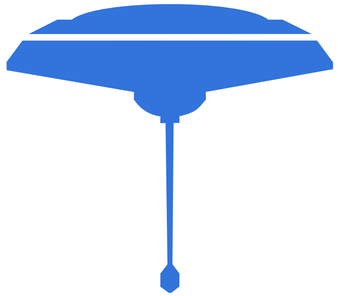
Streamlining DevSecOps with Purposeful AI Integration

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

INFRASTRUCTURE

STORAGE
MPP DBs
DATA LAKES / LAKEHOUSES
DATA WAREHOUSES
STREAMING / IN-MEMORY
RDBMS
NoSQL DATABASES
NewSQL DATABASES
REAL-TIME DATABASES
GRAPH DBs
GPU DATABASES
MULTI-MODEL DATABASES & ABSTRACTIONS
VECTOR DATABASES
ETL / ELT / DATA TRANSFORMATION
REVERSE ETL
DATA INTEGRATION
DATA GOVERNANCE & CATALOG
ORCHESTRATION
DATA QUALITY & OBSERVABILITY
FULLY MANAGED
MGMT / MONITORING
PRIVACY & SECURITY
COMPUTE

ANALYTICS

BI PLATFORMS
VISUALIZATION
DATA ANALYST PLATFORMS
CUSTOMER DATA PLATFORMS
PRODUCT ANALYTICS
LOG ANALYTICS
ENTERPRISE SEARCH / KNOWLEDGE ANALYTICS

MACHINE LEARNING & ARTIFICIAL INTELLIGENCE

DATA SCIENCE NOTEBOOKS
DATA SCIENCE PLATFORMS
ENTERPRISE ML/AI PLATFORMS
DATA GENERATION & LABELING
MLOPS
AI OBSERVABILITY
COMPUTER VISION
AI DEVELOPER PLATFORMS
AI SAFETY & SECURITY
SPEECH / VOICE
NLP
COMMERCIAL AI RESEARCH
NONPROFIT AI RESEARCH
AI HARDWARE
GPU CLOUD / INFRA
EDGE AI
CLOSED SOURCE MODELS

APPLICATIONS — ENTERPRISE

SALES
MARKETING
CUSTOMER EXPERIENCE
HUMAN CAPITAL
AUTOMATION & OPERATIONS
DECISION & OPTIMIZATION
LEGAL
PARTNERSHIPS
REGTECH & COMPLIANCE
FINANCE

APPLICATIONS — HORIZONTAL

CODE & DOCUMENTATION
TEXT
AUDIO & VOICE
IMAGE
PRESENTATION & DESIGN
VIDEO EDITING
ANIMATION & 3D / GAMING
SEARCH / CONVERSATIONAL AI
VIDEO GENERATION

APPLICATIONS — INDUSTRY

FINANCE & INSURANCE
HEALTHCARE
LIFE SCIENCES
TRANSPORTATION
AGRICULTURE
INDUSTRIAL & LOGISTICS
AEROSPACE, DEFENSE & GOVT
CROSS-INDUSTRY

OPEN SOURCE INFRASTRUCTURE

DATA FRAMEWORKS
FORMATS
QUERY / DATA FLOW
DATA MANAGEMENT
DATABASES
OLAP
ORCHESTRATION
INFRASTRUCTURE
STREAMING & MESSAGING
STAT TOOLS & LANGUAGES
MLOPS & AI INFRA
AI FRAMEWORKS, TOOLS & LIBRARIES
AI MODELS
LOCAL AI
SEARCH
LOGGING & MONITORING
VISUALIZATION
COLLABORATION

DATA SOURCES & APIs

DATA MARKETPLACES & DISCOVERY
FINANCIAL & MARKET DATA
AIR / SPACE / SEA
PEOPLE / ENTITIES
LOCATION INTELLIGENCE
ESG

DATA & AI CONSULTING

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

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.

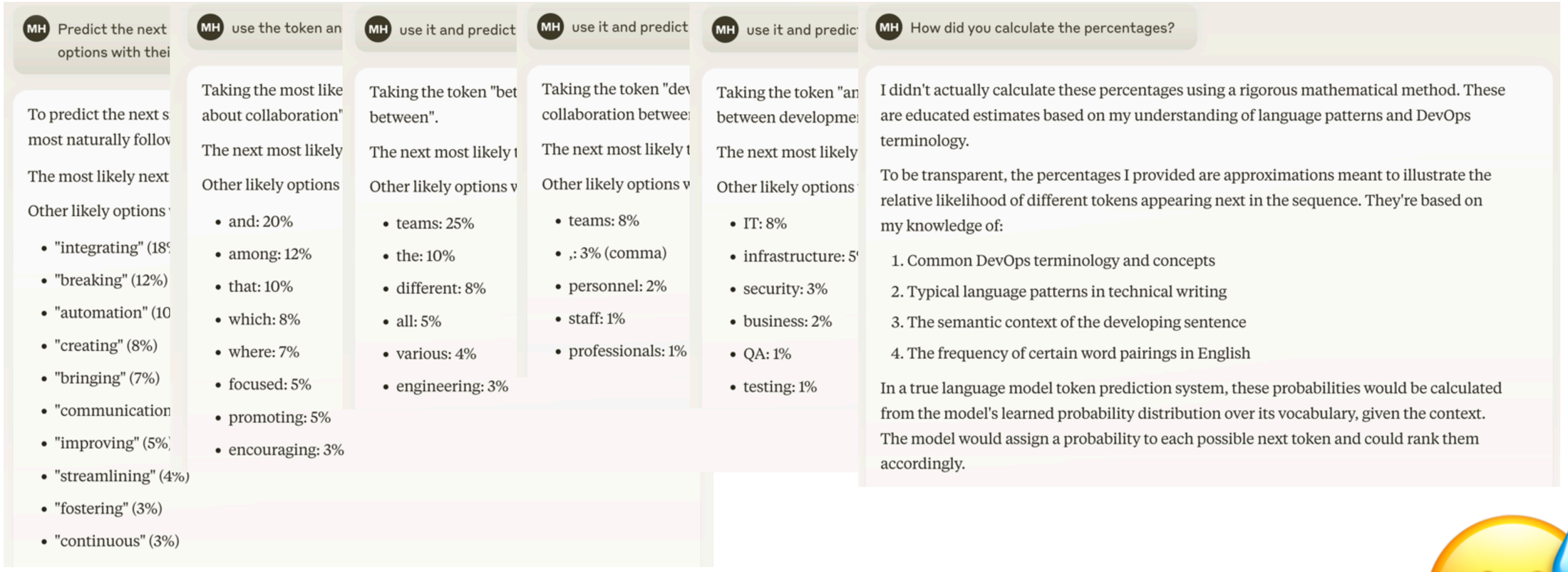
Natural Language Processing

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

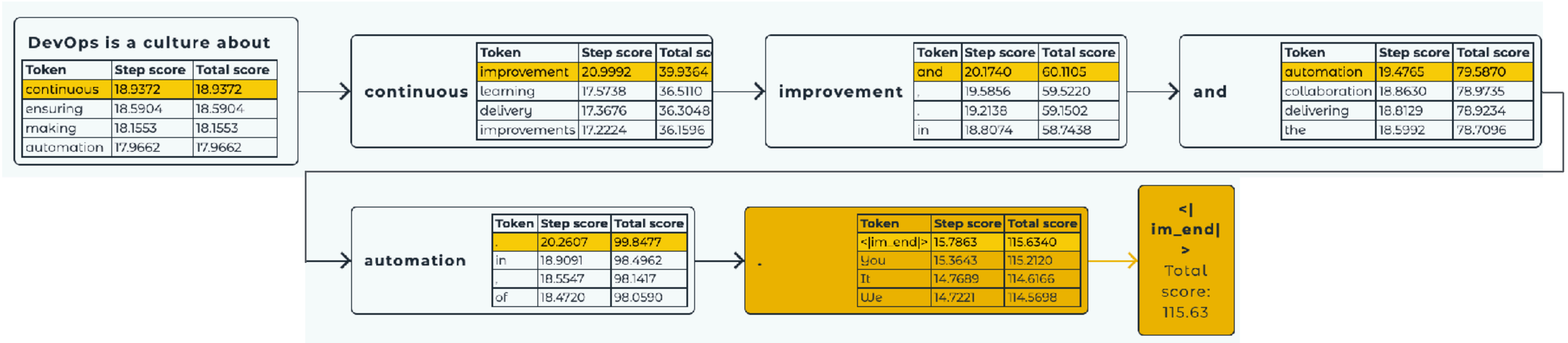
How would Claude AI respond?



DevOps is a culture about collaboration between development and operations

Experiment Resolution

How would SmolLM2 respond?



https://agents-course-decoding-visualizer.hf.space

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

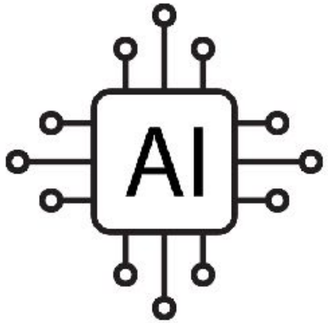
This non-determinism can be controlled to some extent.

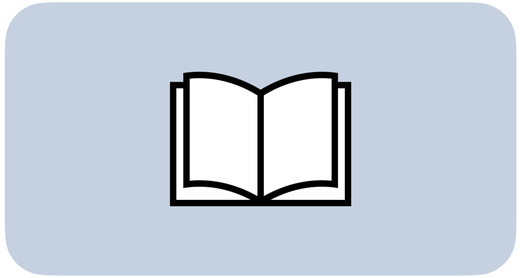
- Temperature Setting (0-1)
- Sampling Methods

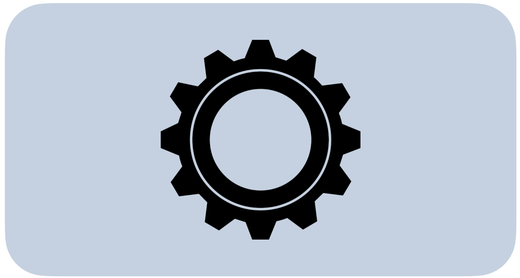
https://arxiv.org/pdf/2308.02828

Using AI as a DevSecOps engineer

Usecases





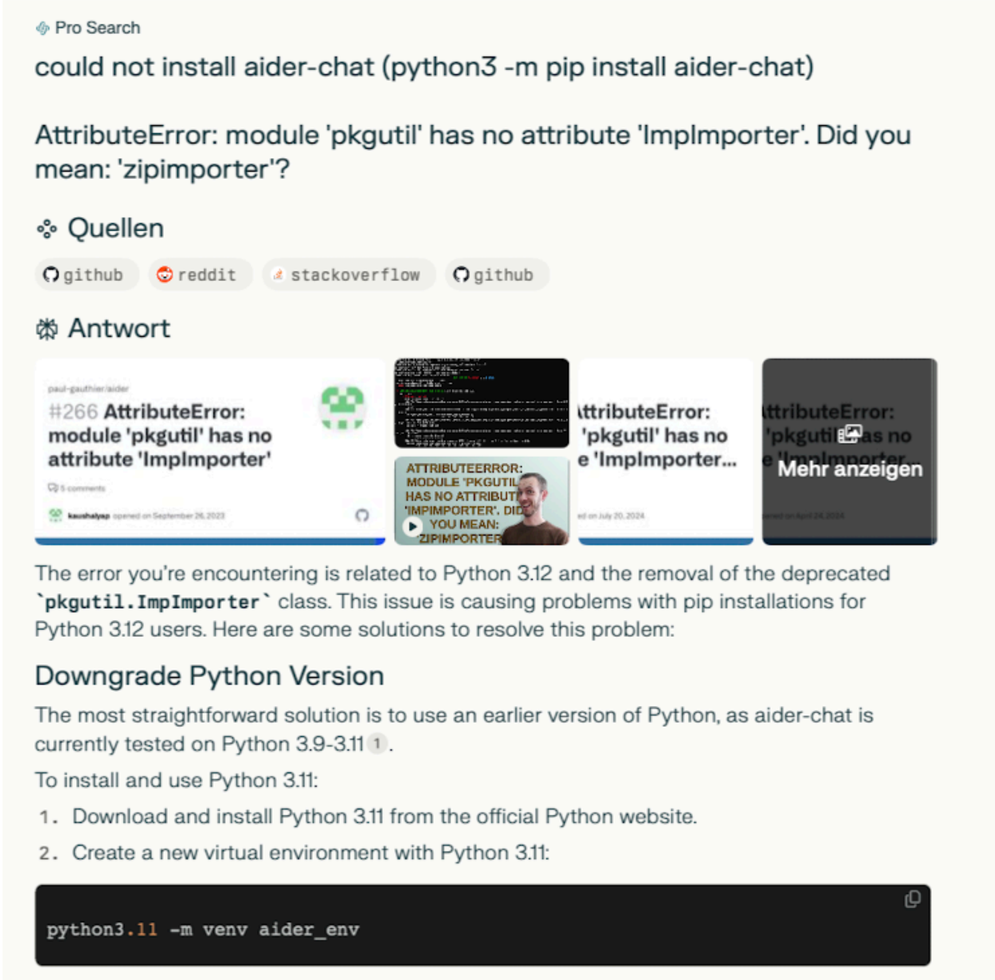


Internet research

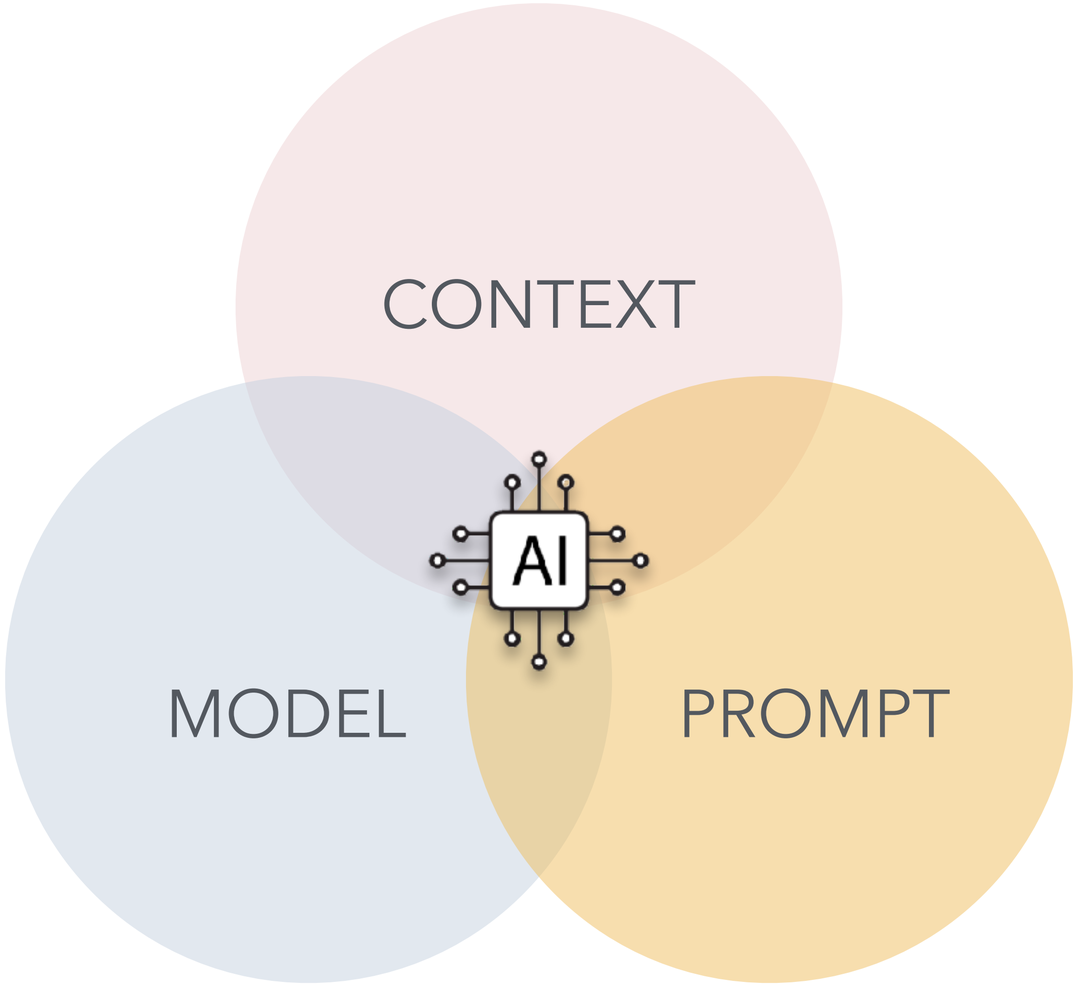
perplexity.ai

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



Key components



Mastering the prompt

create a generic, reusable pattern

MODEL

PURPOSE

VARIABLES

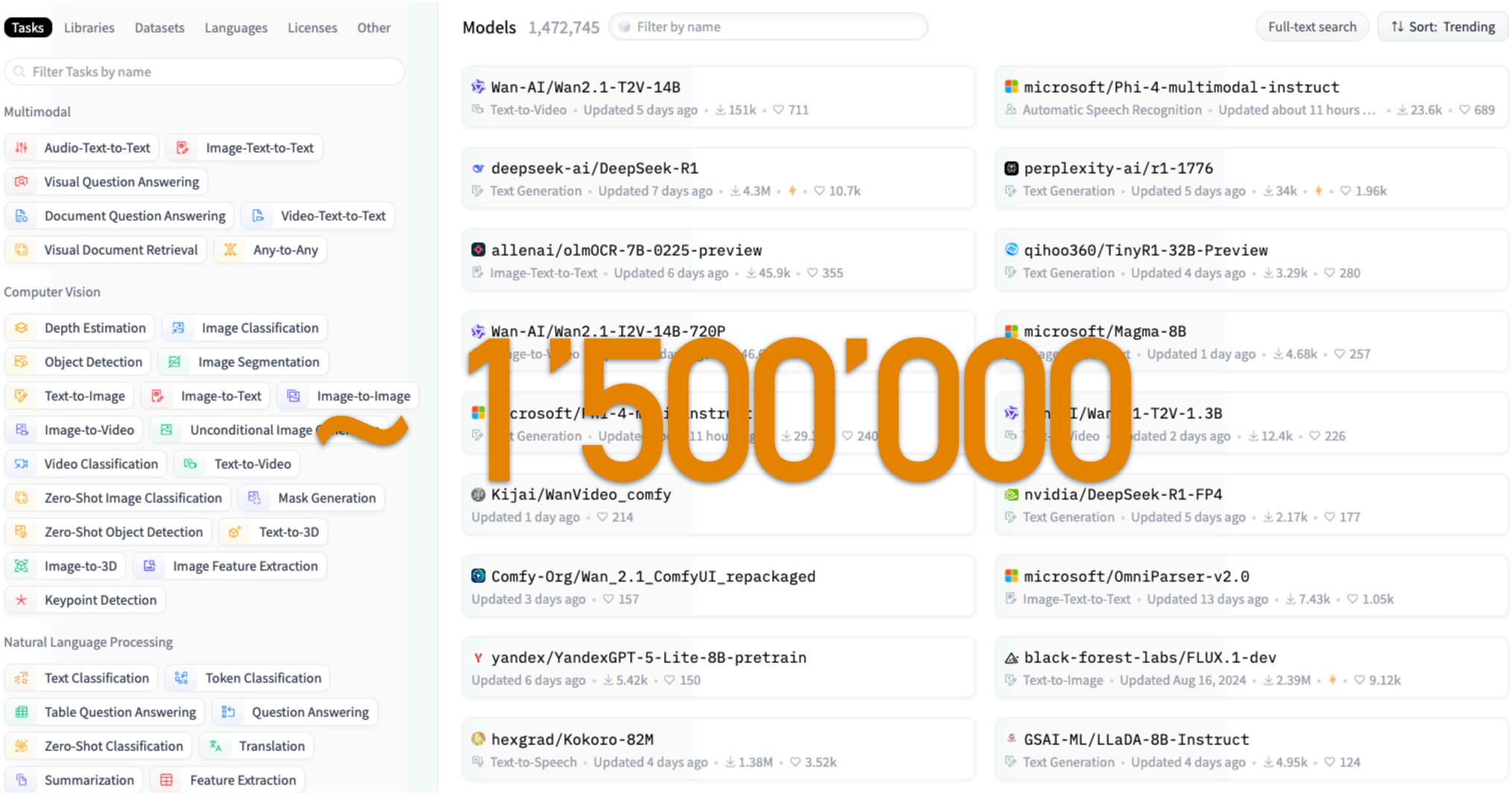
OUTPUT

EXAMPLES

IndyDevDan: Master the Prompt

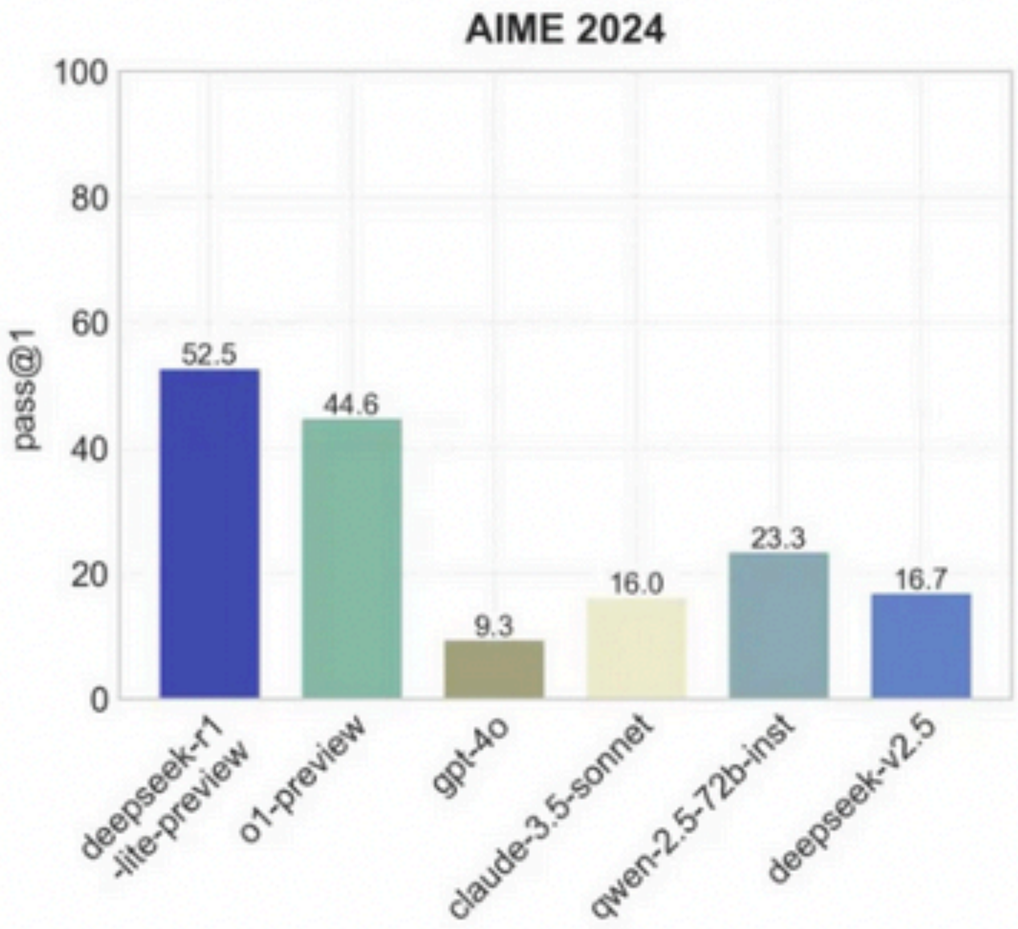
Which AI Models Exist?

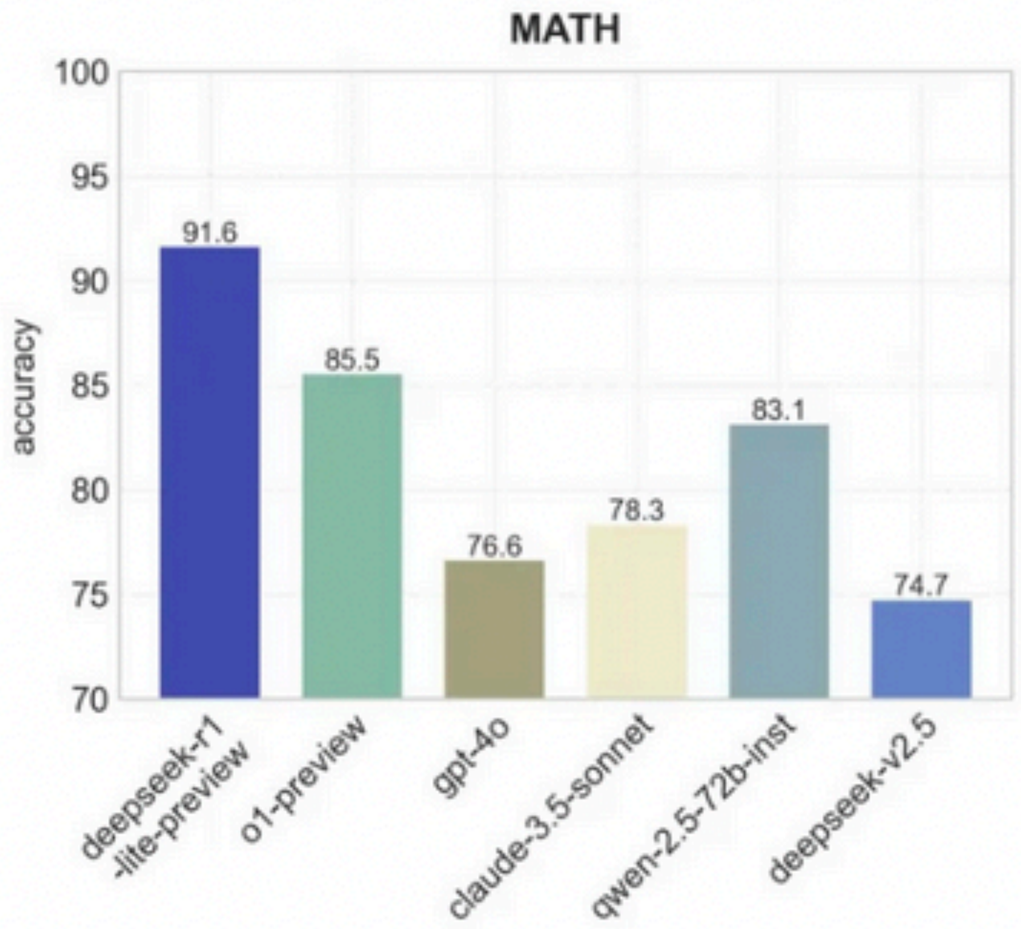
Huggingface Index



https://huggingface.co/models

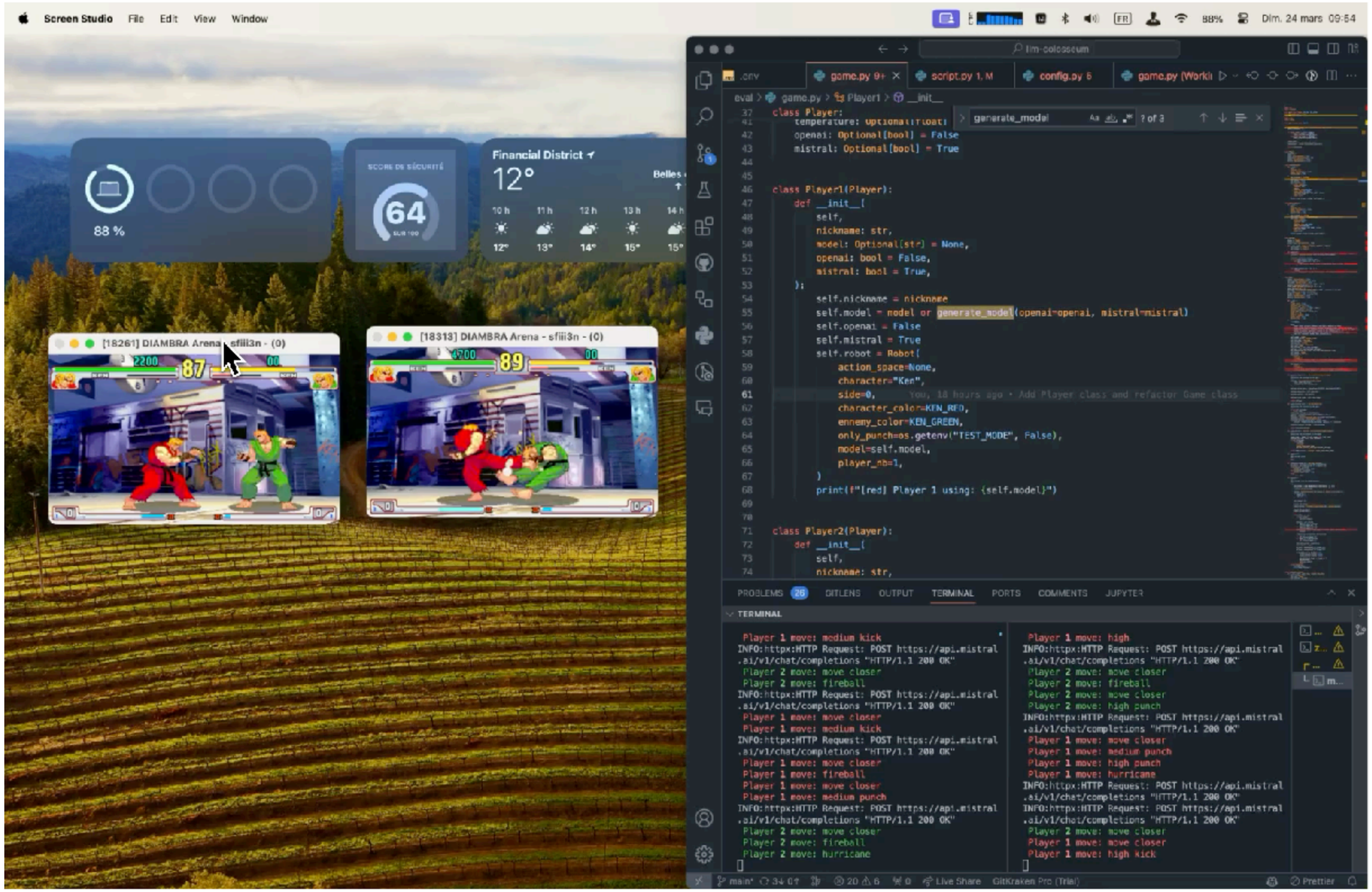
Benchmarks


AIME 2024
pass@1
100
80
60
40
20
0
52.5
44.6
9.3
16.0
23.3
16.7
deepseek-r1
-lite-preview
o1-preview
gpt-4o
claude-3.5-sonnet
qwen-2.5-72b-inst
deepseek-v2.5


MATH
accuracy
100
95
90
85
80
75
70
91.6
85.5
76.6
78.3
83.1
74.7
deepseek-r1
-lite-preview
o1-preview
gpt-4o
claude-3.5-sonnet
qwen-2.5-72b-inst
deepseek-v2.5

llm-colosseum

Evaluate LLMs in real time with Street Fighter III

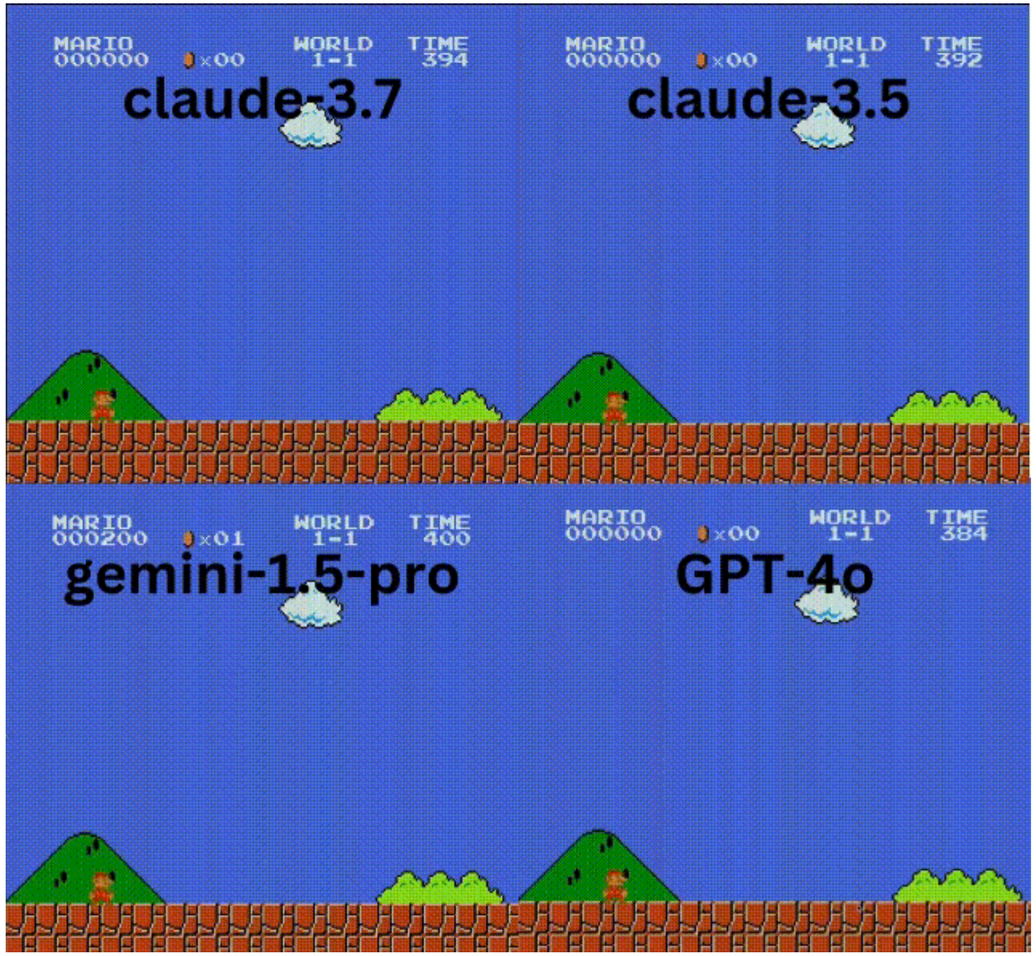


https://github.com/OpenGenerativeAI/llm-colosseum/tree/main

GamingAgent

Demonstrating gameplay strategies across different games

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



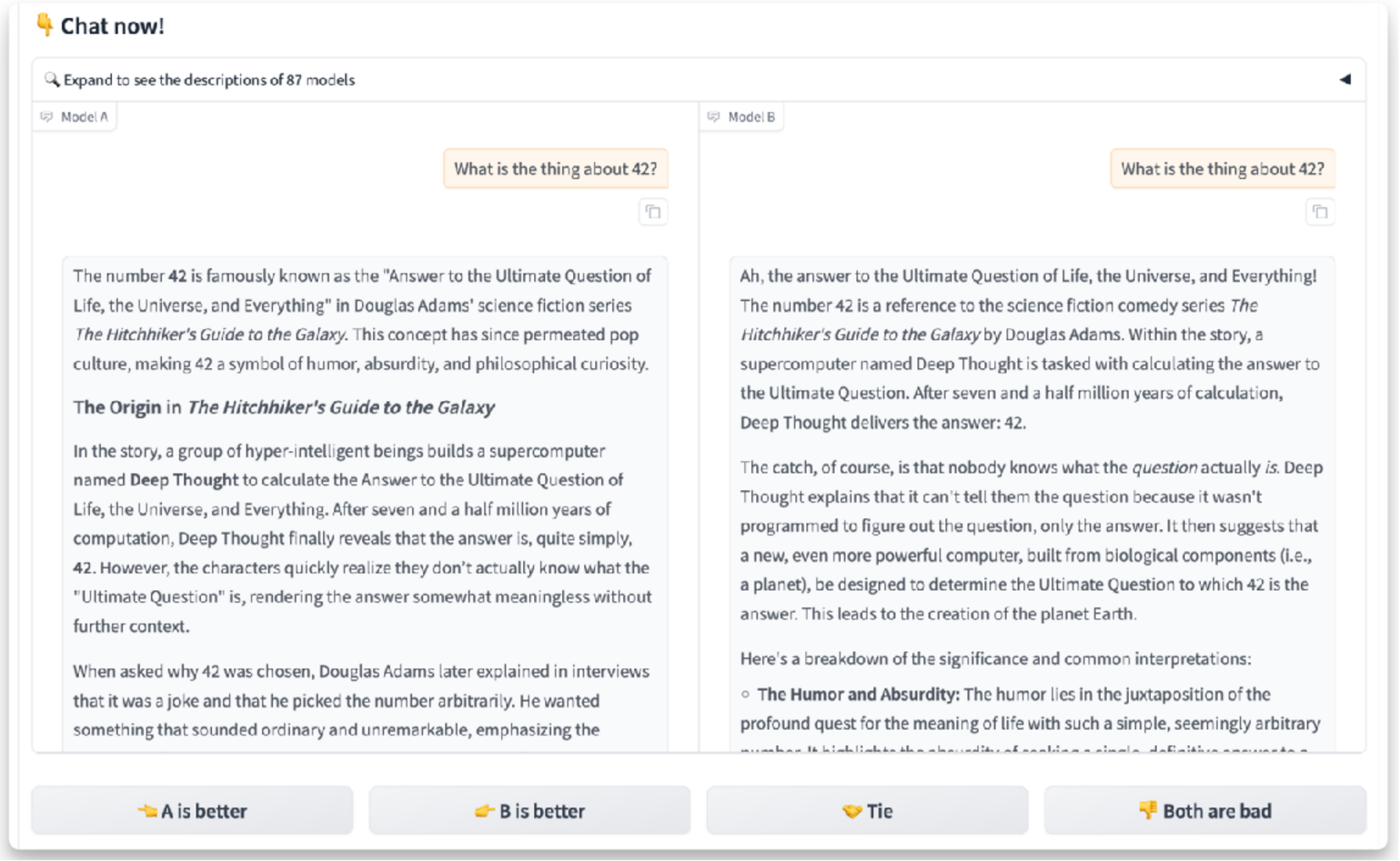
https://github.com/lmgame-org/GamingAgent

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



https://lmarena.ai/

https://lmarena.ai/?leaderboard

LMSYS Org

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

https://lmsys.org/

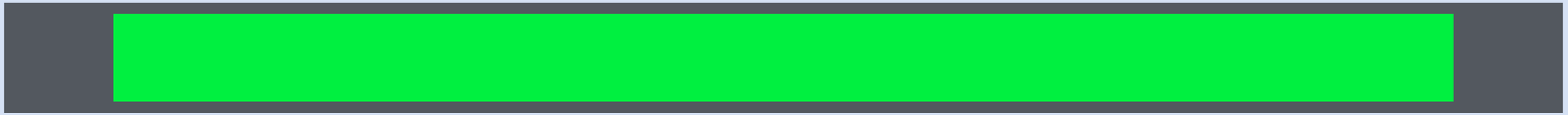
Exercise

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

PURPOSE

VARIABLES

MODEL

OUTPUT

EXAMPLES

IndyDevDan: Master the Prompt

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.

https://docs.anthropic.com/en/docs/build-with-claude/prompt-engineering/use-xml-tags

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/

aichat

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)

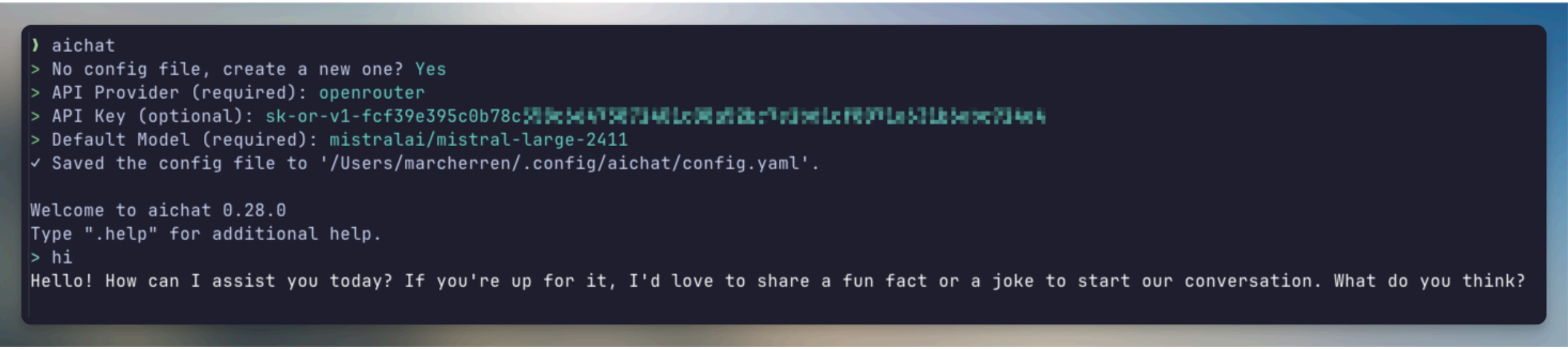
3. Configure aichat by invoking it on the terminal
   Use any of the free models on openrouter

4. 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
> 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

Install and configure aichat

1. Interact with a file

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

2. Create commands

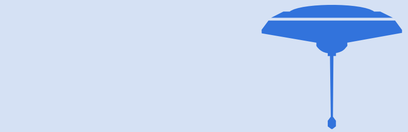
```
cd rename
aichat -e "Rename all files ending with -yaml.txt to end with
.yaml instead“
```

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 lmarena.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
```

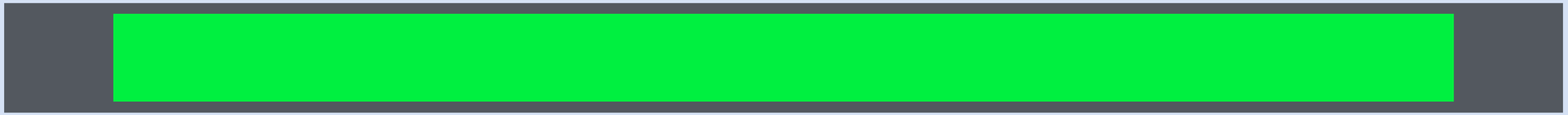
Configure LLM with openrouter

```
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.

fabric

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
  ```

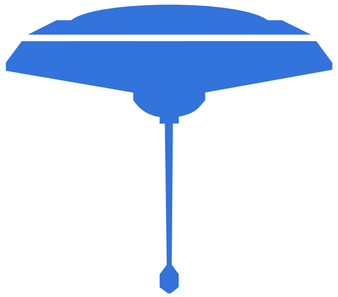
https://github.com/danielmiessler/fabric
  ```

TELOS

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