

Code, Chaos, and Copilots

Thinking in Tandem: Integrating AI into Daily Development

From Stable Diffusion to Unstable Decisions

This is information dense

The deck will be made available for offline ingestion

Happy to take Q&A at the end of the talk

If you find errors in this deck (and there will be), please let me know!

"👏PRs Accepted👏"

Obligatory Who Am I Slide

Hi, I'm Sam

- 19~ Years in tech 🧑💻😭🙌😂🤖
 - Tech -> Unix Systems Engineer -> Linux "" -> Operations Lead -> Platform Engineering Lead -> Platform Engineering Lead Consultant
- Started using what we know think of as "AI" in early 2020 an IDE integration, moved onto the Github Copilot private beta in 2021
 - Got *really* into AI/LLMs in late 2022 / early 2023
 - Use AI/LLMs to augment my capabilities and learning every day - with *great* success
- Geeks out on music, hifi, hardware, automation, AI, open source, cats, things with motors, lists
- Dislikes closed-source, bureaucracy, hierarchy, NPS surveys



Who this guy *isn't!*

Hello. My name is not Inigo Montoya.

Disclaimer

- I'm *not* an expert in any one AI thing within AI/ML
- I'm *not* a data or ML engineer / scientist
- I'm *not* a programmer
- I'm clearly not a comedian

If you find errors in this deck (and there will be), please let me know!

"👏PRs Accepted👏"

<https://smcleod.net>



I've tried a few tools this year

Here's the ones I remember

Hugging Face
Transformers

deepseek

open-webui

stability.ai

groq

crewai

Danswer

tabbyAPI

code2prompt

microWakeWord

Anything LLM

openWakeWord

lobe-chat

aider

ggml-org

r/LocalLLaMA

LLaMA C++

kohya_ss

BIG-AGI

ElevenLabs

Kerlig

Jan

MISTRAL AI

cohere

facefusion

refact.ai

kagi

Mindmac

Closed AI

Msty

Qwen

OpenRouter

GitHub Copilot

ExLlama U2

mistral.rs

TOPAZ LABS

BoltAI

Gollama

GPTQ

Langflow

Verba

Perplexica

MoA

Ollama-autocoder

faster-whisper

Apple Silicon

NVIDIA

invoke

LLaMA-Factory
Easy and Efficient LLM Fine-Tuning

COMFY UI

PIPER

VLLM

Textgen WebUI

Jupyter

H2O.ai

Langfuse

LM Studio

Tooling

Rating tools, techniques and libraries as I try them

Last Updated: 2024-07-14

Clients	Bolt	BigAGI	Open WebUI	Jan	LM-Studio	MindMac	Msty	Anything LLM	GPT4All	Librechat	Lobe Chat	Mods	Khoj
Codegen	Code2Prompt	Github Copilot	Continue.dev	Aider	Llama Autocoder								
Data and Training	Kohya_SS	Unsloth	Augment Toolk										
Imagegen	Invokeai	Stable Diffusion XL	Topaz Video A					Stable Swarm	Automatic1111	Draw Things	Pinokio	Sogni	Foocus
Libraries	Litellm	Langfuse	llm-scrapers										
Methods and Techniques	MoA	MoE	RAG										
Multimodal	Piper	Text-generation-webui	Whisper	MicroWakeWord	OpenWakeWord	TTS-generation-webui	Llavavision						
Observability / Utils	gollama	nvapi	parllama										
Pipelines / Integrations	Danswer	Crewai	Flowise	HomeAssistant AI Integration	Perplexica	Autogen	Fabric	Jupyter Notebook	Langflow	Verba			
SaaS Providers	Github Copilot	TogetherAI	Claude	DeepSeek AI	Elevenlabs	Groq	OpenRouter	Kagi AI	OpenAI	Suno	01.ai		
Servers	Llama.cpp	ExLlamav2	Ollama	TabbyAPI	llama.cpp RPC	Aphrodite engine	vLLM	Tensorrt-LLM	Localai	Mistralrs			

Open WebUI (Clients)
 Rating: 7/10
 Rapidly being developed, Fast UI, RAG, Configurable, Diagram generation. Cons: Commonly used configuration is buried away in 'advanced' settings. ***Highly recommend!***
[Click For More Info](#)

Note: Light blue cells with dashed outlines indicate items to check.

Heatmap Key (<-- 10 = Best | 1 = Worst -->)



<https://smcleod.net/2024/07/rating-ai-tools/>

AI:
Over-
or
Under-
Hyped



AI:
Over-
&
Under-
Hyped

I think
it's both

**"Check out our new AI
powered chatbot!"**

I just want to speak to support

**"Our product is enhanced
with AI"**

Extraordinary claims require extraordinary

***Most* products are over-hyped**

The CEO/CIOs have learnt a new buzzword

**"Do the impossible with
Atlassian Intelligence"**

How well are you really going to do AI if you
can't even build search that works?

100%
*"Of companies trust
our product"*

**"Your data will never be
used for training"**

I've heard that before

**The technological and
social implications
are
under-hyped**

We're facing an unprecedented global
reconfiguration of work and skills across industries

"It's just another hype cycle like cryptocurrency"

"Isn't it all just hallucinations?"

"What if it introduces 'bad' code?"

"I'm busy doing real work, I don't have time to learn new tools!"

**Doubt is rampant, much of it due
to AI tooling and tool fluency
being in its infancy**

"Your power bill must be high!"

"I tried it, but it's not that 'smart!'"

"I don't want to send all my data to <vendor>"

"Why care about local AI when there's ChatGPT?"

**However, I get value out of using
AI/LLMs every. single. day.**

This deck aims to **show some of the ways I use AI/LLMs to augment my capabilities**

Outline

- What I use AI/LLMs for
- Prompting tips
- Codegen workflow
- Picking the right models
- Model formats
- Context windows
- Quantisation
- Model servers
- Inference parameters
- Clients & tools
- Getting started cheatsheet

What I'm not covering today

Because I already have too many slides...

- <Insert SaaS AI vendor product here>
- Multimodal Models
- Classification Models
- Embeddings
- ImageGen
- RAG
- Agents / Agentic Workflows
- Frameworks / Libraries
- Training / Fine tuning
- Model censorship
- Security / Privacy
- Model Server Deployment / 'MLOps'
- IoT + LLMs
- AI Ethics / Politics / Debates
- My Predictions On Where We're Heading

but these are all fun things ...maybe a future talk?

How I use AI/LLMs

Everyday

- IDE integration
- Chat interfaces
- OS integration
- Home automation
- Article summarisation
- Document querying
- Web search augmentation
- Peer review
- Reviewing and rewriting text
- PKI / Notes integration

Sometimes

- Thought experiments
- Idea challenging
- Generating "art"
 - T-shirt designs
 - Wallpapers
 - Logos
- Making sick memes
- Swapping my face onto the entire cast of full house



How I use AI/LLMs

Prompting

My Number One Prompting Tip

Treat every (chat) interaction as if it's the first time you're meeting someone that knows nothing about you

Context Matters

Prompt Crafting

1. What is the objective and why are we doing this?
2. What are the constraints?
3. What is the expected output?
4. Can you provide an example?
5. Who is the intended audience?
6. Iterate

Create a simple Golang app that exposes a HTTP endpoint that outputs the CPU temperature.
The HTTP endpoint will be accessed remotely by monitoring software to alert on high CPU temperatures.

The program should:

- Take a http request from a remote host (e.g. via curl).
- Use off-the-shelf golang libraries where possible.
- Have consistent error handling.
- Have an (optional) configurable auth token to allow operations, verify the auth token if provided.
- Have the ability to log at a info, warn and debug level and appropriate logging statements.
- Contain comments for code that is complex or not obvious.
- The name of the application should be something fun, short, catchy and related to it's functionality.

The program should not crash if it cannot access the CPU temperatures.

The program should not require external (non-golang) dependencies to be installed on the host.

Your output code must be 100% feature complete, with no placeholders, unfinished code or TODOs.

The provided code should run when compiled without warnings or errors.

The program should return a JSON response with the cpu temperature (in Celsius), for example:

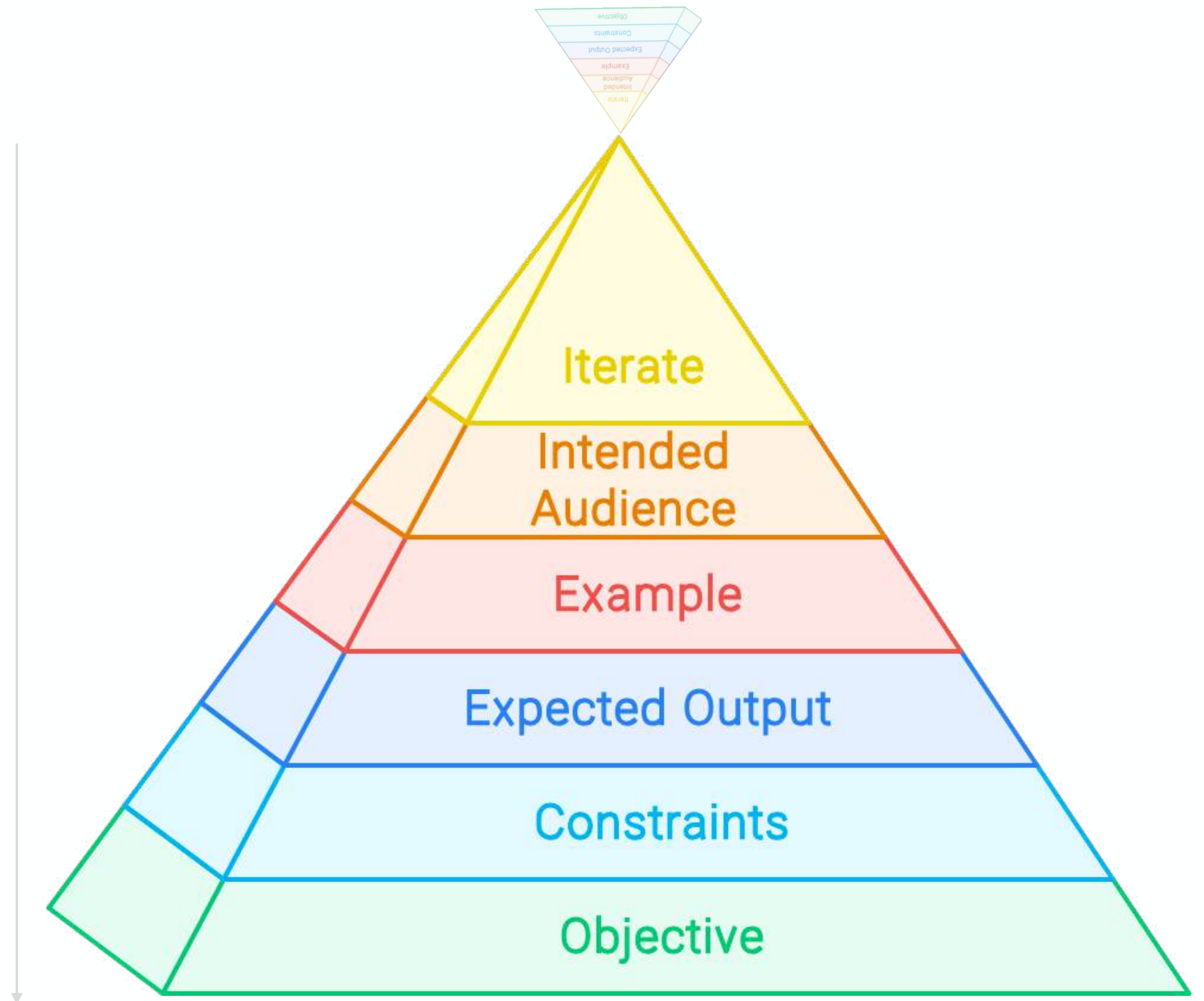
```
```shell
curl -X GET http://localhost/temperature
{
 "cpu_temperature": 55.3,
 "unit": "c"
}
```
```

You should provide a README.md suitable for sharing on Github that explains how to build and run the application.
Any documentation should be concise and targeted at a technical audience.

After you have finished consider the completed application as a whole and make a list of high value improvements.
Now complete the task, ensuring all requirements are met.

Prompt Crafting

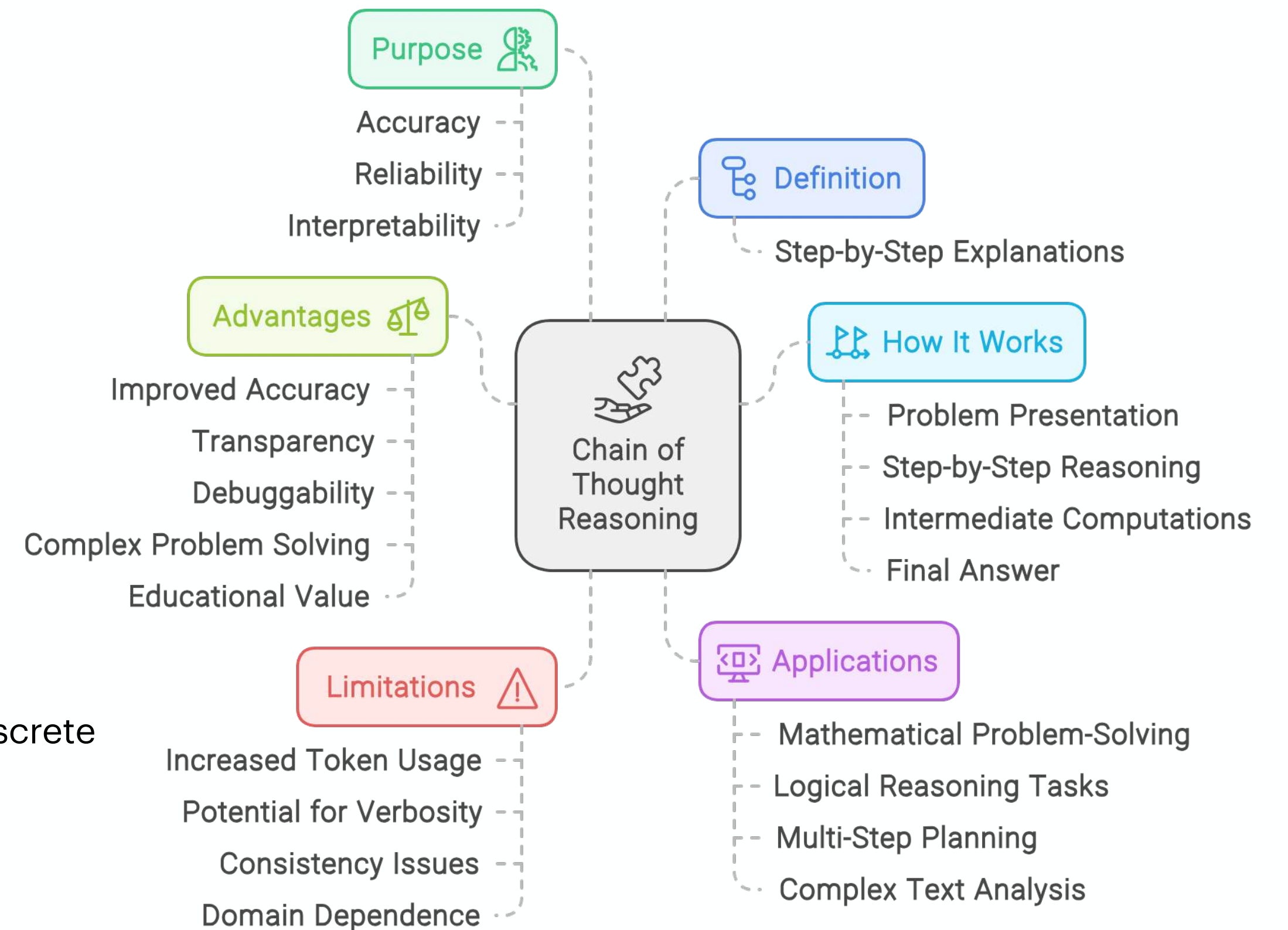
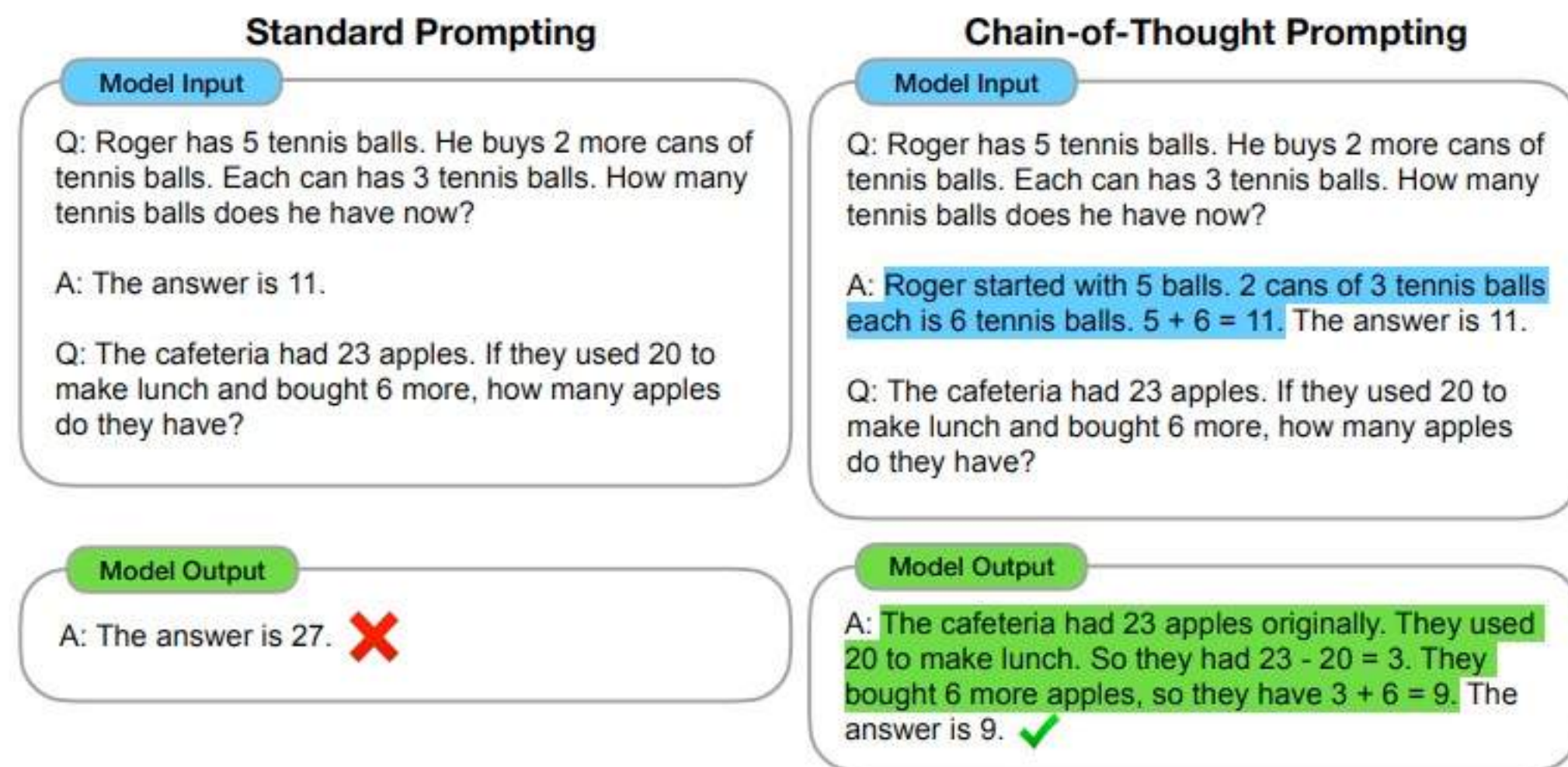
1. What is the objective and why are we doing this?
2. What are the constraints?
3. What is the expected output?
4. Can you provide an example?
5. Who is the intended audience?
6. Iterate



Prompt Crafting

Chain of Thought (CoT) Reasoning

Chain of Thought reasoning involves prompting an AI model to provide step-by-step explanations of its thought process while solving a problem or answering a question

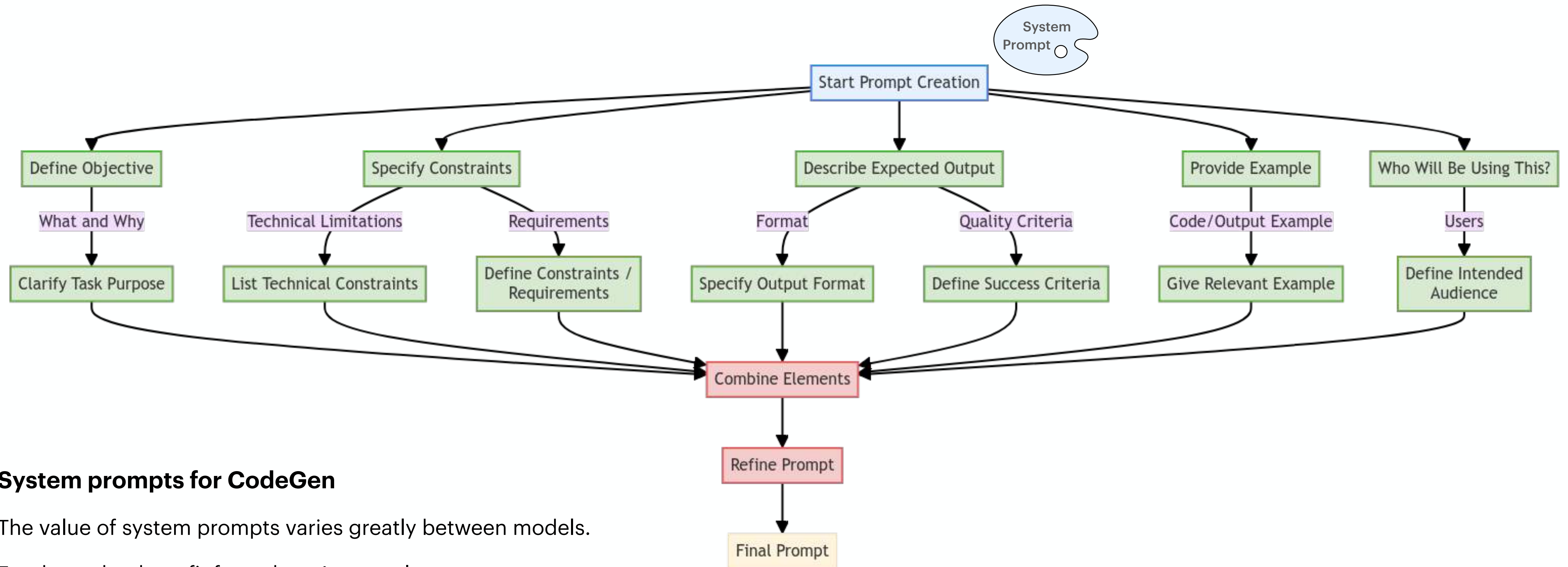


How it works

1. Problem Presentation: The user presents a problem or question to the AI model.
2. Step-by-Step Reasoning: The model is prompted to break down its thought process into discrete steps.
3. Intermediate Computations: The AI shows its work, including any calculations or logical deductions.
4. Final Answer: The model provides a conclusion based on its reasoning chain.

Prompt Crafting

Code Generation Example (Chat Based)



System prompts for CodeGen

The value of system prompts varies greatly between models.

For those that benefit from them I currently use:

"Before outputting any code list a few key bullet points (2-5 items at most) concisely stating (in 3-6 words) the steps you will take, then carry out the request in full."

How I use AI/LLMs

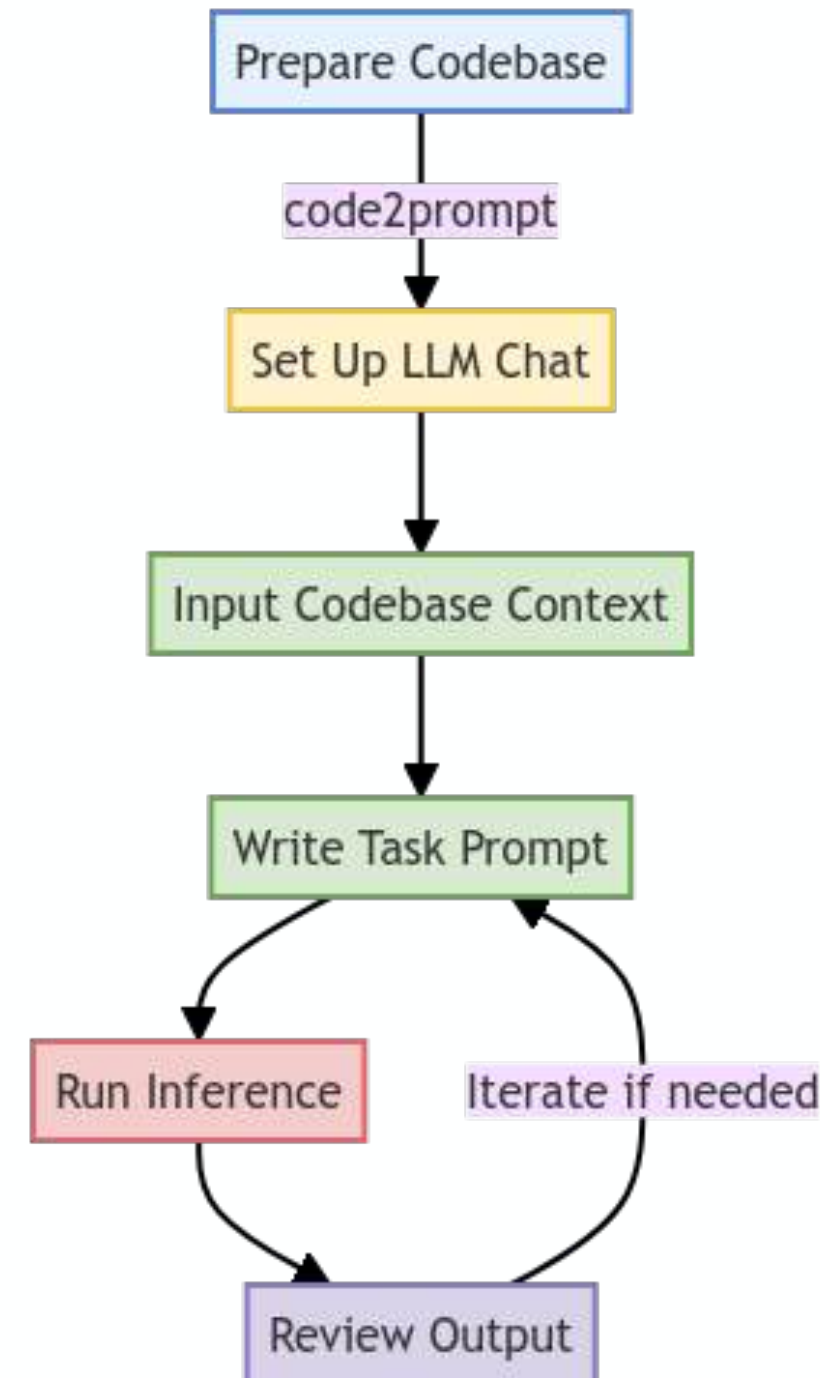
Code generation

IDE Integration

- Tab complete 2.0
- Select function -> Ask to refactor
- Generate code from comments
- Generate comments from code

Methods

- One shot
 - "Create a function that does X"
- Many shot
 - "Start by writing an app in X..."
 - "Now add Y..."
- MoA (Mixture of Agents)
 - Model A: "Generate X"
 - Model B: "Generate X"
 - Model C: "Review code generated by Model A and B, take the best parts from both..."



Using clients / chat interfaces

- Writing a program from scratch
- Code documentation
 - "Here is my codebase, write a README.md suitable for Github"
 - Generating doc-blocks
- Language translation
 - "The following is a Python script that does X, rewrite it in Golang"
- Refactoring
 - "Review my code, suggest improvements"
 - "Given the following, what is a better way to do X?"

How I use AI/LLMs

Code generation from DocBlocks

Write functions with a DocBlock

You can use AI integration in your IDE to write functions by first providing a DocBlock.

Start by defining:

- The functions name
- Expected and optional parameters
- Error handling
- Return values
- Optionally add a usage example

```
/**
 * Add a lifecycle rule to the repository based on the number of images to keep.
 * @name addLifecycleRuleNum
 * @param rulePriority The priority of the rule.
 * @param tagPrefixList The list of tag prefixes to apply the rule to.
 * @param maxImageCount The maximum number of images to keep.
 * @example addLifecycleRule(21, ['deployed-prod'], 15) { ... }
 */
const addLifecycleRuleNum = (rulePriority: number, tagPrefixList: string[], maxImageCount: number) => {
  const rule =
    tagPrefixList.length > 0
      ? { rulePriority, tagPrefixList, maxImageCount }
      : { rulePriority, maxImageCount };

  this.repository.addLifecycleRule(rule);
}
```

Then let Copilot/Continue.dev/whatever suggestions do the heavy lifting to write the function.

Copilot

```
slides.go 1, U
slides > slides.go > ReasonsNotToUseJava
1 package slides
2
3 // A golang function that lists all the reasons why you should not use Java in 2024
4 // Inputs (none)
5 // Outputs (slice of strings)
6 func ReasonsNotToUseJava() []string {
7     return []string{
8         "Java is verbose",
9         "Java is slow",
10        "Java is not modern",
11        "Java is not cool",
12        "Java is not fun",
13        "Java is not hip",
14        "Java is not trendy",
15        "Java is not sexy",
16        "Java is not hot",
17        "Java is not lit",
18        "Java is not fire",
19        "Java is not dope",
20        "Java is not rad",
21        "Java is not gnarly",
22        "Java is not sick",
23        "Java is not phat",
24        "Java is not fresh",
25        "Java is not fly",
26        "Java is not bangin'",
27        "Java is not poppin'",
28        "Java is not on fleek",
29        "Java is not swag",
30        "Java is not turnt",
31        "Java is not lit AF",
32        "Java is not Gucci",
33        "Java is not fire emoji",
34        "Java is not 100",
35        "Java is not the GOAT",
36        "Java is not the 🐐",
37        "Java is not the bomb",
38        "Java is not the
```

```
548
549 // A golang function that lists all the reasons why you should not use Java in 2024
550 // Inputs (none)
551 // Outputs (slice of strings)
552 func listReasonsToAvoidJava() []string {
553     return []string{
554         "1. Performance: Java is generally slower than languages like Go or Rust for certain types of applications.",
555         "2. Complexity: The syntax and verbosity of Java can make it harder to write clear, concise code compared to other languages.",
556         "3. Memory Management: While modern JVMs have improved performance, manual memory management in Java is still a potential source of bugs.",
557         "4. Learning Curve: New developers may find the extensive ecosystem and tooling required for effective Java development daunting.",
558         "5. Vendor Lock-in: Companies that rely heavily on Java might face challenges when trying to move to other platforms or technologies.",
559         "6. Security Flaws: Historical vulnerabilities in the JVM and libraries can pose significant security risks.",
560         "7. Scalability Issues: While Java is widely used, it may not always be the best choice for highly scalable systems due to its runtime overhead.",
561         "8. Community and Support: The popularity of Java means there's a vast amount of information online, but also a large community that can sometimes move slowly with updates.",
562     }
563 }
564
```

Continue

DeepSeek-Coder-V2 Lite

```
549 // A golang function that lists all the reasons why you should not use Java in 2024
550 // Inputs (none)
551 // Outputs (slice of strings)
552 func listReasonsToAvoidJava() []string {
553     return []string{
554         "It's a bloated language with unnecessary features.",
555         "It has poor performance and high memory usage.",
556         "It lacks modern programming paradigms like functional programming.",
557         "Its syntax is verbose and difficult to read.",
558         "It doesn't have built-in support for concurrency or parallelism.",
559         "It requires a large runtime environment that can be slow to start up.",
560         "It has poor tooling support, with few modern IDEs available.",
561         "It lacks strong type checking, leading to common bugs and errors.",
562         "Its ecosystem is fragmented and difficult to navigate.",
563         "It doesn't have a clear path for migrating existing codebases to newer versions.",
564         "It has poor performance on the JVM compared to other languages like Go or Rust.",
```

Go home copilot, you're drunk

```
"Java is not lit AF",  
"Java is not Gucci",  
"Java is not fire emoji",  
"Java is not 100",  
"Java is not the GOAT",  
"Java is not the 🐕",
```

Code Ingestion

ingest

File parsing for LLM ingestion

- Creates a single markdown file for inputting to LLMs
- Works on directories containing files
 1. Applies include/exclude filters
 2. Adds the directory tree
 3. Parses each file
 1. Wraps the (unmodified) content of each file into a markdown code block
 2. Adds the directory path for the file (code block)
 4. Outputs an estimate of tokens
 5. Copies the completed markdown to your clipboard
- Output can be piped to other tools, output to a file or copied to the clipboard (default)
- Can estimate vRAM for a given model, context size and quantisation
- Can directly query Ollama with the output
- Supports templating and exclude patterns

<https://github.com/sammcj/ingest>

```
> go run main.go --llm -p 'explain this code and one of its functions'
" Traversing directory and building tree... [0s] [i] 13,547 Tokens (Approximate)

This is a Go program that contains several utility functions. I'll provide an overview of the code and then focus on explaining one specific function.

Overview

The code consists of two main files: prompts.go and utils/utils.go. The first file defines a struct called Prompt with fields for rendering, output formats, and metadata. It also contains functions for encoding prompts into tokens and printing these prompts in different ways.

The second file (utils/utils.go) provides various utility functions for tasks such as:

• Reading and writing files
• Copying text to the clipboard
• Setting up progress spinners for displaying status updates

Let's take a closer look at one of these functions: SetupSpinner.

SetupSpinner

func SetupSpinner(message string) *progressbar.ProgressBar {
    return progressbar.NewOptions(-1,
        progressbar.OptionSetDescri
        progressbar.OptionSpinnerTy
        progressbar.OptionSetTheme(
            Saucer: "=",
            SaucerHead: ">",
            SaucerPadding: " ",
            BarStart: "[",
            BarEnd: "]"
        ),
    ),
}
```

```
> i --vram --model 'llama3.1:8b-instruct-q6_K' --fits 8
" Traversing directory and building tree.. [0s]
Using Ollama API URL: https://ollama

VRAM Estimation Results:
Model: llama3.1:8b-instruct-q6_K
Estimated vRAM Required: 9.38 GB
Fits Available vRAM: false
Max Context Size: 0
Maximum Quantisation: IQ3_S
VRAM Estimation for Model: llama3.1:8b-instruct-q6_K
```

| QUANT CTX | BPW | 2K | 8K | 16K | 32K | 49K | 64K |
|-----------|------|-----|-----|-----------------|-----------------|-----------------|-----------------|
| IQ1_S | 1.56 | 2.2 | 2.8 | 3.7(3.7,3.7) | 5.5(5.5,5.5) | 7.3(7.3,7.3) | 9.1(9.1,9.1) |
| IQ2_XXS | 2.06 | 2.6 | 3.3 | 4.3(4.3,4.3) | 6.1(6.1,6.1) | 7.9(7.9,7.9) | 9.8(9.8,9.8) |
| IQ2_XS | 2.31 | 2.9 | 3.6 | 4.5(4.5,4.5) | 6.4(6.4,6.4) | 8.2(8.2,8.2) | 10.1(10.1,10.1) |
| IQ2_S | 2.50 | 3.1 | 3.8 | 4.7(4.7,4.7) | 6.6(6.6,6.6) | 8.5(8.5,8.5) | 10.4(10.4,10.4) |
| IQ2_M | 2.70 | 3.2 | 4.0 | 4.9(4.9,4.9) | 6.8(6.8,6.8) | 8.7(8.7,8.7) | 10.6(10.6,10.6) |
| IQ3_XXS | 3.06 | 3.6 | 4.3 | 5.3(5.3,5.3) | 7.2(7.2,7.2) | 9.2(9.2,9.2) | 11.1(11.1,11.1) |
| IQ3_XS | 3.30 | 3.8 | 4.5 | 5.5(5.5,5.5) | 7.5(7.5,7.5) | 9.5(9.5,9.5) | 11.4(11.4,11.4) |
| Q2_K | 3.35 | 3.9 | 4.6 | 5.6(5.6,5.6) | 7.6(7.6,7.6) | 9.5(9.5,9.5) | 11.5(11.5,11.5) |
| Q3_K_S | 3.50 | 4.0 | 4.8 | 5.7(5.7,5.7) | 7.7(7.7,7.7) | 9.7(9.7,9.7) | 11.7(11.7,11.7) |
| IQ3_S | 3.50 | 4.0 | 4.8 | 5.7(5.7,5.7) | 7.7(7.7,7.7) | 9.7(9.7,9.7) | 11.7(11.7,11.7) |
| IQ3_M | 3.70 | 4.2 | 5.0 | 6.0(6.0,6.0) | 8.0(8.0,8.0) | 9.9(9.9,9.9) | 12.0(12.0,12.0) |
| Q3_K_M | 3.91 | 4.4 | 5.2 | 6.2(6.2,6.2) | 8.2(8.2,8.2) | 10.2(10.2,10.2) | 12.2(12.2,12.2) |
| IQ4_XS | 4.25 | 4.7 | 5.5 | 6.5(6.5,6.5) | 8.6(8.6,8.6) | 10.6(10.6,10.6) | 12.7(12.7,12.7) |
| Q3_K_L | 4.27 | 4.7 | 5.5 | 6.5(6.5,6.5) | 8.6(8.6,8.6) | 10.7(10.7,10.7) | 12.7(12.7,12.7) |
| IQ4_NL | 4.50 | 5.0 | 5.7 | 6.8(6.8,6.8) | 8.9(8.9,8.9) | 10.9(10.9,10.9) | 13.0(13.0,13.0) |
| Q4_0 | 4.55 | 5.0 | 5.8 | 6.8(6.8,6.8) | 8.9(8.9,8.9) | 11.0(11.0,11.0) | 13.1(13.1,13.1) |
| Q4_K_S | 4.58 | 5.0 | 5.8 | 6.9(6.9,6.9) | 8.9(8.9,8.9) | 11.0(11.0,11.0) | 13.1(13.1,13.1) |
| Q4_K_M | 4.85 | 5.3 | 6.1 | 7.1(7.1,7.1) | 9.2(9.2,9.2) | 11.4(11.4,11.4) | 13.5(13.5,13.5) |
| Q4_K_L | 4.90 | 5.3 | 6.1 | 7.2(7.2,7.2) | 9.3(9.3,9.3) | 11.4(11.4,11.4) | 13.6(13.6,13.6) |
| Q5_0 | 5.54 | 5.9 | 6.8 | 7.8(7.8,7.8) | 10.0(10.0,10.0) | 12.2(12.2,12.2) | 14.4(14.4,14.4) |
| Q5_K_S | 5.54 | 5.9 | 6.8 | 7.8(7.8,7.8) | 10.0(10.0,10.0) | 12.2(12.2,12.2) | 14.4(14.4,14.4) |
| Q5_K_M | 5.69 | 6.1 | 6.9 | 8.0(8.0,8.0) | 10.2(10.2,10.2) | 12.4(12.4,12.4) | 14.6(14.6,14.6) |
| Q5_K_L | 5.75 | 6.1 | 7.0 | 8.1(8.1,8.1) | 10.3(10.3,10.3) | 12.5(12.5,12.5) | 14.7(14.7,14.7) |
| Q6_K | 6.59 | 7.0 | 8.0 | 9.4(9.4,9.4) | 12.2(12.2,12.2) | 15.0(15.0,15.0) | 17.8(17.8,17.8) |
| Q8_0 | 8.50 | 8.8 | 9.9 | 11.4(11.4,11.4) | 14.4(14.4,14.4) | 17.4(17.4,17.4) | 20.3(20.3,20.3) |

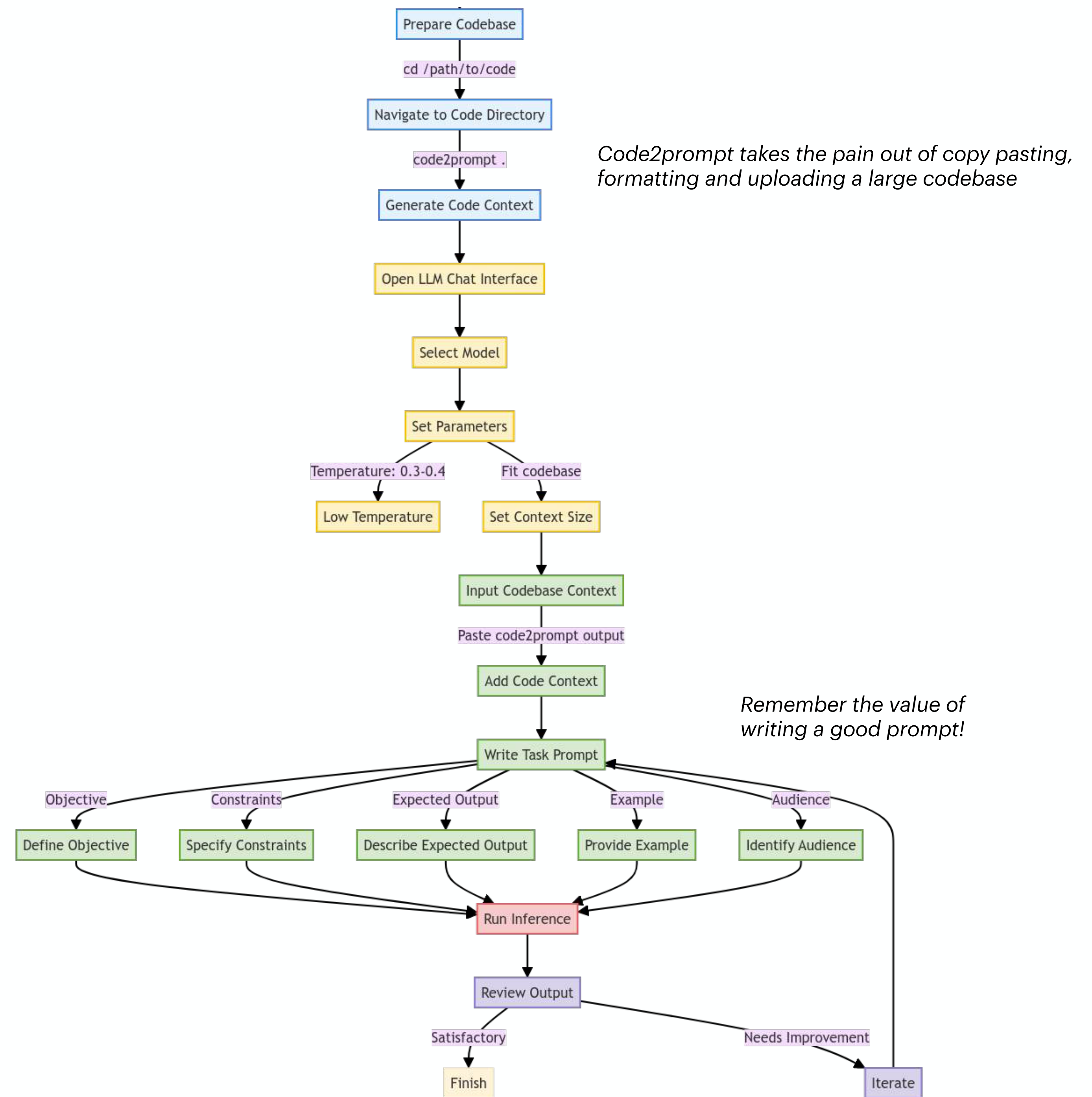
```
[!] Generated content (33795 tokens) exceeds maximum context (0 tokens).
[i] Tokens (Approximate): 33,795

Top 5 largest files (by estimated token count):
1. /Users/samm/git/sammcj/ingest/main.go (4,858 tokens)
2. /Users/samm/git/sammcj/ingest/filesystem/filesystem.go (2,694 tokens)
3. /Users/samm/git/sammcj/ingest/README.md (2,466 tokens)
4. /Users/samm/git/sammcj/ingest/utils/utils.go (948 tokens)
5. /Users/samm/git/sammcj/ingest/config/config.go (884 tokens)
[✓] Copied to clipboard successfully.
```

Code Generation

Workflow with prompt crafting and code2prompt

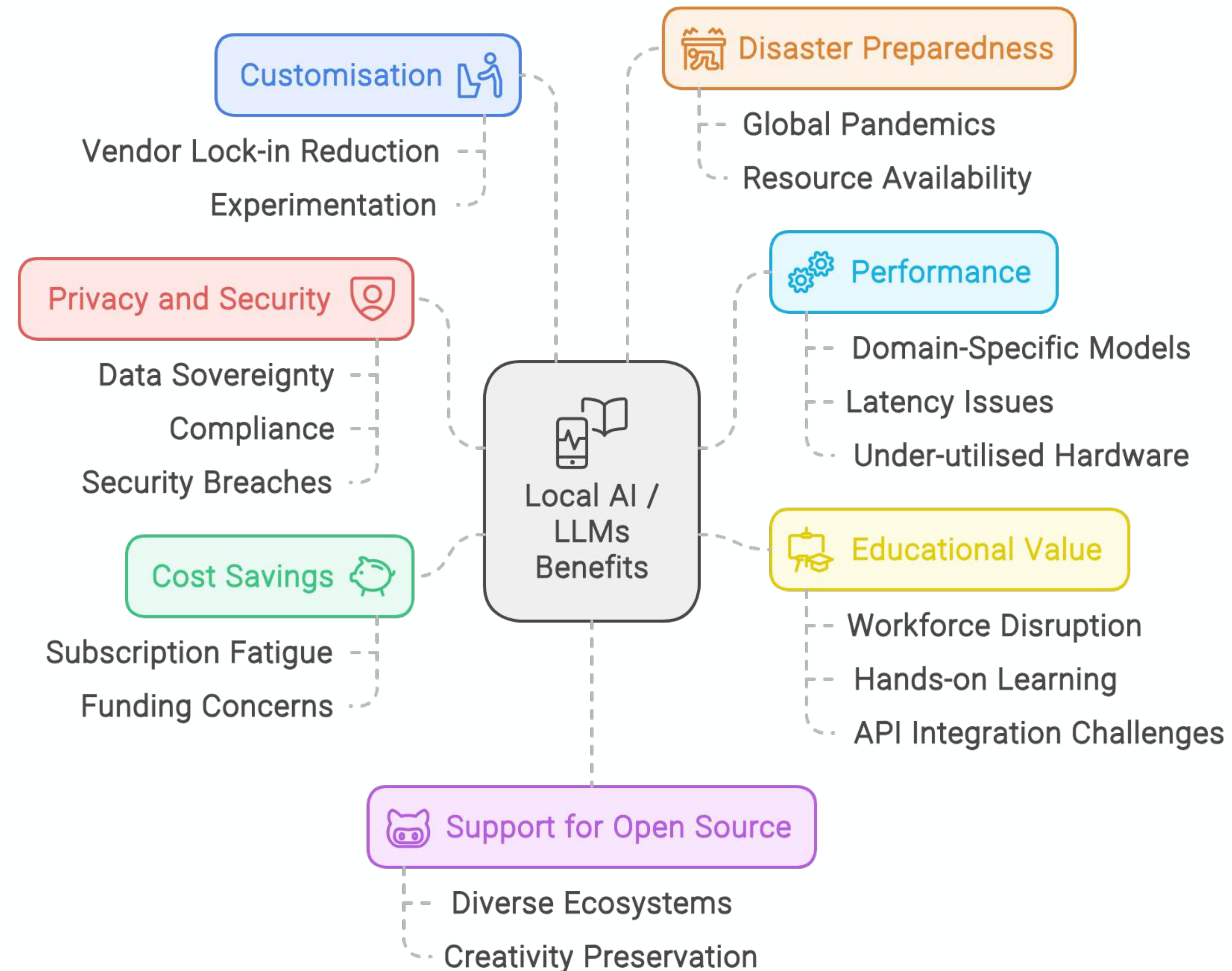
1. Use code2prompt to get the codebase ready to ingest
`cd /path/to/code`
`code2prompt .`
2. Open LLM Chat interface (e.g. Open WebUI)
 - Select the model
 - Set the parameters for coding tasks:
 - Low temperature (0.3-0.4)
 - Context size that will fit the codebase (you can use estimate from code2prompt!)
3. Add something like
'This is my application:'
paste the code2prompt output from the clipboard
Then 'Your task is:'
4. Write a prompt for the task
5. Run inference
6. Review
7. Iterate as needed



How I use AI/LLMs

Clients

Why Local LLMs?



Why Local LLMs?

That's a bit like asking why I wear my own clothes when I could rent costumes every day

- **Price**
 - Not just another bill, but another bill in *USD!*
 - Subscription fatigue is real
 - I don't really want my income to keep funding the 1% in silicon valley
- **Privacy and Security**
 - Your data doesn't leave your network unless you want it to
 - Compliance / data sovereignty / control points
 - *"OpenAI Leaks Sensitive User Data, exposed conversations, personal data, and login credentials."*
- **Agency**
 - Customisation and experimentation
 - Reduce vendor lock-in
- **Disaster Planning**
 - We've just had a global pandemic - what if something like that happens that impacts global stability or resource availability?
- **Performance**
 - Strong domain specific models
 - Latency and throughput (especially on Australian internet)
 - So many people have under-utilised hardware sitting on their desk - why not put it to use!
- **Supporting The Concept of Open Source / Open Weights**
 - We don't want to promote Internet Explorer style global ecosystem
 - Open communities are vibrant, diverse and helpful
 - Monocultures destroy creativity
- **Learning / Education / Fun**
 - We're potentially at the start of the biggest disruption to the global workforce
 - Why not help mitigate the risk of your role in the workforce shifting from creator to consumer?
 - Taking your understanding beyond the surface helps you grasp the system as a whole, embrace learning by breaking things and putting them back together
 - Let's be honest - glueing SaaS APIs together all day isn't always rewarding

Recommended Clients

Pros

- Fast, easy to use, simple UI for those familiar with chatGPT
- Open Source
- Rapid development
- Capable and configurable RAG and Web RAG
- Advance tooling, tool chaining and scripting
- Large community of tools, integrations, templates
- Multimodal
- Many optional features
- Lots of configuration options
- Long term memory (e.g. user prefers en-GB spelling)
- Multi-user, IDP integration, RBAC, Admin settings

Cons

- Commonly used inference parameters buried several clicks deep in the UI (*this is improving*)
- Being a web interface it has no OS integration

Open WebUI

The screenshot shows the Open WebUI interface. On the left is a chat history sidebar with various conversations. The main chat window shows a user message and an AI response. The AI response includes a JSON error log and a detailed explanation of the issue, along with steps to diagnose and resolve it. Below the chat window is a code editor showing a Go function for editing model files with added error handling.

```
AI deepseek-coder-v2-lite-instruct:q6_k_l +
AEST false llama} {registry.icu.io/ollama/sfr-embedding-mistral:q4_k_m 42269e3 4.068426355719566 Q4_K_M 2024-04-16 13:44:00.990859579 +1000 AEST false llama} {explain:7b 2ed9800
3.5631574979051948 Q4_0 2024-04-04 17:14:34.959418635 +1100 AEDT false llama}}\n"}
{"level":"debug","time":"2024-07-04T11:46:24+10:00","message":"Use VSCode: false\n"}

OI deepseek-coder-v2-lite-instruct:q6_k_l
Based on the provided logs and error message, it appears that the issue is occurring when the application attempts to create a new model with the edited model file. The error indicates a nil pointer dereference in the Create method of the Client. This suggests that there might be an issue with how the client is being initialized or passed around within your code.

Here are some steps you can take to diagnose and potentially resolve this issue:

1. Check Client Initialization: Ensure that the client variable is properly initialized before it is used in the editModelFile function. If the client is nil, it should not be passed into this function.
2. Add More Logging: Add more detailed logging around the areas where errors might occur to get a better understanding of what is happening just before the crash.
3. Error Handling: Implement proper error handling and return values from functions to ensure that any issues are caught and handled gracefully.
4. Check for Nil Values: Ensure that no nil pointers are being dereferenced by adding checks for nil values where appropriate.

Here's an updated version of your function with additional logging and error checking:

go
func editModelFile(client *api.Client, modelName string, useVSCode bool) {
    if client == nil {
        log.Println("Error: Client is nil")
        return
    }
    ctx := context.Background()

    listResp, err := client.List(ctx)
    if err != nil {
        log.Printf("Error listing models: %v\n", err)
        return
    }

    // Check if the specific model exists in the server's list of available models
    var modelExists bool
    for _, model := range listResp.Models {
        if model == modelName {
            modelExists = true
            break
        }
    }
}
```


Recommended Clients

Open WebUI

LLM Response Insights

Details of every generated response can be viewed, including external model API insights and comprehensive local model info

Export RLHF (Training) Annotations 👍👎

Enhance the impact of messages by rating them with either a thumbs up or thumbs down, followed by the option to provide textual feedback, facilitating the creation of datasets for Reinforcement Learning from Human Feedback (RLHF)

Utilise messages to train or fine-tune models, all while ensuring the confidentiality of locally saved data

The screenshot displays a chat interface. At the top, a light gray input box contains the text: "Show me a code snippet of a website's sticky header in CSS and JavaScript." Below this, a message from the model is shown with the identifier "OI deepseek-coder-v2-lite-instruct:q6_k_l" and a long string of "G" characters. A row of icons for navigation and actions is visible below the message. A feedback form titled "Tell us more:" is open, featuring five buttons: "Don't like the style", "Not factually correct", "Didn't fully follow instructions", "Refused when it shouldn't have", and "Being lazy", along with an "Other" button. A text input field contains the text "it's drooling...". A green "Submit" button is located at the bottom right of the feedback form.

Recommended Clients

Open WebUI

Workspace

Models Prompts Documents Tools **Functions**

Functions

Q Search Functions

FILTER v0.1.1 AutoTool Filter

autotool_filter A filter that pre-processes a user query to match relevant tools and populates the request with matching tool IDs.



FILTER v0.1.4 Semantic Model Router

semantic_model_router Used to allow small LLMs to make intention decisions and then dynamically switch between larger LLMs for different tasks. Also allows override...



FILTER v1.0 Myzer

myzer Minimise token count by stripping unnecessary white space and padding words, and optionally automatically request a concise answer. Minimising token count will (should)...



PIPE v6.6.6r9 DOOM Pipeline

doom_pipeline Play DOOM in your browser, just don't let the boss catch you!



FILTER v0.1.1 Cerebro Tool Launcher

cerebro_tool_launcher Allows LLMs to launch packages installed by cerebro with the appropriate prompt



FILTER v0.2.0 Cerebro Package Manager

cerebro_package_manager A package manager used to install GUI applets that run directly in chat, such as games, news applets, weather applets, and hopefully soon...



FILTER v0.1 Context Clip Filter

context_clip_filter A filter that truncates chat history to retain the latest messages while preserving the system prompt for optimal context management.



Import Functions

Export Functions

Documents

Q Search Documents



all #bubbletea-examples #testingdemo

#api-package-githubcomollamaollamaapi-go-packagesmd (api package - github.comollamaollamaapi - Go Packages.md)

#apimd (api.md)

Workspace



Models Prompts Documents **Tools** Functions

Tools

Q Search Tools

TOOL v0.1 Web Search

web_search Web Search using SearXNG and Scrap first N Pages

TOOL Convert to JSON

convert_to_json Converts data added to the chat into JSON then returns it in a markdown code block.

TOOL Wikidata

wikidata SPARQL Wikidata queries

TOOL v0.1 Home Assistant Tool

home_assistant_tool Simple device requests like turn on/off lights, pause media players, etc.

TOOL v0.2.1 User Info Retrieval Tool

user_info_retrieval_tool Get the User, Email and ID from the user object. This tool also serves as an example for setting up an Open WebUI tool with Valves & Event Emitters.

TOOL v0.0.2 Web Scrape

web_scrape Web Scraper using Jina Reader

Made by OpenWebUI Community



Discover a function

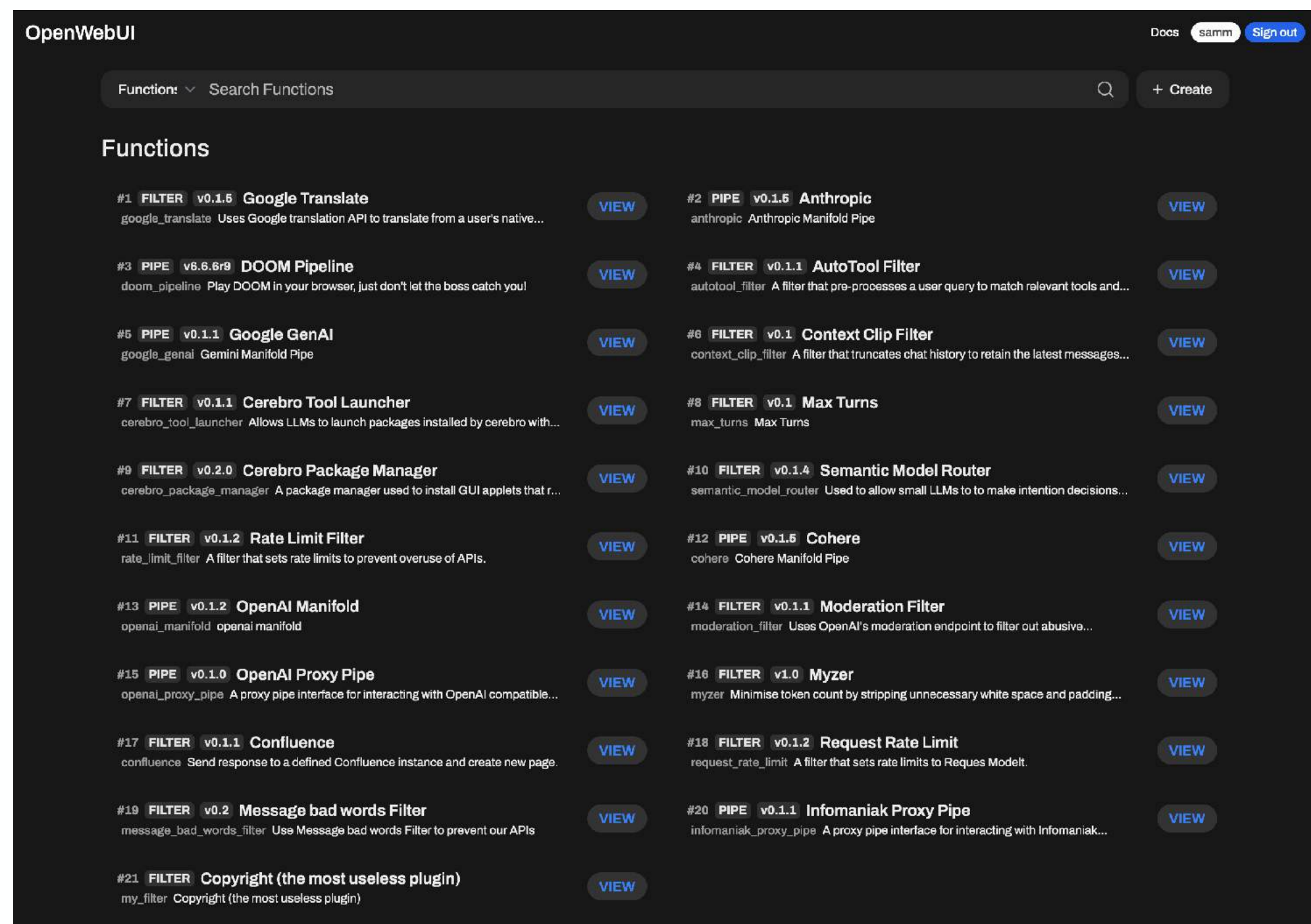
Discover, download, and explore custom functions

<https://github.com/open-webui/open-webui>

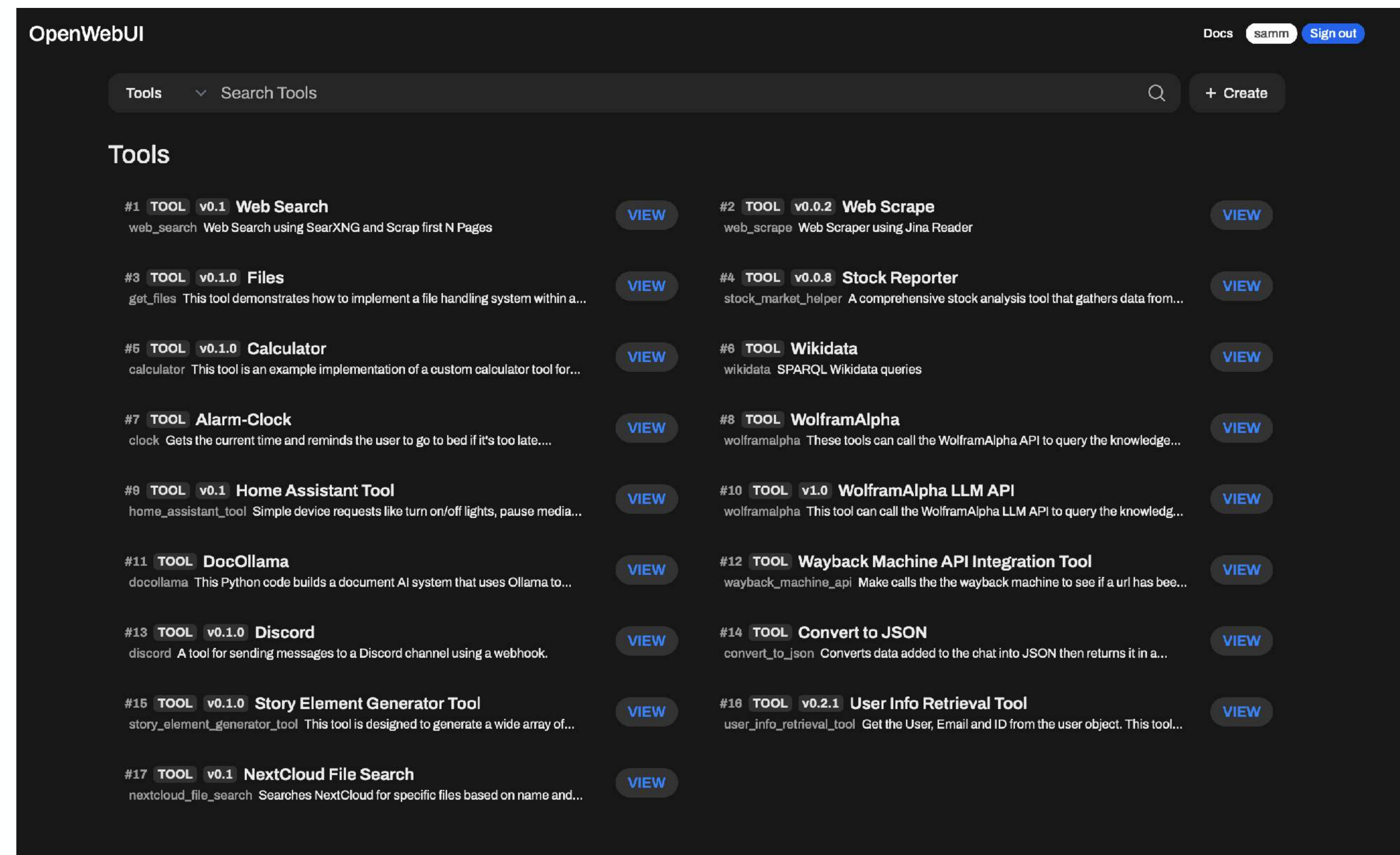
<https://openwebui.com/tools>

Recommended Clients

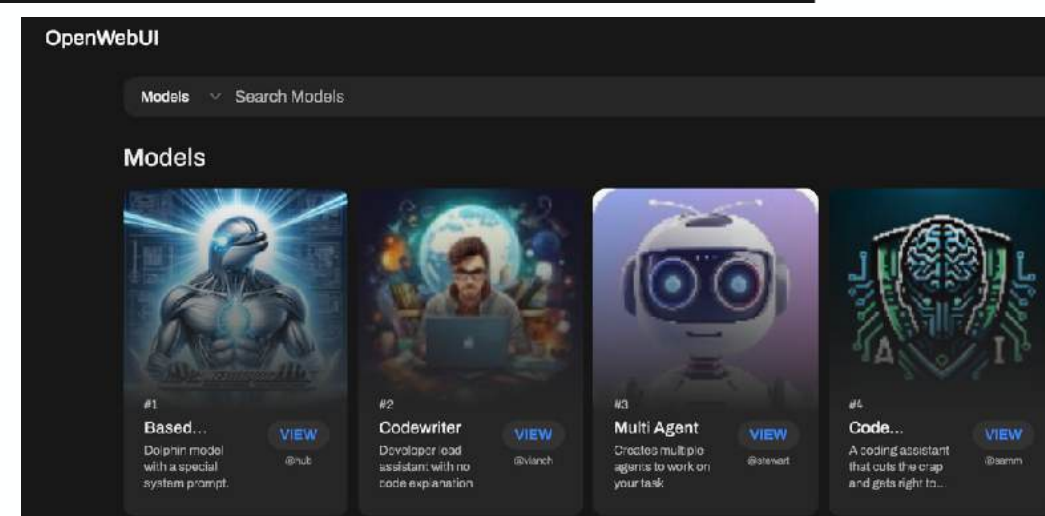
Open WebUI



The screenshot shows the 'Functions' page in OpenWebUI. At the top, there is a search bar labeled 'Function: Search Functions' and a '+ Create' button. Below the search bar, a list of 21 functions is displayed in a grid. Each function entry includes a number, a category (FILTER or PIPE), a version number, a title, a brief description, and a 'VIEW' button. The functions include: Google Translate, Anthropic, DOOM Pipeline, AutoTool Filter, Google GenAI, Context Clip Filter, Cerebro Tool Launcher, Max Turns, Cerebro Package Manager, Semantic Model Router, Rate Limit Filter, Cohere, OpenAI Manifold, Moderation Filter, OpenAI Proxy Pipe, Myzer, Confluence, Request Rate Limit, Message bad words Filter, Infomaniak Proxy Pipe, and Copyright (the most useless plugin).



The screenshot shows the 'Tools' page in OpenWebUI. At the top, there is a search bar labeled 'Tools Search Tools' and a '+ Create' button. Below the search bar, a list of 17 tools is displayed in a grid. Each tool entry includes a number, a category (TOOL), a version number, a title, a brief description, and a 'VIEW' button. The tools include: Web Search, Web Scrape, Files, Stock Reporter, Calculator, Wikidata, Alarm-Clock, WolframAlpha, Home Assistant Tool, WolframAlpha LLM API, DocOllama, Wayback Machine API Integration Tool, Discord, Convert to JSON, Story Element Generator Tool, User Info Retrieval Tool, and NextCloud File Search.



The screenshot shows the 'Models' page in OpenWebUI. At the top, there is a search bar labeled 'Models Search Models'. Below the search bar, a list of 4 models is displayed in a grid. Each model entry includes a number, a title, a brief description, and a 'VIEW' button. The models include: Based..., Codewriter, Multi Agent, and Code... (with a description: 'A coding assistant that can help you with your code and get it right.').

<https://openwebui.com/tools>
<https://openwebui.com/functions>
<https://openwebui.com/models>

<https://github.com/open-webui/open-webui>

Recommended Clients

Open WebUI

And yes, it can run doom

The screenshot shows the Open WebUI interface. On the left is a chat history sidebar with various prompts. The main area features a chat window with a terminal output for the command `/play doom`. The terminal shows the following download logs:

```
console
Downloading WAD..... ID: 610458fe-b7a3-417d-80e1-e7b7bf59bb2c DONE
Downloading CFG..... ID: 33267d5c-9dfe-4c2b-ba75-0bc362476214 DONE
Downloading WASM..... ID: d27c36b4-4988-4f0f-b3c3-77138e8bbce2 DONE
Downloading JS..... ID: 5048c79e-4195-4011-b833-1d1c4ce0503e DONE
Downloading HTML..... ID: c677f775-4353-4022-8f4c-eb120726b14a DONE
```

Below the terminal is a screenshot of the Doom game, showing the player's perspective in a dark, industrial environment. The player's status bar at the bottom indicates 49 AMMO, 100% HEALTH, 2/3/4 ARMS, and 100% ARMOR. A red text overlay at the top of the game screen reads "PICKED UP THE ARMOR."

At the bottom of the interface, there is a "Send a Message" input field and a footer with the text "LLMs can make mistakes. Verify important information." and the URL <https://github.com/open-webui/open-webui>.

Recommended Clients

Big AGI

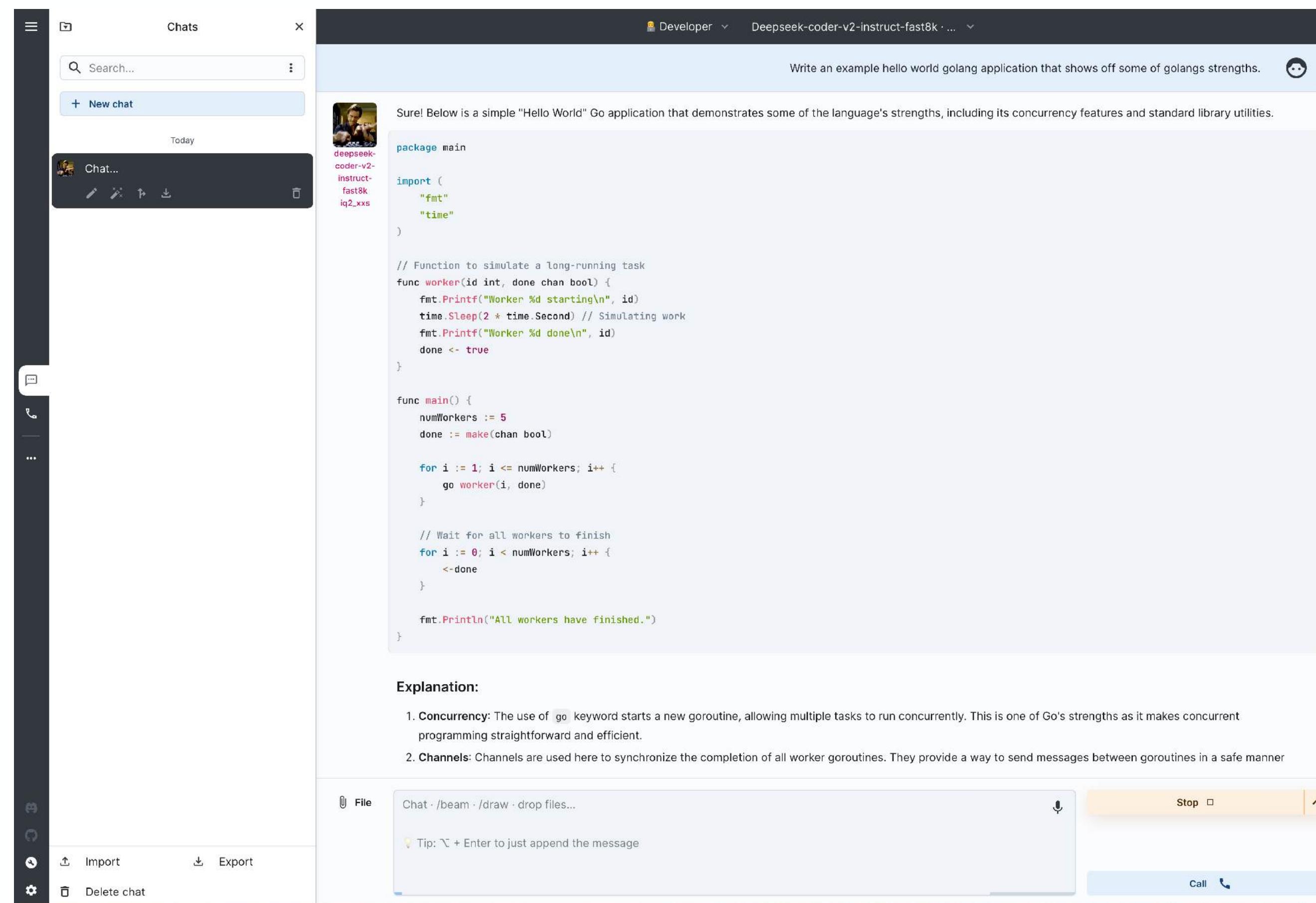
Pros

- Selling feature is "Beam" which can use multiple models to greatly improve reasoning, augment capabilities, fork and merge conversations, think MoA before it was cool
- Multi-model conversation branching
- Open Source
- Web search
- Very fast UI
- Great for codegen
- Great for generation diagrams
- Multimodal

Cons

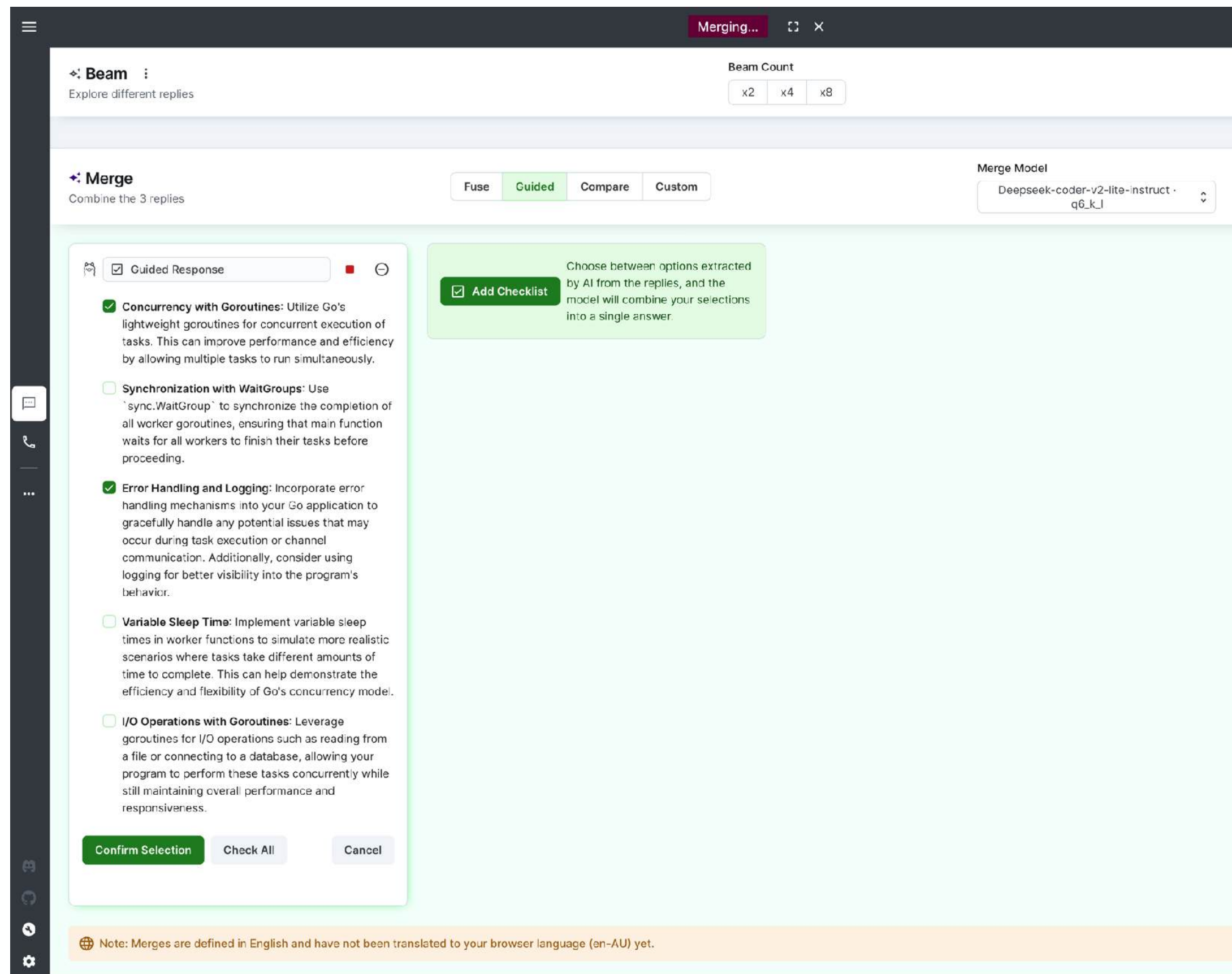
- Limited inference parameters
- Being a web interface it has no OS integration
- Saves config/data in the browser
- Development active, but slow

<https://github.com/enricoros/big-agi>



Recommended Clients

Big AGI



Beam Count: x2, x4, x8

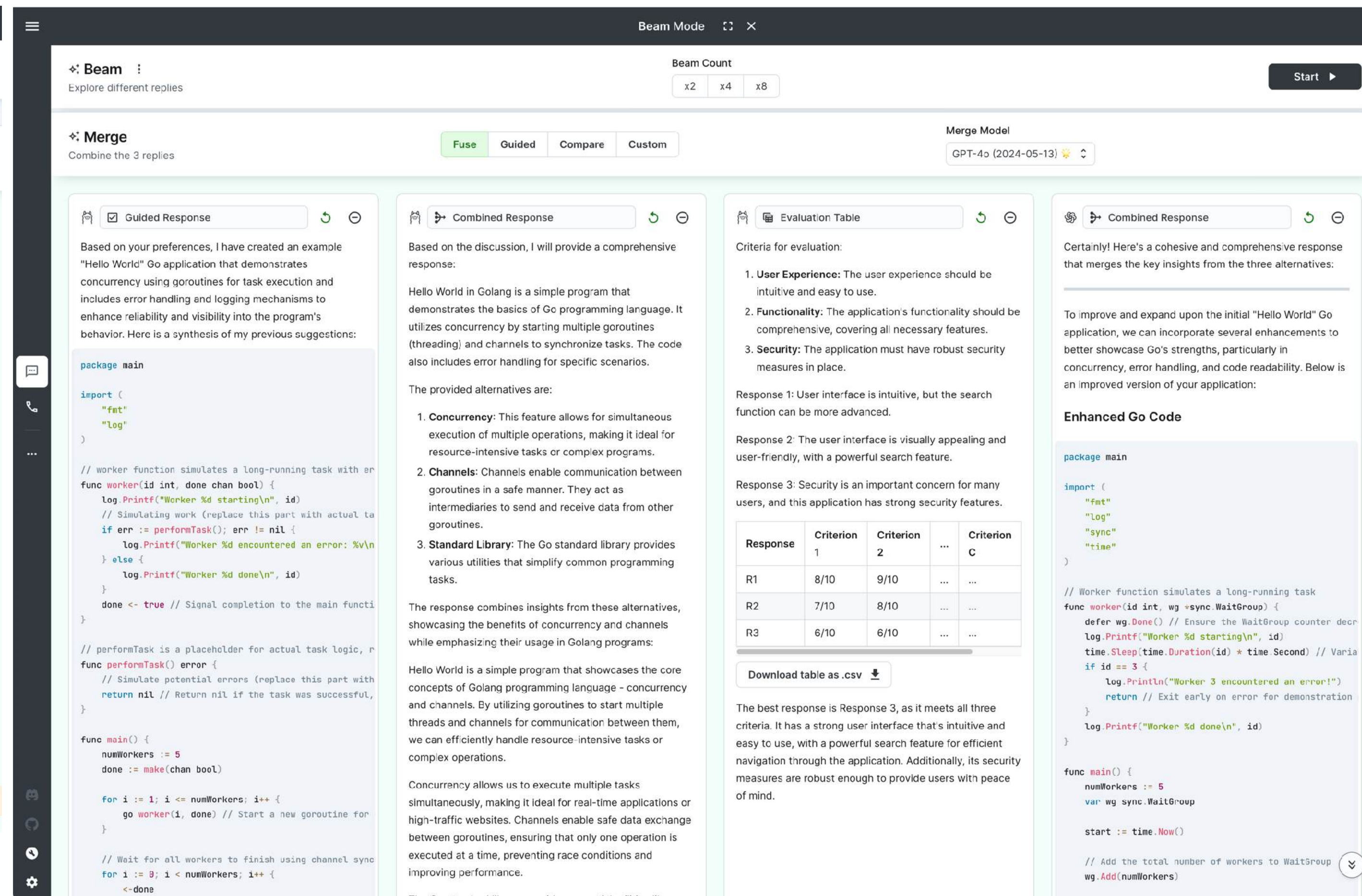
Merge Model: Deepseek-coder-v2-lite-instruct-q6_k_l

Options: Fuse, **Guided**, Compare, Custom

- Concurrency with Goroutines:** Utilize Go's lightweight goroutines for concurrent execution of tasks. This can improve performance and efficiency by allowing multiple tasks to run simultaneously.
- Synchronization with WaitGroups:** Use `sync.WaitGroup` to synchronize the completion of all worker goroutines, ensuring that main function waits for all workers to finish their tasks before proceeding.
- Error Handling and Logging:** Incorporate error handling mechanisms into your Go application to gracefully handle any potential issues that may occur during task execution or channel communication. Additionally, consider using logging for better visibility into the program's behavior.
- Variable Sleep Time:** Implement variable sleep times in worker functions to simulate more realistic scenarios where tasks take different amounts of time to complete. This can help demonstrate the efficiency and flexibility of Go's concurrency model.
- I/O Operations with Goroutines:** Leverage goroutines for I/O operations such as reading from a file or connecting to a database, allowing your program to perform these tasks concurrently while still maintaining overall performance and responsiveness.

Buttons: Confirm Selection, Check All, Cancel

Note: Merges are defined in English and have not been translated to your browser language (en-AU) yet.



Beam Count: x2, x4, x8

Merge Model: GPT-4o (2024-05-13)

Options: Fuse, **Guided**, Compare, Custom

Guided Response

Based on your preferences, I have created an example "Hello World" Go application that demonstrates concurrency using goroutines for task execution and includes error handling and logging mechanisms to enhance reliability and visibility into the program's behavior. Here is a synthesis of my previous suggestions:

```
package main

import (
    "fmt"
    "log"
)

// worker function simulates a long-running task with error handling
func worker(id int, done chan bool) {
    log.Printf("Worker %d starting\n", id)
    // Simulating work (replace this part with actual task logic)
    if err := performTask(); err != nil {
        log.Printf("Worker %d encountered an error: %v\n", id, err)
    } else {
        log.Printf("Worker %d done\n", id)
    }
    done <- true // Signal completion to the main function
}

// performTask is a placeholder for actual task logic, it can simulate errors
func performTask() error {
    // Simulate potential errors (replace this part with actual task logic)
    return nil // Return nil if the task was successful, or an error otherwise
}

func main() {
    numWorkers := 5
    done := make(chan bool)

    for i := 1; i <= numWorkers; i++ {
        go worker(i, done) // Start a new goroutine for each worker
    }

    // Wait for all workers to finish using channel sync
    for i := 0; i < numWorkers; i++ {
        <-done
    }
}
```

Combined Response

Based on the discussion, I will provide a comprehensive response:

Hello World in Golang is a simple program that demonstrates the basics of Go programming language. It utilizes concurrency by starting multiple goroutines (threading) and channels to synchronize tasks. The code also includes error handling for specific scenarios.

The provided alternatives are:

- Concurrency:** This feature allows for simultaneous execution of multiple operations, making it ideal for resource-intensive tasks or complex programs.
- Channels:** Channels enable communication between goroutines in a safe manner. They act as intermediaries to send and receive data from other goroutines.
- Standard Library:** The Go standard library provides various utilities that simplify common programming tasks.

The response combines insights from these alternatives, showcasing the benefits of concurrency and channels while emphasizing their usage in Golang programs:

Hello World is a simple program that showcases the core concepts of Golang programming language - concurrency and channels. By utilizing goroutines to start multiple threads and channels for communication between them, we can efficiently handle resource-intensive tasks or complex operations.

Concurrency allows us to execute multiple tasks simultaneously, making it ideal for real-time applications or high-traffic websites. Channels enable safe data exchange between goroutines, ensuring that only one operation is executed at a time, preventing race conditions and improving performance.

The Go standard library provides essential utilities like

Evaluation Table

Criteria for evaluation:

- User Experience:** The user experience should be intuitive and easy to use.
- Functionality:** The application's functionality should be comprehensive, covering all necessary features.
- Security:** The application must have robust security measures in place.

| Response | Criterion 1 | Criterion 2 | ... | Criterion C |
|----------|-------------|-------------|-----|-------------|
| R1 | 8/10 | 9/10 | ... | ... |
| R2 | 7/10 | 8/10 | ... | ... |
| R3 | 6/10 | 6/10 | ... | ... |

Download table as .csv

The best response is Response 3, as it meets all three criteria. It has a strong user interface that's intuitive and easy to use, with a powerful search feature for efficient navigation through the application. Additionally, its security measures are robust enough to provide users with peace of mind.

Enhanced Go Code

```
package main

import (
    "fmt"
    "log"
    "sync"
    "time"
)

// Worker function simulates a long-running task with error handling
func worker(id int, wg *sync.WaitGroup) {
    defer wg.Done() // Ensure the WaitGroup counter decrements
    log.Printf("Worker %d starting\n", id)
    time.Sleep(time.Duration(id) * time.Second) // Variable sleep time
    if id == 3 {
        log.Println("Worker 3 encountered an error!")
        return // Exit early on error for demonstration
    }
    log.Printf("Worker %d done\n", id)
}

func main() {
    numWorkers := 5
    var wg sync.WaitGroup

    start := time.Now()

    // Add the total number of workers to WaitGroup
    wg.Add(numWorkers)
```

Recommended Clients

Pros

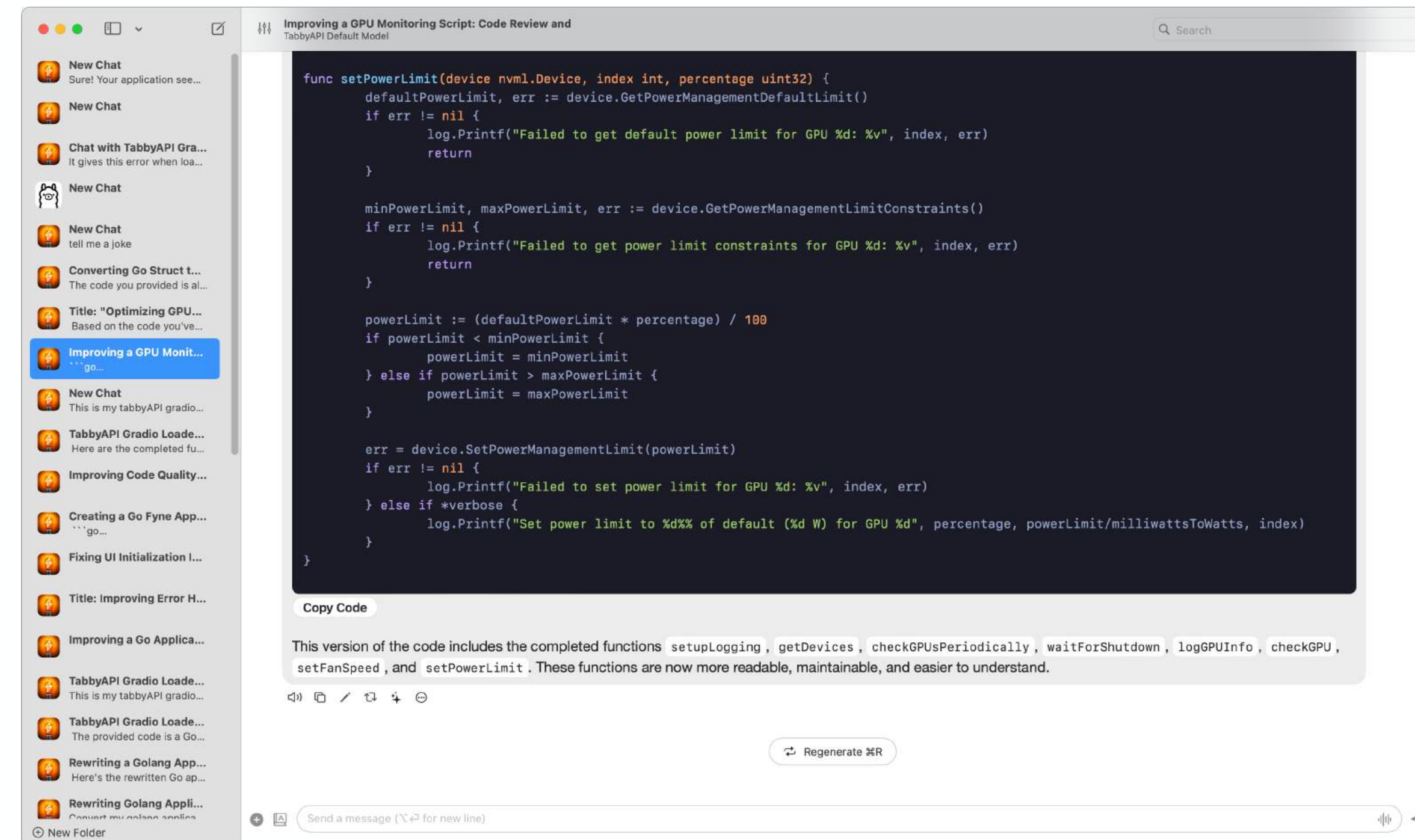
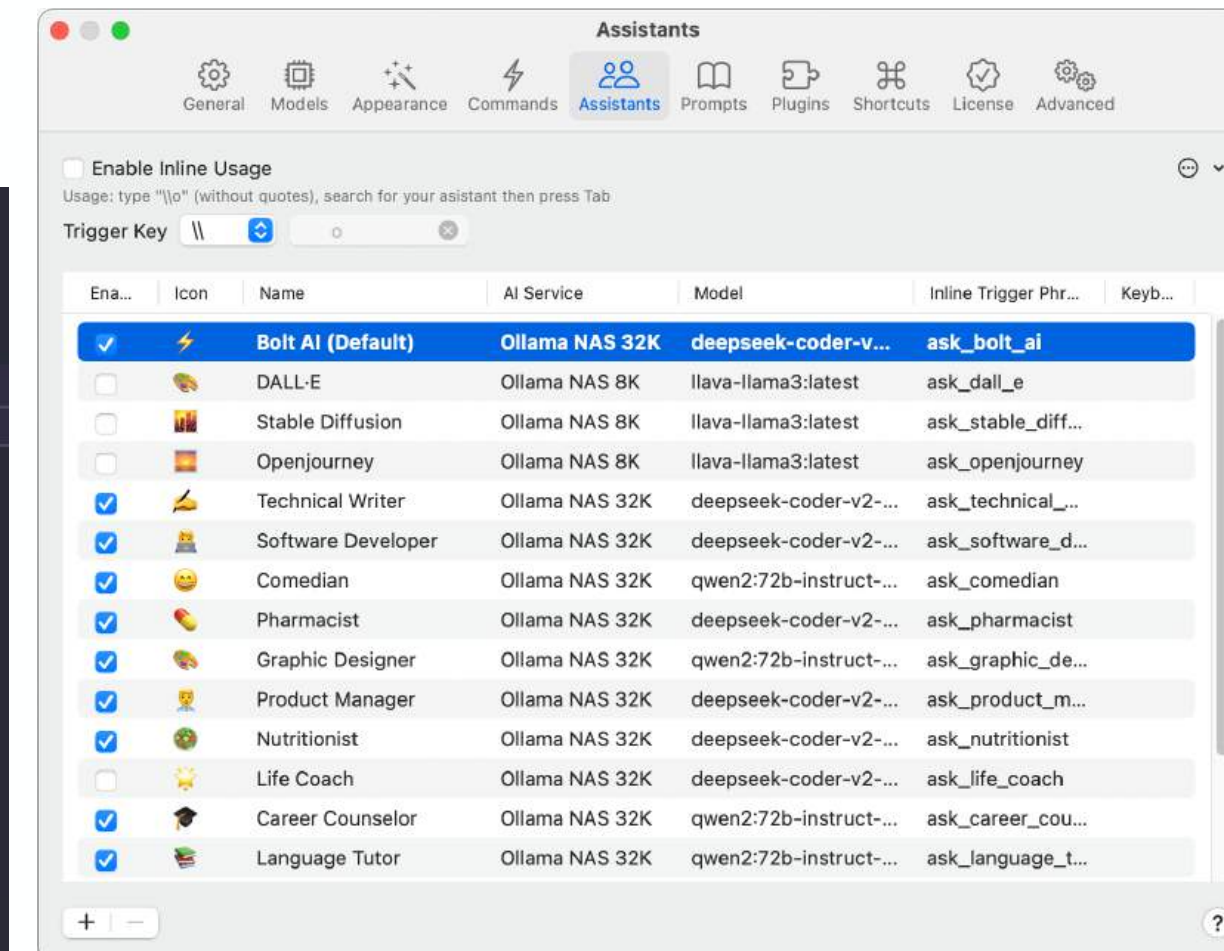
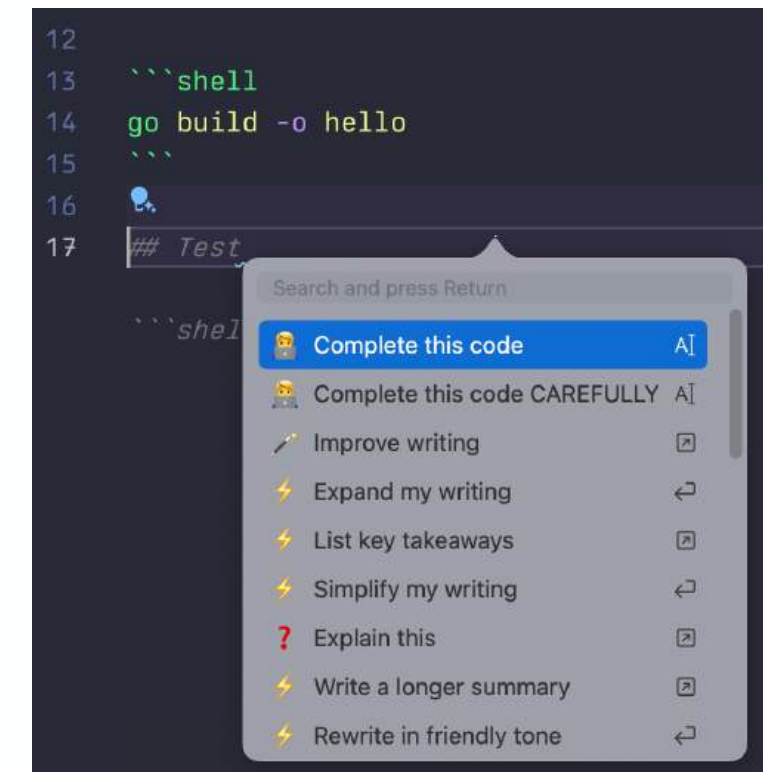
- Fast, native app
- OS integrations (automation, hotkeys, inference from and into any app)
- Templating of assistants, quick functions
- Web search/scraping
- Responsive, friendly (solo) developer
- Integrated text to speech and speech to text

Cons

- Limited data / tooling connectors
- Limited inference parameters
- Currently only supports OpenClosedAI for embeddings
- Closed Source
- Costs money (but worth it IMO)

<https://boltai.com>

Bolt

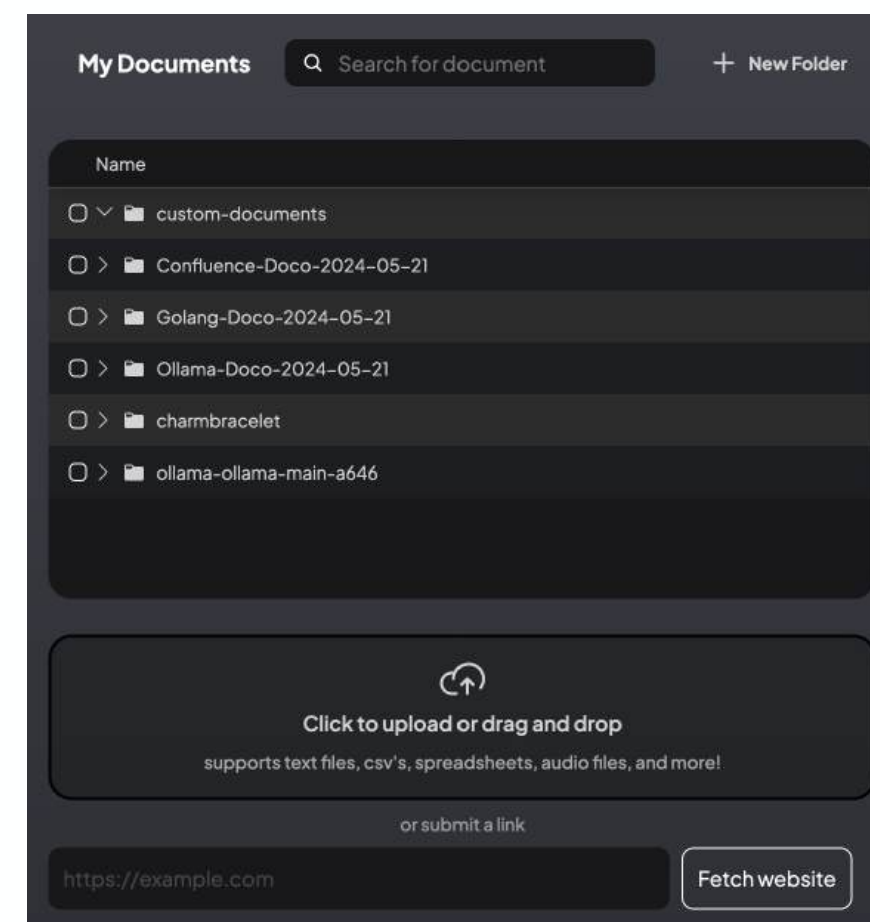


Recommended Clients

AnythingLLM

Pros

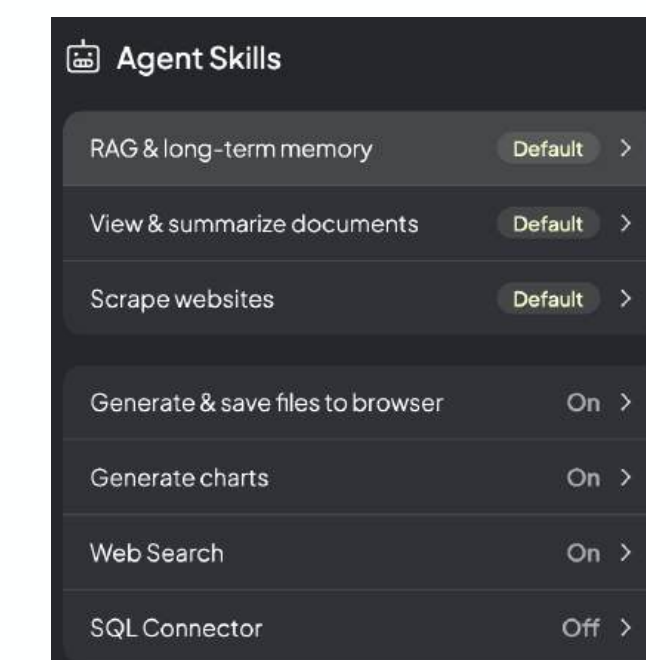
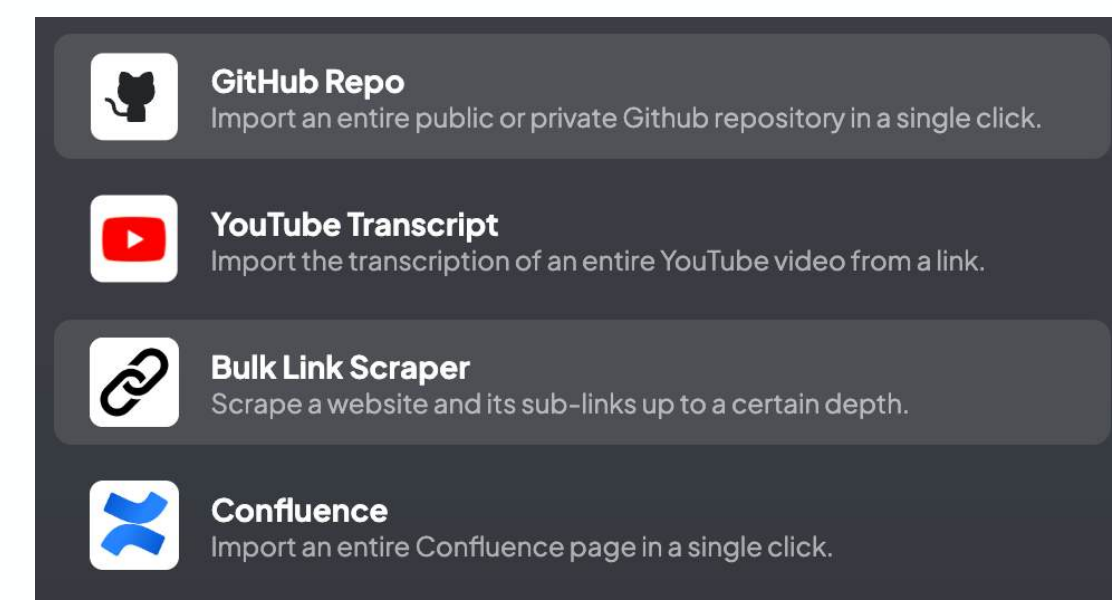
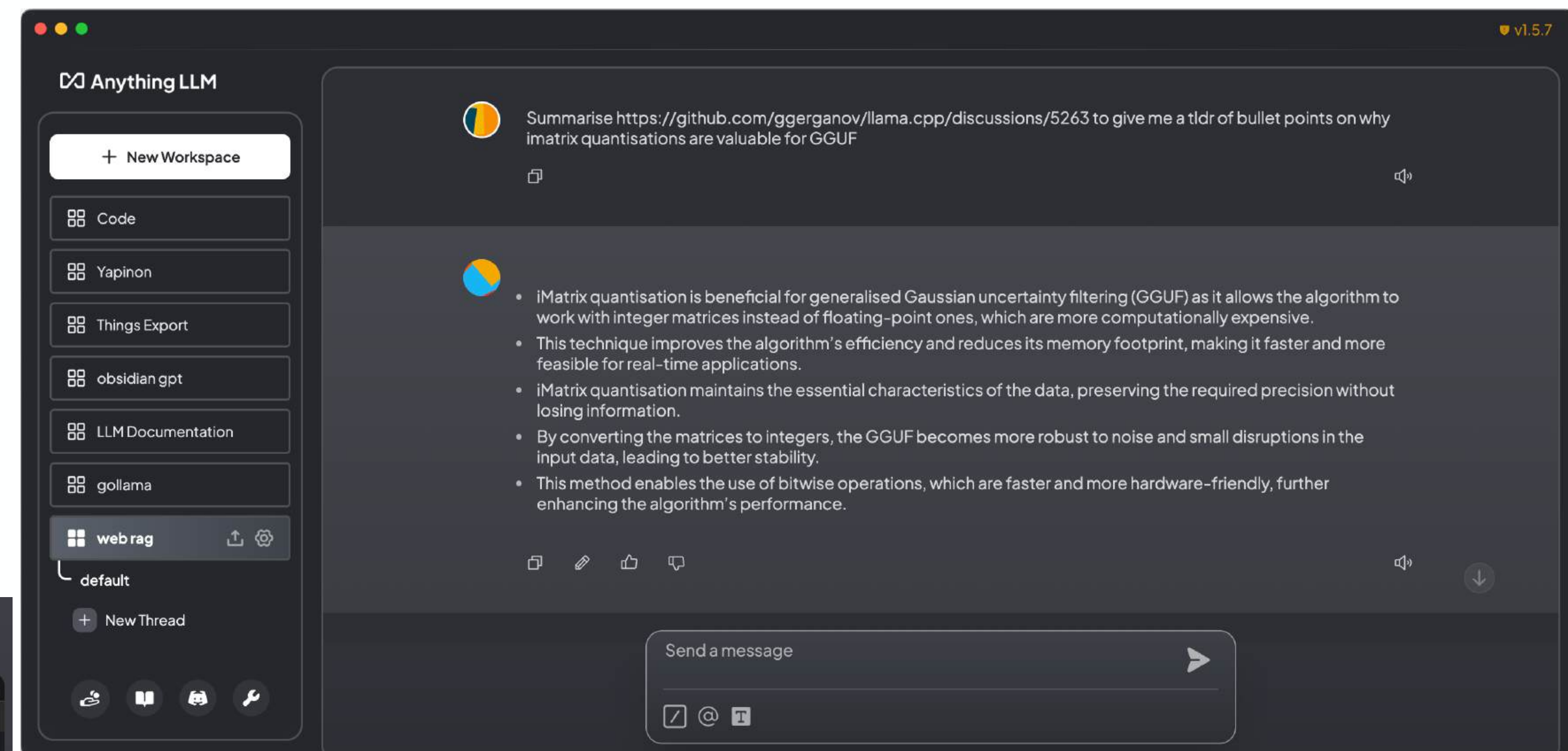
- Very capable with RAG and tooling
- Open Source
- Data connectors for Confluence, Github, Youtube, SQL, Web crawling
- Can run as a GUI app, or as a web interface
- Has an API server
- Supports many LLM backends
- Integrated text to speech



Cons

- *Seriously* ugly
- Limited inference parameters

<https://useanything.com>



Recommended Clients

Pros

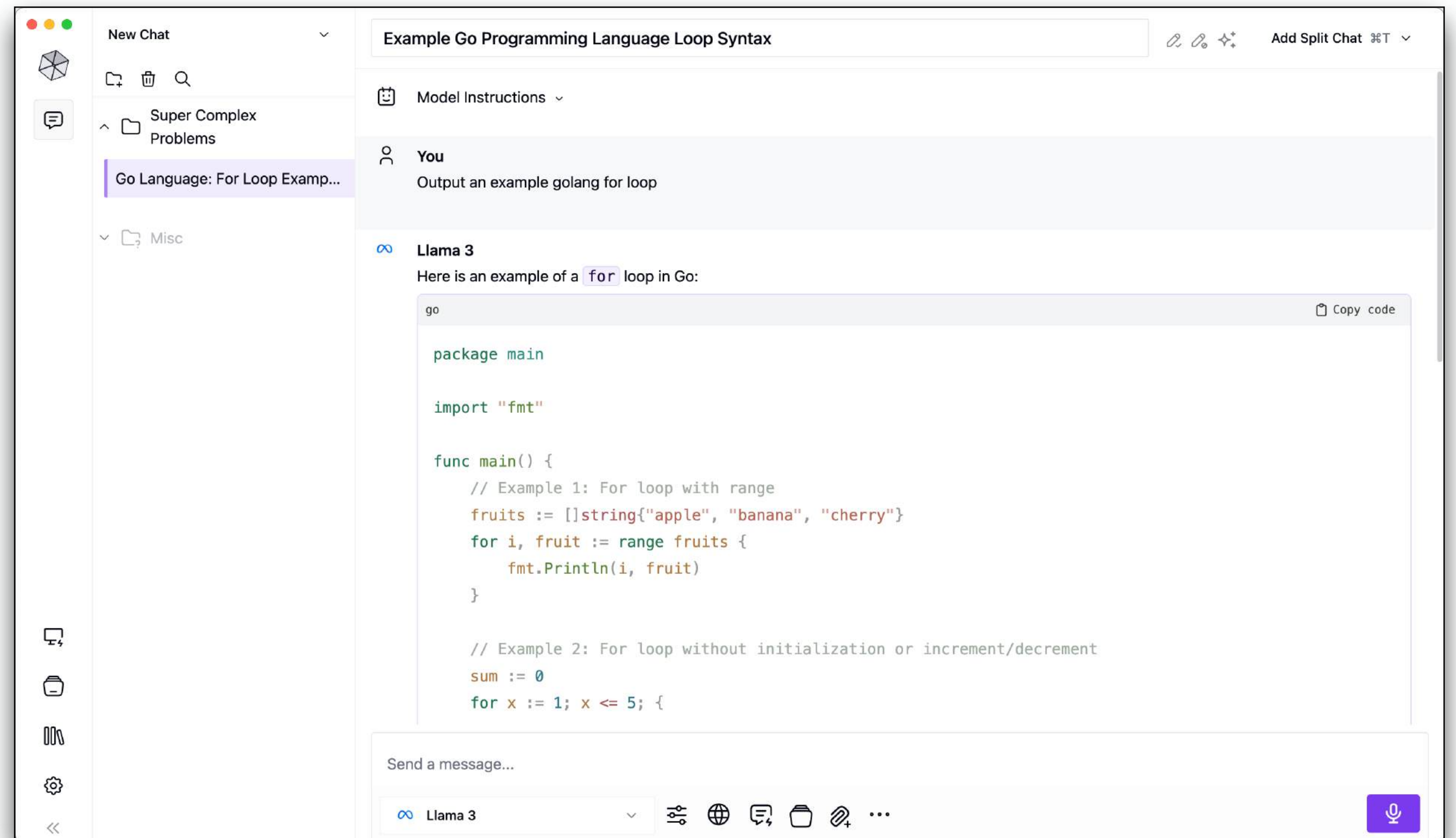
- Visually pleasing UI
- Conversation branching
- Data connectors for Confluence, Github, Youtube, SQL, Web crawling
- Supports a few different LLM backends
- Capable RAG with document collections
- Integrated text to speech
- Prompt library
- Multimodal
- Optional inbuilt llama.cpp server

Cons

- Limited inference parameter settings
- Not open source
- Commercial use requires a yearly license (but it's pretty affordable)

<https://msty.app>

Msty



Recommended Clients

LM Studio

Pros

- Very easy to use
- Easy to setup ("just works" out of the box)
- Uses llama.cpp for inference
- Inbuilt OpenAI compatible API
- Online model browser and downloader

Cons

- Not open source
- No plugins/extensions
- No RAG (yet)
- Slower release cycle means it sometimes lags behind llama.cpp features

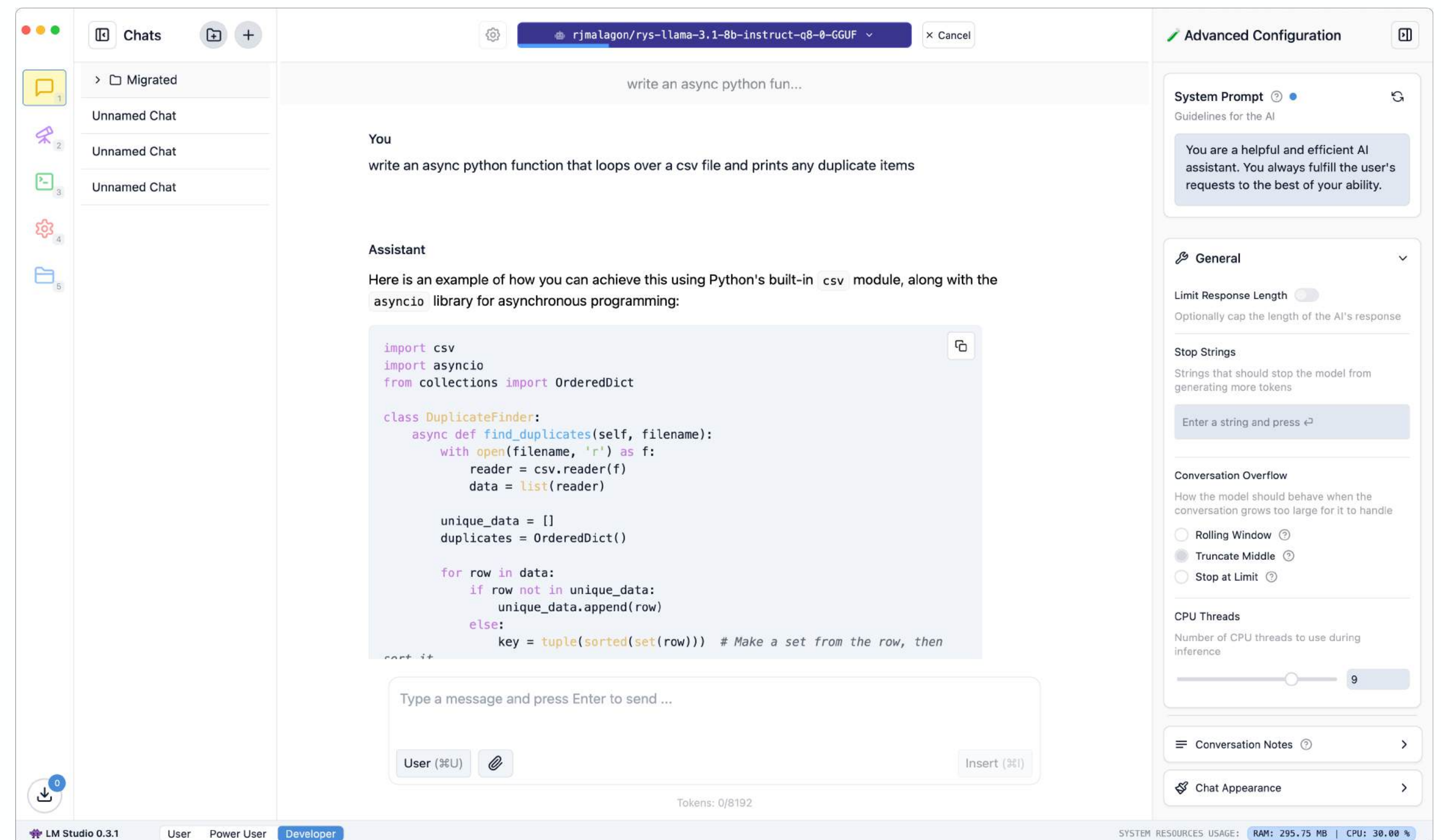


Image Generation / Editing

InvokeAI

I use InvokeAI for almost all image generation tasks

Pros

- Excellent UI / web interface
- Has a strong concept of workflow for both generation and editing
- Inbuilt model and adapter downloading
- Easy to install and run (locally, on a server, containerised etc...)
- Optional advanced node-based workflows
- Excellent inference performance
- Active and engaged development team
- Open source and runs on both Linux and macOS
- Easy to use

Cons

- Some common settings are buried under "advanced" option drop-downs

<https://www.invoke.com> | <https://github.com/invoke-ai/InvokeAI>







Choosing The Right Models

Model Formats, Context Sizes, Parameter Size and Quantisation




Pick the right model, know it's strengths

You wouldn't recommend a chef to do your plumbing

Know your task, do you need speciality skills?

-  Coding
-  General reasoning and logic
-  Tool use
-  Multimodal (audio/video/text)

Does your use case favour being served more by  or  ?

-  Wise but slower (reasoning, logic, translation, refactoring, bug fixing, documentation)
-  Lean and fast (runs locally with ease, IDE tab completion, fast feedback cycles)
-  Reasoning

Finding the right model

Models Are Improving Quickly

Newer is often (but not always) better

The rate of breakthroughs and general performance / quality improvements with LLMs is nothing short of astounding.

A model that is state of the art one month, might be quickly superseded by another that's faster, smaller, more resource efficient and with new features the next.

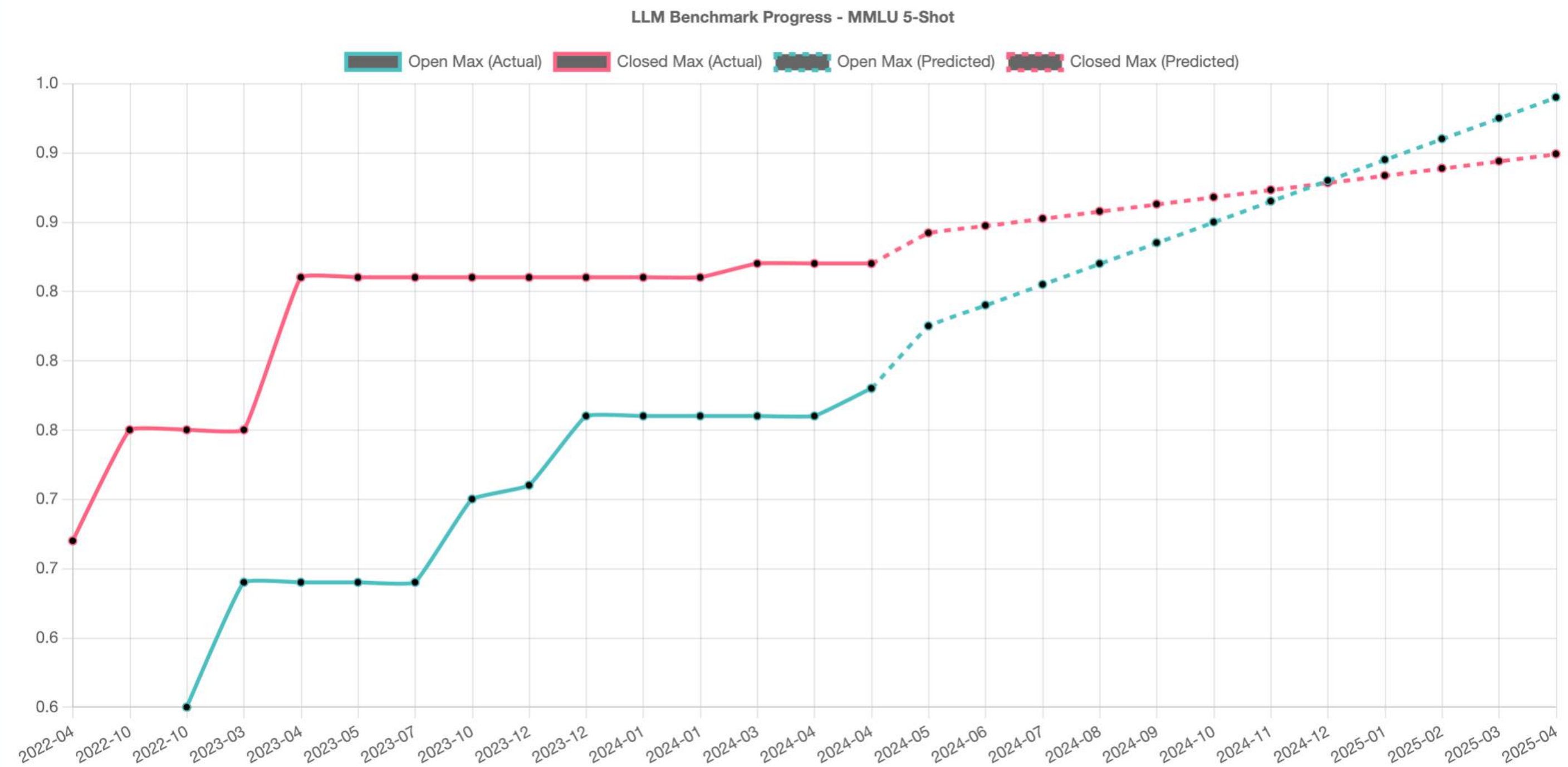
It is worth regularly questioning if there are better models available than what you're using.

Maintain a healthy scepticism of claims and benchmarks made in model releases, especially if the claimed results come from custom benchmarks, or if the data is not shared on a public leaderboards.

Closing the gap

Open LLMs have been quickly catching up to closed providers in a number of areas, this is especially relevant if you have focused tasks that benefit from specialised or specific skills.

Open vs Closed LLMs - Closing the gap

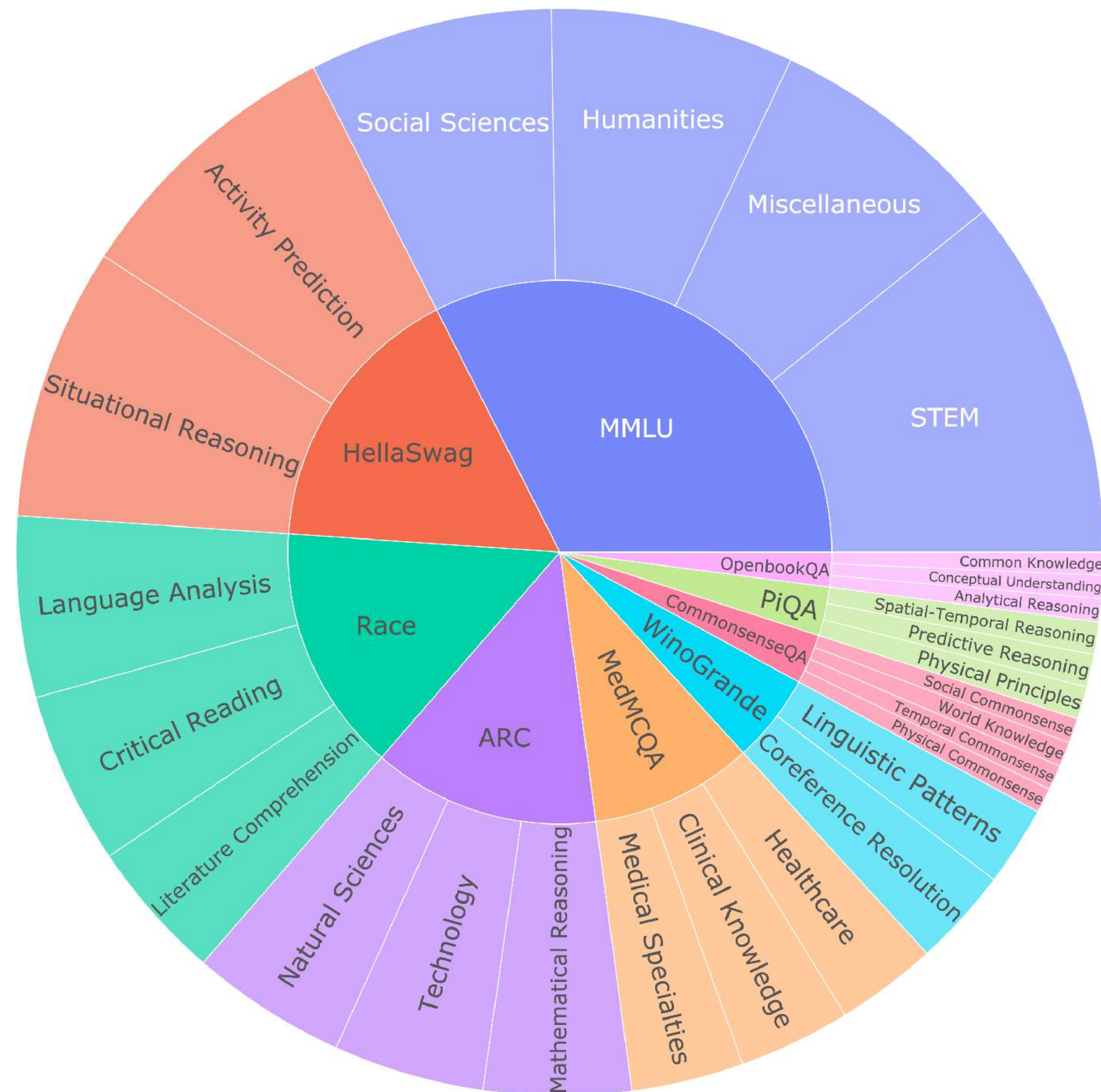


<https://github.com/sammcj/closing-the-gap>

Finding the right model

Check relevant leaderboards for models with strong domain specific capabilities

But keep in mind that benchmarks do not tell the whole story



- HF's Open LLM LB - ([Link](#))
 - Probably the best general all rounder
- BigCode's BigCodeBench LB - ([Link](#))
- Can AI Code LB - ([Link](#))
 - Very good, sometimes takes a while to get the latest models
- EvalPlus LB - ([Link](#))
 - Good, make sure to select "Average"
- LiveCodeBench - ([Link](#))
- EQ-Bench LB - ([Link](#))
- Aider Code Editing LB - ([Link](#))
- MTBE (Embedding Benchmark) LB - ([Link](#))
- HF Big Benchmarks Collection - ([Link](#))
- Berkley Function Calling LB - ([Link](#))
- CyberSecEval - ([Link](#))
- OpenVLM (vision) LB - ([Link](#))
- QBench (vision) - ([Link](#))
- HHEM Hallucinations LB - ([Link](#))
- LMSys Chatbot Area LB - ([Link](#))
- ZebraLogicBench - ([Link](#))
- LiveBench - ([Link](#))

Note:

Many benchmarks are heavily criticised for GPT trained bias, favouring closed source models and often missing fine tuned models. Benchmarks like MMLU, while useful, contain an lot of questions (tests) that aren't truly relevant to how people use LLMs.

Consider checking what people are saying on [r/locallama](#)
I find it to be the best local LLM community and news source!

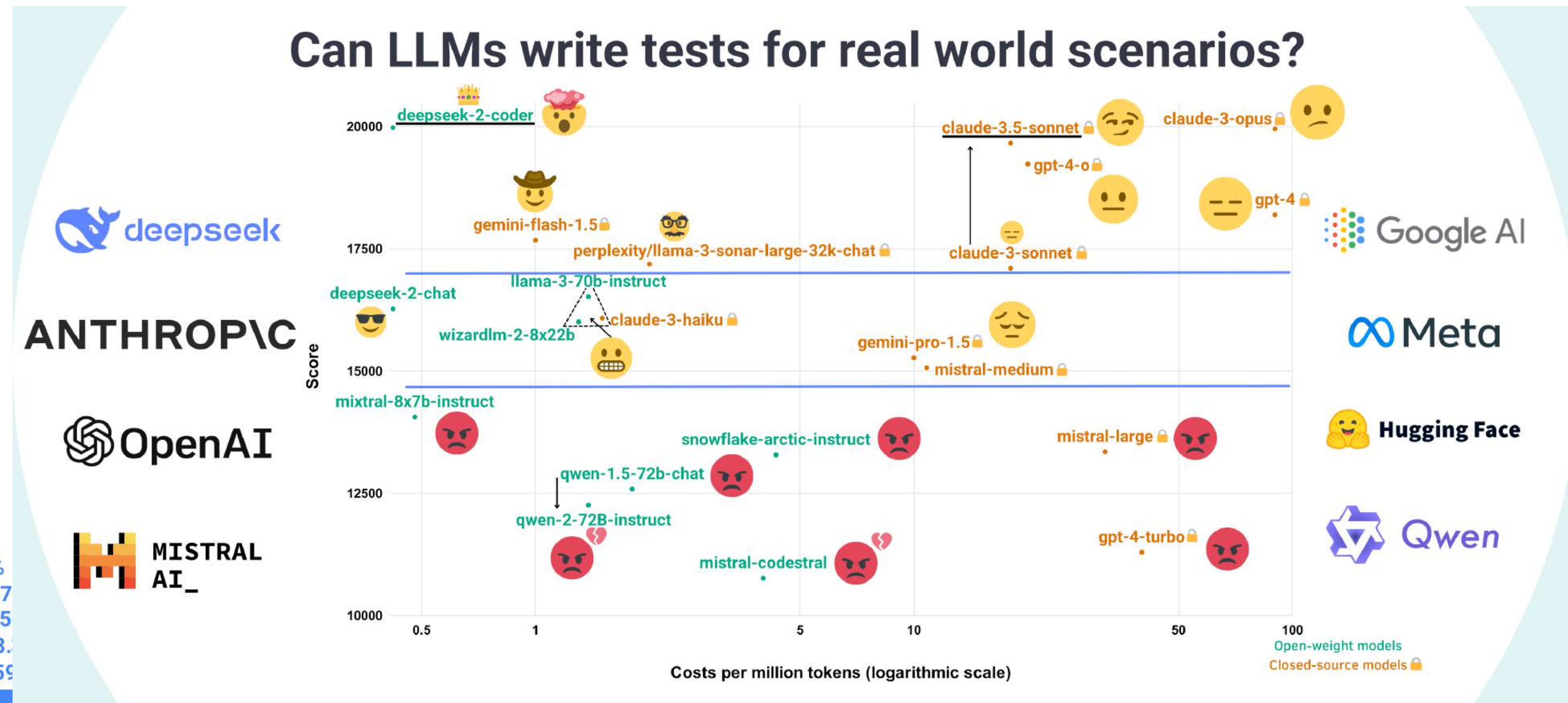
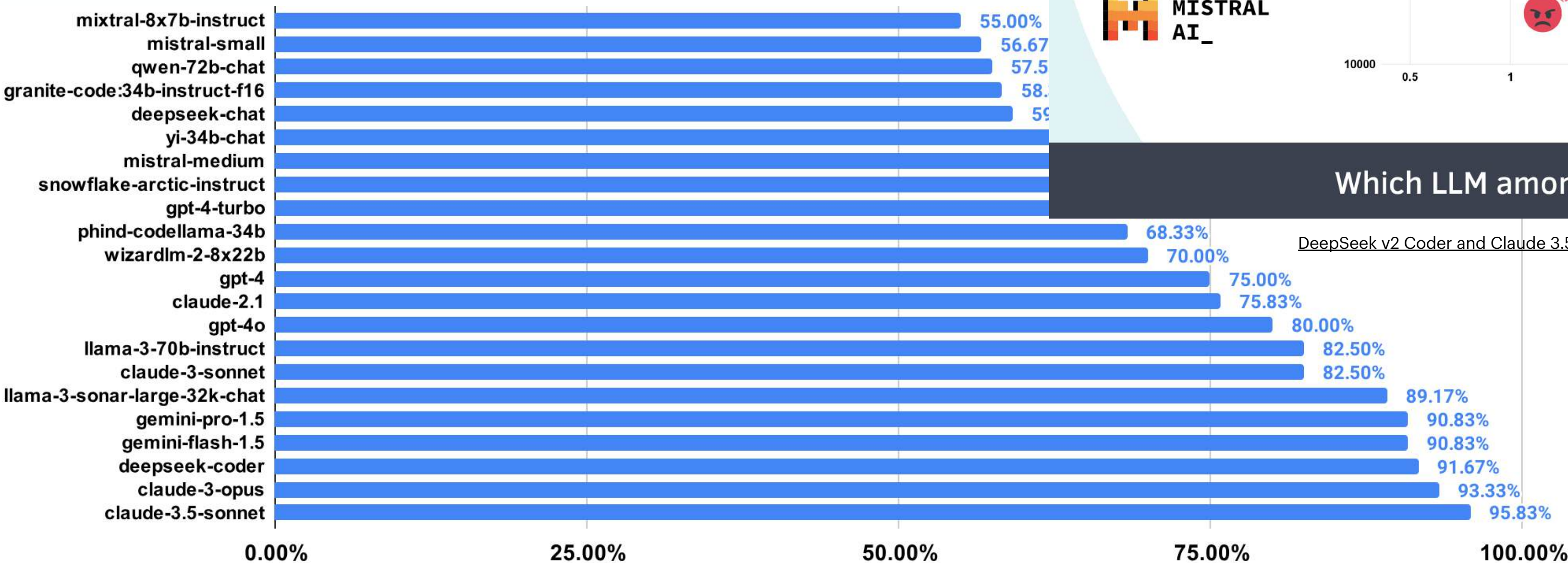
Finding the right model

Cost considerations

Performance benchmarks often don't take the cost of inference into account

Taking into account the cost of API / hardware (if any) vs the model performance can greatly impact which models or model services might be fit for your needs

- Do you already have hardware at your disposal that could be put to use?
- Whats the latency, throughput and concurrency?
- Is there a rate limit and what happens when you're rate limited?
- Power consumption (costs for self-hosted inference are generally lower than people assume)
- The cost in risk of sending data to a third party



Which LLM among +180 can write tests for Java and Go?

DevQualityEval v0.5.0

DeepSeek v2 Coder and Claude 3.5 Sonnet are more cost-effective at code generation than GPT-4o! (Deep dives from the DevQualityEval v0.5.0) <https://github.com/symflower/eval-dev-quality>

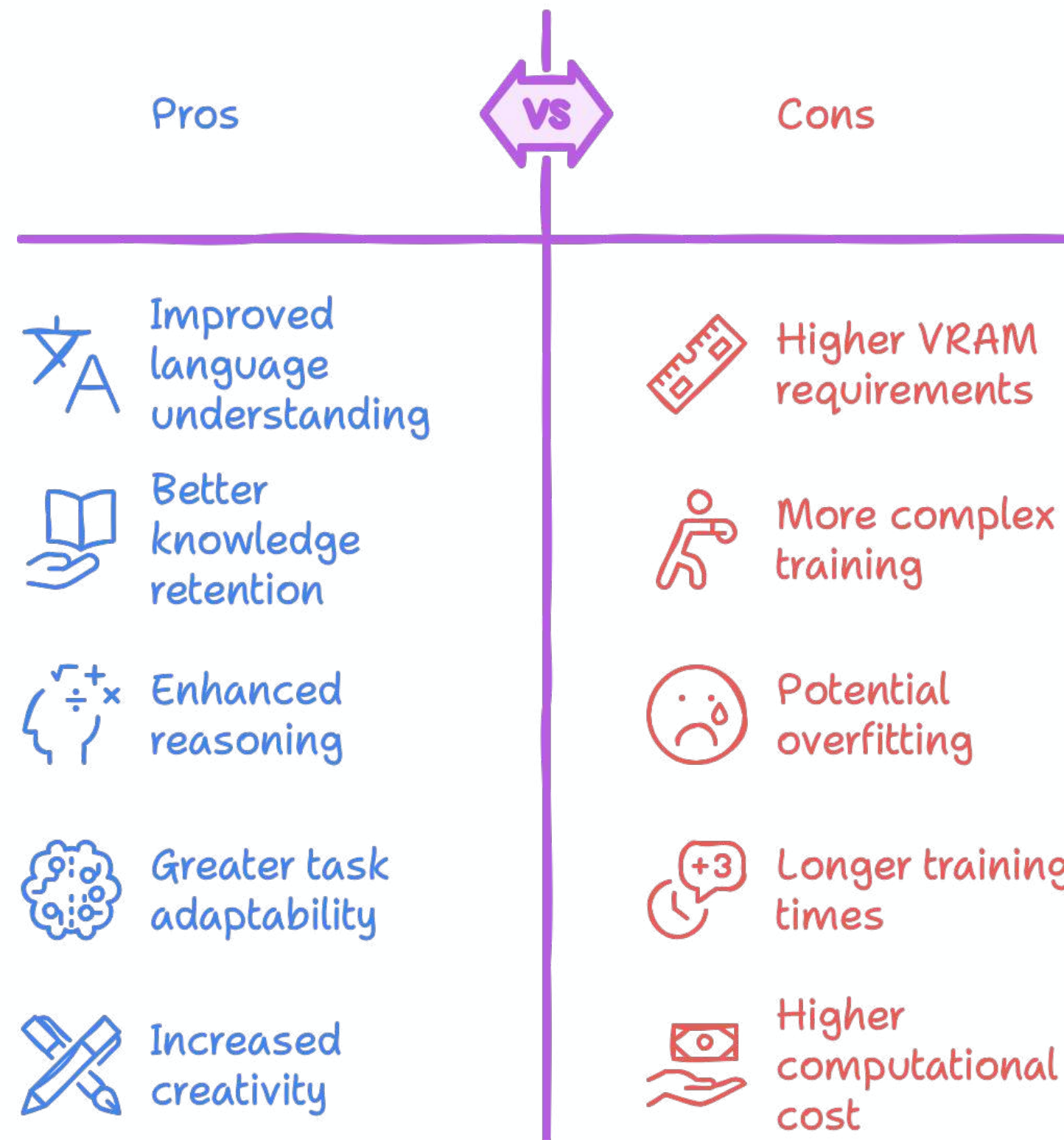


GPU metrics exposed via <https://github.com/sammcj/nvapi>

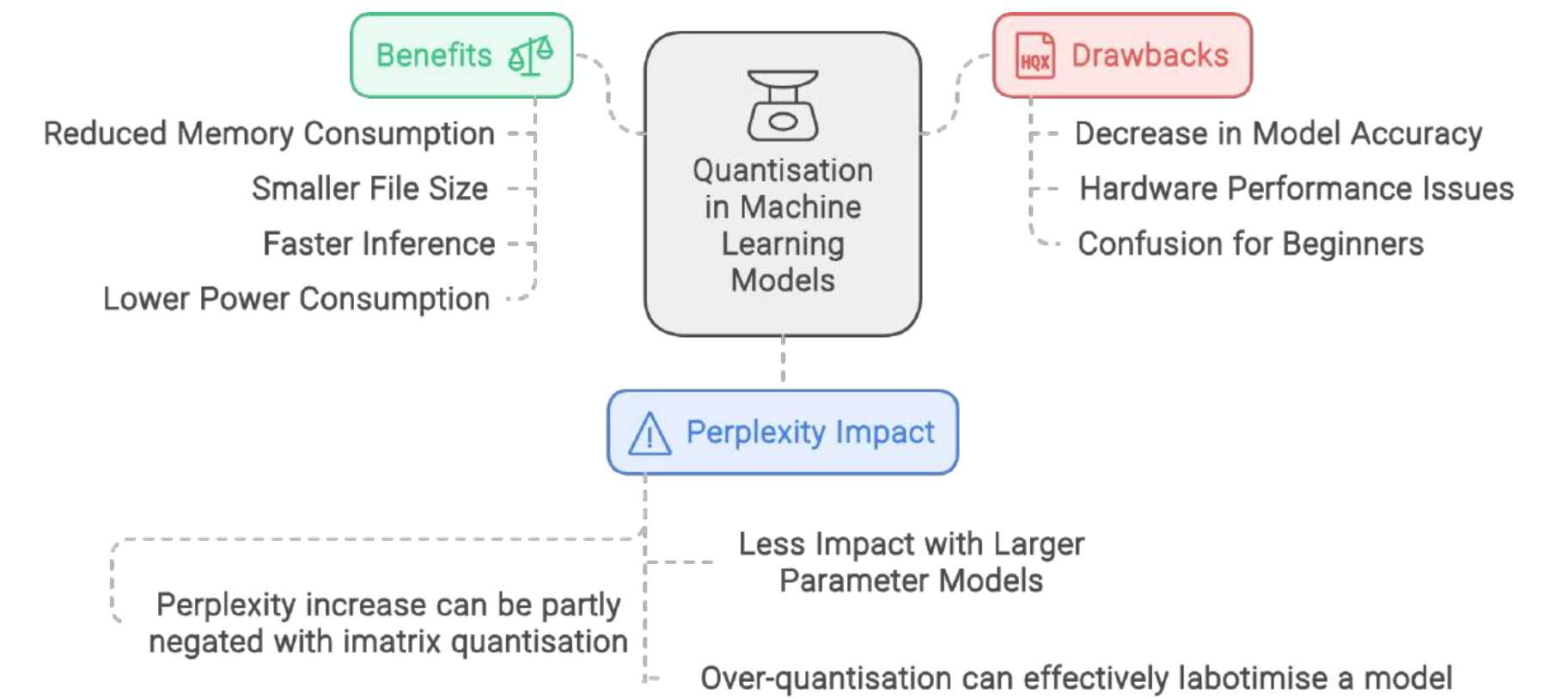
Finding the right model

Parameter Size

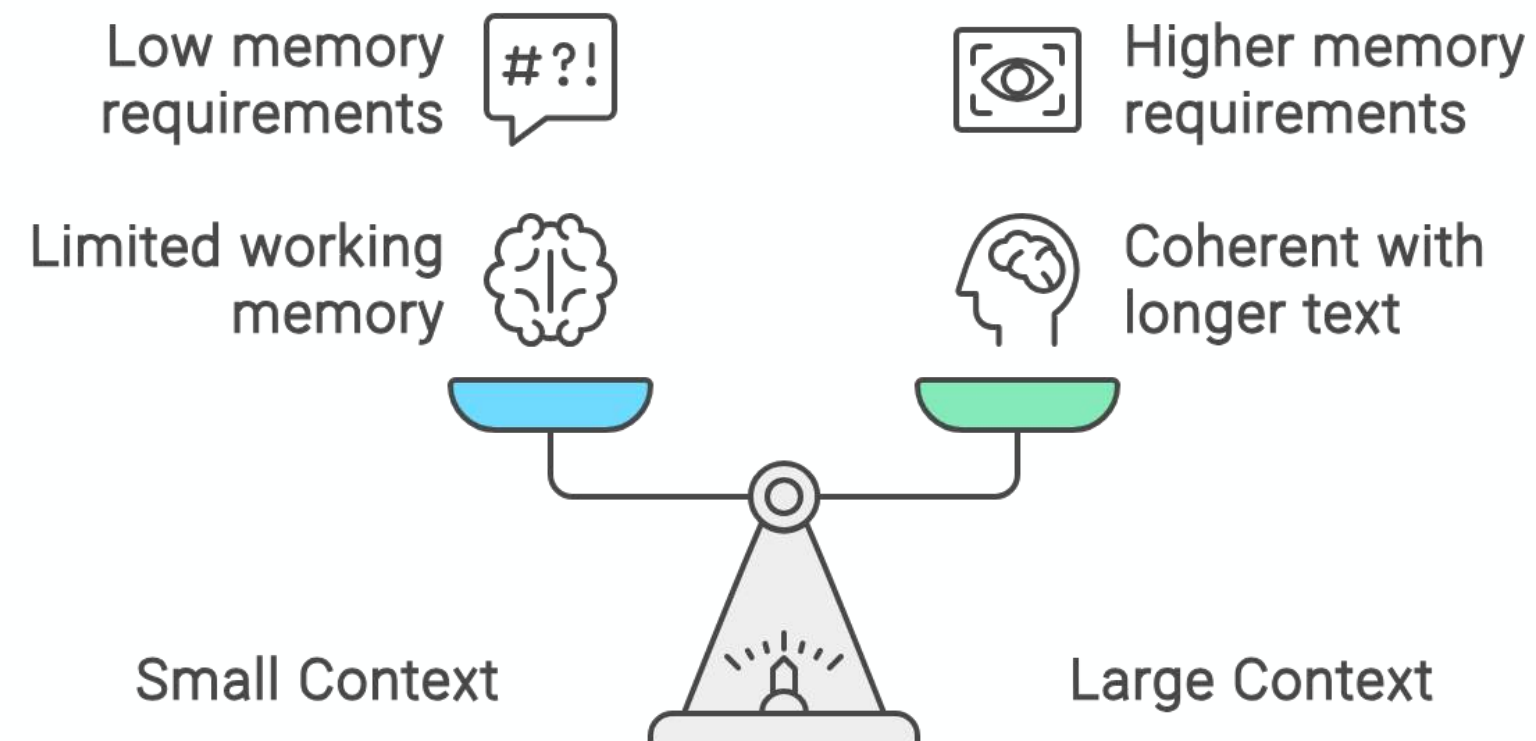
Larger parameter sizes in LLMs

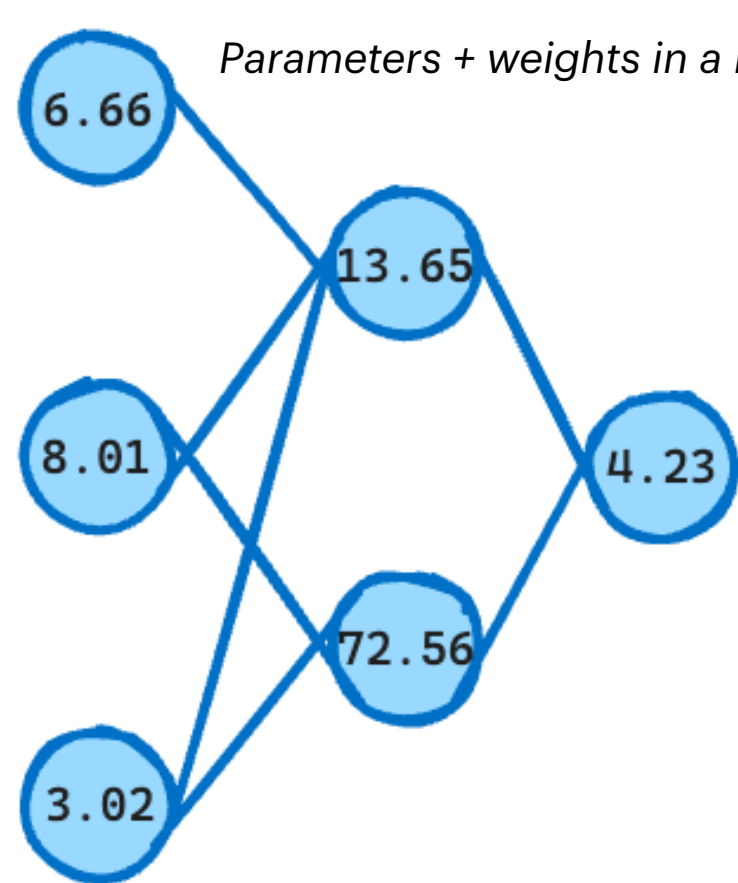


Quantisation



Context Size





Finding the right model

Parameter Size

Parameters

E.g. llama-**8b**, qwen2-**110b** etc...

- Parameters are numerical values that define the strength of connections between neurons in a neural network.
- In LLMs, they capture patterns, relationships, and knowledge from the training data.
- They're adjusted during training to minimise the difference between the model's predictions and actual outcomes.
- More parameters = larger model, e.g:
 - Unquantised (FP16/FP32): 1B parameters ≈ 4GB VRAM
 - Quantised: 1B parameters ≈ 0.5 - 1.5GB VRAM

Models with larger parameter tend to provide improved:

- Language understanding: Grasp context, nuance, and meaning in text.
- Knowledge retention: Store and recall factual information from training data.
- Reasoning: Make logical inferences and connections between concepts.
- Task adaptability: Apply learned knowledge to various tasks (e.g., translation, summarisation).
- Creativity: Generate novel ideas or content based on learned patterns.

LLM Recipe Book

Recipe: Generate Text

- 500 ml of context vectors (20.5, 15.3, ...)
- 15 g of grammar rules (0.8, 1.2, 0.5)
- 85 g of vocabulary embeddings (45.1, 22.7, ...)
- 1 g of style transfer (0.2, 0.3, 0.1)
- 10 ml of tone adjustment (1.5, 0.7, 1.1)

Each boxed ingredient represents a group of parameters.
Numbers in parentheses are example parameter values.

Neural Network

Parameters as Recipe Ingredients:

Each boxed ingredient and its measurement represents a group of parameters in the LLM. The neural network "mixes" these to produce the final output.

Generated Text

The result of combining all parameters (ingredients) through the neural network

You can have lots of ingredients and detailed instructions - if your ingredients are spoiled = your meal will be bad

If the recipe information (training data) is incorrect or of poor quality, the output (generated text) will also be of poor quality.

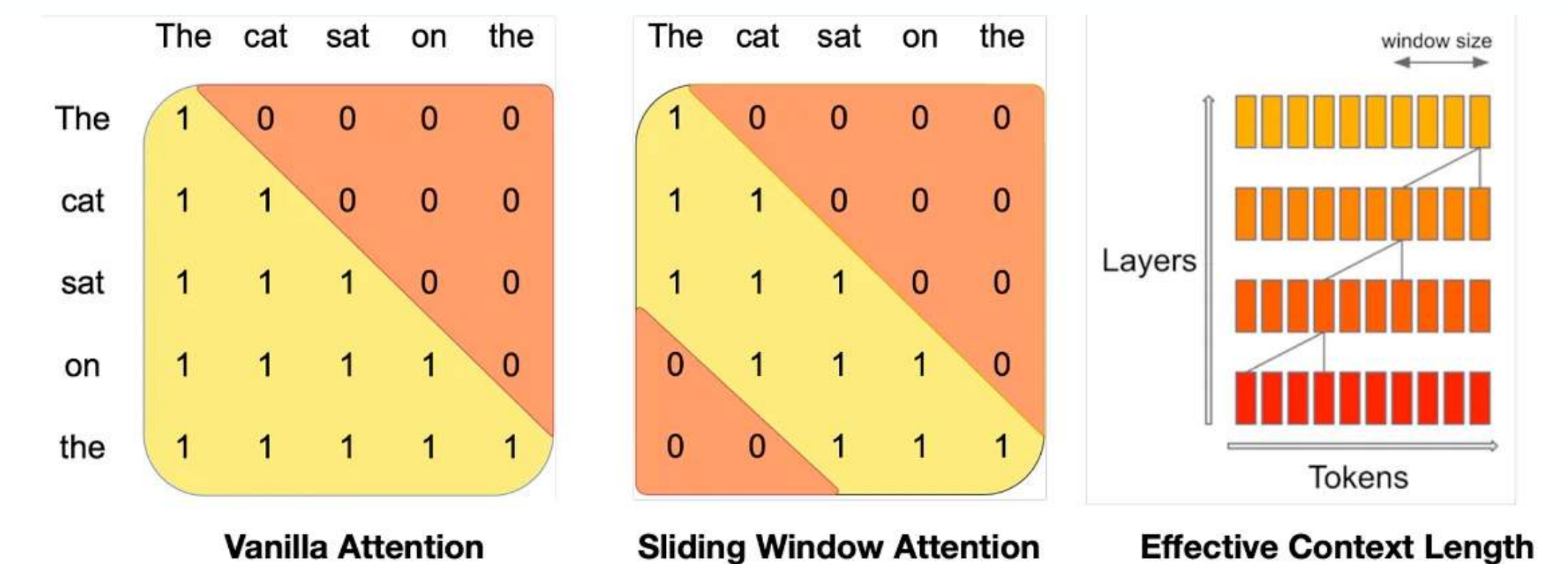
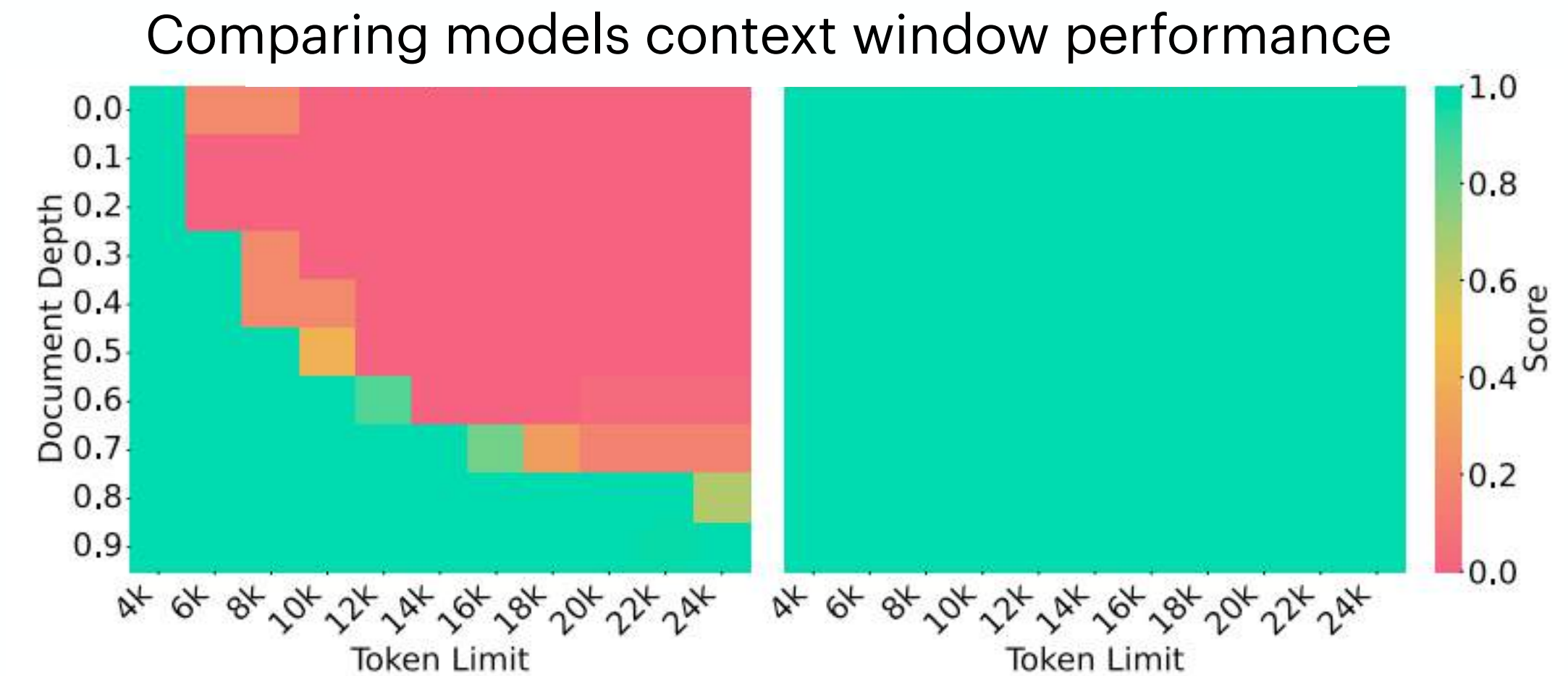
Finding the right model

Model Context Size

Context Size

- Think of the context like working memory
- Small context sizes (under 16K) are fine for tasks that don't need a lot of working memory but will give you gibberish beyond the limits
- For loading in codebases I aim for models with a least 32K context, ideally 64K*
- Make sure the model doesn't have a sliding window attention (SWA) that's smaller than the context you need.
 - With SWA working knowledge may become truncated without you knowing.
(e.g. Gemma 2 is "8K" context, but has a 4K sliding window)

**I often run them at 32-40K for performance*



Finding the right model

Model Context Size

Finding a models context size

Huggingface

- In a model's GGUF metadata label on HF
- In the model's config.json (sometimes)

Ollama

- In the model info on ollama.com/library (sometimes)
- Note: Ollama defaults to creating models with just a 2K context limit so you cannot rely on the ollama show alone

CLI

- For safetensors: In the model's config.json
- For GGUF: using gguf-dump from llama.cpp

Huggingface GGUF metadata

| File | Value |
|--------------------------------|--------------------|
| Qwen2-72B-Instruct-Q4_K_M.gguf | |
| Metadata | |
| version | 3 |
| tensor_count | 963 |
| kv_count | 25 |
| general.architecture | qwen2 |
| general.name | Qwen2-72B-Instruct |
| general.file_type | 15 |
| general.quantization_version | 2 |
| qwen2.block_count | 80 |
| qwen2.context_length | 32768 |
| qwen2.embedding_length | 8192 |
| qwen2.feed_forward_length | 29568 |
| qwen2.attention.head_count | 64 |
| awen2.attention.head count kv | 8 |

Hugging Face

Model's config.json

```

1 {
2   "architectures": [
3     "Gemma2ForCausalLM"
4   ],
5   "attention_bias": false,
6   "attention_dropout": 0.0,
7   "attn_logit_softcapping": 50.0,
8   "bos_token_id": 2,
9   "cache_implementation": "hybrid",
10  "eos_token_id": 1,
11  "final_logit_softcapping": 30.0,
12  "head_dim": 128,
13  "hidden_act": "gelu_pytorch_tanh",
14  "hidden_activation": "gelu_pytorch_tanh",
15  "hidden_size": 4608,
16  "initializer_range": 0.02,
17  "intermediate_size": 36864,
18  "max_position_embeddings": 8192,
19  "model_type": "gemma2",
20  "num_attention_heads": 32,
21  "num_hidden_layers": 46,
22  "num_key_value_heads": 16,
23  "pad_token_id": 0,
24  "query_pre_attn_scaler": 144,
25  "rms_norm_eps": 1e-06,
26  "rope_theta": 10000.0,
27  "sliding_window": 4096,
28  "sliding_window_size": 4096,
29  "torch_dtype": "bfloat16",
30  "transformers_version": "4.42.0.dev0",
31  "use_cache": true,
32  "vocab_size": 256000,
33  "_attn_implementation": "eager"
34 }
35

```

Model info on ollama.com (also in model's README.md)

ollama run qwen2

| Model | arch | qwen2 | parameters | 7.62B | quantization | 04_0 | 4.4GB |
|----------|---|-------|------------|-------|--------------|------|-------|
| params | {"stop": ["< im_start >", "< im_end >"]} | | | | | | |
| template | {% if .System %}< im_start >system {% .System %}< im_end > _ | | | | | | |
| license | Apache License Version 2.0, January 2004 http://www.apache... | | | | | | |

Readme

Qwen2

Qwen2 is trained on data in 29 languages, including English and Chinese.

It is available in 4 parameter sizes: 0.5B, 1.5B, 7B, 72B.

In the 7B and 72B models, context length has been extended to 128k tokens.

| Models | Qwen2-0.5B | Qwen2-1.5B | Qwen2-7B | Qwen2-72B |
|----------------|------------|------------|----------|-----------|
| Params | 0.49B | 1.54B | 7.07B | 72.71B |
| Non-Emb Params | 0.35B | 1.31B | 5.98B | 70.21B |
| GQA | True | True | True | True |
| Tie Embedding | True | True | False | False |
| Context Length | 32K | 32K | 128K | 128K |

Ollama's show command (shows configured context limit)

```

> ollama show qwen2:72b-instruct-q3_K_S
Model
  arch          qwen2
  parameters    72.7B
  quantization  Q3 K S
  context length 32768
  embedding length 8192

Parameters
  stop "<|im_start|>"
  stop "<|im_end|>"

```

llama.cpp's gguf-dump

```

$ git clone --depth=1 https://github.com/ggerganov/llama.cpp.git
$ llama.cpp/gguf-py/scripts/gguf-dump.py Qwen2-72B-Instruct-v0.1.IQ4_XS.gguf \
  | grep context_length

INFO:gguf-dump:* Loading: Qwen2-72B-Instruct-v0.1.IQ4_XS.gguf
qwen2.context_length = 32768

```

Finding the right model

Model Formats

Which format is right for you? **as of July 2024*

GGUF

Models 154

gguf qwen2

Qwen/Qwen2-0.5B-Instruct-GGUF

Text Generation • Updated 27 days ago • ↓ 256k • ♥ 38

MazyarPanahi/Qwen2-1.5B-Instruct-GGUF

Text Generation • Updated Jun 7 • ↓ 121k • ♥ 7

MazyarPanahi/Qwen2-72B-Instruct-v0.1-GGUF

Text Generation • Updated 17 days ago • ↓ 21.7k • ♥ 1

- Formally known as GGML (which is the underlying format)
- Used by llama.cpp and all tools that make use of llama.cpp (e.g. Ollama, LM Studio)
- Most common quantised model format
- Runs on CPU, GPU and just about anywhere
- Rapid development and big community
- Quants look like "Q4_K_M, Q6_K, IQ3_XXS etc..."

Use GGUF if you can't run EXL2 - or simply for the tooling convenience

EXL2

Models 80

exl2 qwen2

bartowski/Qwen2-72B-Instruct-exl2

Text Generation • Updated Jun 7 • ↓ 44 • ♥ 2

LoneStriker/Qwen2-7B-Instruct-4.0bpw-h6-exl2

Text Generation • Updated Jun 7 • ↓ 24 • ♥ 1

LoneStriker/Qwen2-72B-Instruct-6.0bpw-h6-exl2

Text Generation • Updated Jun 7 • ↓ 19

- Used by exllamav2
- Highly efficient format with advanced KV/context caching and quantisation
- Probably the second most common
- Runs on GPU
- One of the highest performing formats
- Commonly used with TextGen WebUI, TabbyAPI, vLLM (Mistral.RS coming soon!)
- Quants look like "3_0bpw, 4_25bpw etc...")

Use EXL2 if you have an Nvidia GPU

Other Common Formats

Qwen/Qwen2-7B-Instruct

Text Generation • Updated Jun 7 • ↓ 202k • ♥ 387

Qwen/Qwen2-1.5B-Instruct-AWQ

Text Generation • Updated Jun 7 • ↓ 1.44k • ♥ 2

PrunaAI/Qwen-Qwen2-7B-Instruct-HQQ-4bit-smashed

Text Generation • Updated 2 days ago • ↓ 2

Qwen/Qwen2-57B-A14B-Instruct-GPTQ-Int4

Text Generation • Updated 27 days ago • ↓ 4.81k • ♥ 17

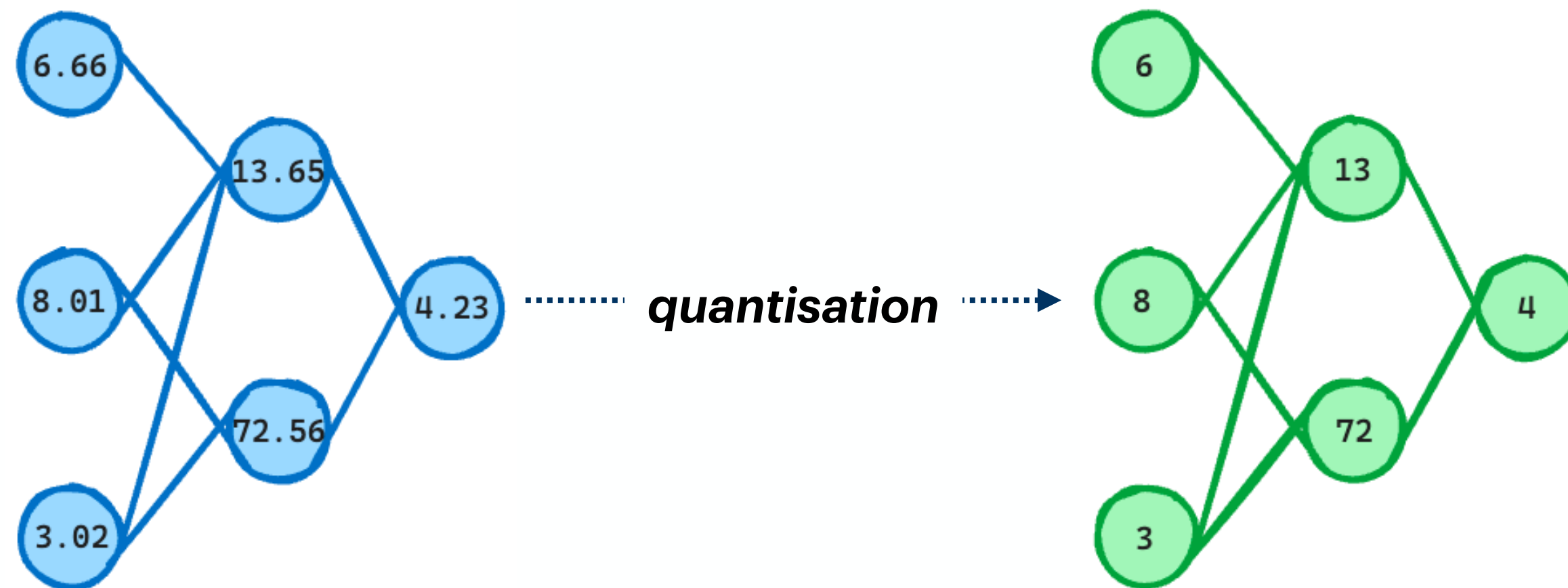
- AWQ
- GPTQ
- Safetensors
 - Unquantised
 - FP32
 - BF16
 - FP16
 - INT8
 - INT4
 - QLoRA
- HQQ

Quantisation & Model Formats

Finding the right model

Quantisation

Quantisation refers to the process of converting model weights from *higher to lower* precision data types (e.g. floating point -> integer)



Benefits

- Significantly reduced memory consumption
- Smaller file size
- Faster inference
- Lower power consumption

Drawbacks

- Decrease in model accuracy
- Some quant types perform poorly on certain hardware
- It can be confusing for people when they first start out and aren't sure which model to download

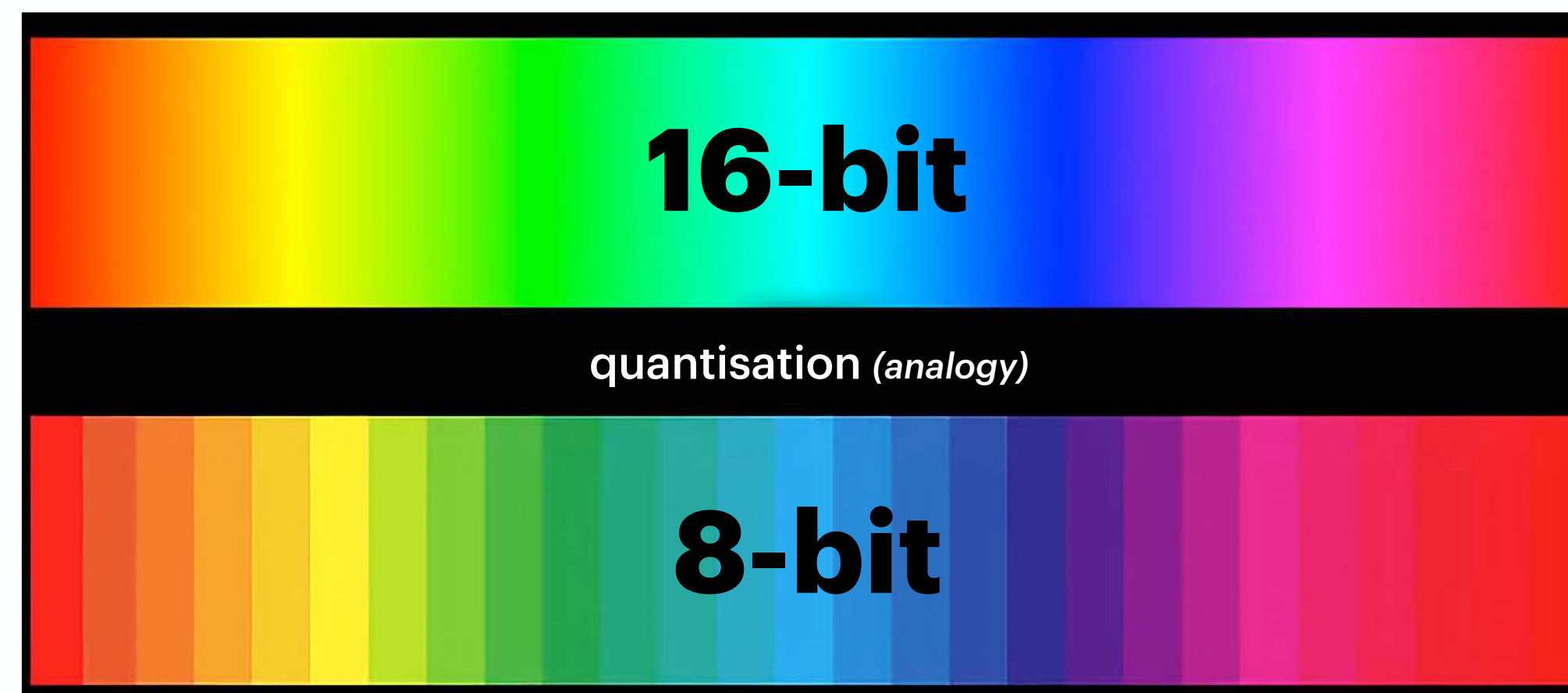
Colour Spectrum Analogy

Imagine the model data to be the colour spectrum (pictured as 16 bits here)

If we quantise the data to 8 bits we are removing (thus compressing) parts of the data based on a set of rules

We can still see a wide range of "colours" but we lose some of the detail

Note: This is a crude analogy, modern quantisation techniques have smarts that selective quantise parts of the model to varying degrees to reduce the loss



Finding the right model

Quantisation

Quantisation Types

Legacy 'non-k' Quants

Examples: Q4_0, Q5_1

- Less efficient compression
- Larger files
- Higher memory usage
- Lower quality output
- Mostly static quantisation - (e.g. all weights/layers are the same quantisation no matter their importance)



K Quants

Examples: Q4_K_M, Q5_K_L, Q6_K

Current/Previous Generation Format

Compared to legacy, non-K format:

- Improved compression
- Smaller files
- Lower memory usage (for the quality)
- Higher quality output
- Supports mixing quantisation levels so that the more important weights/layers can be higher quality
- Supports creation with iMatrix to prevent over-fitting



IQ Quants

Examples: IQ2_XXS, IQ3_M, IQ4_NL

Current Generation Format

Compared to K Quants:

- Slower inference on Metal and CPU - making IQ quants best suited to machines with Nvidia CUDA/AMD ROCm
- Even more efficient compression
- Even smaller files
- Even lower memory usage (for the quality)
- Even higher quality output - so much so that it makes lower quant sizes (e.g. IQ2_XS) on 70b+ models very usable

Don't use these

Best option for Metal, Good option for Nvidia/AMD

Best option for Nvidia/AMD

Quants created using an iMatrix

Examples (sometimes): <model>-iMat-Q4_K_M, <model>-i1-Q4_K_M

Improves the above quant types

iMatrix is a *method* of improving K and IQ quants, compared to quants created without iMatrix:

- Higher quality output, especially for smaller models
- Helps prevent over-fitting a model (where the model doesn't handle new/generalised use cases)
- Can confuse newcomers who think it's related to IQ quant types - it's a *method* not a *type* of quantisation
- As people don't always put i1 or iMat in the model title - it's not clear which quants are created with an iMatrix

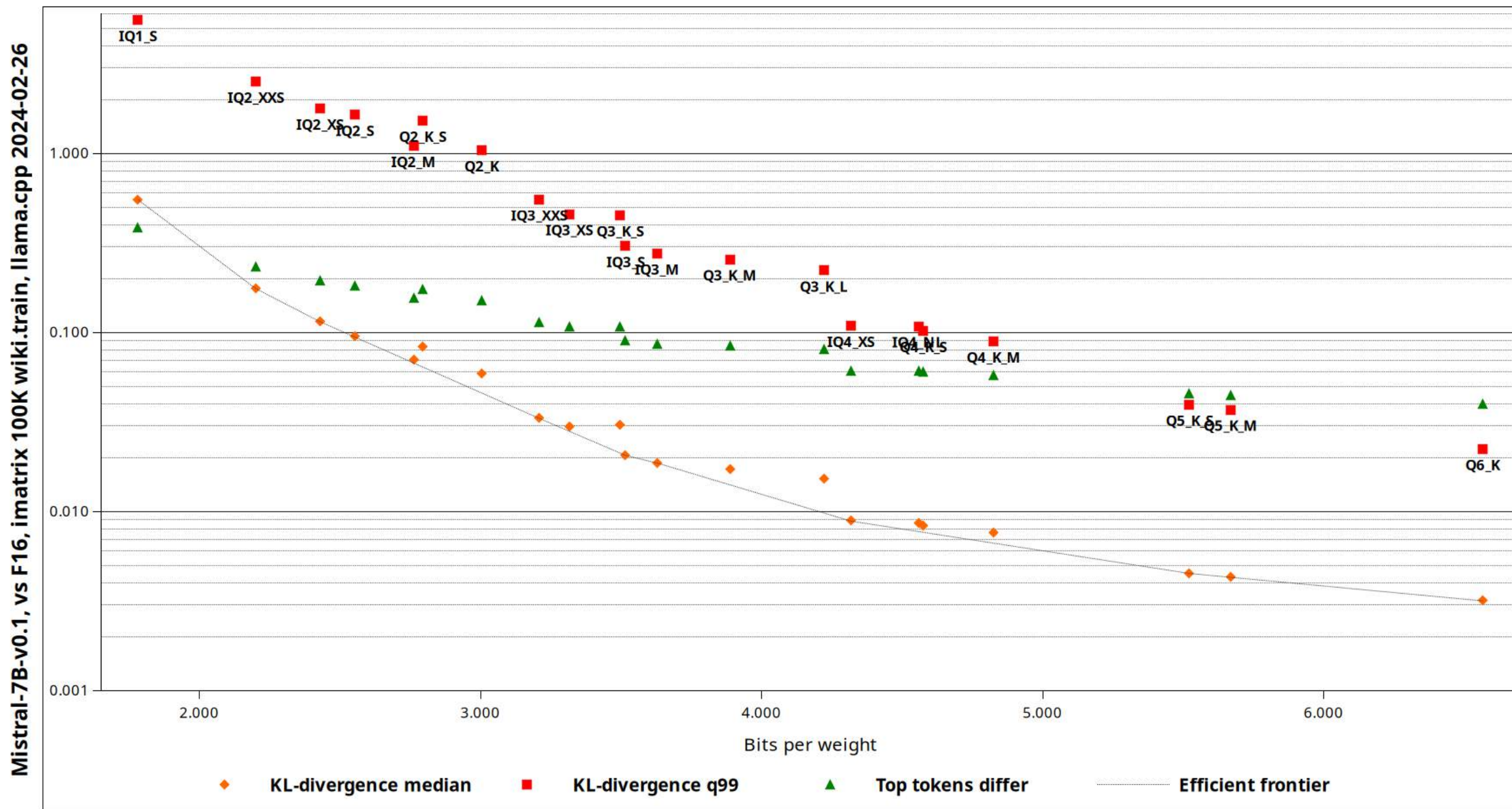
See also:

- gist.github.com/Artefact2/b5f810600771265fc1e39442288e8ec9
- <https://github.com/ggerganov/llama.cpp/discussions/5263>

Finding the right model

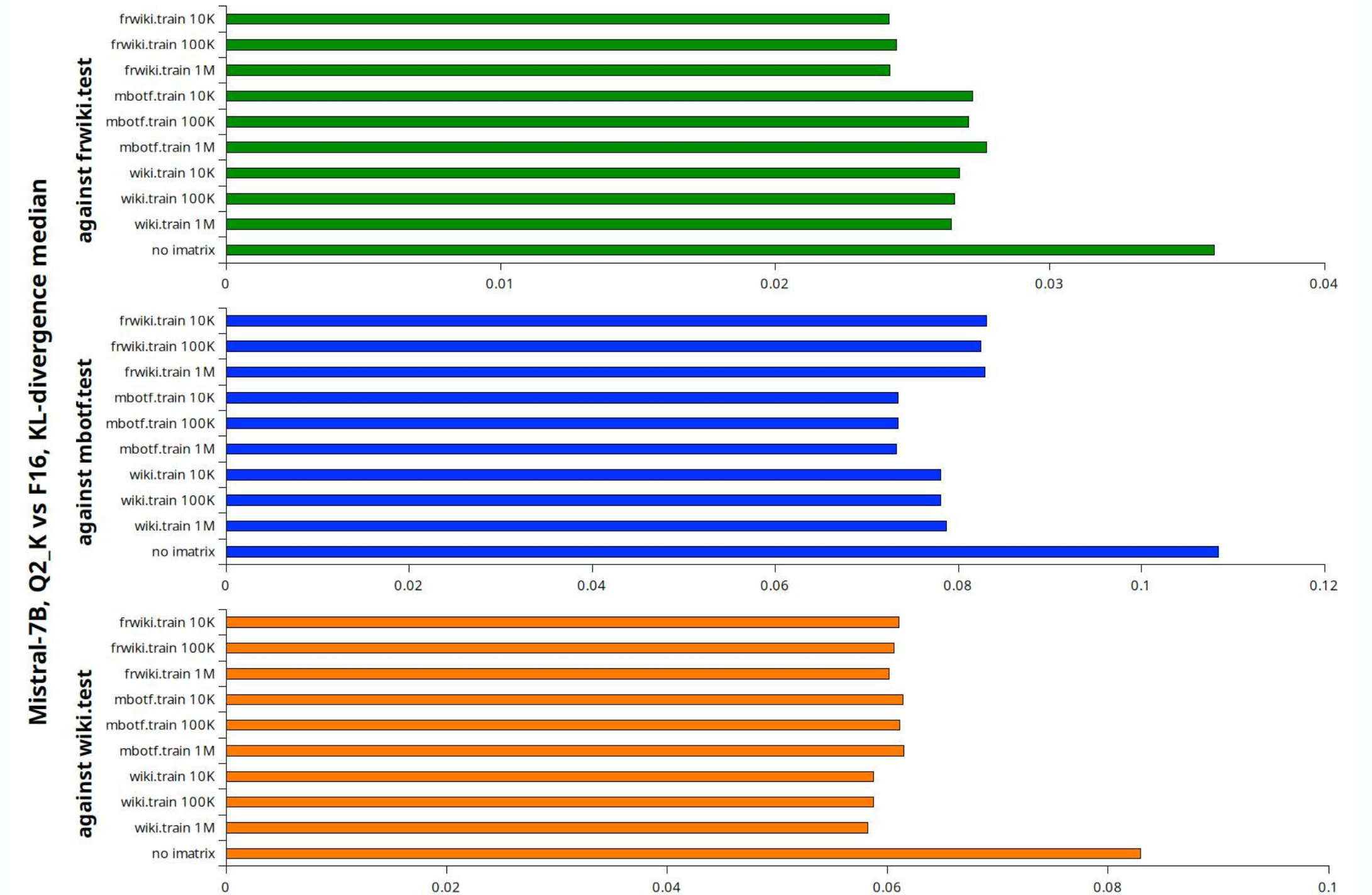
Quantisation

Quantisation Types
Size vs Quality



Lower = Higher Quality
 Left = Smaller Size
 Lower + Left = Better

With vs Without iMatrix

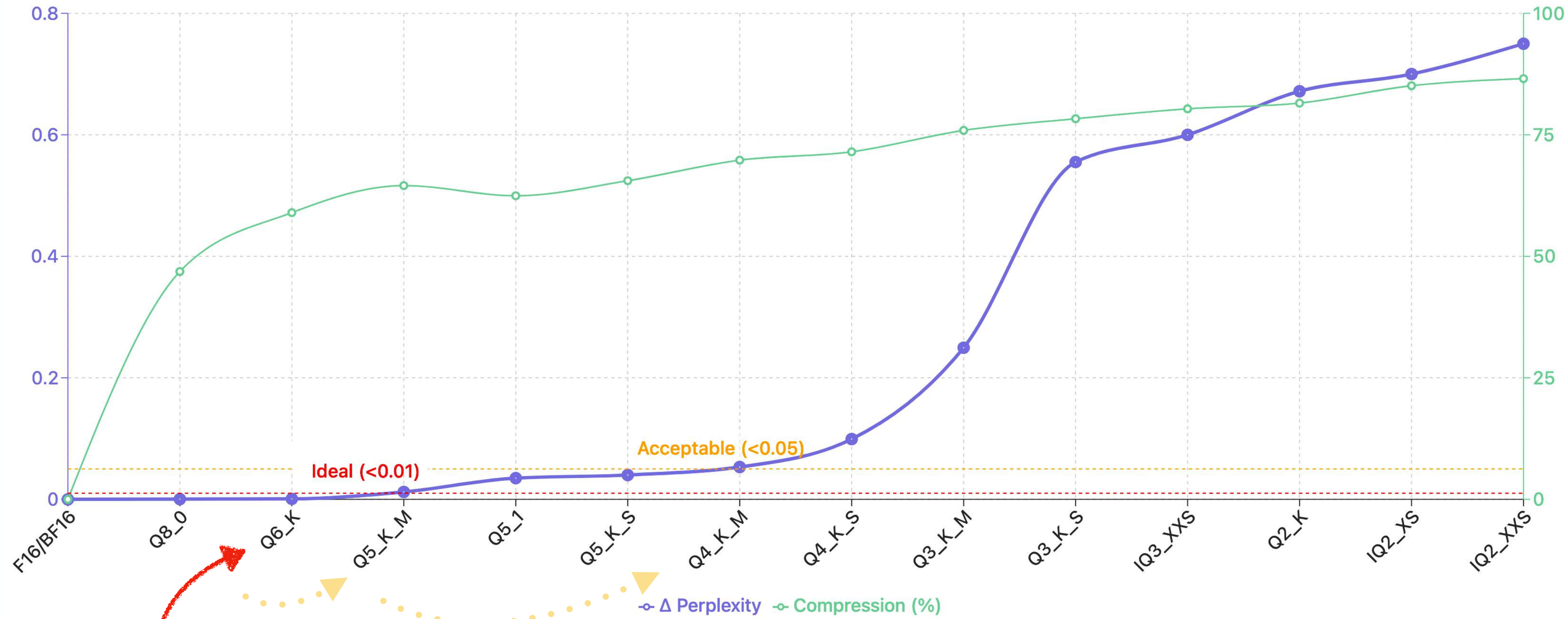


Lower = Better

Finding the right model

Quantisation ^{*GGUF}

Lower **perplexity** is better, Higher **compression** is better



Perplexity (lower = better)

Perplexity is a statistical measure used to evaluate the performance of language models

It quantifies how well a language model can predict a given text

Different quantisation sizes and variants both affect perplexity

It's always a trade off between performance/quality and model size

If I can fit the model 100% in vRAM (along with the desired context size), I choose **Q6_K** quants.

If *not* -

- Q5_K_M, then Q4_K_M
- *then*:
 - CUDA: IQ3_M, then IQ2_XS, then a smaller model
 - Metal/CPU: Q3_K_M, then a smaller model

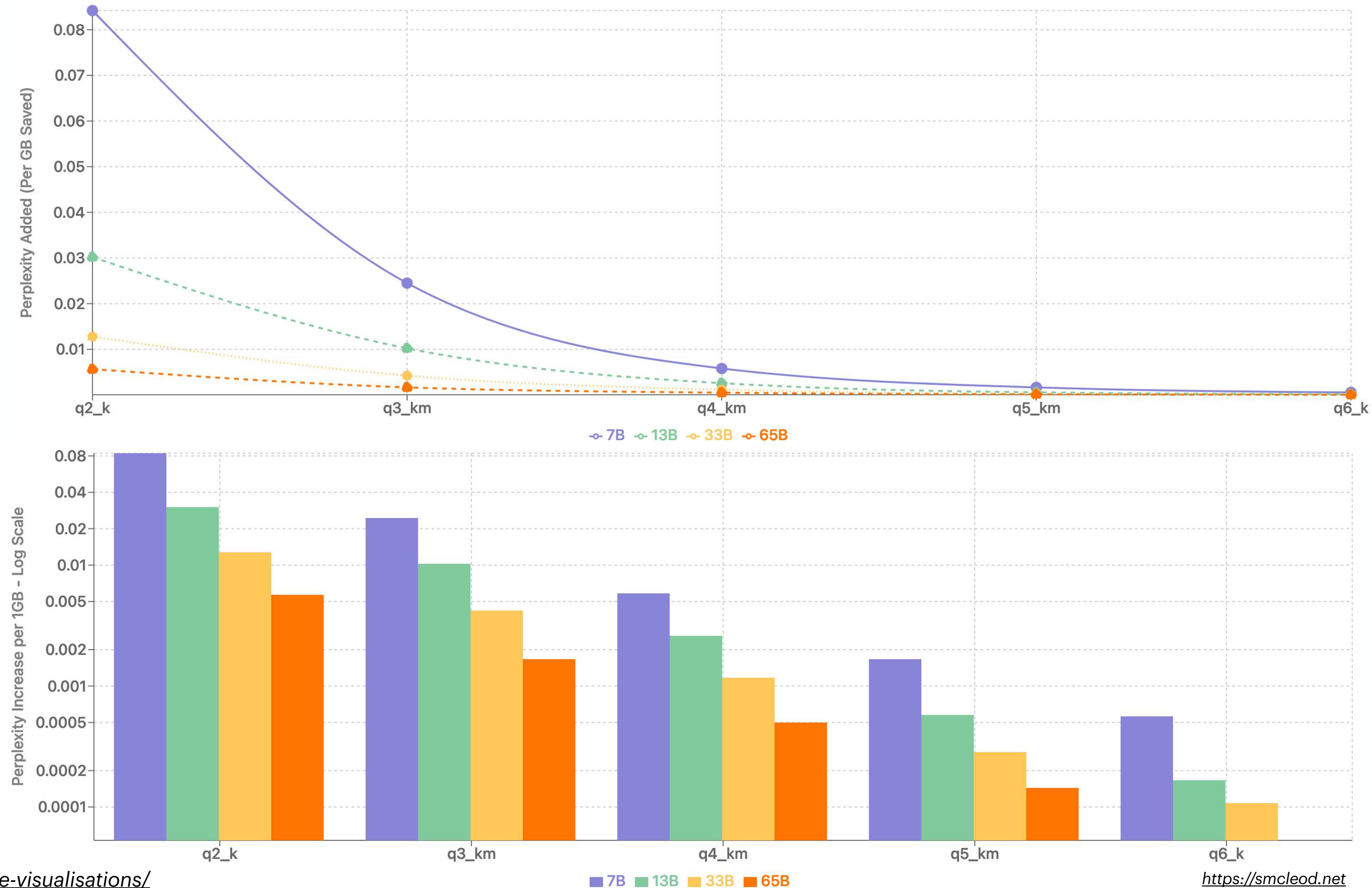
Finding the right model

Quantisation ^{*GGUF}

Smaller parameter models suffer more perplexity at lower quants

- Larger models still show perplexity increases with quantisation, but the relative impact varies across model sizes.
- The efficiency of quantisation (perplexity increase per GB saved) improves significantly as model size increases.
- For smaller models (e.g., 7B), the jump from Q4 to Q3 quantisation results in a more significant perplexity increase compared to larger models.
- Q6_K quantisation consistently provides excellent quality across all model sizes, with minimal perplexity increase.
- The benefits of using higher quality quantisation (e.g., Q5_K_M, Q6_K) start to diminish for larger models, potentially allowing for more aggressive compression without significant quality loss.

Perplexity Added (Per GB Saved) in Quantisation



Finding the right model

Quantisation ^{*GGUF}

Cheat Sheet

Which quant should I pick?

- While this is not 100% accurate (there's edge cases, model/feature specific details that can impact the (v)RAM usage) - it's a starting point

GGUF Quantisation Sweet Spots (8K Context)

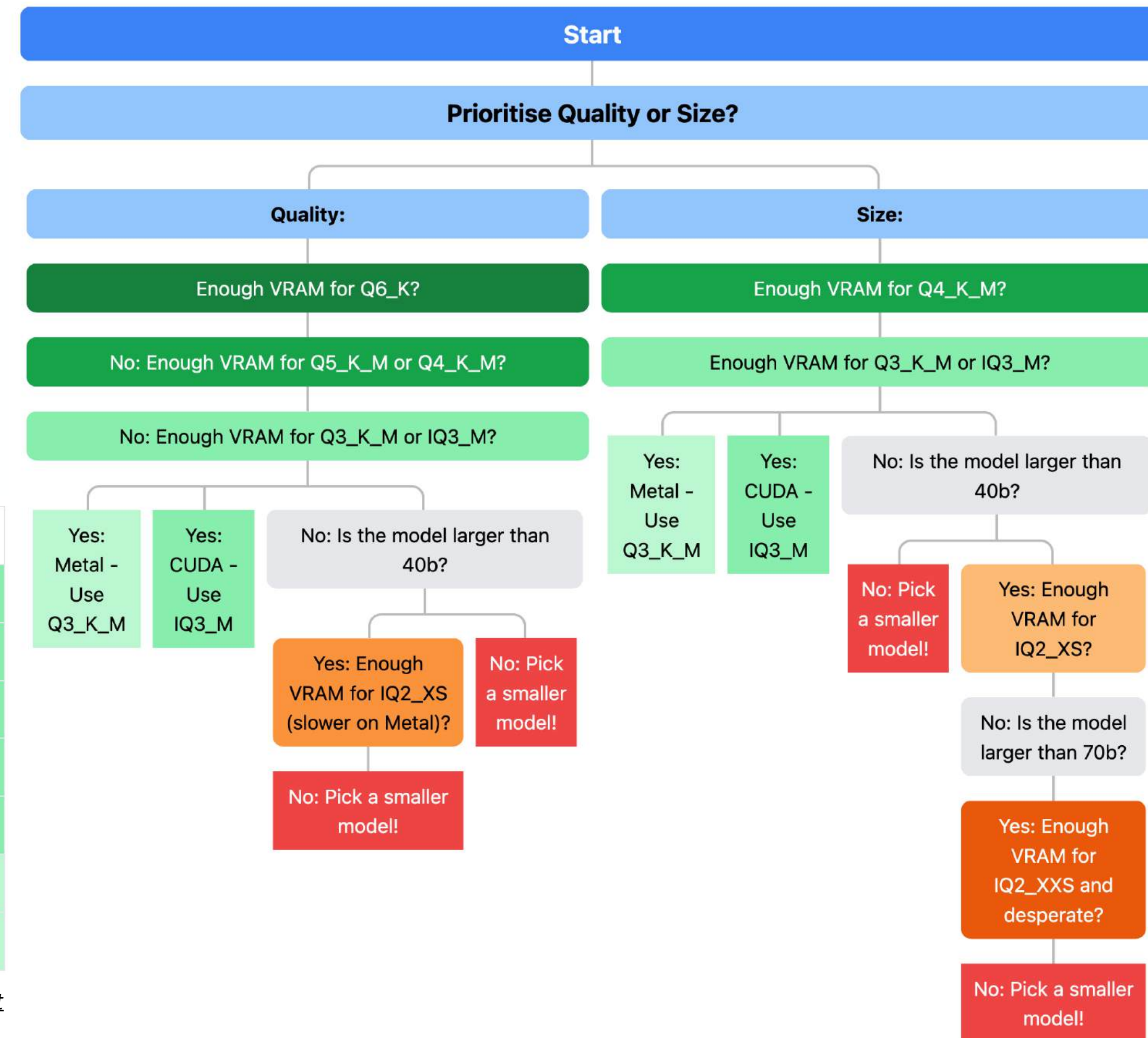
| Model Size | 8GB VRAM | 12GB VRAM | 16GB VRAM | 20GB VRAM | 24GB VRAM | 32GB VRAM | 48GB VRAM | 64GB VRAM |
|------------|----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| 3B | IQ3_M | Q4_K_M | Q5_K_M | Q6_K | Q6_K | Q6_K | Q6_K | Q6_K |
| 8B | IQ2_XXS | IQ3_M | Q4_K_M | Q5_K_M | Q6_K | Q6_K | Q6_K | Q6_K |
| 14B | N/A | IQ2_XXS | IQ3_M | Q4_K_M | Q5_K_M | Q6_K | Q6_K | Q6_K |
| 30B | N/A | N/A | IQ2_XXS | IQ3_M | Q4_K_M | Q5_K_M | Q6_K | Q6_K |
| 50B | N/A | N/A | N/A | IQ2_XXS | IQ3_M | Q4_K_M | Q5_K_M | Q6_K |
| 70B | N/A | N/A | N/A | N/A | IQ2_XXS | IQ3_M | Q4_K_M | Q5_K_M |
| 110B | N/A | N/A | N/A | N/A | N/A | IQ2_XXS | IQ3_M | Q4_K_M |

- Recommended
- Might be possible but depends on use case
- Not recommended / Not possible

<https://smcleod.net>

GGUF Quantisation Decision Tree

<https://smcleod.net>



<https://smcleod.net/2024/07/understanding-ai/llm-quantisation-through-interactive-visualisations/>

Finding the right model

Quantisation ^{*GGUF}

Cheat Sheet

Common quantisation types and their relative size, quality, and performance

- While this is not 100% accurate (there's edge cases, model/feature specific details that can impact the (v)RAM usage) - it's a starting point

| Quant Type | Size | Quality | Performance (CUDA) | Performance (Metal) | Notes |
|------------|--------------|---------------|--------------------|---------------------|---|
| IQ1_XS | Smallest | Unusable | Excellent | OK | Basically a jabbering idiot |
| Q2_K_S | Smallest | Unusable | Excellent | Excellent | Likely generates lots of errors, not very useful |
| Q2_K_M | Smallest | Very-Very-Low | Excellent | Excellent | Likely generates lots of errors, not very useful |
| IQ2_XXS | Very Small | Very-Low | Excellent | OK | Surprisingly usable for the GPU poor if you have CUDA |
| IQ2_XS | Very Small | Low | Very Good | Not Great | Surprisingly usable for the GPU poor if you have CUDA |
| Q3_K_S | Small | Low | Excellent | Excellent | Usable and quick but has had a few head injuries |
| Q4_0 | Small | Medium-Low | Excellent | Excellent | Legacy Quant Type - Not recommended |
| IQ3_XXS | Small | Medium-Low | Very Good | Poor | As good as K4_K_S but smaller |
| Q4_K_S | Medium-Small | Medium-Low | Excellent | Excellent | You may as well use Q4_K_M, or IQ3_X(X)S if you have CUDA |
| Q5_1 | Medium | Medium-Low | Excellent | Excellent | Legacy Quant Type - Not recommended |
| Q4_K_M | Medium | Medium | Excellent | Excellent | Balanced mid range quant |
| Q5_K_S | Medium-Large | Medium | Excellent | Excellent | Slightly better than Q4_K_M |
| Q5_K_M | Medium-Large | Medium-High | Excellent | Excellent | A nice little upgrade from Q4_K_M |
| Q6_K | Large | Very-High | Very Good | Very Good | Best all-rounder, quality-to-size ratio for systems with enough VRAM |
| Q8_0 | Very Large | Overkill | Good | Good | Large file size, usually overkill and practically indistinguishable from full precision for inference |

Understanding GGUF Quantisation

GGUF quantisation is a technique used to reduce the size and memory footprint of large language models while attempting to maintain performance. Here are some key points to remember:

- Quants without IQ or K (e.g. Q4_0, Q5_1) are *legacy* and not recommended, use K-quants instead
- IQ-quants offer better quality-to-size ratios but may have slower CPU/Metal performance.
- K-quants provide a good balance of quality, size, and performance across different hardware.
- Higher number quants (e.g., Q6_K) offer better quality but at the cost of larger file sizes.
- Q8_K and higher are generally overkill for inference and may not offer noticeable quality improvements over Q6_K.
- The larger the model parameters, the more usable it is likely to be at lower quantisation levels.
- Always consider your specific hardware capabilities and quality requirements when choosing a quantisation type.

<https://smcleod.net/2024/07/understanding-ai/llm-quantisation-through-interactive-visualisations/>

Finding the right model

Quantisation ^{*GGUF}

Find The Sweet Spot

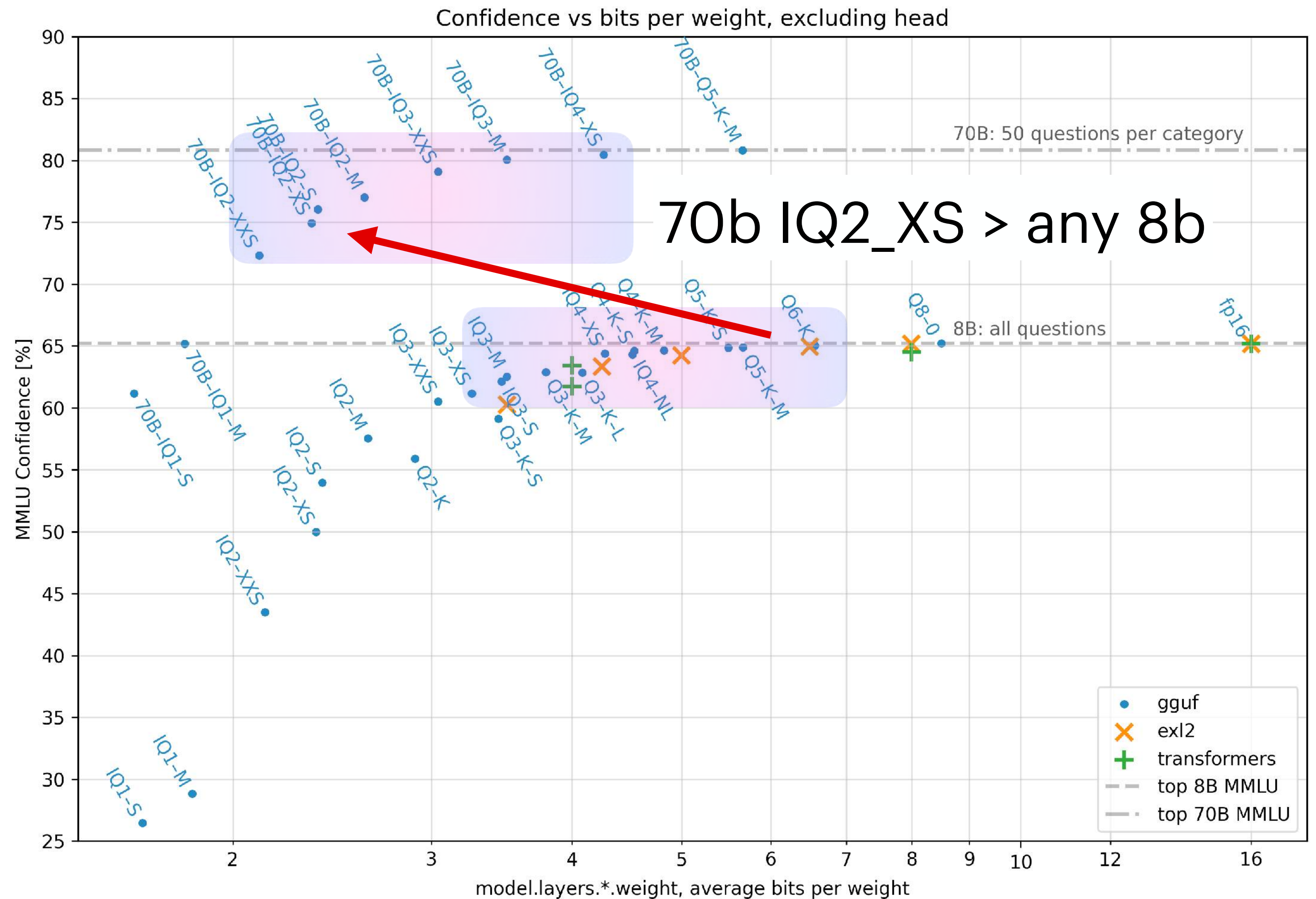
Larger parameter model with lower quant

Vs

Smaller parameter model with higher quant

When making a decision on the parameter and quant size that suits your needs, weigh up the:

- Performance (throughput, latency)
- Quality (perplexity)
- Hardware requirements (vRAM, storage, power)
- Model capabilities



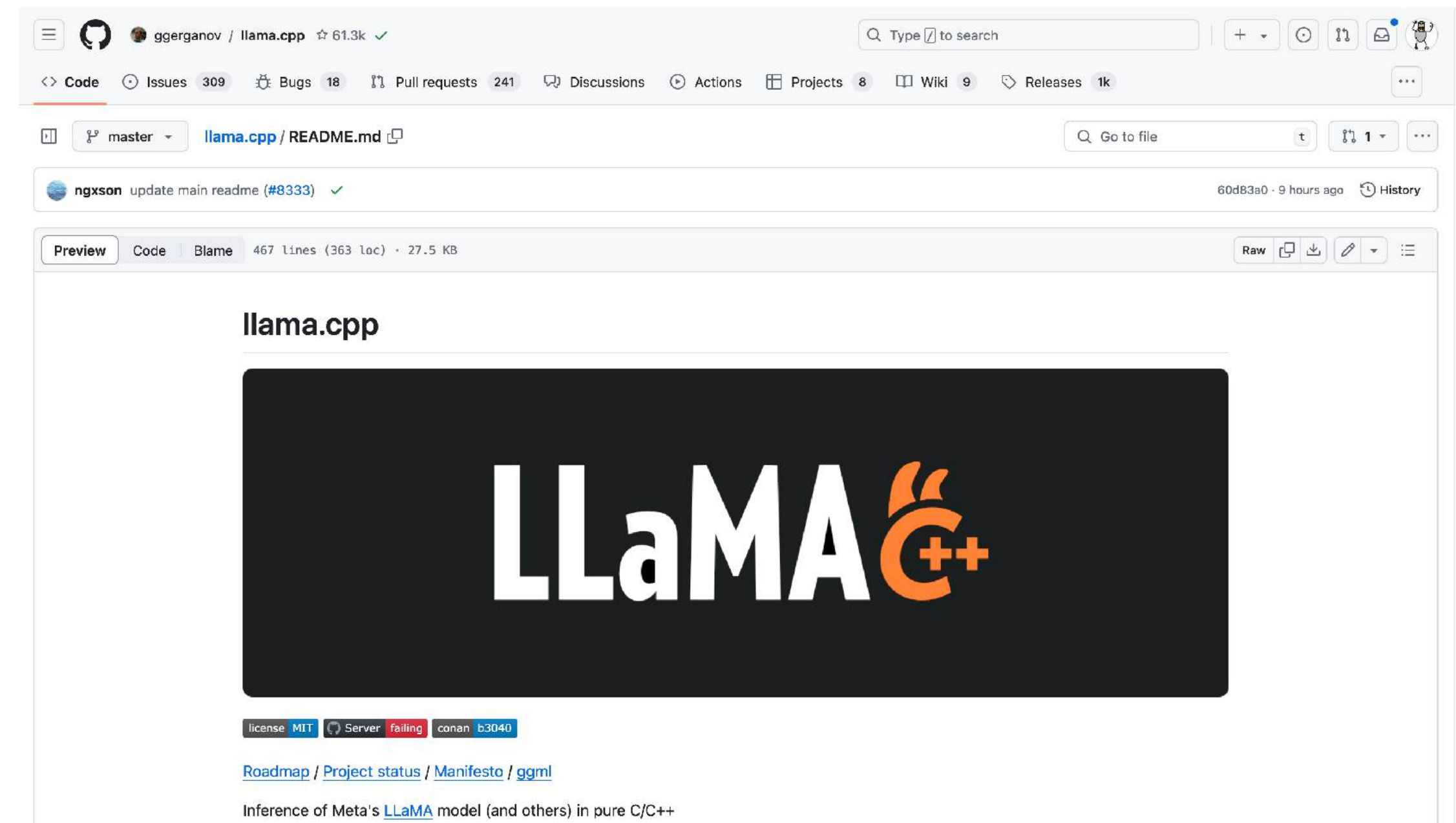
Tooling & Tuning

Model Servers

Pros

- Rapid development (probably the fastest software project I've seen since the Linux kernel)
 - Projects can gain performance improvements from updating this library frequently
- Huge community and widely used
- Works well on both macOS and Linux
- Great performance (although not the fastest option)
- Heavily customisable
- Large range of quantisation options
- The largest number of supported models
- Development decisions driven by data and community feedback
- Open Source
- Works with both GPU and CPU
- For CPU only inference it's the fastest server
- Offers an experimental RPC server for distributed inference across non-heterogenous hosts
- Responsive developer(s) with public roadmap

llama.cpp



Cons

- While performance is great, there are some performance focused servers that can offer higher throughput and advanced caching
- Limited to GGUF and HF Safetensors model formats

Model Servers

Ollama

Pros

- Uses llama.cpp for inference
- Easy to use
- Provides a docker-like experience
- Fast development
- Huge community
- Public model library
- Runs models in parallel and concurrently
- Multi-GPU
- Works well on both macOS and Linux
- Very well supported by clients
- Both native and OpenAI compatible API
- Function calling, embeddings, tool use
- Open Source
- Supports LoRAs ("adapters")

Cons

- Off-the-shelf builds sometimes lag slightly behind llama.cpp's features
- PRs can take some time to be reviewed
- OpenAI API endpoint doesn't support embeddings - but PR is up and should be merged soon
- Modelfile format has limited configuration

<https://ollama.com>

```
# ollama create deepseek-coder-v2-instruct-vfast4k:iq2_xxs
FROM ../DeepSeek-Coder-V2-Instruct.IQ2_XXS.gguf

TEMPLATE """{{ if .System }}{{ .System }}

{{ end }}{{ if .Prompt }}User: {{ .Prompt }}

{{ end }}Assistant:{{ .Response }}<|end_of_sentence|>"""

PARAMETER stop "User:"
PARAMETER stop "Assistant:"
PARAMETER stop "<|end_of_sentence|>"

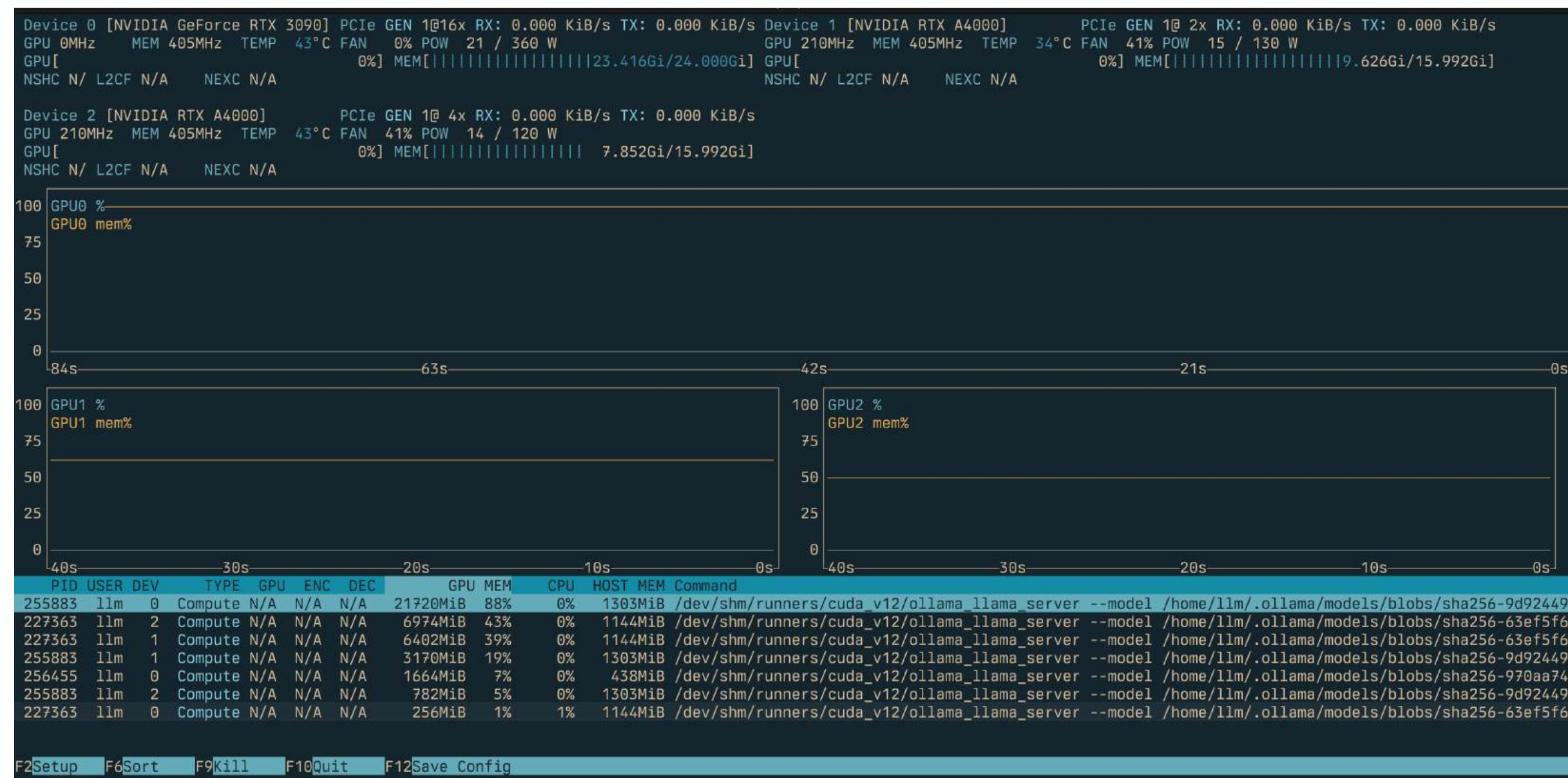
### Tuning ###
PARAMETER num_batch 2048

## For Codegen ##
PARAMETER num_ctx 40960
PARAMETER temperature 0.35
```

Ollama Modelfile

| Name | Size (GB) | VRAM (GB) | Until |
|--|-----------|-----------|---------------------|
| deepseek-coder-v2-lite-instruct:q6_k_1 | 31.71 GB | 31.71 GB | 2024-07-07 02:15:10 |
| nomic-embed-text:latest | 1.01 GB | 1.01 GB | 2024-07-07 02:15:06 |
| internlm2:7b-chat-v2.5-q6_K | 21.47 GB | 21.47 GB | 2024-07-07 02:07:05 |

Press 'q' or `esc` to return to the main view.



Multi-GPU, Multi-model concurrency

Model Servers

ExLlamaV2

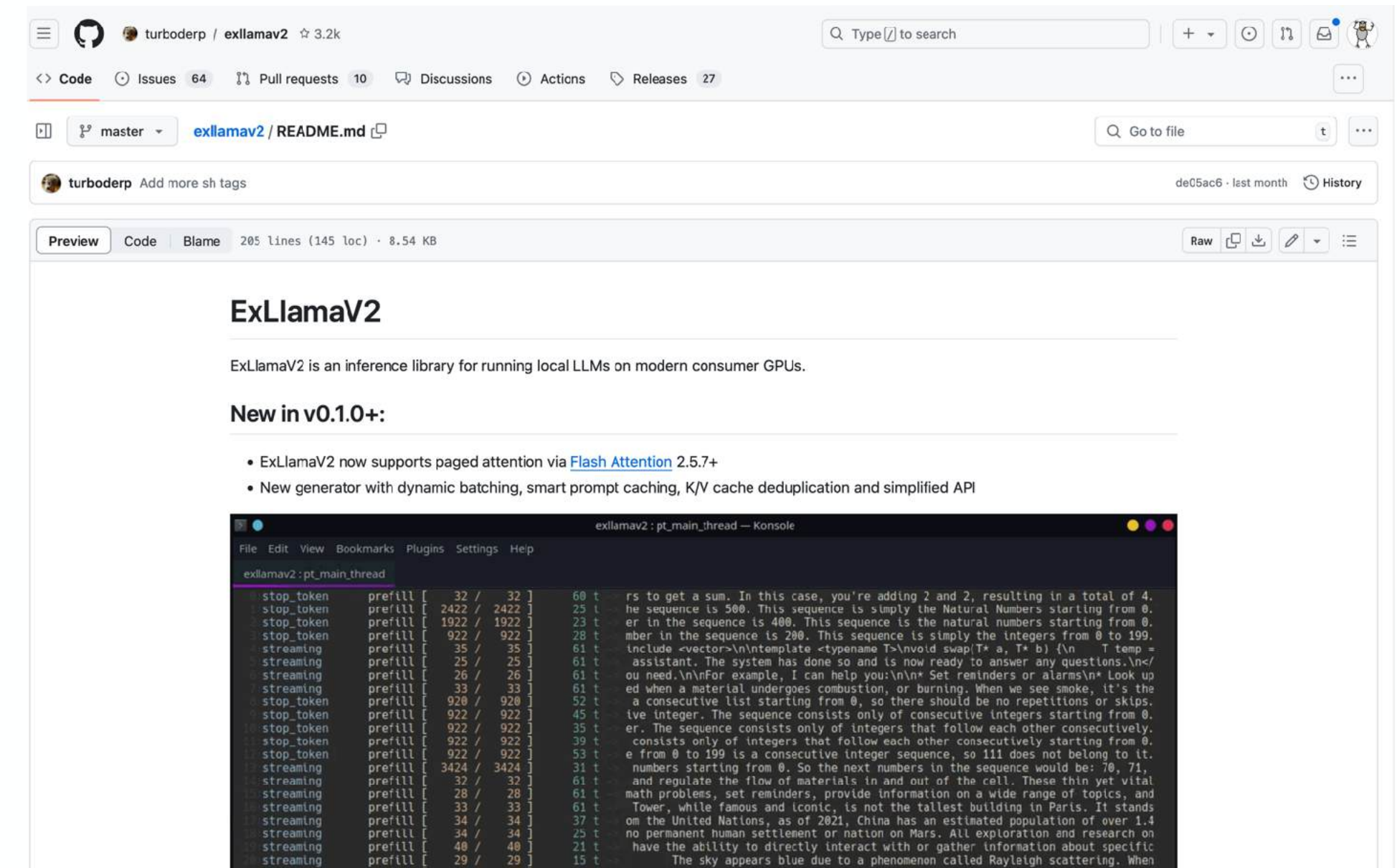
Pros

- Excellent performance for CUDA based model serving
- State of the art context quantisation and caching
- Supports LoRAs
- Open Source
- Resource efficient
- Efficient parallelisation and Context caching
- Multi-GPU
- Advance model

- Supports speculative decoding (small model in front of larger model to increase performance)
- Uses second most common model format (exl2)
- Responsive developer(s)

Cons

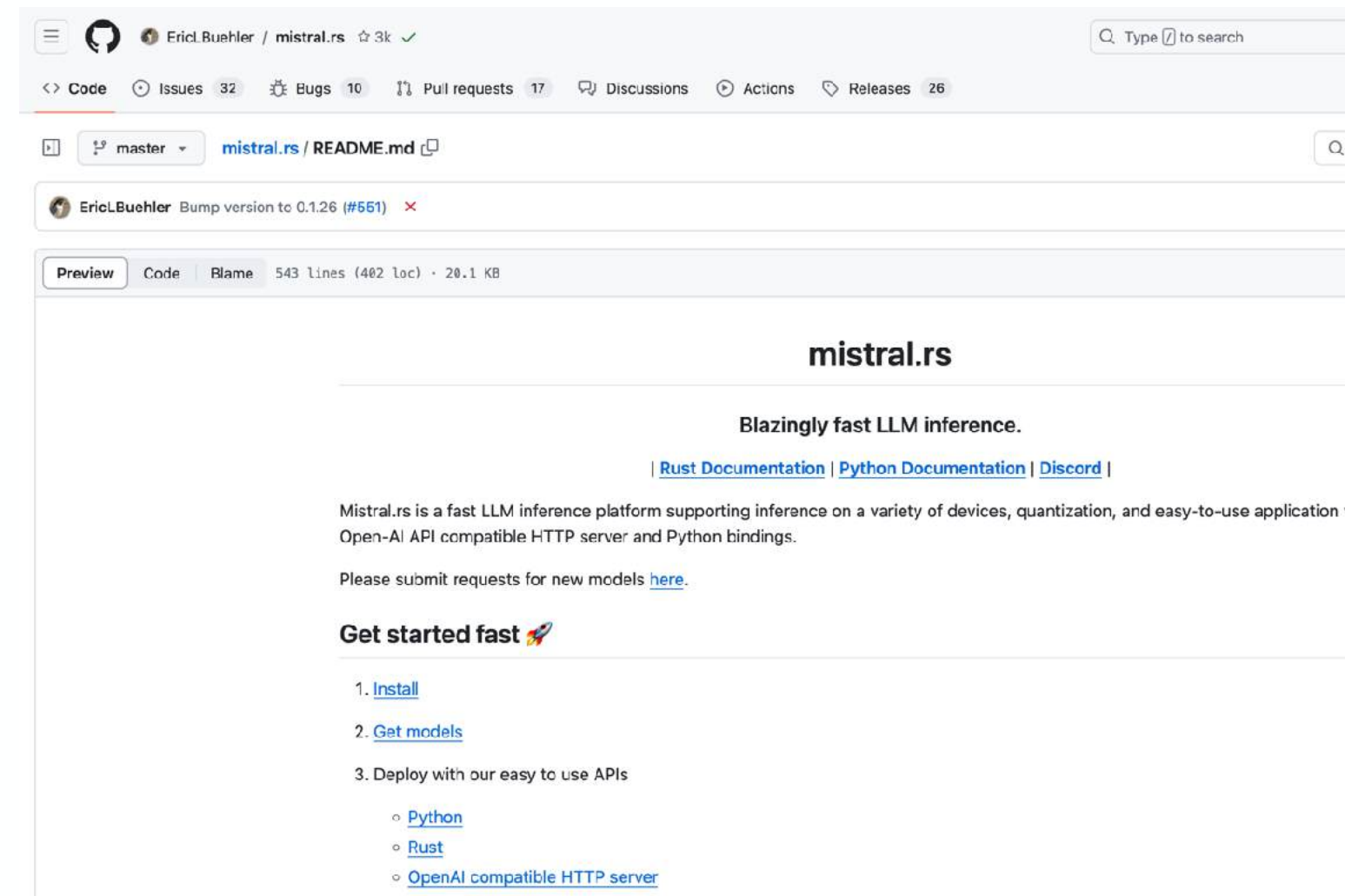
- Only works with Nvidia CUDA
- Limited to exl2 and GPTQ model formats
- Needs a server application to provide an API (e.g. Textgen WebUI, TabbyAPI, vLLM, Aphrodite Engine, [mistral.rs](#) support coming soon!)
- Not all model architectures supported (e.g. [DeepSeek Coder V2](#))



Model Servers

Mistral.rs

The one to watch!



Support matrix

Note: See [supported models](#) for more information

| Model | Supports quantization | Supports adapters | Supports device mapping | Supported by AnyMoE |
|------------------------|-----------------------|-------------------|-------------------------|---------------------|
| Mistral v0.1/v0.2/v0.3 | ✓ | ✓ | ✓ | ✓ |
| Gemma | ✓ | ✓ | ✓ | ✓ |
| Llama 2/3 | ✓ | ✓ | ✓ | ✓ |
| Mixtral | ✓ | ✓ | ✓ | |
| Phi 2 | ✓ | ✓ | ✓ | ✓ |
| Phi 3 | ✓ | ✓ | ✓ | ✓ |
| Qwen 2 | ✓ | | ✓ | ✓ |
| Phi 3 Vision | ✓ | | ✓ | ✓ |
| Idefics 2 | ✓ | | ✓ | ✓ |
| Gemma 2 | ✓ | ✓ | ✓ | ✓ |
| StarCoder 2 | ✓ | ✓ | ✓ | ✓ |
| LLaVa Next | ✓ | | ✓ | ✓ |
| LLaVa | ✓ | | ✓ | ✓ |

Cons

- Only supports GGUF and HF Safetensors models at present
 - GPTQ support coming soon
 - EXL2 support coming (after GPTQ has been added)
- OpenAI compatible API does not have embeddings yet but is coming soon
- Not all model architectures supported (e.g. DeepSeek Coder V2)

Pros

- State of the art performance
- State of the art context quantisation and caching
- Open Source, written in Rust
- Multi-GPU + CPU offloading
- Parallelisation
- Speculative Decoding: Mix models with the same vocabulary as the draft and target model to achieve massive speedups no loss in accuracy
- AnyMoE: Build a memory-efficient MoE model from anything, in seconds
- Continuous batching
- PagedAttention, mistral.rs is the first platform with GGUF and PagedAttention support
- Prefix caching
- CPU, CUDA, Metal, Accelerate support
- Ships with lightweight OpenAI API compatible HTTP server and provides a python API
- Grammar support with Regex and Yacc
- ISQ (In situ quantisation): run Safetensors models directly from HF quantising them after loading instead of creating a GGUF file.
- LoRA and X-LoRA support with dynamic adapter swapping at runtimeFast LoRA support with weight merging.
- Approachable and very responsive developer

Model Servers

Mistral.rs

The one to watch!

A message from Eric Buehler, Author of Mistral.rs

We are looking for collaborators!

Mistral.rs is written in Rust but anyone with experience in GPU optimisations/kernels, or LLMs can help!

I am looking for collaborators who are able to help out the process of adding models, optimising things, and general maintenance such as issue triaging. Progress can be slow when it is just one person (me) maintaining the project, so any collaborators would really make a difference.

Create an issue stating you want to help out:

<https://github.com/EricLBuehler/mistral.rs>

Or contact Eric on the mistral.rs Discord <https://discord.gg/SZrecqK8qw>

Model Servers

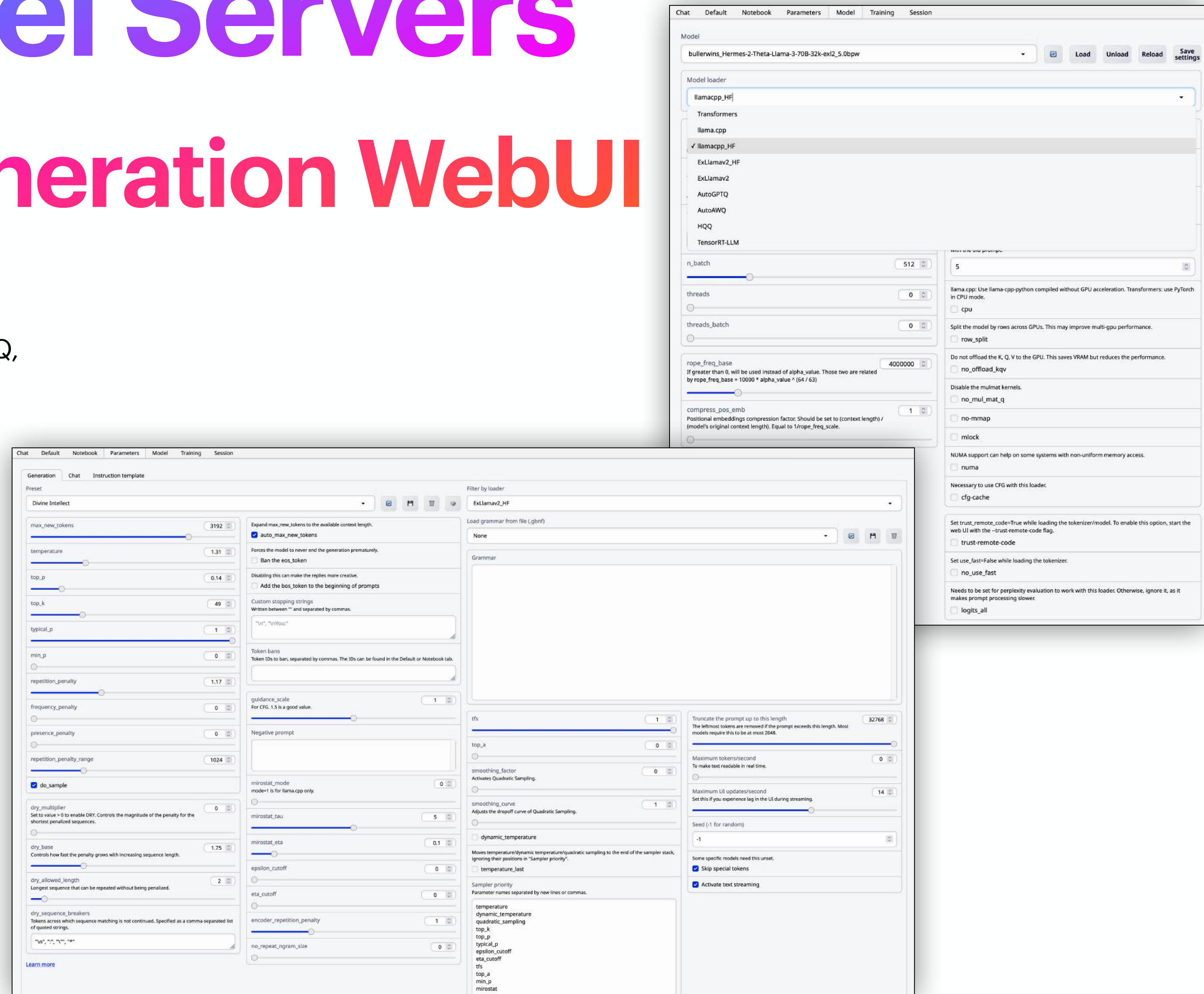
Text Generation WebUI

Pros

- Supports many model servers built in (llama.cpp, ExLlamav2, HQQ, AutoGPTQ, AutoAWQ, Transformers, TensorRT-LLM)
- Advanced server settings
- Open Source
- Large community
- Plugin ecosystem
- Inbuilt client with many advanced inference parameters
- LoRA training capabilities
- Multimodal (Image, Vision, Voice)
- Basic OpenAI Compatible API

Cons

- Gradio based UI can be clunky, especially for chat client
- Some users can be overwhelmed by the number of configuration options (as it is positioned as an advanced tool)
- OpenAI Compatible API doesn't support embeddings



Parameter Tuning

Parameter Tuning

Sampling Parameters For Codegen Tasks

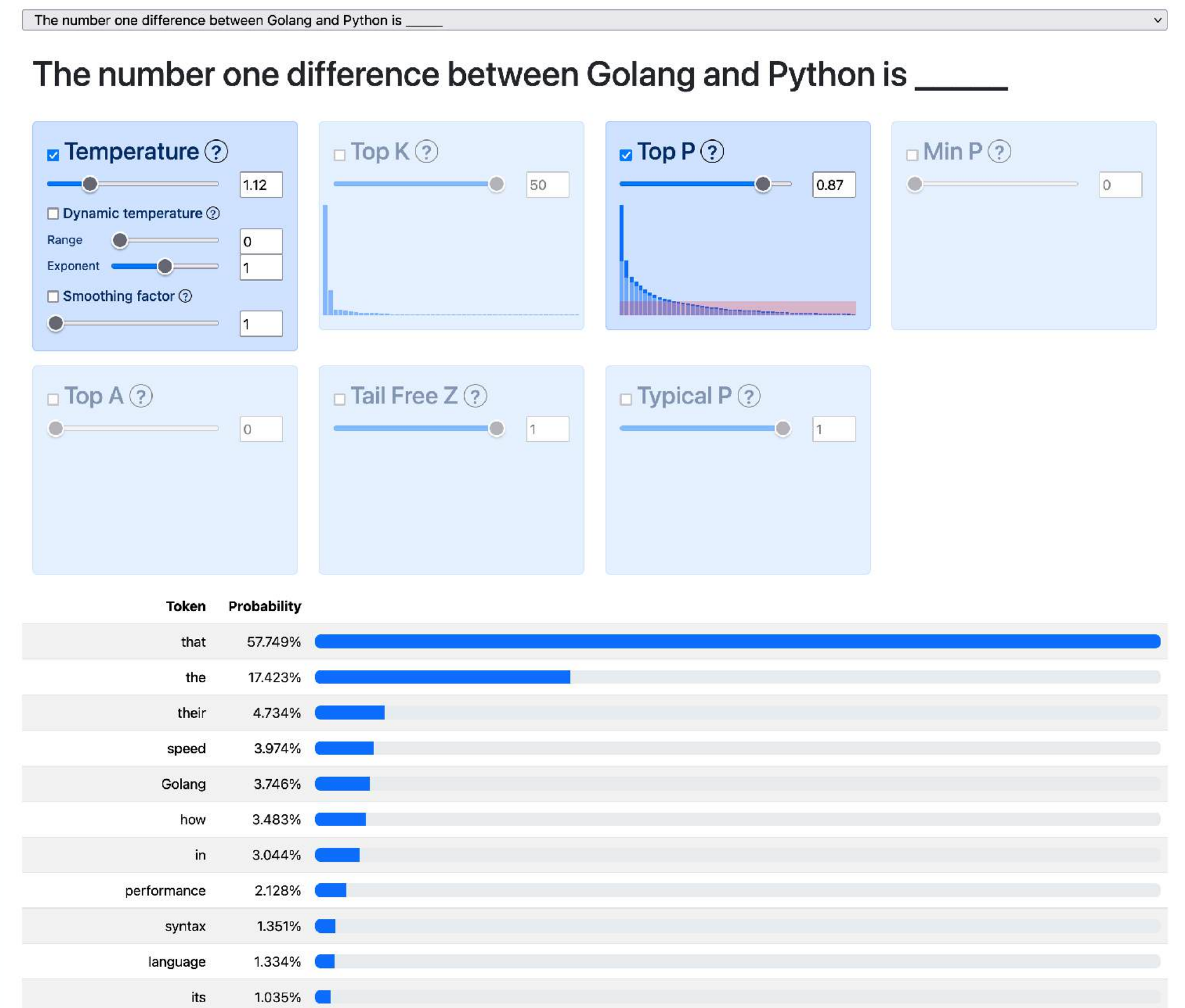
temperature vs top_p

Temperature increases variety, while top_p (and the lesser used top_k) reduce variety and focus samples on the model's top predictions, they interact with each other.

For codegen I start with temperature: 0.35 and top_p: 0.9

- **Low Temperature and High Top_p for Code Generation:** This setting tends to produce more conservative, predictable, and syntactically correct code. It's ideal for generating standard implementations, following best practices, or adhering to specific coding styles. The code will be reliable but may lack creative problem-solving approaches.
- **High Temperature with Low Top_p for Code Generation:** This configuration can lead to more experimental or unconventional code. It might introduce novel solutions or unexpected coding patterns, but may also result in syntax errors or logically inconsistent implementations. This setting could be useful for brainstorming alternative approaches or generating code snippets for creative coding exercises.

LLM Sampling Playground



Parameter Tuning

Advanced Sampling Parameters For Codegen Tasks

Codegen temperature 0.35 and top_p: 0.9

Temperature: 0.35 (Low)

- This low temperature setting promotes more predictable and consistent code output.
- It reduces the likelihood of unexpected or creative but potentially erroneous code.
- The generated code is more likely to follow common patterns and best practices.

Top_p: 0.9 (High)

- This relatively high top_p allows for a broader vocabulary and more diverse code constructs.
- It provides some room for the model to consider less common but potentially useful coding solutions.

Combined effect for code generation

1. Reliable and consistent: The low temperature ensures that the generated code will generally be syntactically correct and logically sound.
2. Slight flexibility: The high top_p allows for some variation, potentially introducing less common but valid coding patterns or solutions.
3. Balanced creativity: While predominantly conservative, this setting might occasionally suggest slightly novel approaches within the bounds of established practices.
4. Error reduction: The low temperature significantly reduces the chance of syntax errors or logically flawed code.

top_p (Nucleus Sampling)

- Definition: Involves choosing the smallest possible set of tokens whose cumulative probability exceeds a threshold 'p'. It ensures the sum of probabilities is at least 'p', focusing on a variable number of high-probability tokens.
- Effect: Controls diversity in a dynamic manner. Lower values (e.g, 0.75) lead to less diverse but more predictable outputs. Higher values (e.g. 0.95) increase diversity.

temperature

- Definition: Controls the randomness of predictions by scaling the logits before applying the softmax function. Values range from 0 to 1, where 1 means no scaling and lower values reduce randomness.
- Effect: Lower values (e.g., 0.2) make the output more deterministic, while higher values (e.g., 1.0) increase randomness and creativity.

repeat_last_n

- Definition: Number of tokens to look back for applying repeat penalties.
- Effect: Can be adjusted based on the typical length of generation. Generally leave as default, however can be tweaked if experiencing repetition or problems maintaining context (values like 20-40 can be a good starting point).

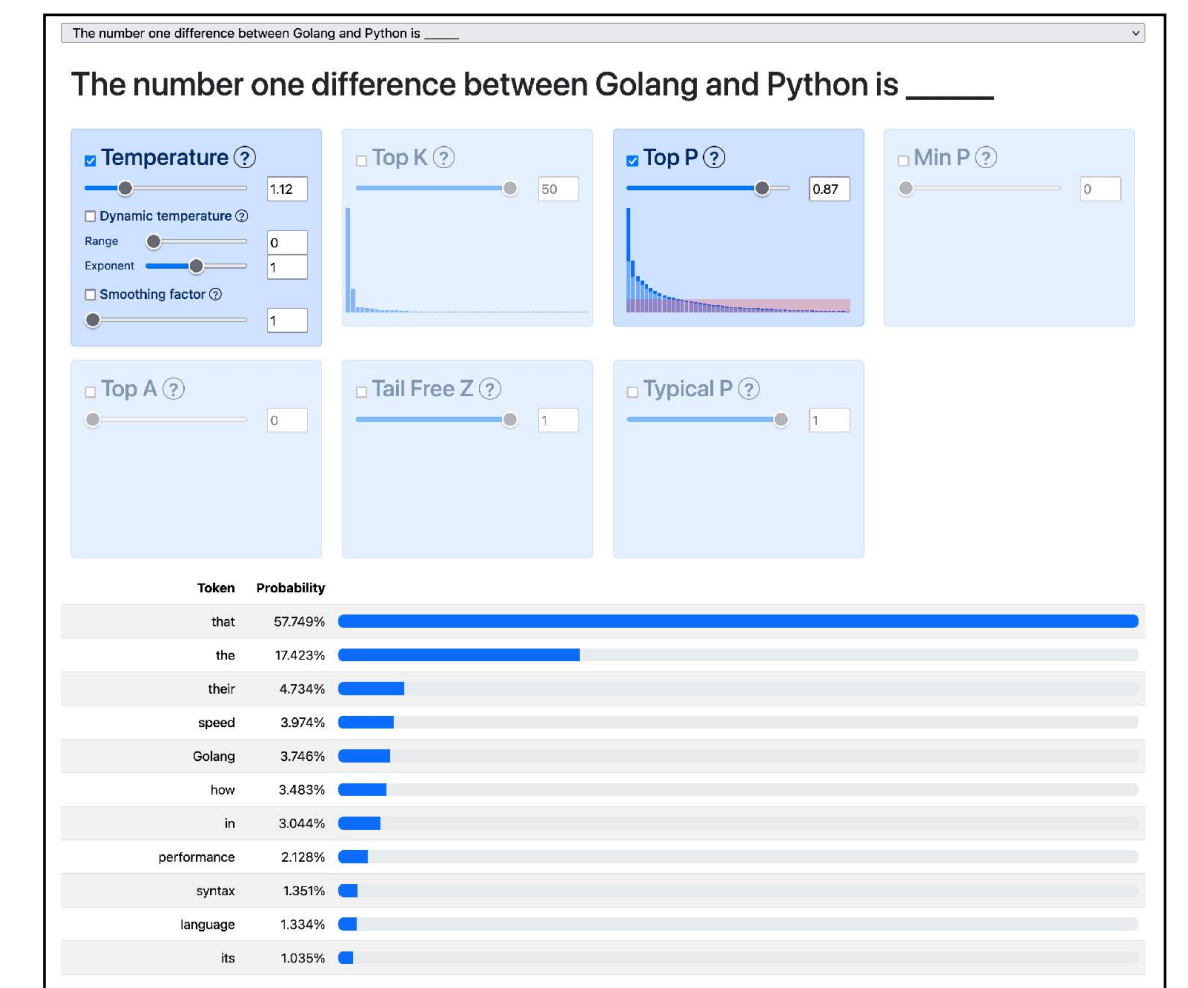
frequency_penalty

- Definition: Penalises the model for generating tokens that appear frequently in the context, reducing repetitiveness.
- Effect: Higher values (e.g., 0.5) make the output less repetitive by discouraging frequent token repetitions.

presence_penalty

- Definition: Penalises the model for generating tokens that have already appeared in the context, encouraging the introduction of new topics.
- Effect: Higher values (e.g., 0.5) make the model more likely to introduce new content rather than repeating itself.

LLM Sampling Playground



<https://artefact2.github.io/llm-sampling/index.xhtml>

Fork w/ code examples <https://sammcj.github.io/llm-sampling/index.xhtml>

Parameter Tuning

Tuning for performance and memory requirements

- **Batch size** (num_batch)
 - Ollama defaults to 512, which is smaller than llama.cpp upstream's default of 2048. This allows for reduced (v)RAM usage at the cost of some performance. **My default: 1024-2048.**
- **Parallelisation** (OLLAMA_NUM_PARALLEL, tensor_replicas etc...)
 - Some model servers such as Ollama allow loading models with parallelisation to improve performance at the cost of additional (v)RAM usage. **My default: auto** (new default)
 - Ollama now manages this setting by default 🎉
- **Concurrency** (OLLAMA_MAX_LOADED_MODELS, num_servers etc...)
 - Some model servers allow you to load multiple models that can respond at the same time. **My default: 4-6**
- **Context size** (num_ctx, max_tokens)
 - Large impact on (v)RAM usage
 - Context (v)RAM usage can be greatly reduced using KV/context caching (supported by exllamav2 🎉, but not llama.cpp)
- **Flash Attention** (flash_attn, -fa, OLLAMA_FLASH_ATTENTION etc...)
 - Greatly reduces (v)RAM usage as the context size increases
 - Llama.cpp has a custom implementation which works on GPU+CPU
 - ExllamaV2 and some others support this on Nvidia cards only

Example setting Ollama server environment (macOS)

```
launchctl setenv OLLAMA_NUM_PARALLEL 2
launchctl setenv OLLAMA_MAX_LOADED_MODELS 4
```

Example
Ollama
API request

```
curl http://localhost:11434/api/generate -d '{
  "model": "deepseek-coder-v2-lite-instruct:q6_k_1",
  "prompt": "Output a for loop example in Bash",
  "options": {
    "num_ctx": 32768,
    "num_batch": 2,
    "num_gpu": 1,
    "main_gpu": 0,
    "low_vram": false,
    "use_mmap": true,
    "use_mlock": false,
    "num_thread": 8
  }
}'
```

Example Ollama Modelfile

```
FROM ../DeepSeek-Coder-V2-Lite-Instruct.Q6_K_L.gguf

### Tuning ###
PARAMETER num_batch 2048

## For Codegen ##
PARAMETER num_ctx 32768
PARAMETER temperature 0.4
PARAMETER num_keep 256
```

Parameter Tuning

Parameters that affect model performance and memory requirements

ExLlamaV2 Servers

- Max sequence length (Context size)
- Cache size
- Cache mode (quantised K/V cache)
- Chunk size
- If your model is MoE
 - Number of active MoE "experts" per token
- Whether you're using a draft model (for speculative decoding)
 - Draft cache mode (quantised draft K/V cache)

The screenshot shows the configuration interface for ExLlamaV2 servers. Several parameters are highlighted with red boxes:

- Select Model:** NobodySpecial_dolphin_292_qwen2_72b_exl2_4_0bpw
- Max Sequence Length:** 32768
- Override Base Sequence Length:** 1
- Cache Size:** 65536
- Rope Scale:** 1
- Rope Alpha:** 1
- Speculative Decoding:**
 - Select Draft Model:** blockblockblock_dolphin-2.9.2-qwen2-7b-bpw4-exl2
- Draft Rope Scale:** (empty)
- Draft Rope Alpha:** (empty)
- Draft Cache Mode:** Q4 (selected), Q6, Q8, FP16
- Cache Mode:** Q4 (selected), Q6, Q8, FP16
- GPU Split Auto:**
- Use Fasttensors:**
- Auto-split Reserve:** 96
- Number of experts per token (MoE only):** 2
- Chunk Size:** 2048

Parameter Tuning

Prompt and Parameter A/B Testing

Ollama Grid Search

A handy tool for quickly A/B testing prompts and inference parameters

- Iterates over different models, prompts and parameters to generate inferences
- Allows multiple iterations for each combination of parameters
- Supports concurrent requests
- Experiments can be exported (JSON) for analysis or use in training / fine tuning

<https://github.com/dezoito/ollama-grid-search>

The screenshot shows the Ollama Grid Search web application. The interface is divided into two main sections: a configuration panel on the left and a documentation panel on the right.

Configuration Panel (Left):

- Models:** 36 available on http://localhost:11434 - Ollama v.v0.2.5-16-g987dbab0. A search bar "Filter Models by Name" is present. Selected models include llama3:latest, phi3-128k:Q8_0, and phi3:3.8b-mini-128k-instruct-q6_K.
- Prompts:** Two prompts are defined:
 - Prompt 1:** "Tell me a short joke that doesn't include tomatoes, chickens, bicycles or scientists"
 - Prompt 2:** "Tell me a short joke"
- System Prompt:** "You are an unhelpful AI assistant."
- Generations per parameter set:** Set to 2.
- Temperature List:** 0.3,0.7,1.0
- Repeat Penalty List:** 1.1,1.3

Documentation Panel (Right):

- Grid Search:** Explains that grid search involves systematically exploring different combinations of parameters within predefined ranges to optimize model performance.
- A/B Testing:** Explains that A/B testing involves comparing the performance of different models when making inferences on the same parameters or data.
- Experiment History:** Explains that users can visualize past experiments, export them in JSON format, or re-run them with the same parameters and configurations.

At the bottom of the configuration panel, there are two buttons: "Start Experiment" (black) and "Stop Experiment" (red).

Testing two prompts across three different models and multiple inference parameters

Model Management

Downloading Models

Can I run it?

Selecting quant / param sizes that fits on your hardware

(v)RAM calculators

There are a number of handy tools that assist with estimating the required amount of (v)RAM to run a given model at a given quantisation.

Some even have parameters various caching and context settings

- <https://huggingface.co/spaces/NyxKrage/LLM-Model-VRAM-Calculator>
- <https://huggingface.co/spaces/Vokturz/can-it-run-llm>
- <https://llm.extractum.io>
- https://rahulschand.github.io/gpu_poor/
- <https://huggingface.co/spaces/hf-accelerate/model-memory-usage>

Remember to tune model server parameters as required

1. Context size
2. Batch size
3. Parallelism
4. Caching

Memory Requirement:

| Total | 53840 MB |
|-------------------------|----------|
| Breakdown | |
| KV Cache | 5841 MB |
| Model Size | 46111 MB |
| Activation Memory | 983 MB |
| Grad & Optimizer memory | 0 MB |
| cuda + other overhead | 906 MB |
| selected GPUs needed | 3 |

Model Memory Calculator

This tool will help you calculate how much vRAM is needed to train and perform big model inference on a model hosted on the Hugging Face Hub. The minimum recommended vRAM needed for a model is denoted as the size of the "largest layer", and training of a model is roughly 4x its size (for Adam). These calculations are accurate within a few percent at most, such as bert-base-cased being 413.66 MB and the calculator estimating 413.18 MB. When performing inference, expect to add up to an additional 20% to this as found by EleutherAI. More tests will be performed in the future to get a more accurate benchmark for each model. Currently this tool supports all models hosted that use transformers and timm. To use this tool pass in the URL or model name of the model you want to calculate the memory usage for, select which framework it originates from ("auto" will try and detect it from the model metadata), and what precisions you want to use.

| dtype | Largest Layer or Residual Group | Total Size | Training using Adam (1 |
|-------|---------------------------------|------------|------------------------|
| int4 | 594.0 MB | 33.59 GB | N/A |
| int8 | 1.16 GB | 67.18 GB | N/A |

Memory usage for 'Qwen/Qwen2-72B-Instruct'

| dtype | Largest Layer or Residual Group | Total Size | Training using Adam (1 |
|-------|---------------------------------|------------|------------------------|
| int4 | 594.0 MB | 33.59 GB | N/A |
| int8 | 1.16 GB | 67.18 GB | N/A |

Model Name or URL: Qwen/Qwen2-72B-Instruct

Library: auto transformers timm

Model Precision: float32 float16/bfloat16 int8 int4

API Token: Optional (for gated models)

Calculate Memory Usage

Can you run it? LLM version

Information:

- GPU information comes from TechPowerUp GPU Specs
- Mainly based on Model Memory Calculator by hf-accelerate using transformers library
- Inference is calculated following EleutherAI Transformer Math 101, where it is estimated as $Memory_{inference} \approx Model\ Size \times 1.2$
- For LoRA Fine-tuning, I'm assuming a 16-bit dtype of trainable parameters. The formula (in terms of GB) is $Memory_{LoRA} \approx (Model\ Size + \# trainable\ Params_{bits} \times \frac{16}{8} \times 4) \times 1.2$

NousResearch/Meta-Llama-3-8B-Instruct (7.6B)

Number of GPUs required for GeForce RTX 3090 (24.0 GB, 2020)

| Variable | float32 | float16/bfloat16 | int8 | int4 |
|--------------------------|---------|------------------|-------|-------|
| Total Size (GB) | 27.96 | 13.98 | 6.99 | 3.49 |
| Inference (GB) | 33.55 | 16.77 | 8.39 | 4.19 |
| Training using Adam (GB) | 111.83 | 55.92 | 27.96 | 13.98 |
| LoRA Fine-Tuning (GB) | 94.99 | 48.22 | 9.83 | 5.83 |

Legend: float32 (red), float16/bfloat16 (orange), int8 (green), int4 (blue)

Finding the right model

Memory Requirements

Use a calculator!

LLM Model, Can I run it?

GPU (optional)
GeForce RTX 3090 - 24 GB

Model (unquantized)
deepseek-ai/DeepSeek-Coder-V2-Instruct

Context Size
16384

Quant Format
 GGUF EXL2 GPTQ (coming soon)

BPW
3

KV Cache 4 bit

Submit

Model Size (GB)
82.33

Context Size (GB)
8.79

Total Size (GB)
91.12

Models are often available in multiple parameter sizes (7b,14b,70b)

Ideally pick a model that 100% fits in your vRAM

Assuming a given set of models are all the same family and release:

Generally larger size models will perform better than smaller - there are a few additional factors to take into account:

- Performance, larger usually means slower
- Desired context size
- Quantisation (more on this shortly)

LLM Model, Can I run it?

GPU (optional)
GeForce RTX 3090 - 24 GB

Model (unquantized)
deepseek-ai/DeepSeek-Coder-V2-Lite-Instruct

Context Size
16384

Quant Format
 GGUF EXL2 GPTQ (coming soon)

BPW
3

KV Cache 4 bit

Submit

Model Size (GB)
5.49

Context Size (GB)
1.36

Total Size (GB)
6.85

263b vs 16b

MoE Models

MoE ("Mixture of Experts") models are made up of multiple, smaller models internally with a router

| Model | #Total Params | #Active Params | Context Length |
|---------------------------------|---------------|----------------|----------------|
| DeepSeek-Coder-V2-Lite-Base | 16B | 2.4B | 128k |
| DeepSeek-Coder-V2-Lite-Instruct | 16B | 2.4B | 128k |
| DeepSeek-Coder-V2-Base | 236B | 21B | 128k |
| DeepSeek-Coder-V2-Instruct | 236B | 21B | 128k |

- You'll still need enough (v)RAM to load the #Total Params
- But you'll only use the compute required for the #Active Params during inference

Model Management - Ollama



Gollama *Shameless plug

Interactive TUI with edit, sorting & filtering

- List available models
- Display metadata such as size, quantisation, model family, and modified date
- Edit / update a model's Modelfile
- Sort models by name, size, modification date, quantisation level, family etc
- Select and delete models
- Run and unload models
- Inspect model for additional details
- Link models to LM Studio
- Copy / rename models
- Push models to a registry
- Update (pull) existing models
- Pull new models
- Show running models
- Has some cool bugs (PRs appreciated!)

```
Ollama Models
74 items
```

| | | | | | |
|--|---------|---------|-----------|------------|---------|
| phi3:14b-medium-4k-instruct-q6_K | 10.67GB | F16 | phi3 | 2024-07-04 | 73b4386 |
| phi3-128k:Q8_0 | 3.78GB | Q8_0 | llama | 2024-07-04 | 673c136 |
| llama3:8b | 4.34GB | Q4_0 | llama | 2024-07-04 | 6be7220 |
| phi3:3.8b-mini-128k-instruct-q6_K | 2.92GB | Q6_K | phi3 | 2024-07-04 | 6d414ac |
| internlm2:7b-chat-v2.5-q6_K | 5.91GB | Q6_K | internlm2 | 2024-07-03 | d1e1c05 |
| deepseek-coder-v2-instruct-vfast4k:iq2_xxs | 57.28GB | IQ2_XXS | deepseek2 | 2024-07-03 | 278fdbf |
| deepseek-coder-v2-lite-instruct:q6_k_1 | 13.56GB | Q6_K | deepseek2 | 2024-07-03 | 1d6bc4d |
| registry.ollama.ai/ollama/deepseek-coder-v2-lite-instruct:q6_k_1 | 13.56GB | Q6_K | deepseek2 | 2024-07-03 | 1d6bc4d |
| gemma-2-9b-it-sppo-iter3:q5_k_M | 6.19GB | Q5_K_M | gemma2 | 2024-07-03 | 9039599 |
| llava-llama3:latest | 5.16GB | Q4_K_M | llama | 2024-07-03 | 44c161b |
| rjmalagon/gte-qwen2-1.5b-instruct-embed-f16:latest | 3.31GB | F16 | qwen2 | 2024-07-02 | 90ba362 |
| gemma-2-27b-it:q6_k_1 | 22.11GB | Q6_K | gemma2 | 2024-06-30 | 606d77e |
| qwen2:1.5b-instruct-q4_K_M | 0.92GB | Q4_K_M | qwen2 | 2024-06-28 | 0d2a504 |
| qwen2:1.5b-instruct-q6_K | 1.19GB | Q6_K | qwen2 | 2024-06-28 | 1fcc9e4 |
| smaug-llama-3-70b-instruct-32k:q4_k_m | 39.60GB | Q4_K_M | llama | 2024-06-25 | 590844a |
| llava:13b | 7.46GB | Q4_0 | llama | 2024-06-24 | 0d0eb4d |
| llava:7b | 4.41GB | Q4_0 | llama | 2024-06-24 | 8dd30f6 |
| mistral:7b | 3.83GB | Q4_0 | llama | 2024-06-24 | 2ae6f6d |
| gemma:7b | 4.67GB | Q4_0 | gemma | 2024-06-24 | a72c7f4 |
| gemma:2b | 1.56GB | Q4_0 | gemma | 2024-06-24 | b50d6c9 |
| qwen2:1.5b | 0.87GB | Q4_0 | qwen2 | 2024-06-24 | f6daf2b |
| command-r-plus-qwp4w3hyb:iq2_xxs | 26.65GB | IQ2_XXS | command-r | 2024-06-24 | 388d3b7 |
| smaug-llama-3-70b-instruct-32k:iq3_xxs | 25.58GB | IQ3_XXS | llama | 2024-06-23 | da3e259 |
| registry.ollama.ai/ollama/replete-coder-qwen-1.5b:q6_k_1 | 1.44GB | Q6_K | qwen2 | 2024-06-23 | 675cf7e |
| Replete-Coder-Qwen-1.5b:q6_k_1 | 1.44GB | Q6_K | qwen2 | 2024-06-23 | 675cf7e |
| deepseek-coder-v2-instruct-fast8k:iq2_xxs | 57.28GB | IQ2_XXS | deepseek2 | 2024-06-22 | f22f1d4 |
| deepseek-coder-v2-lite-instruct-fast8k:q6_k_1 | 13.56GB | Q6_K | deepseek2 | 2024-06-22 | d386662 |
| deepseek-coder-v2-lite-base:q6_k_1 | 13.56GB | Q6_K | deepseek2 | 2024-06-20 | 60ac6a7 |
| hermes-2-theta-llama-3-8b-16k_1024batch_ef16:q6_k | 7.30GB | Q6_K | llama | 2024-06-19 | 7e6a953 |
| hermes-2-theta-llama-3-8b-16k_512batch_ef16:q6_k | 7.30GB | Q6_K | llama | 2024-06-19 | 466793f |
| Llama-3-70b-262k-Uncensored-Tess-gradient:iq2_m | 29.74GB | IQ2_S | llama | 2024-06-17 | 4ad77da |
| hermes-2-theta-llama-3-8b-32k_ef16:q6_k | 7.30GB | Q6_K | llama | 2024-06-17 | 93e3586 |
| codestral-22b_ef16:q6_k | 17.44GB | Q6_K | llama | 2024-06-17 | 738c024 |
| hermes-2-theta-llama-3-8b-32k:q6_k | 6.14GB | Q6_K | llama | 2024-06-15 | 7b42db1 |
| nous-hermes-2-mixtral-8x7b-dpo.i1:q5_k_m | 30.95GB | Q5_K_M | llama | 2024-06-15 | fba2f1c |

And yes, I did write it with AI

That was a lot to take in!

How about a one pager on where to get started?

TLDR;

1. 🦙 Install and run a model server

```
brew install ollama  
open /Applications/Ollama.app
```

2. 💾 Download a model

```
ollama pull llama3:8b-instruct-q6_K
```

3. 🧑💻 Install a client

```
brew install msty (or anythingllm, jan, lm-studio)  
open /Applications/Msty.app
```

4. 🎉 Solve the worlds problems

Getting Started Cheatsheet

2024-07

Starting Clients and Models

These apps/models should provide a decent starting point

| | Try These | Download | Links |
|---------------------|-------------------------|---|--|
| Model Server | ★ Ollama | ★ brew install ollama
★ Download from ollama.com | • https://ollama.com |
| Client Apps | ★ Msty
★ AnythingLLM | ★ brew install anythingllm
★ Download from useanything.com | • https://msty.app/
• https://useanything.com |

Scenario

- You don't want to spend any money
- You don't want to send data to third parties
- You want a GUI client
- You want to run LLMs on your laptop
- Your laptop has at least
 - 16GB of (v)RAM (*small*)
 - 24GB of (v)RAM (*medium*)

| | | | |
|-------------------------------|--|---|---|
| General Models (Small) | ★ Qwen2.5 7b Instruct | ★ ollama pull qwen2.5:7b-instruct-q6_K
(q4_k_m if you can't spare 7-8GB vRAM) | • https://ollama.com/library/qwen2.5:7b-instruct-q6_K
• https://ollama.com/NightFuryTimo/hermes-2-theta-llama-3-8b:Q6_K |
| Coding Models (Small) | ★ qwen2.5-coder:7b
★ qwen2.5-coder:14b | ★ ollama pull qwen2.5-coder:7b-instruct-q6_K
★ ollama pull qwen2.5-coder:14b-instruct-q6_K | • https://ollama.com/library/qwen2.5-coder:7b-instruct-q6_K
• https://ollama.com/library/qwen2.5-coder:14b-instruct-q6_K |
| Coding Models (Medium) | ★ qwen2.5-coder:32b
★ deepseek-coder-v2-lite
★ codestral | ★ ollama pull qwen2.5-coder:32b-instruct-q4_K_M
(q6_k if you have >24GB vRAM)
★ ollama pull deepseek-coder-v2:16b-lite-instruct-q6_k
★ ollama pull codestral:22b-v0.1-q5_K_M | • https://ollama.com/library/qwen2.5-coder:32b-instruct-q4_K_M
• https://ollama.com/library/deepseek-coder-v2:16b-lite-instruct-q6_K
• https://ollama.com/library/starcoder2:7b-q6_K |
| Vision Models | ★ llava-llama3:8b-v1.1-fp16
★ llama3-llava-next-8b | ★ ollama pull llava-llama3:8b-v1.1-fp16
(if you have < 24GB (v)RAM pick a smaller quant)
★ ollama pull mapler/llama3-llava-next-8b:latest | • https://ollama.com/library/deepseek-coder-v2:16b-lite-instruct-q6_K
• https://ollama.com/library/starcoder2:7b-q6_K
• https://ollama.com/mapler/llama3-llava-next-8b:latest |
| Embedding Models | ★ mxbai-embed-large f16 | ★ ollama pull mxbai-embed-large | • https://ollama.com/library/mxbai-embed-large:latest |

My Home Lab Setup

Note: You do not need to invest in a setup like this to make practical use of AI/LLMs!

Everyday Software / Containerised Apps

- Model Servers
 - Ollama for GGUF serving
 - TabbyAPI and Tabbyloader for exllamav2 serving
 - Piper for STT
 - Whisper-Faster for TTS
- Hosted Clients
 - BigAGI
 - Open-WebUI & Open-WebUI Pipelines
 - InvokeAI for ImageGen
 - Home Assistant for IoT+LLMs
- Utils
 - Playwright for web scraping/automation
 - NVApi for GPU stats
 - Container registry & registry-ui for private Ollama registry
 - SearxNG for search API
 - Traefik for networking
 - Authentik for auth
 - Caching proxies to speed up builds
 - apt caching proxy
 - npm caching proxy
 - pip caching proxy



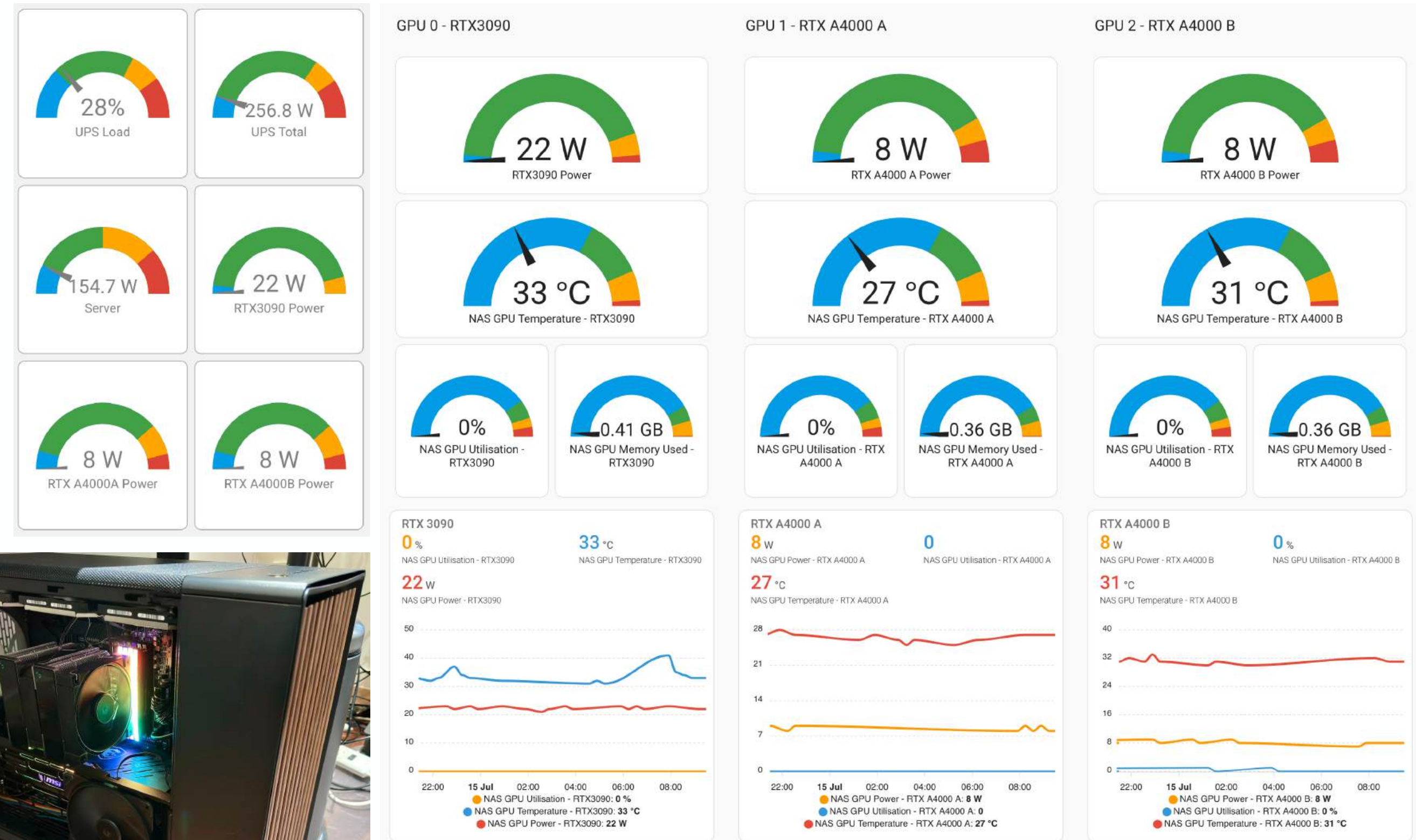
LLM powered IoT voice assistant

Ad-hoc Software / Containerised Apps

- Text Generation WebUI for experimenting with model serving and advanced inference params
- ComfyUI (for testing out new ImageGen models/adapters/methods)
- Flowise (for node based low coding)
- Training
 - Llama Factory
 - Kohya_SS for ImageGen

Hardware / Server

- My home server is used for many things (not just AI/LLMs), runs Fedora Linux and is a Ryzen 9 9900K, with 192GB DDR5, a bunch of NVMe and rotational drives, UPS, PiKVM and:
 - 2x RTX 3090 (2x 24GB)
 - 1x RTX A4000 (1x 16GB)
- Power consumption:
 - Standard / non-LLM: 90-100W
 - Inference: 250-550W
 - Training / maxed out: 400-850W
- ESP-S3-Box-3 for LLM powered IoT voice assistant
- On-device LLMs on MacBook



GPU metrics exposed via <https://github.com/sammcj/nvapi>

Links & Community

Where to find models, answers, and questions you didn't know you needed to ask

Community

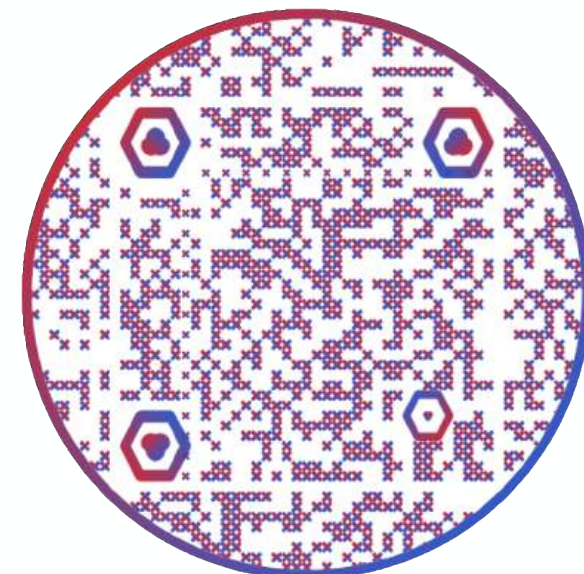
- [r/locallama](#) - the *best* online community for LLMs
- [Huggingface](#) - models, datasets, demo spaces, blog posts
- Discord is terrible, but every tool seems to have it's own discord server

Models / Quants

- [bartowski](#)
- [MaziyarPanahi](#)
- [mradermacher](#)
- [LoneStriker](#)
- [NousResearch](#)
- [QuantFactory](#)
- [cognitivecomputations](#)
- [mlx-community](#)
- [turboderp](#)
- [dranger003](#)
- [lmstudio-community](#)
- [EXL2 quants](#)

Me

- Blog - <https://smcleod.net>
 - Tooling ratings - ([Link](#))
 - Quantisation visualised - ([Link](#))
- LinkedIn - <https://www.linkedin.com/in/sammcj>
- Github - <https://github.com/sammcj>



Clients

- Anything LLM - <https://useanything.com>
- Open WebUI - <https://github.com/open-webui/open-webui>
- BigAGI - <https://github.com/enricoros/big-agi>
- BoltAI - <https://boltai.com>
- Msty - <https://msty.app>
- JanAI - <https://github.com/janhq/jan>
- LM Studio - <https://lmstudio.ai>
- InvokeAI - <https://invoke.com>

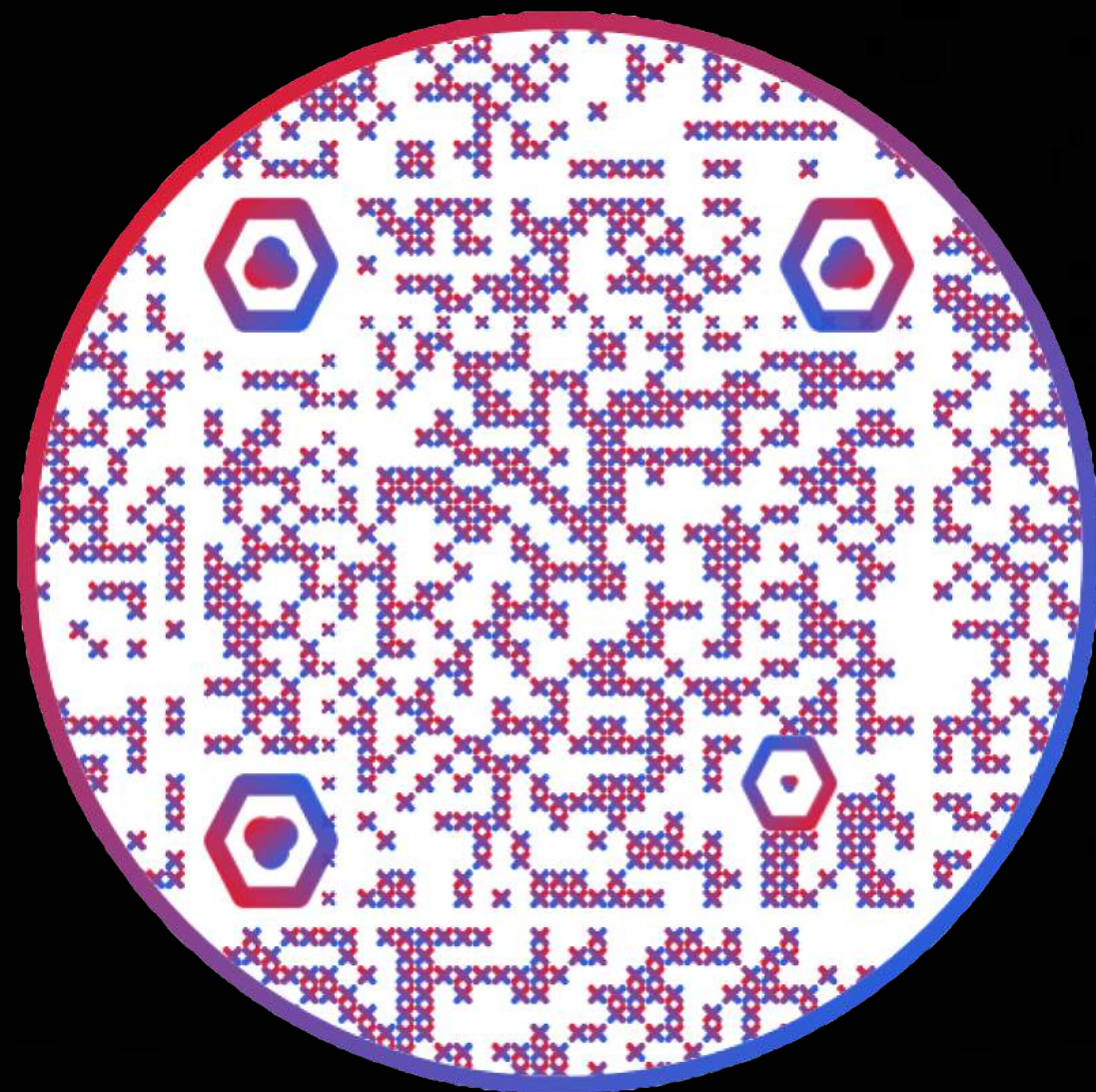
Servers & Server Tools

- Ollama - <https://ollama.com>
 - Gollama - <https://github.com/sammcj/gollama>
 - Ollama Grid Search - <https://github.com/dezoiito/ollama-grid-search>
- ExLlamaV2 - <https://github.com/turboderp/exllamav2>
 - TabbyAPI - <https://github.com/theroyallab/tabbyAPI>
 - Tabby Loader - <https://github.com/theroyallab/tabbyAPI-gradio-loader>
- mistral.rs - <https://github.com/EricLBuehler/mistral.rs>
- Text Generation WebUI - <https://github.com/oobabooga/text-generation-webui/>
- NVApi - <https://github.com/sammcj/nvapi>
- LlamaFactory - <https://github.com/hiyouga/LLaMA-Factory>
- HFDownloader - <https://github.com/bodaay/HuggingFaceModelDownloader>

Thanks!

When you find errors in this deck, please let me know!

"👋PRs Accepted👋"



<https://smcleod.net/2024/07/code-chaos-and-copilots-ai/llm-talk-july-2024/>

<https://smcleod.net>

Copyright © Sam McLeod 2024