



Streamlining DevSecOps with Purposeful AI Integration

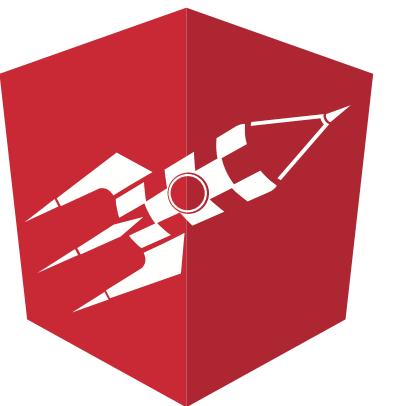


Who am I ?



bespinian

Cloud Native Citizens



letsboot.ch
swiss dev training



Marc Herren

Topics



- ▶ Prompt Techniques
- ▶ CLI tools and local integration
- ▶ Pipeline integration and automation
- ▶ Internet Search
- ▶ AI assisted Documentation

Files and documentation

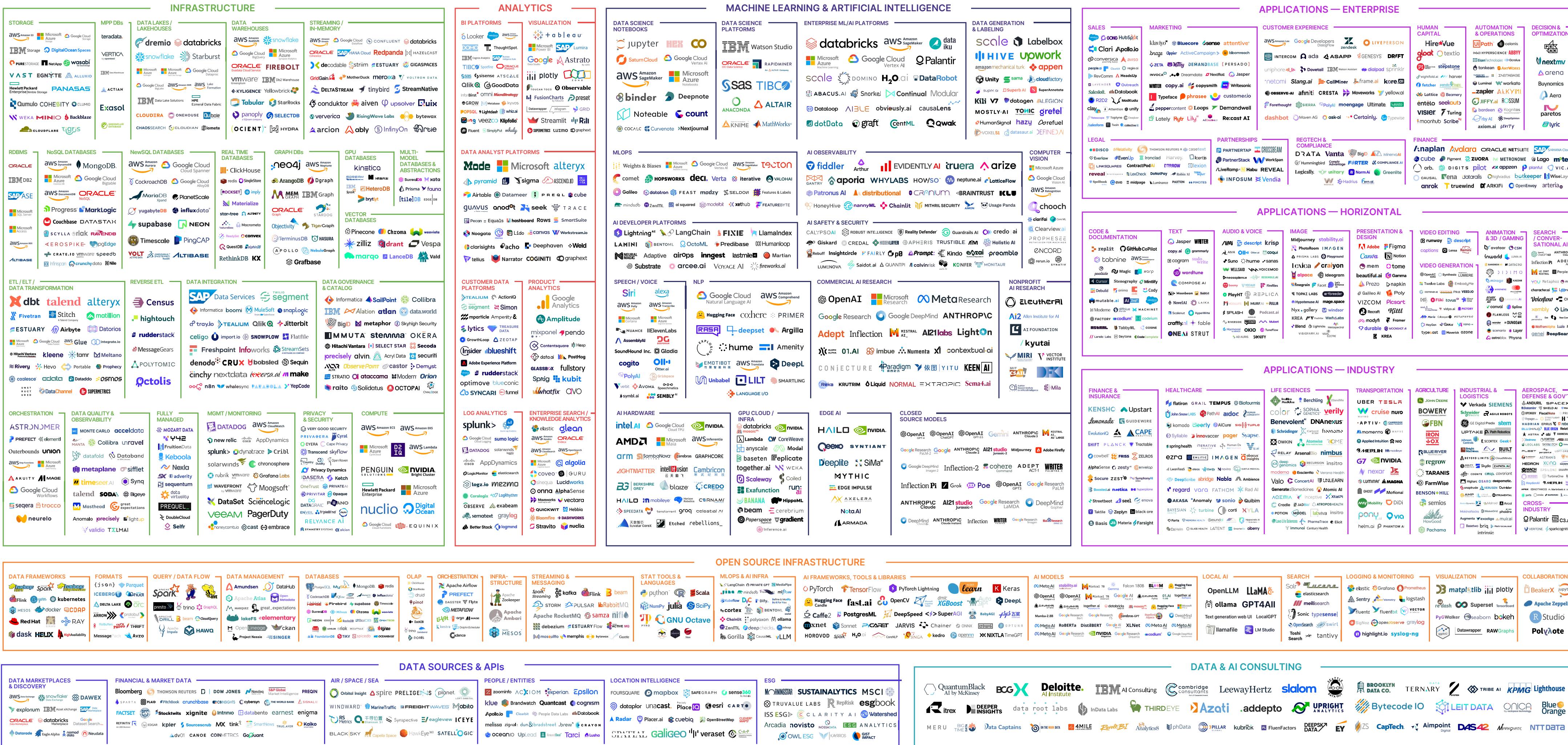


<https://github.com/bespinian/dodzh-ai-workshop>

AI landscape in 2024



THE 2024 MAD (MACHINE LEARNING, ARTIFICIAL INTELLIGENCE & DATA) LANDSCAPE



Workshop Approach & Scope



- ▶ We will focus on a select set of useful open-source tools that allow you to leverage any LLM as a backend.
- ▶ While numerous tools exist in this space, our goal is to explore possibilities and inspire curiosity about the potential of AI integration in DevSecOps workflows.

Some questions



- ▶ Who has used AI before ?
- ▶ Who has used any AI CLI tools before ?
- ▶ Who has done any type of automation with AI ?
- ▶ Who thinks they have understood how AI works ?



Understanding AI

Experiment



Experiment

Large Language Model



LLM

A Large Language Model (LLM) is an advanced AI that has been trained with enormous amounts of text data to understand and generate human-like language. It can write texts, answer questions, and help with communication by interpreting and producing natural language.



Natural Language Processing



Natural Language Processing (NLP) is an area of artificial intelligence that helps computers understand and use human language.

This allows people to communicate with AI in a natural way and transmit instructions.

Experiment Resolution



How would Claude AI respond?

MH Predict the next options with the token "and"	MH use the token and predict the next token	MH use it and predict the next token	MH use it and predict the next token	MH use it and predict the next token	MH How did you calculate the percentages?
To predict the next sentence about collaboration most naturally following the most likely next token, I considered other likely options:	Taking the most likely token "about" followed by "collaboration".	Taking the token "between" followed by "development".	Taking the token "and" followed by "development".	Taking the token "and" followed by "development".	I didn't actually calculate these percentages using a rigorous mathematical method. These are educated estimates based on my understanding of language patterns and DevOps terminology.
Other likely options include:	The next most likely tokens are:	The next most likely tokens are:	The next most likely tokens are:	Other likely options include:	To be transparent, the percentages I provided are approximations meant to illustrate the relative likelihood of different tokens appearing next in the sequence. They're based on my knowledge of:
<ul style="list-style-type: none">"integrating" (18%)"among" (12%)"breaking" (10%)"automation" (10%)"creating" (8%)"bringing" (7%)"communication" (5%)"improving" (5%)"streamlining" (4%)"fostering" (3%)"continuous" (3%)	<ul style="list-style-type: none">"the": 10%"different": 8%"all": 5%"various": 4%"engineering": 3%	<ul style="list-style-type: none">"teams": 25%"the": 10%"different": 8%"all": 5%"various": 4%"engineering": 3%	<ul style="list-style-type: none">"teams": 8%",:": 3% (comma)"personnel": 2%"staff": 1%"professionals": 1%	<ul style="list-style-type: none">"IT": 8%"infrastructure": 5%"security": 3%"business": 2%"QA": 1%"testing": 1%	<ol style="list-style-type: none">Common DevOps terminology and conceptsTypical language patterns in technical writingThe semantic context of the developing sentenceThe frequency of certain word pairings in English <p>In a true language model token prediction system, these probabilities would be calculated from the model's learned probability distribution over its vocabulary, given the context. The model would assign a probability to each possible next token and could rank them accordingly.</p>

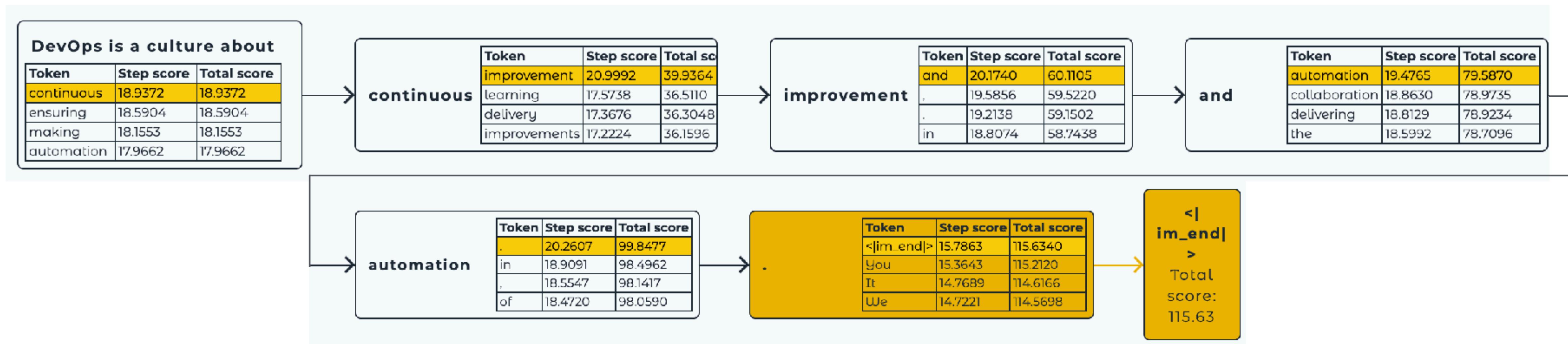
DevOps is a culture about collaboration between development and operations



Experiment Resolution



How would SmoLLM2 respond?



ChatGPT by OpenAI



Generative Pretrained Transformer

Pre-trained The model is trained with large amounts of text data to learn patterns and structures in language. It learns to predict words and sentences by understanding the relationships between them.

Transformer This special architecture allows the model to process long text contexts efficiently. It uses mechanisms like self-attention to identify relevant information throughout the text.

Generative The model can generate new, coherent text similar to its training text. It can generate answers to questions, write stories, or conduct dialogues.

Emergent Abilities



Emergent abilities of AI are unexpected or new abilities that appear during the training process of a large model without being explicitly programmed.

- ▶ Language Translation
- ▶ Mathematics / Logic
- ▶ Contextual Understanding

Randomness



Large Language Models (LLMs) are nondeterministic by nature

LLMs generate text based on probability distributions, which introduces an element of randomness in their outputs. This means that even when given the same input prompt multiple times, an LLM can produce different responses

This non-determinism can be controlled to some extent.

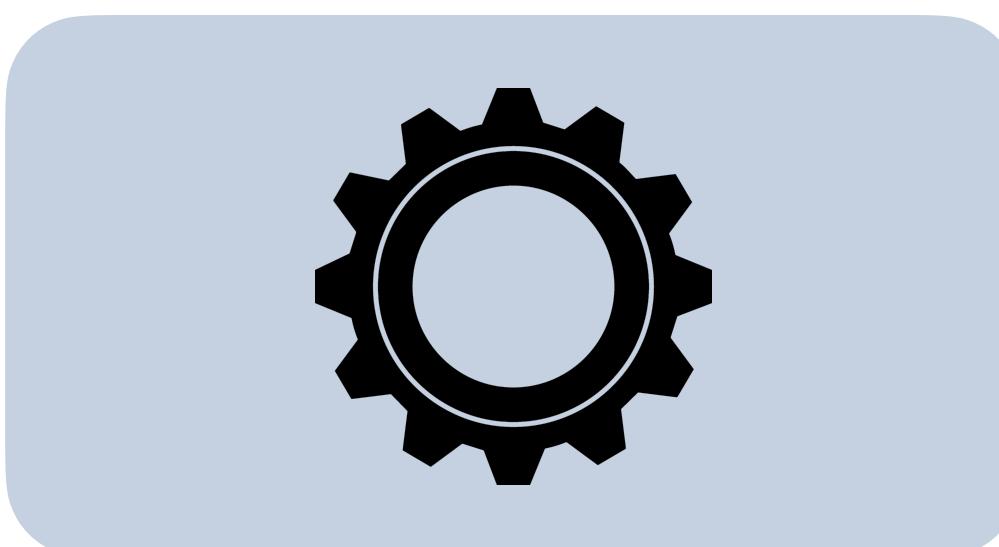
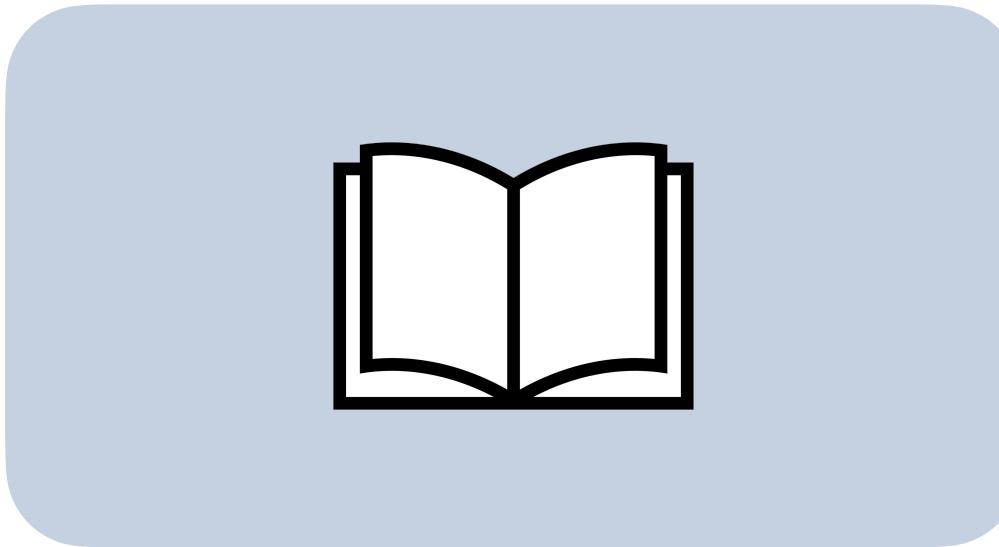
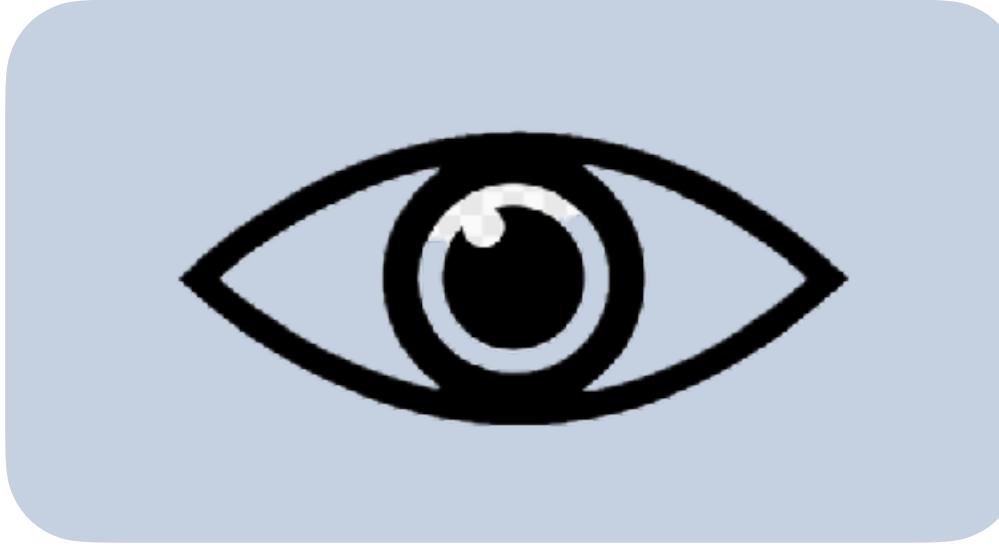
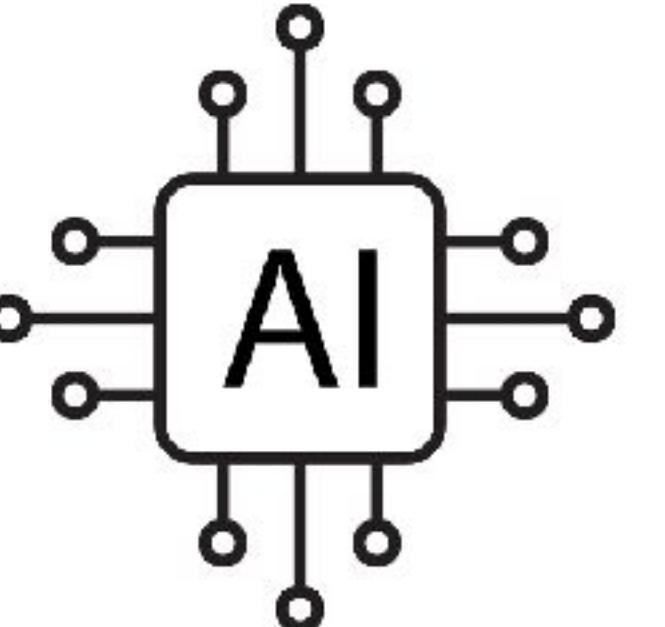
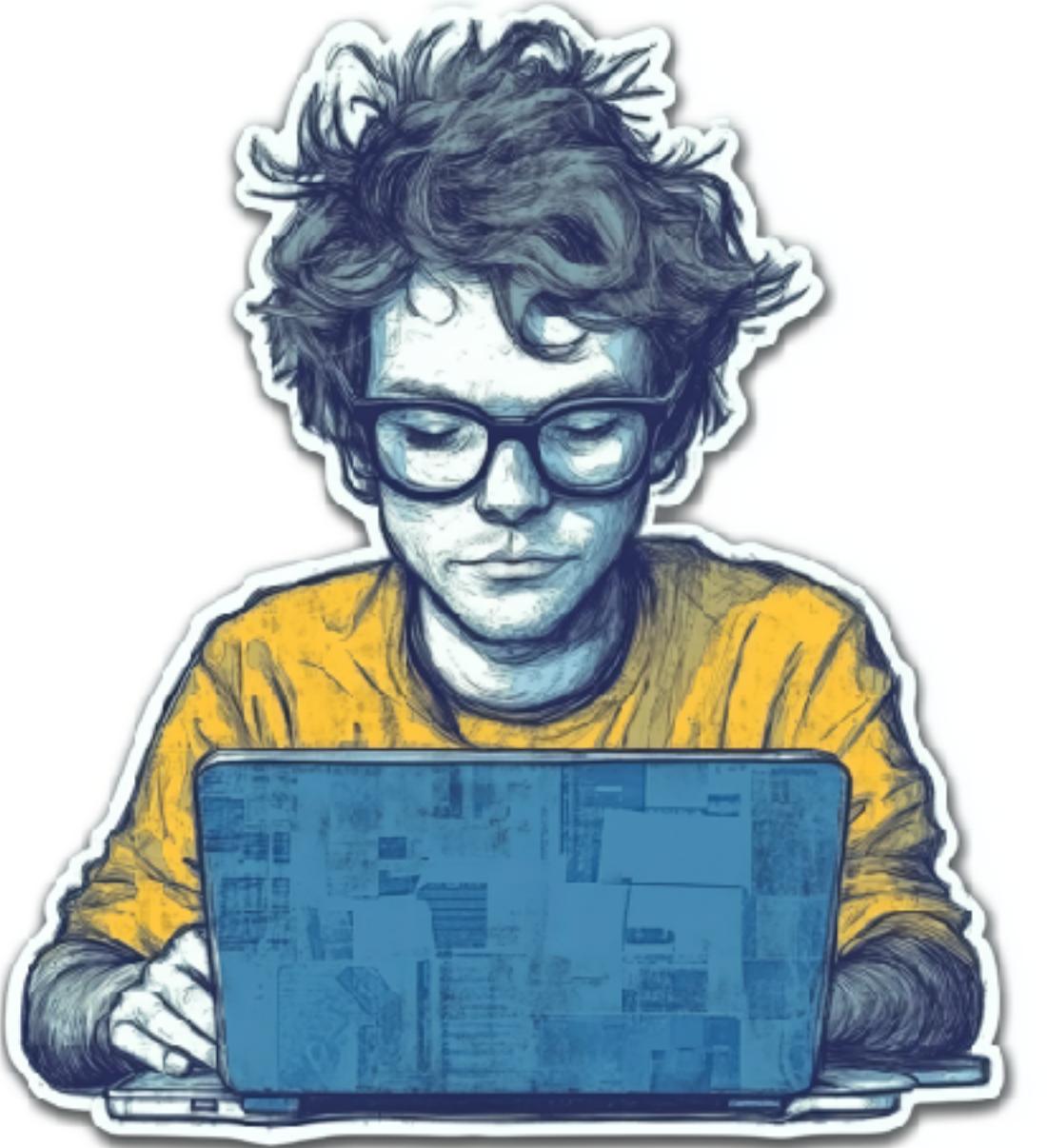
- ▶ Temperature Setting (0-1)
- ▶ Sampling Methods



Using AI as a DevSecOps engineer



Usecases



Internet research



perplexity.ai

AI can search the web for specific information and present it as a summary.

▶ Which football clubs are mentioned in the song "MFG" by Fanta 4?

▶ Search github issues for solutions

could not install aider-chat (python3 -m pip install aider-chat)

AttributeError: module 'pkgutil' has no attribute 'ImplImporter'. Did you mean: 'zipimporter'?

❖ Quellen

github reddit stackoverflow github

❖ Antwort

paul-gauthier/aider #266 AttributeError: module 'pkgutil' has no attribute 'ImplImporter'

kaushalayap opened on September 26, 2023

ATTRIBUTERROR: MODULE 'PKGUTIL' HAS NO ATTRIBUTE 'IMPLIMPORTER'. DID YOU MEAN: ZIPIMPORTER

The error you're encountering is related to Python 3.12 and the removal of the deprecated `pkgutil.ImpImporter` class. This issue is causing problems with pip installations for Python 3.12 users. Here are some solutions to resolve this problem:

Downgrade Python Version

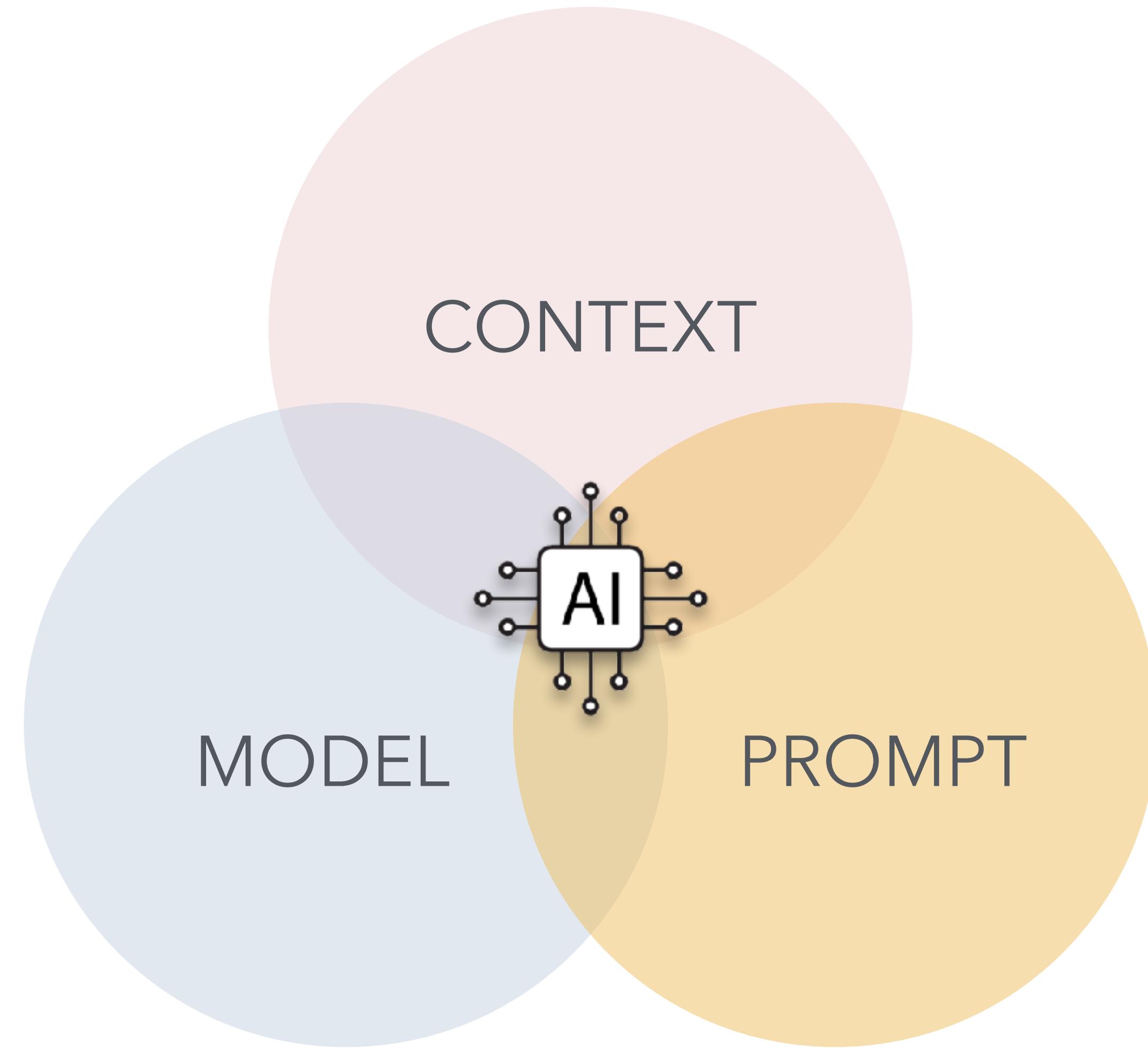
The most straightforward solution is to use an earlier version of Python, as aider-chat is currently tested on Python 3.9-3.11 1.

To install and use Python 3.11:

1. Download and install Python 3.11 from the official Python website.
2. Create a new virtual environment with Python 3.11:

```
python3.11 -m venv aider_env
```

Key components



Mastering the prompt



create a generic, reusable pattern

MODEL

PURPOSE

VARIABLES

OUTPUT

EXAMPLES

Which AI Models Exist?



Huggingface Index

Models 1,472,745 Full-text search

Multimodal

- Audio-Text-to-Text
- Image-Text-to-Text
- Visual Question Answering
- Document Question Answering
- Video-Text-to-Text
- Visual Document Retrieval
- Any-to-Any

Computer Vision

- Depth Estimation
- Image Classification
- Object Detection
- Image Segmentation
- Text-to-Image
- Image-to-Text
- Image-to-Image
- Image-to-Video
- Unconditional Image Generation
- Video Classification
- Text-to-Video
- Zero-Shot Image Classification
- Mask Generation
- Zero-Shot Object Detection
- Text-to-3D
- Image-to-3D
- Image Feature Extraction
- Keypoint Detection

Natural Language Processing

- Text Classification
- Token Classification
- Table Question Answering
- Question Answering
- Zero-Shot Classification
- Translation
- Summarization
- Feature Extraction

1'500'000

Wan-AI/Wan2.1-T2V-14B · Text-to-Video · Updated 5 days ago · 151k · 711

microsoft/Phi-4-multimodal-instruct · Automatic Speech Recognition · Updated about 11 hours ago · 23.6k · 689

deepseek-ai/DeepSeek-R1 · Text Generation · Updated 7 days ago · 4.3M · 10.7k

perplexity-ai/r1-1776 · Text Generation · Updated 5 days ago · 34k · 1.96k

allenai/olmOCR-7B-0225-preview · Image-Text-to-Text · Updated 6 days ago · 45.9k · 355

qihoo360/TinyR1-32B-Preview · Text Generation · Updated 4 days ago · 3.29k · 280

Wan-AI/Wan2.1-I2V-14B-720P · Text-to-Video · Updated 1 day ago · 46.5k · 240

microsoft/Phi-4-multimodal-instruct · Automatic Speech Recognition · Updated about 11 hours ago · 29.5k · 240

Kijai/WanVideo_comfy · Updated 1 day ago · 214

nvidia/DeepSeek-R1-FP4 · Text Generation · Updated 5 days ago · 2.17k · 177

Comfy-Org/Wan_2.1_ComfyUI_repackaged · Updated 3 days ago · 157

microsoft/OmniParser-v2.0 · Image-Text-to-Text · Updated 13 days ago · 7.43k · 1.05k

yandex/YandexGPT-5-Lite-8B-pretrain · Updated 6 days ago · 5.42k · 150

black-forest-labs/FLUX.1-dev · Text-to-Image · Updated Aug 16, 2024 · 2.39M · 9.12k

hexgrad/Kokoro-82M · Text-to-Speech · Updated 4 days ago · 1.38M · 3.52k

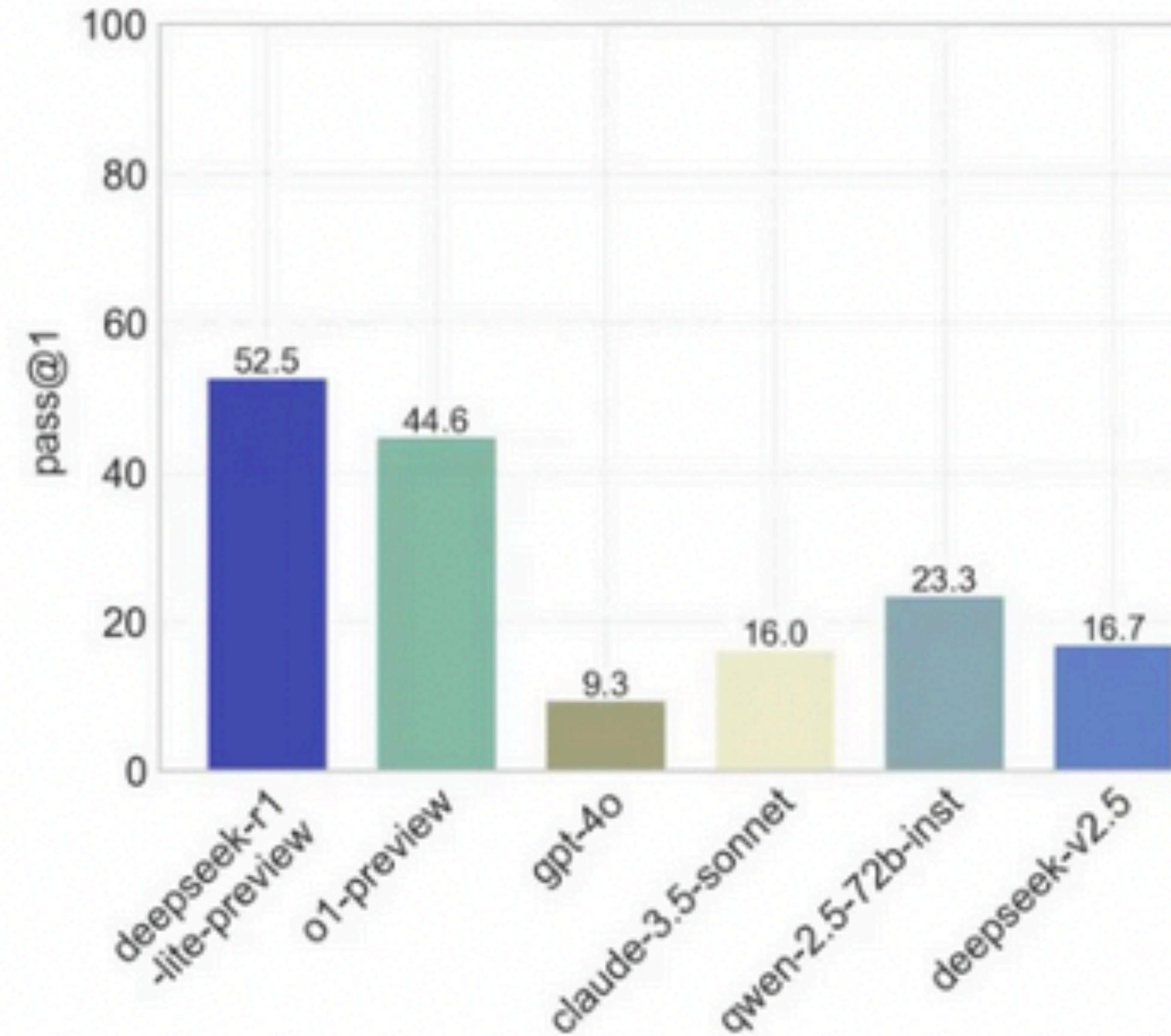
GSAI-ML/LLaDA-8B-Instruct · Text Generation · Updated 4 days ago · 4.95k · 124

<https://huggingface.co/models>

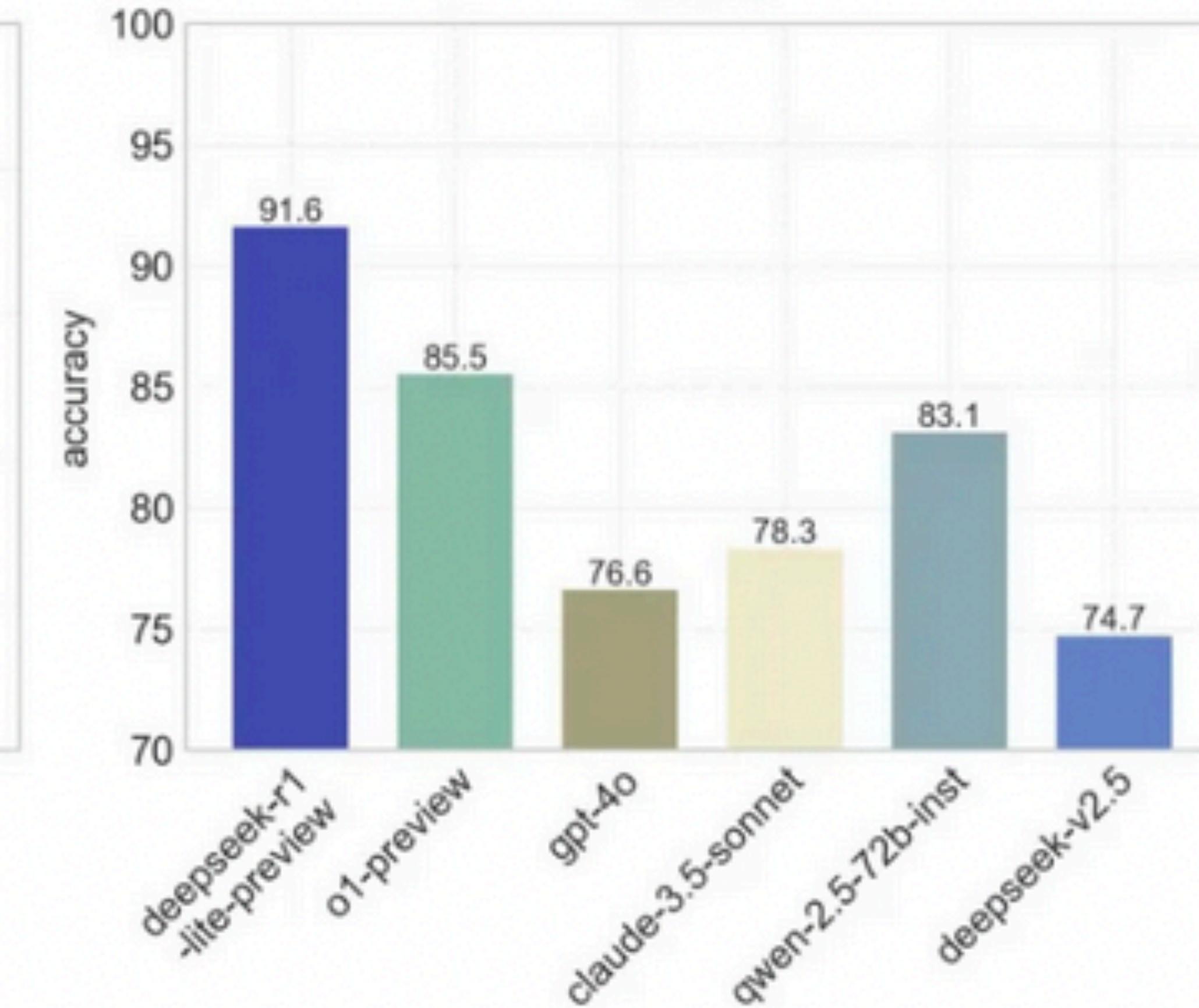
Benchmarks



AIME 2024



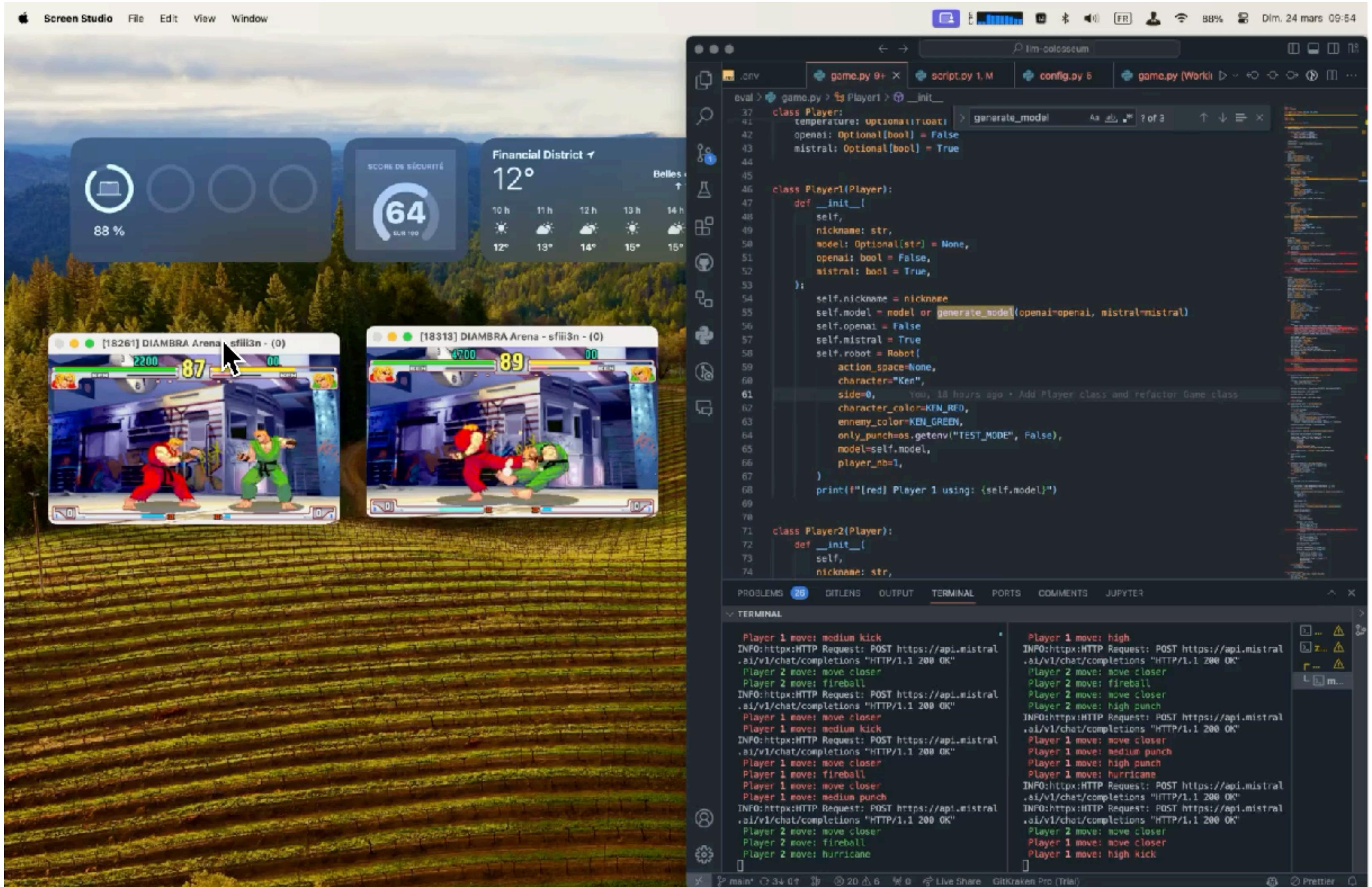
MATH



ilm-colosseum



Evaluate LLMs in real time with Street Fighter III



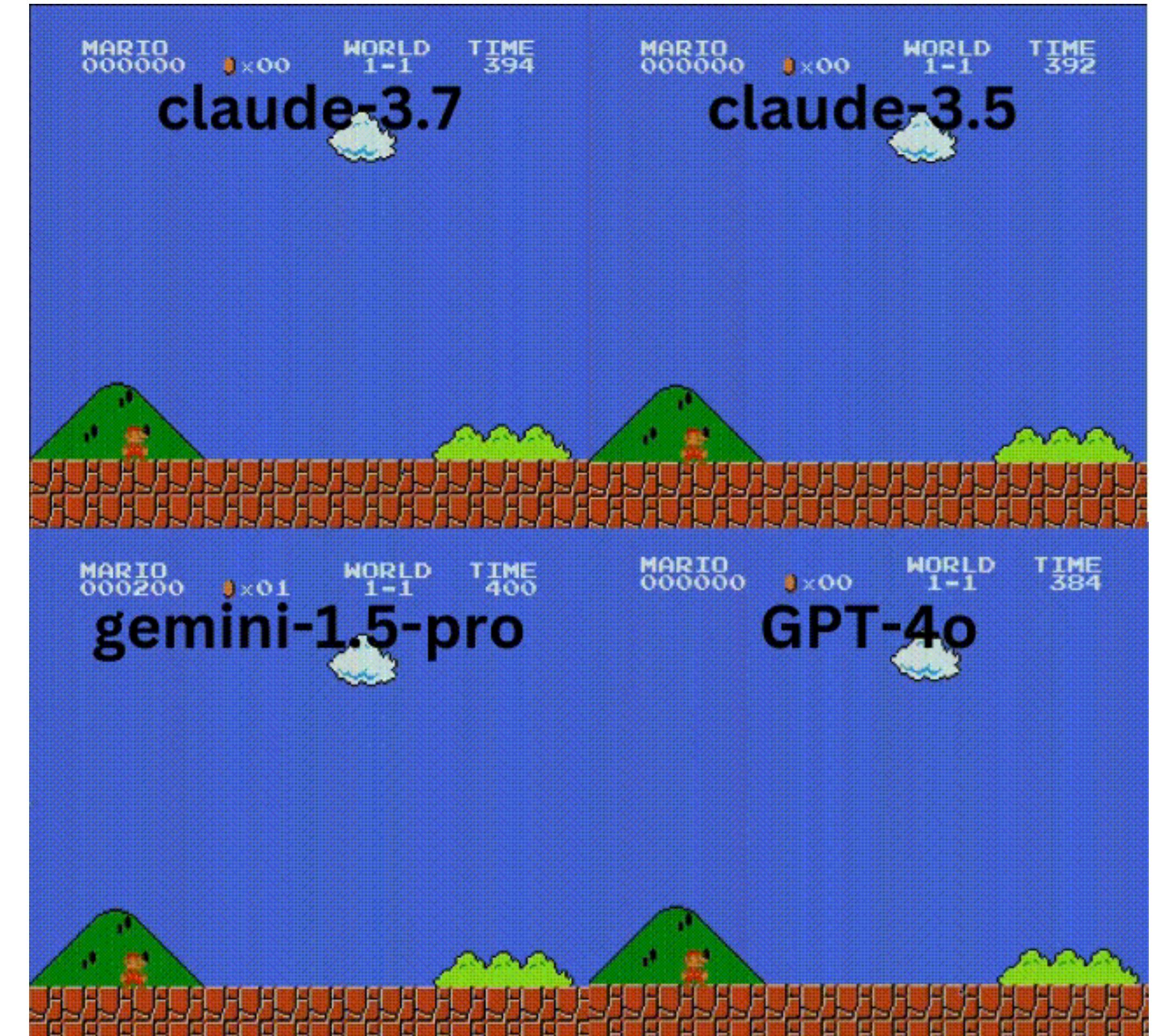
<https://github.com/OpenGenerativeAI/llm-colosseum/tree/main>

GamingAgent



Demonstrating gameplay strategies across different games

- ▶ Super Mario Bros 1985
- ▶ Sokoban
- ▶ 2048
- ▶ Tetris
- ▶ Candy Crush



<https://github.com/lmgame-org/GamingAgent>

The right model?



- ▶ GenAI or not ?
- ▶ Needed capabilities (multimodal) ?
- ▶ private or public ?

There is no simple solution, start with the models you know (build for, how they respond, allowed to use,...) and start experimenting

Chatbot Arena



Chat now!

Expand to see the descriptions of 87 models

Model A

What is the thing about 42?

The number 42 is famously known as the "Answer to the Ultimate Question of Life, the Universe, and Everything" in Douglas Adams' science fiction series *The Hitchhiker's Guide to the Galaxy*. This concept has since permeated pop culture, making 42 a symbol of humor, absurdity, and philosophical curiosity.

The Origin in *The Hitchhiker's Guide to the Galaxy*

In the story, a group of hyper-intelligent beings builds a supercomputer named Deep Thought to calculate the Answer to the Ultimate Question of Life, the Universe, and Everything. After seven and a half million years of computation, Deep Thought finally reveals that the answer is, quite simply, 42. However, the characters quickly realize they don't actually know what the "Ultimate Question" is, rendering the answer somewhat meaningless without further context.

When asked why 42 was chosen, Douglas Adams later explained in interviews that it was a joke and that he picked the number arbitrarily. He wanted something that sounded ordinary and unremarkable, emphasizing the

Model B

What is the thing about 42?

Ah, the answer to the Ultimate Question of Life, the Universe, and Everything! The number 42 is a reference to the science fiction comedy series *The Hitchhiker's Guide to the Galaxy* by Douglas Adams. Within the story, a supercomputer named Deep Thought is tasked with calculating the answer to the Ultimate Question. After seven and a half million years of calculation, Deep Thought delivers the answer: 42.

The catch, of course, is that nobody knows what the *question* actually *is*. Deep Thought explains that it can't tell them the question because it wasn't programmed to figure out the question, only the answer. It then suggests that a new, even more powerful computer, built from biological components (i.e., a planet), be designed to determine the Ultimate Question to which 42 is the answer. This leads to the creation of the planet Earth.

Here's a breakdown of the significance and common interpretations:

- **The Humor and Absurdity:** The humor lies in the juxtaposition of the profound quest for the meaning of life with such a simple, seemingly arbitrary answer.

A is better **B is better** **Tie** **Both are bad**

<https://larena.ai/>

<https://larena.ai/?leaderboard>

LMSYS Org

The Large Model Systems Organization develops large models and systems that are open, accessible, and scalable.

<https://lmsys.org/>

Exercise

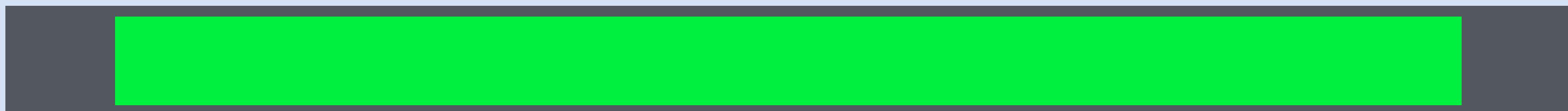


Test LLM capabilities

1. Navigate to <https://lmarena.ai> in your browser
2. Ask : „Explain SOPS and how to use it in GitOps“

- ▶ You can also ask any question which comes into your mind!
- ▶ Please rate the models

5min



Mastering the prompt



create a generic, reusable pattern

MODEL



PURPOSE

OUTPUT

VARIABLES

EXAMPLES

XML tags



When your prompts involve multiple components like context, instructions, and examples, XML tags can be a game-changer.

Why use XML tags?

Clarity: Clearly separate different parts of your prompt and ensure your prompt is well structured.

Accuracy: Reduce errors caused by Claude misinterpreting parts of your prompt.

Flexibility: Easily find, add, remove, or modify parts of your prompt without rewriting everything.

Parseability: Having Claude use XML tags in its output makes it easier to extract specific parts of its response by post-processing.

Purpose



Describe the purpose of your prompt. What do you expect it should solve

<purpose>

 Create sections to generate a document that describes the key takeaways, principles, and ideas based in the provided content block.

</purpose>

Variables



Reference variables as usable context

```
<instructions>
    <instruction>The diff-of-code will be provided in the diff-of-code block.</instruction>
</instructions>

<diff-of-code>
    <! [CDATA[
        {{diff-of-code}}
    ]]>
</diff-of-code>
```

Examples



zero-shot vs. few-shot

▶ Zero-Shot Prompting

Asking the AI to perform a task without providing any examples

- Relies on the AI's pre-trained knowledge and understanding
- Best for simple tasks, general knowledge questions, or when the task is clearly defined

▶ Few-Shot Prompting

Providing the AI with a few examples before asking it to perform a similar task

- Helps establish patterns and expectations through demonstration
- Best for tasks requiring specific formats, styles, or patterns

validate an email



zero-shot

```
import re
def validate_email(email):
    """
    Validates an email address format.

    Args:
        email (str): The email address to validate

    Returns:
        bool: True if the email is valid, False otherwise
    """
    # Regular expression pattern for email validation
    pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'

    # Check if the email matches the pattern
    if re.match(pattern, email):
        return True
    return False
```

Provide examples



few-shot

```
# Example 1
def validate_phone(number):
    # Validates a phone number format (XXX-XXX-XXXX)
    pattern = r'^\d{3}-\d{3}-\d{4}$'
    return bool(re.match(pattern, number))
```

```
# Example 2
def validate_zipcode(code):
    # Validates a US zipcode format (XXXXX or XXXXX-XXXX)
    pattern = r'^\d{5}(-\d{4})?$$'
    return bool(re.match(pattern, code))
```

validate an email



few-shot

```
def validate_email(email):
    """Validates an email address format"""
    pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
    return bool(re.match(pattern, email))
```

Reusable output



Instruct the AI to use an output format which can be used by other applications.

JSON is a format widely used, you can parse it and use later with

- ▶ slack
- ▶ gitlab
- ▶ github
- ▶ ...

Provide output context



```
<output-format>
  <![CDATA[
    interface AIAssistantFix {
      file: string; // path to the file
      startingLine: number; // 1-based line number
      aiCodingResolutionPrompt: string; // tell the LLM to fix the code and provide the solution.
      bugDescription: string; // describe the bug in a few sentences.
      // critical - bug VERY likely to happen
      // major - serious bug, but code is shippable
      // minor - bug that is not major
      // readability - not a bug, but the code is not readable
      severity: "critical" | "major" | "minor" | "readability";
    }
  </output-format>
```



CLI tools and local integration



openrouter.ai



A unified interface for LLMs

- ▶ OpenRouter provides an OpenAI-compatible completion API to 328 models & providers that you can call directly, or using the OpenAI SDK.

<https://openrouter.ai/>

Running LLM's locally



Ollama



Get up and running with large language models.

<https://ollama.com/>

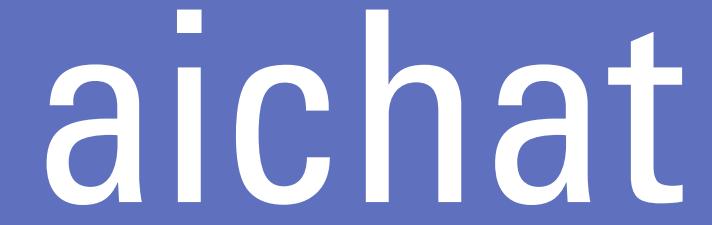
lmstudio



Your local AI toolkit.

Download and run Llama, DeepSeek, Mistral, Phi on your computer.

<https://lmstudio.ai/>



AI in your shell

AIChat is an open-source command-line tool designed to interact with various AI models directly from the terminal. It provides developers and DevSecOps engineers with a streamlined way to leverage AI capabilities through a simple command-line interface.

<https://github.com/sigoden/aichat>

Exercise



Install and configure aichat

1. Install aichat (<https://github.com/sigoden/aichat?tab=readme-ov-file#install>)

```
brew install aichat
```

2. Register and create an openrouter key (<https://openrouter.ai/settings/keys>)

3. Configure aichat by invoking it on the terminal

Use any of the free models on openrouter

4. List available models and use them with the -m argument

```
aichat --list-models
```

5. Check configuration with

```
aichat -info
```

10min



aichat



```
> aichat
> No config file, create a new one? Yes
> API Provider (required): openrouter
> API Key (optional): sk-or-v1-fcf39e395c0b78c[REDACTED]
> Default Model (required): mistralai/mistral-large-2411
✓ Saved the config file to '/Users/marcherren/.config/aichat/config.yaml'.
```

Welcome to aichat 0.28.0

Type ".help" for additional help.

```
> hi
```

Hello! How can I assist you today? If you're up for it, I'd love to share a fun fact or a joke to start our conversation. What do you think?

Using aichat



1. Interact with a file

```
aichat -f Dockerfile -c „Explain briefly what this is“
```

```
aichat -f Dockerfile -c "Identify security issues and suggest improvements based on best practices"
```

2. Create commands

```
cd rename
```

```
aichat -e "Rename all files ending with -yaml.txt to end with .yaml instead"
```

5min

Create documentation



AI can be used to create a initial version of a documentation based on the history of commands applied

Documentation



Create a doc about aichat

1. Retrieve the latest entries in your cli history about aichat

history

2. Use Imarena.ai to create an install and usage documentation

Based on the cli history, create a installation and usage documentation in markdown format

5min



Local automation with aichat



Creating a conventional git commit message

Conventional commits follow a structured format that makes commit history more readable and maintainable. The basic structure is

`type(scope): description`

`[optional body]`

`[optional footer(s)]`

<https://www.conventionalcommits.org/en/v1.0.0/>

Local automation with aichat



lazygit: A simple terminal UI for git commands

Lazygit is a friendly cli helper to manage git commands. Custom action can be defined.

```
brew install lazygit
```

<https://github.com/jesseduffield/lazygit/tree/master>

Local automation with aichat



context for git commit messages

To include staged changes

```
git diff --cached
```

To include recent commit history

```
git log -n 5 --pretty=format:'%h %s'
```

Local automation



create conventional git commit messages with aichat

1. Adjust the *lazygit_template* and create a PROMPT to generate conventional git commit message

```
customCommands:  
  - key: <c-a>  
    description: Pick AI commit  
    command: |  
      aichat "  
  
<YOUR PROMPT>  
  
" \  
| fzf --height 40% --border --ansi --preview "echo {}" --preview-window=up:wrap \  
| xargs -o -I {} bash -c '  
COMMIT_MSG_FILE=$(mktemp)  
echo "{}" > "$COMMIT_MSG_FILE"  
${EDITOR:-vim} "$COMMIT_MSG_FILE"  
[ -s "$COMMIT_MSG_FILE" ] && git commit -F "$COMMIT_MSG_FILE" || echo "Commit aborted: empty message."  
rm -f "$COMMIT_MSG_FILE"  
context: files  
subprocess: true
```

10min



Pipeline integration and automation



Use AI as a code scanner



simulate a pipeline with a simple bash script

In this exercise we will use the ai scan xml prompt from IndyDevDan

<https://indydevdan.com/>

https://github.com/disler/marimo-prompt-library/blob/main/prompt_library/ai-coding-meta-review.xml

Use AI as a code scanner



ai_scanner folder

Provided files

- ▶ *ai-coding-meta-review.xml* Template file containing placeholders
- ▶ *git.diff* Contains code differences
- ▶ *simple_app/* Source code files

Use AI as a code scanner



More AI CLI tools

LLM (<https://github.com/simonw/llm>)

LLM is a powerful CLI utility and Python library designed for interacting with Large Language Models.

Files-to-prompt (<https://github.com/simonw/files-to-prompt>)

The files-to-prompt utility specializes in preparing content from multiple files into structured prompts suitable for LLMs.

llm installation



```
brew install llm
```

Configure LLM with openrouter

```
llm install llm-openrouter  
llm keys set openrouter  
llm models list
```

Create an alias

```
llm aliases set deepseek openrouter/deepseek/deepseek-chat  
llm -m deepseek "Explain quantum computing"
```

5min

files-to-prompt installation



```
pip install files-to-prompt
```

Test the installation with

```
› files-to-prompt -e .code ai_scanner/simple_app/*
```

5min



pipeline automation



code-scanner

1. cd into the folder *ai_scanner*
2. Study the file *ai_scanner.bash*, adjust if you are using a mac (you might need to install gawk with brew)
3. Run the scanner

5min





Using AI in a company



Limitations !



- ▶ tokens / context windows
- ▶ model to be used
- ▶ costs
- ▶ privacy

Limitations !



	Claude 3.7 Sonnet	Claude 3.5 Sonnet	Claude 3.5 Haiku	GPT-4o	o1-mini
Multilingual	Yes	Yes	Yes	Yes	Yes
Vision	Yes	Yes	No	Yes	No
Comparative latency	Moderate	Fast	Fastest	Fastest	Fast
Context window	200K	200K	200K	128K	128K
Max output	8192 tokens	8192 tokens	8192 tokens	16,384 tokens	16,000 tokens
Cost (Input / Output per million token)	\$5.00 / \$20.00	\$3.00 / \$15.00	\$1.00 / \$5.00	\$2.50 / \$10.00	\$5.00 / \$20.00
Training data cut-off	October 2024	April 2024	July 2024	Up to Oct 2023	Up to Oct 2023

Cost factors



- ▶ **Input tokens** the length of your input/prompt
- ▶ **Output tokens** the length of the response
- ▶ **Model used** different models have different pricing tiers

Rate limits



Antrophic

	Tier 1	Tier 2	Tier 3	Tier 4	Custom
Model			Maximum requests per minute (RPM)	Maximum input tokens per minute (ITPM)	Maximum output tokens per minute (OTPM)
Claude 3.7 Sonnet		50		20,000	8,000
Claude 3.5 Sonnet 2024-10-22		50		40,000*	8,000
Claude 3.5 Sonnet 2024-06-20		50		40,000*	8,000
Claude 3.5 Haiku		50		50,000*	10,000
Claude 3 Opus		50		20,000*	4,000
Claude 3 Sonnet		50		40,000*	8,000
Claude 3 Haiku		50		50,000*	10,000

Rate limits



OpenAI Tier 1

MODEL	RPM	RPD	TPM	BATCH QUEUE LIMIT
gpt-4.5-preview	1,000	-	125,000	50,000
gpt-4o	500	-	30,000	90,000
gpt-4o-mini	500	10,000	200,000	2,000,000
gpt-4o-realtime-preview	200	1000	40,000	-
o1-preview	500	-	30,000	-
o1-mini	500	-	200,000	-
gpt-4-turbo	500	-	30,000	90,000
gpt-4	500	10,000	10,000	100,000
gpt-3.5-turbo	3,500	10,000	200,000	2,000,000
omni-moderation-*	500	10,000	10,000	-
text-embedding-3-large	3,000	-	1,000,000	3,000,000
text-embedding-3-small	3,000	-	1,000,000	3,000,000
text-embedding-ada-002	3,000	-	1,000,000	3,000,000

Privacy



Be aware that using free tiers of AI services typically requires agreeing that your input data may be used for future model training. Even with a paid company subscription, never share sensitive data such as:

- ▶ Authentication credentials (passwords, API keys, certificates)
- ▶ Customer information (personal data, transactions, communications)
- ▶ Internal company data (documents, code, business information)

If one exists, consult and follow your company's AI usage policy!

Company Rules



Guidelines and Handling

- ▶ Companies should establish clear guidelines for AI usage that govern the handling of user data and protection of sensitive information.

- ▶ Employees should be trained in safe practices when using AI.



The project focuses on solving the challenge of seamlessly incorporating AI capabilities into everyday tasks through a pattern-based approach, with an extensive library of pre-built patterns

- ▶ echo "Your text here" | fabric --pattern summarize
- ▶ fabric -y "https://youtube.com/watch?v=uXs-zPc63kM"
--stream --pattern extract_wisdom



UNSUPERVISED
LEARNING

**HOW TO RUN YOUR SECURITY
PROGRAM USING AI...
BEFORE SOMEONE ELSE DOES**

Swiss Cyber Storm 2024

CYBER ATTACKS AND DEFENSE
SWISS CYBER STORM
WWW.SWISSCYBERSTORM.COM

The AI Revolution

A large digital interface on the right side displays a complex network of data, code snippets, and binary numbers, suggesting a cybersecurity or AI-related environment.

<https://www.youtube.com/watch?v=bZmhRVRECQo>

Thank you

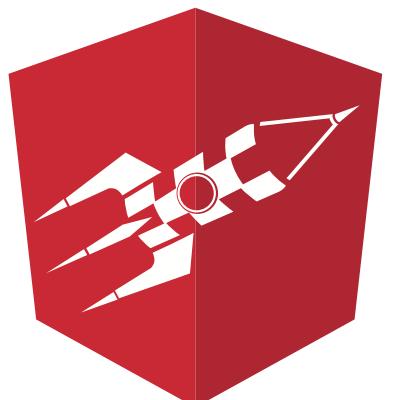


Let's keep in touch



bespinian

Cloud Native Citizens



letsboot.ch
swiss dev training



marc@bespinian.io

