stories), the best choices are going to be **textgen-webui** by Oobabooga, and **koboldcpp**.
Alternatively, you can use **ollama-webui**

Cost Comparison with GPU Servers

4 bit quant test

GPU	VRAM (GB)	SPEED (T/S)	PRICE (\$/HR)	VALUE (T/\$)
RTX A6000	48	52.77	0.79	240,508.60
RTX 6000 Ada	48	68.05	1.14	214,894.17
A40	48	45.17	0.79	205,840.98
L40	48	56.87	1.14	179,614.82
A100 SXM	80	61.96	2.29	97,413.77
RTX4090	24	12.61	0.74	61,362.23
H100 SXM5	80	43.00	4.69	33,008.47
RTX A5000	24	1.59	0.44	13,084.85

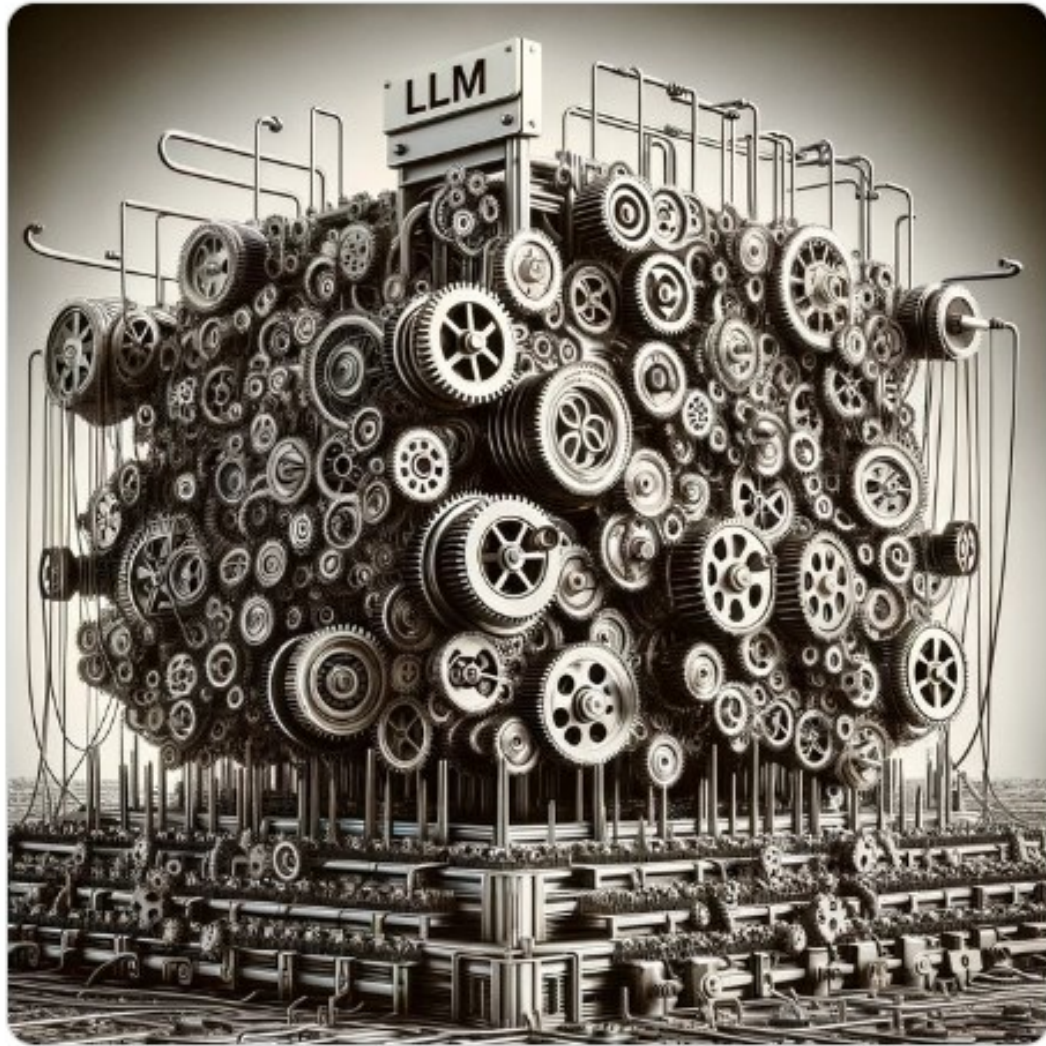
ChatGPT 4: \$20.00/Million tokens

ChatGPT 3.5: \$1.00/Million tokens

Mistral Medium: \$5.39/Million tokens (mistral.ai)

Re-test with lower quant for low vram GPUs

GPU	VRAM (GB)	SPEED (T/S)	PRICE (\$/HR)	VALUE (T/\$)	QUANT	MODEL
RTX 4000 Ada	20	46.75	0.39	431,538.46	2	mixtral:8x7b-instruct-v0.1-q2_K
A5000	24	49.45	0.44	404,590.91	2	mixtral:8x7b-instruct-v0.1-q2_K
RTX4090	24	75.88	0.74	369,145.95	2	mixtral:8x7b-instruct-v0.1-q2_K
RTX4090	24	69.01	0.74	335,724.32	3	mixtral:8x7b-instruct-v0.1-q3_K_S
A5000	24	38.25	0.44	312,954.55	3	mixtral:8x7b-instruct-v0.1-q3_K_S
A6000	48	57.67	0.79	262,800	2	mixtral:8x7b-instruct-v0.1-q2_K
RTX 6000 Ada	48	80.12	1.14	253,010.52	2	mixtral:8x7b-instruct-v0.1-q2_K
A40	48	51.05	0.79	232,632.91	2	mixtral:8x7b-instruct-v0.1-q2_K



Prompting for LLMs

These systems can be thought of as a massive set of gears. Answers are stuck at the intersections of certain gears. If you throw in the wrong inputs, you never get the right answer. The large gears are "concepts", and the smaller gears are "anchor points" defined by propositional logic.

https://twitter.com/kenshin9000_/status/1734238018343457082

Challenging Example

Take all information into account. Think backwards step by step.

My flight departs at 6:30 AM.

```
### Instruction:
Take all information into account. Think backwards step by step.

My flight departs at 6:30 AM.

Before leaving, I need an hour to get ready and have breakfast.

The journey to the airport will take approximately 45 minutes.

I should arrive at the airport an hour prior to departure.

Please create a schedule for me that outlines the time I need to wake up and when to leave in the vehicle, ensuring I reach the ai
rport in time for my 6:30 AM flight. The plan should be organized step-by-step.
<|im_end|>
<|im_start|>assistant
Alright, let's create a schedule for you to follow so you can make it to your 6:30 AM flight on time! We need to think backwards s
tep by step.

1. You need to arrive at the airport an hour before your flight (6:30 AM), which means you should be at the airport by 5:30 AM.
2. The journey to the airport takes about 45 minutes, so you need to leave your home around 5:30 AM - 45 minutes = 4:45 AM.
3. Before leaving, you want to spend an hour getting ready and having breakfast. So, you should start preparing at 4:45 AM - 1 hou
r = 3:45 AM.
4. Finally, this means you need to wake up at 3:45 AM - 10 minutes (to get yourself fully awake) = 3:35 AM.

So, your schedule should be:
- Wake up at 3:35 AM
- Get ready and have breakfast from 3:45 AM to 4:45 AM
- Leave your home around 4:45 AM
- Arrive at the airport by 5:30 AM
- Board your flight at 6:30 AM
```


LLM DEMO

1. llama.cpp

- Chat/Summary (<https://huggingface.co/TheBloke/Yi-34B-200K-DARE-megamerge-v8-GGUF>, <https://huggingface.co/mlabonne/Beyonder-4x7B-v2>)
- Image analysis - llava with GUI (https://huggingface.co/mys/ggml_bakllava-1)

2. LM Studio

- Programming (<https://huggingface.co/TheBloke/WizardCoder-33B-V1.1-GGUF>)

3. IntelliJ integration with coding LLM

- Local, Remote (<https://huggingface.co/TheBloke/CodeLlama-7B-Python-GGUF>)

BACKUP



Configure Running Local LLMs

max_new_tokens: Maximum number of tokens to be generated.

temperature: Diversity of the generated text (<0.5 ensures more deterministic results, while >0.5 increases creativity and randomness).

do_sample: Select the next token from a subset of the most probable next tokens instead of always selecting the token with the highest probability.

top_k: Select the next token from the k most likely next tokens.

top_p: Select the next output token from a subset of all the most probable next tokens, where the cumulative probability of the subset is greater than p. If top_k is also defined, the subset can contain a maximum of k tokens.

LLM reasoning

FPham · 4 hr. ago

Me: if I put plate on a banana in kitchen and then take the plate to the living room. Where is the banana.

Chat GPT 3.5: If you put a plate with a banana on it in the kitchen and then take the plate to the living room, the banana will still be on the plate in the living room. Moving the plate to a different room doesn't change the fact that the banana is on the plate.

Qwen: The banana is still in the kitchen. When you took the plate from the kitchen to the living room, you did not move the banana along with it.

throwawayplop · 46 min. ago

I'm officialy dumber that a 14B LLM

<https://www.reddit.com/r/LocalLLaMA/>



MINIMAN10001 · 3 hr. ago

It took me two read through of everything to catch it lol.

You shoved a plate on top of a banana but then chose to pick the plate up and run off with it.

This meant gpt 3.5 was wrong but the local LLMs were right

↑ 9 ↓ Reply Share ...



SteerPike · 2 hr. ago

Congratulations, you can now conclusively say that you rank somewhere between gpt-3.5 and qwen in terms of reasoning capability

PyTorch vs. Llama.cpp/GGUF

The Llama2 family of LLMs are typically trained and fine-tuned in PyTorch. Hence, they are typically distributed as PyTorch projects on Huggingface. However, when it comes to inference, it is much more attractive to use the GGUF model format for three reasons.

1. Python is not a great stack for AI inference. It is beneficial to remove PyTorch and Python dependency in production systems. GGUF can support very efficient zero-Python inference using tools like llama.cpp.
2. The Llama2 models are trained with 16-bit floating point numbers as weights. It has been demonstrated that models can be scaled down to 4-bit integers for inference without losing much knowledge, but saving a large amount of computing resources (expensive GPU RAM in particular). This process is called quantization.
3. The GGUF format is specifically designed for LLM inference. It supports LLM tasks like programming, completion, question answering, language encoding and decoding, making it more versatile, faster and easier to use than PyTorch.

Text generation web UI

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

Features

- 3 interface modes: default (two columns), notebook, and chat
- Multiple model backends: [transformers](#), [llama.cpp](#), [ExLlama](#), [ExLlamaV2](#), [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>

LLM Inference Everywhere

GPU or CPU server:

- xwin-lm-13b-v0.1.Q5_K_M.gguf
- phind-codellama-34b-v2.Q4_K_M.gguf (<https://huggingface.co/TheBloke/Phind-CodeLlama-34B-v2-GGUF>)

LMSTUDIO on your laptop

- codellama-7b.Q5_K_M.gguf

LLM Inference Browser

- <https://huggingface.co/spaces/radames/Candle-Phi-1.5-Wasm>

LLM Background (1)

temp (Temperature):

- Temperature is a scaling factor applied to the logits (the raw output values from the last layer of the model) before converting them into probabilities.
- A value of 1.0 means no change (default behavior).
- Values greater than 1.0 make the output more random, while values less than 1.0 (like 0.2) make the output more deterministic and confident, with the model more likely to pick the most probable word.
- In essence, lower temperatures make the model's outputs sharper, and higher values make it more random.
- By adjusting these parameters, one can influence the generated text's diversity, randomness, and overall behavior. The optimal values might differ based on the specific application or desired output characteristics.

LLM Background (1)

top_k (Top-K sampling):

- During text generation, at each step, the model predicts the next word (or token) by assigning a probability to each word in its vocabulary.
- top_k sampling restricts the next word selection to the top K probable words. So if top_k is set to 40, the model will only consider the top 40 most probable words for its next word prediction.
- It introduces some randomness in the generation process since the model doesn't always pick the most probable word, but one from the top K words.

LLM Background (1)

top_p (Top-P sampling or nucleus sampling):

- Instead of selecting the top K probable words, top_p sampling selects the smallest set of words such that their cumulative probability exceeds a specified threshold P.
- For instance, if top_p is set to 0.95, it will consider the smallest set of words whose combined probability is at least 0.95. This could be 5 words, 50 words, or any other number depending on the probability distribution.
- This method provides more dynamic sampling compared to top_k because the number of words considered can change based on the context.

Instructions, Completion and Infilling

https://github.com/facebookresearch/codellama/blob/main/example_instructions.py

- `{ "role": "system", "content": "Provide answers in JavaScript", },`
`{ "role": "user", "content": "Write a function that computes the set of sums of all contiguous sublists of a given list." }`

https://github.com/facebookresearch/codellama/blob/main/example_completion.py

- # For these prompts, the expected answer is the natural continuation of the prompt

https://github.com/facebookresearch/codellama/blob/main/example_infilling.py

- # Installation instructions:
- ````bash`
- `<FILL>`
- `````
- This downloads the LLaMA inference code and installs the repository as a local pip package.

Comparison

GPT-4

Ko

Analyze this step by step: I've put a plate on top of an apple, and take the plate from kitchen to dining room. Where is the apple now



Let's break down and analyze the statement step by step:

1. **"I've put a plate on top of an apple,"**

This means that the apple now has a plate resting on top of it.

2. **"...and take the plate from kitchen to dining room."**

This statement tells us that the plate, which was previously in the kitchen, has been transported to the dining room.

Now let's make some inferences based on these steps:



Bard

Analyze this step by step: I've put a plate on top of an apple, and take the plate from kitchen to dining room. Where is the apple now

View ot



Here is a step-by-step analysis of the following question:

I've put a plate on top of an apple, and take the plate from kitchen to dining room. Where is the apple now?

codellama-7b-instruct.Q5_K_M.gguf

USER

Analyze this step by step:

I've put a plate on top of an apple, and take the plate from kitchen to dining room. Where is the apple now

The apple is still in the kitchen.

AI

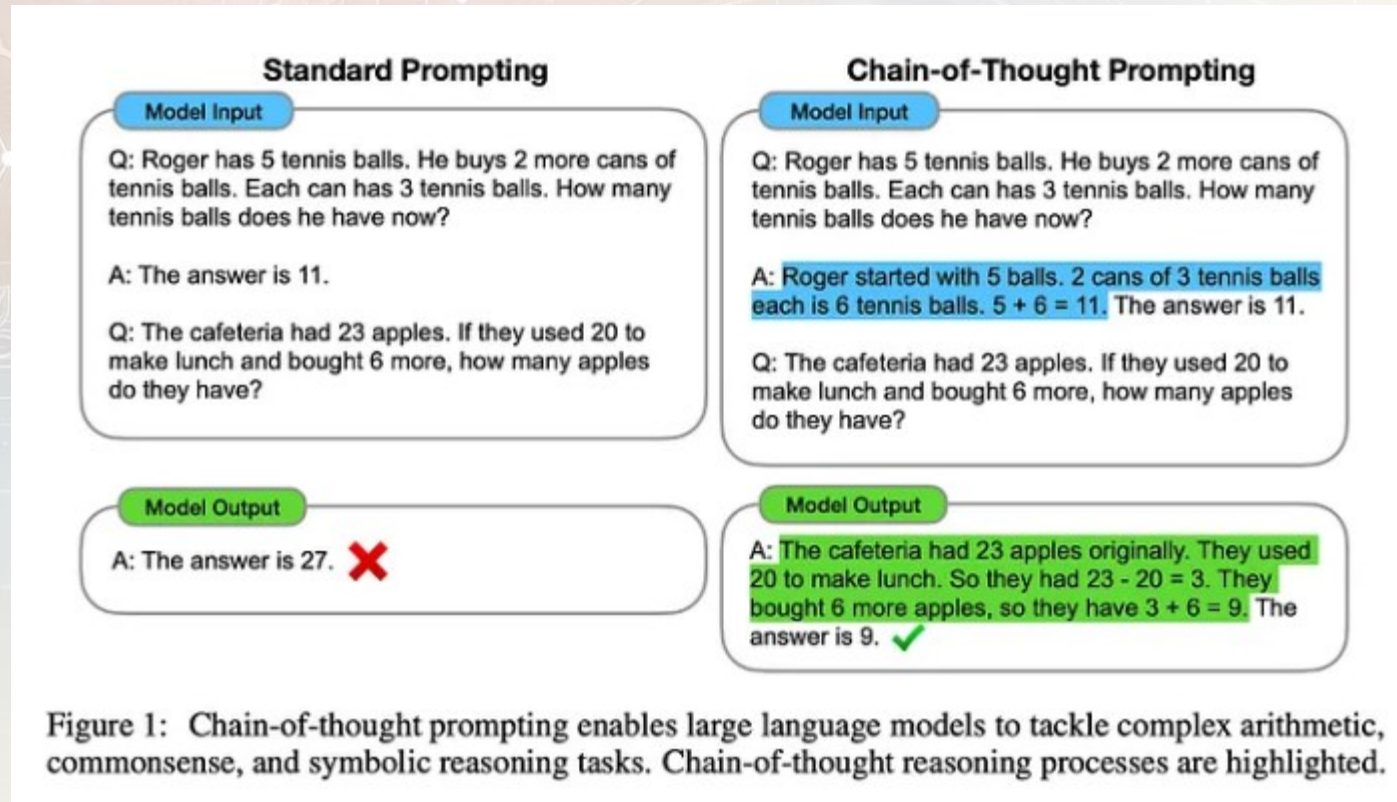
Explanation:

The plate was placed on top of the apple in the kitchen. When you took the plate from the kitchen to the dining room, the apple remained in the kitchen.

ow in the dining room.

ble together. The apple is not being left behind in the kitchen.

Prompting



<https://towardsdatascience.com/advanced-prompt-engineering-f07f9e55fe01>

Why local LLMs

1. Why host your own LLM. Is it a privacy concern or something else?
2. Do people load multiple LLMs on top of each other or just choose one?
3. How do you know which one to choose and why it's best?
4. When hosting your own LLM can you interact with it via voice and it respond accordingly? If so how? :)
5. With new LLMs out almost weekly will people swap them as they go and will their model lose context of that user (like staring all over again)
6. Anyone have a good YouTube link to show how a basic bitch like myself can install a modest model for experimentation with on a M2 Mac.

I like being able to switch between nearly any of the latest models whenever I want. I also feel comfortable using my own AI for work, as sending my workplace's code to OpenAI/MS is considered a no-no. Self-hosting an AI is also very rewarding and a great learning experience.

I have a few of my favorite LLMs saved to disk, but I only ever have one loaded and ready. Since I'm the only user, I can just change the models out whenever I please. I do have configurations for loading two models at once if I ever needed to, but I really haven't had to.

The best model is the one that pisses me off the least. I usually use Mixtral 8x7b, but I'm always trying out other models. Generally speaking, bigger is better. If the models are the same size, it's usually down to your personal tastes. You've just gotta try them all man.

Yes. See `whisper.cpp`

No. AI does not have any memory after it has been trained. People give AI memories by saving the old conversations to plain ol' text files. These text files are compatible with all AI models.

`gpt4all.io` is piss-simple to get up and running. There are many many more options out there. My favorite and arguably the best for developers is `llama.cpp`

https://www.reddit.com/r/LocalLLaMA/comments/18xasd4/home_llm_why/

Self-Extend – increase Context Length

<https://github.com/ggerganov/llama.cpp/pull/4815>,
<https://github.com/ggerganov/llama.cpp/issues/4886>

First, you set `-c` to the context that you want to achieve - let's say `-c 8192`.

Next, given that the original training context of the model is T (let's assume $T = 2048$), you want to set $G \geq 8192 / T$, so in this case: `--grp-attn-n 4` or `--grp-attn-n 8`.

The `--grp-attn-w` corresponds to W from the paper. I think the authors generally used 512, but I think you can go up to $T/2$ - so in this case `--grp-attn-w 1024`.

Additionally, G has to be multiple of W

LLM GUIs

Here's what I like using so far:

- [Mikupad](#)
- [chatbot-ui](#) (Note: this one takes some modification in order to work with llama.cpp, but there's gonna be a huge [update](#) for it any day now.)
- [llama.vim](#)
- [llama.sh](#) (best piece of software to ever be written *wink*)
- [continue.dev](#) for VSCode

https://www.reddit.com/r/LocalLLaMA/comments/18xnsar/what_all_front_ends_exist_for_connecting_to_llm/

LMStudio User Interface

The screenshot displays the LMStudio application interface. At the top, it shows system information: RAM Usage 2.57 GB and CPU: 0%. The current model is identified as 'TheBloke • codellama instruct 7B q5_k_m gguf'. The interface is divided into several sections:

- Chats:** A list of chat sessions on the left, including titles like 'Analyze this step by step...', 'In Bash, how do I list al...', and 'Untitled Chat'. Each entry shows memory usage and token counts.
- Chat Window:** The main area shows a conversation. The user asks, 'Analyze this step by step: I've put a plate on top of an apple, and take the plate from kitchen to dining room. Where is the apple now?'. The AI responds, 'The apple is still in the kitchen.' The user then asks, 'why - explain step by step in detail'. The AI responds, 'Because I took the plate from the kitchen to the dining room, but I did not take the apple with it.'
- Settings:** A panel on the right with sections for 'Preset' (set to 'WizCoder1'), 'Model Configuration' (with 'Save as Preset' and 'Export to JSON' buttons), 'Inference Parameters' (with sliders for 'temp' at 0.1, 'n_predict' at -1, and 'repeat_penalty' at 1.1), and 'Prompt Format'.
- Bottom Bar:** Contains buttons for 'Export as Screenshot', 'Regenerate', and 'Continue'. It also displays performance metrics: 'time to first token: 6.96s', 'gen t: 7.43s', 'speed: 3.23 tok/s', 'stop reason: completed', 'gpu layers: 24', 'cpu threads: 4', 'mlock: true', and 'token count: 121/1800'.

Other

<https://www.fuzzylabs.ai/blog-post/exploring-the-landscape-of-open-source-language-models-llms>

Code Assistant Services

- <https://codeium.com/>
- <https://cursor.sh/>

GPT-4

- According to rumors, gpt4 is a mixture of a few 220B parameters models.
- Even if it's a single 220B LLM, then that's 15x this 14B model and it's 3x the Llama2-70B models.