Open-source infrastructure for LLMs

Philipp Krähenbühl, UT Austin

Parts of a model

- Architecture
- Weights
- Training code / data

Public models

- **Closed models**
  - Only model outputs accessible via online servers
- **Open models**
  - Only weights released
- **Open-source models**
  - Training code + datasets + weights released

Public models

Closed models

Only model outputs available

- GPT 3.5 / 4
- Claude 1 / 2 / 3
- Gemini / Bard
- Mistral Medium / Large

Open models

Model weights available

- LLaMA 1 / 2 / 3
- Mistral 7B / Mixtral
- Qwen
- Gemma
- Grok
- Falcon

and many more....

Open-source models

Training code, datasets, weights released

- Vicuna
- OpenChat
- OLMo
- OpenFlamingo
- BLIP / InstructBLIP
- LLaVA

and many more....

Public models

Closed models

Only model outputs available

- GPT 3.5 / 4
- Claude 1 / 2 / 3
- Gemini / Bard
- Mistral Medium / Large

Open models

Model weights available

- LLaMA 1 / 2 / 3
- Mistral 7B / Mixtral
- Qwen
- Gemma
- Grok
- Falcon

and many more....

Open-source models

Training code, datasets, weights released

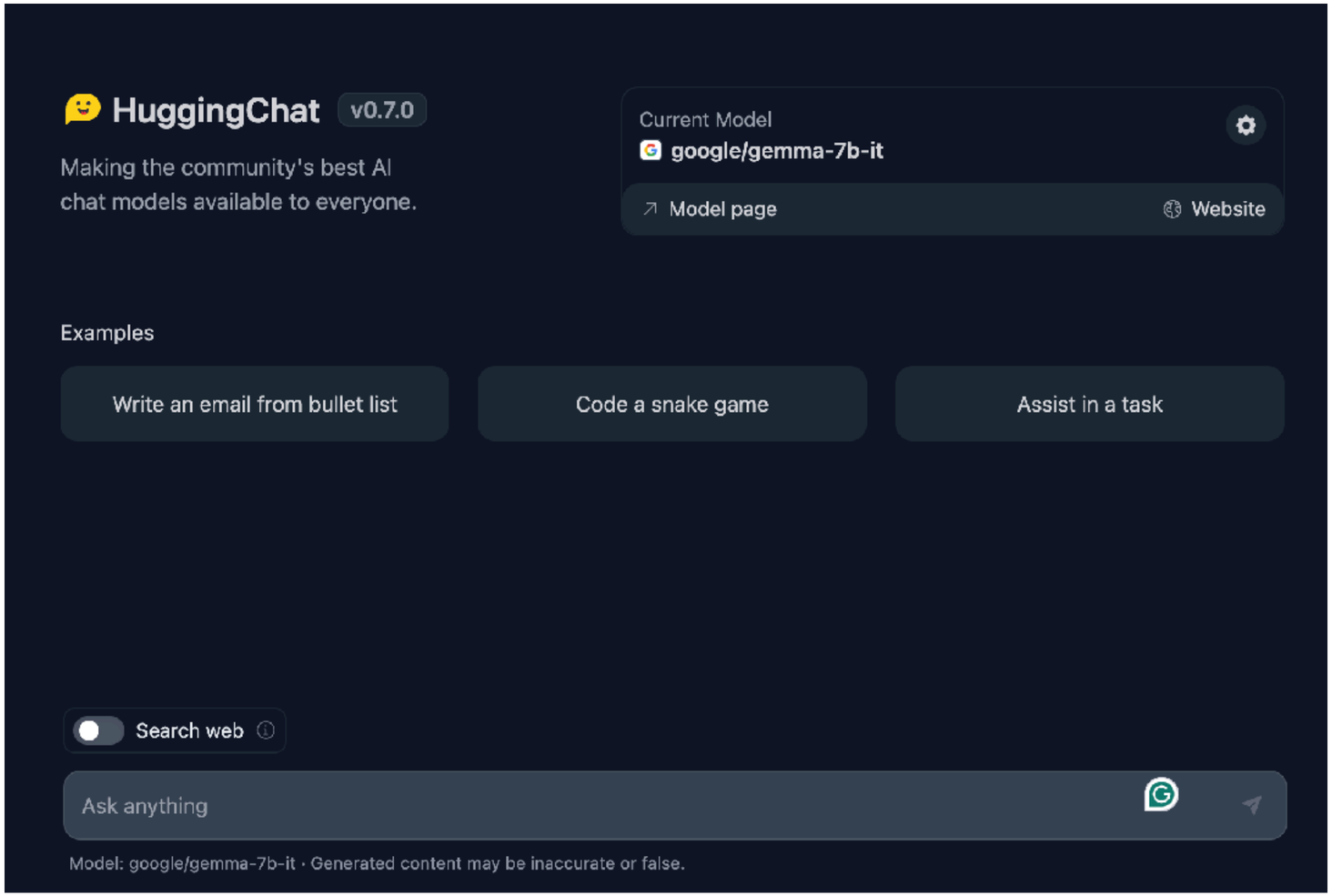
- Vicuna
- OpenChat
- OLMo
- OpenFlamingo
- BLIP / InstructBLIP
- LLaVA

and many more....

https://huggingface.co/models

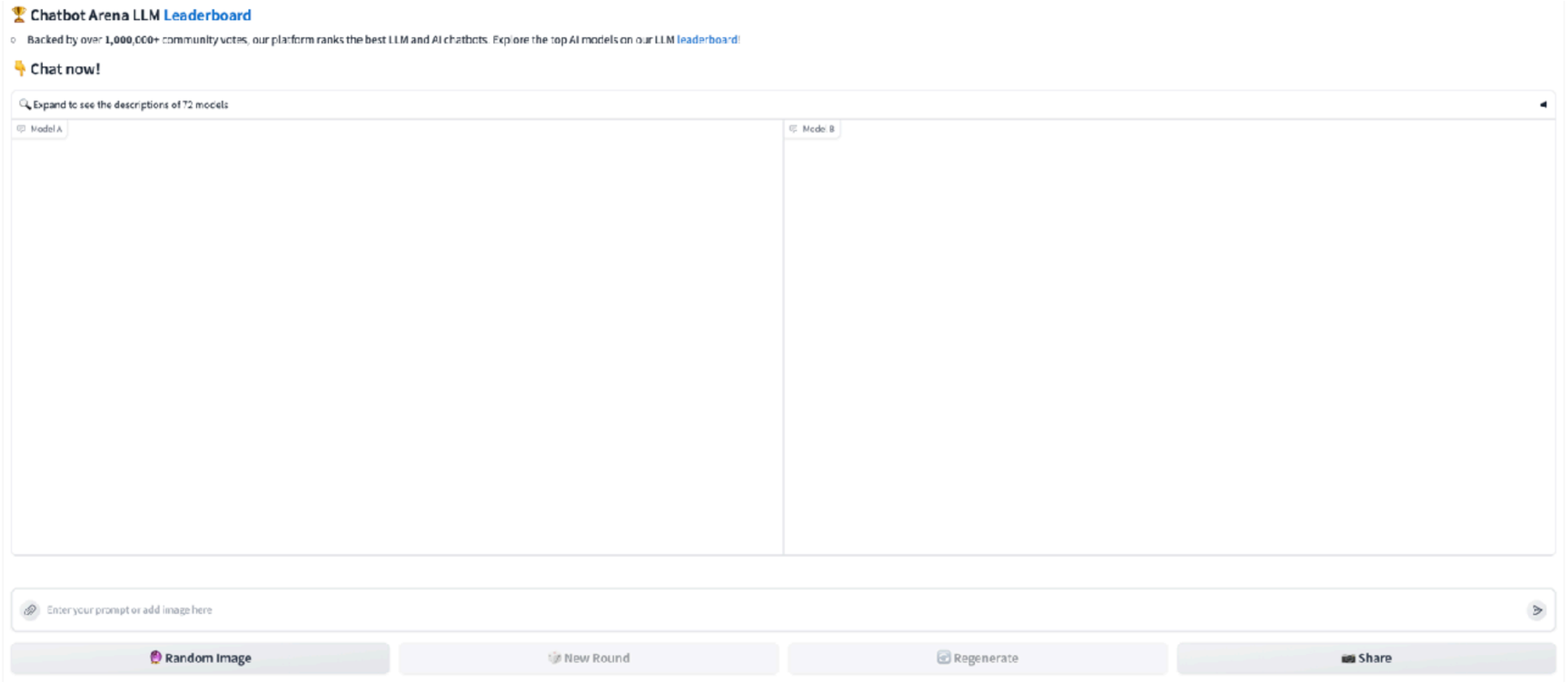
Generation - Online chat UI

- HuggingChat

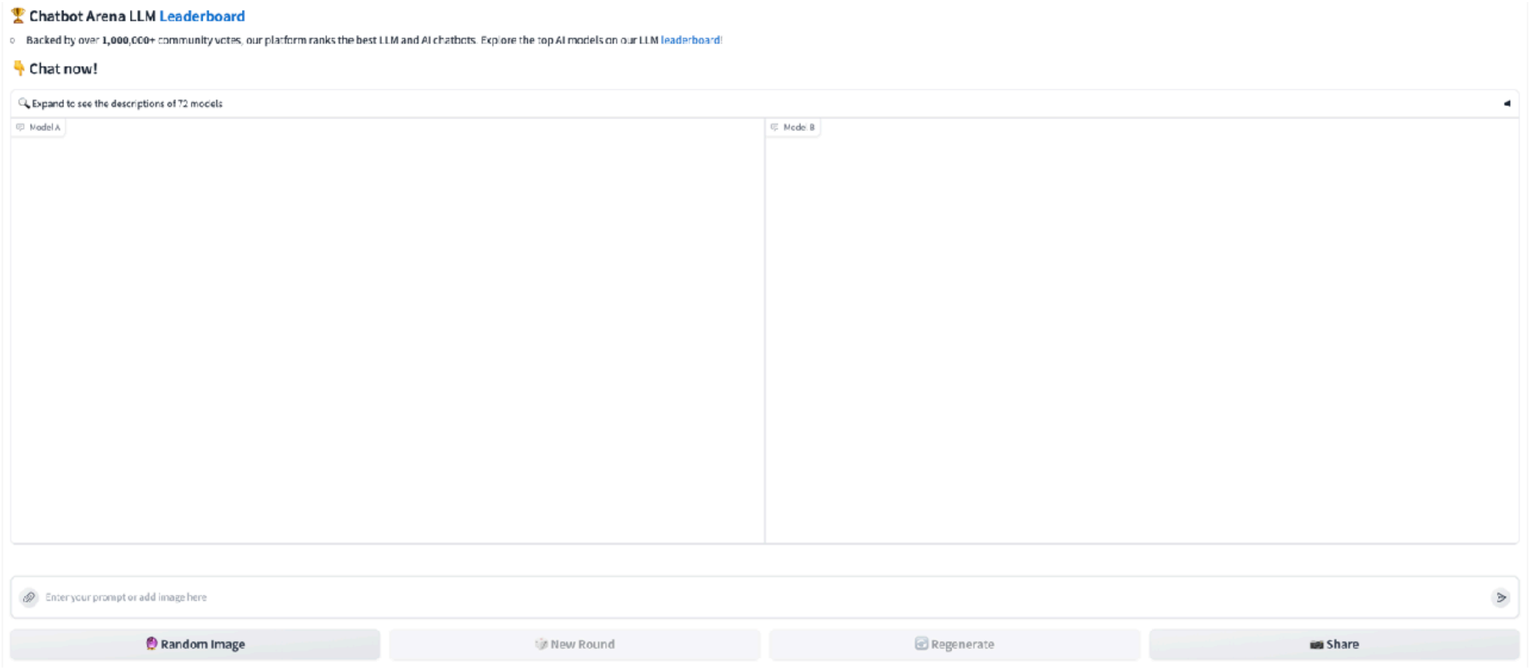


Generation - Online chat UI

- LMSys https://lmarena.ai/



Online chat UI



- **Pros:**
  - Quick testing
  - Good reference
- **Cons:**
  - Chat history not private
  - Rate limitations
  - Limited models
  - Limited programmatic testing

Generation - Local Chat UI

- Ollama: https://github.com/ollama/ollama
- llama_cpp: https://github.com/ggerganov/llama.cpp

Get up and running with large language models.

Run Llama 3.2, Phi 3, Mistral, Gemma 2, and other models. Customize and create your own.

Download ↓

Available for macOS, Linux, and Windows

LLaMA C++

Generation - Local Chat UI

Python API

```python
import ollama
response = ollama.chat(model='llama3.1', messages=[
  {
    'role': 'user',
    'content': 'Why is the sky blue?',
  },
])
print(response['message']['content'])
```

- Ollama: https://github.com/ollama/ollama

- llama_cpp: https://github.com/ggerganov/llama.cpp
  https://github.com/abetlen/llama-cpp-python

```python
from llama_cpp import Llama

llm = Llama(
      model_path="./models/7B/llama-model.gguf",
      # n_gpu_layers=-1, # Uncomment to use GPU acceleration
      # seed=1337, # Uncomment to set a specific seed
      # n_ctx=2048, # Uncomment to increase the context window
)
output = llm(
      "Q: Name the planets in the solar system? A: ", # Prompt
      max_tokens=32, # Generate up to 32 tokens, set to None to generate up to the end of the
      stop=["Q:", "\n"], # Stop generating just before the model would generate a new question
      echo=True # Echo the prompt back in the output
) # Generate a completion, can also call create_completion
print(output)
```

Local Chat UI

- **Pros:**
  - Quick testing
  - Good reference
  - Private (disable telemetry!!)
  - Fast on GPUs and M1+ Macs
- **Cons:**
  - Limited scaling: single request only

Get up and running with large language models.

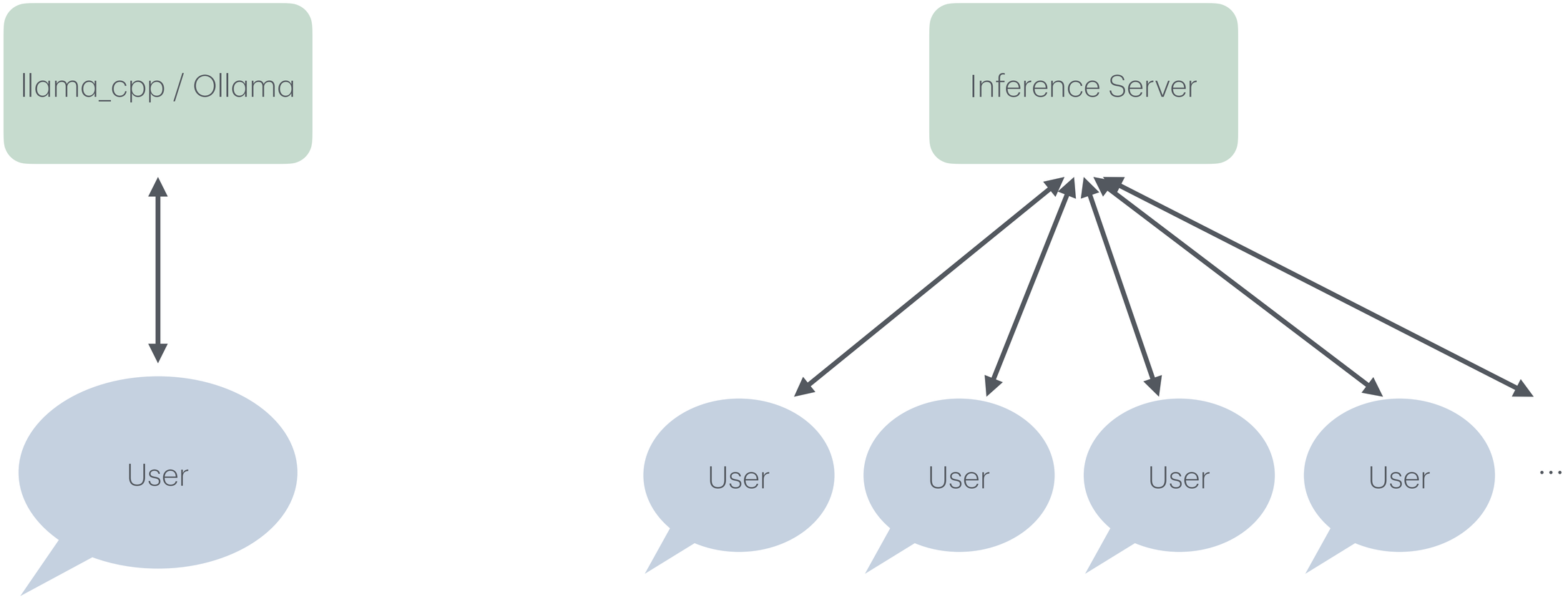
Run Llama 3.2, Phi 3, Mistral, Gemma 2, and other models. Customize and create your own.

Download ↓

Available for macOS, Linux, and Windows

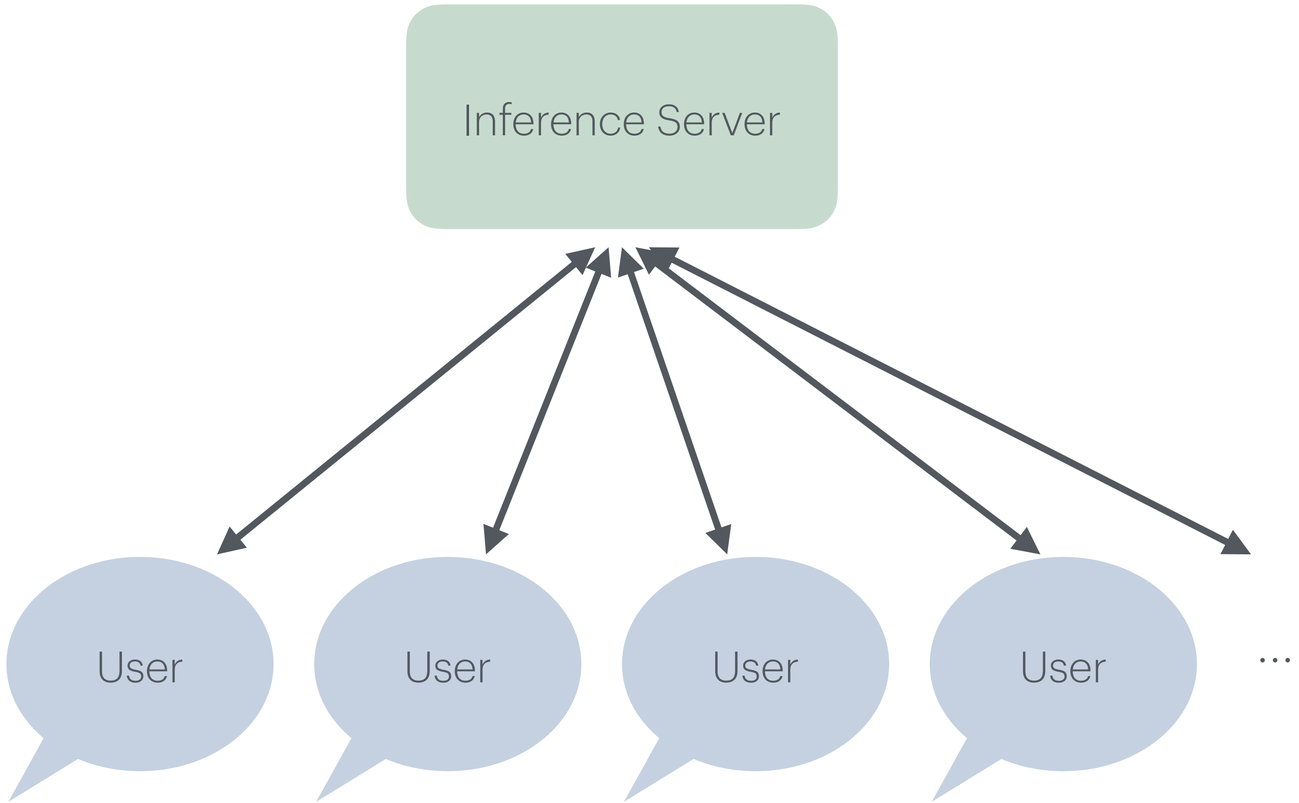
LLaMA C++

Generation - Large scale generation



Large scale generation - Inference Servers

- HuggingFace TGI
- VLLM
- FastChat



VLLM

- Install locally
- Run server
- Use curl / python for requests

```
# Install library
> pip install vllm

# Download model and and host inference server
python -m vllm.entrypoints.openai.api_server \
       --model mistralai/Mixtral-8x7B-Instruct-v0.1 \
       --tensor-parallel-size 8
```

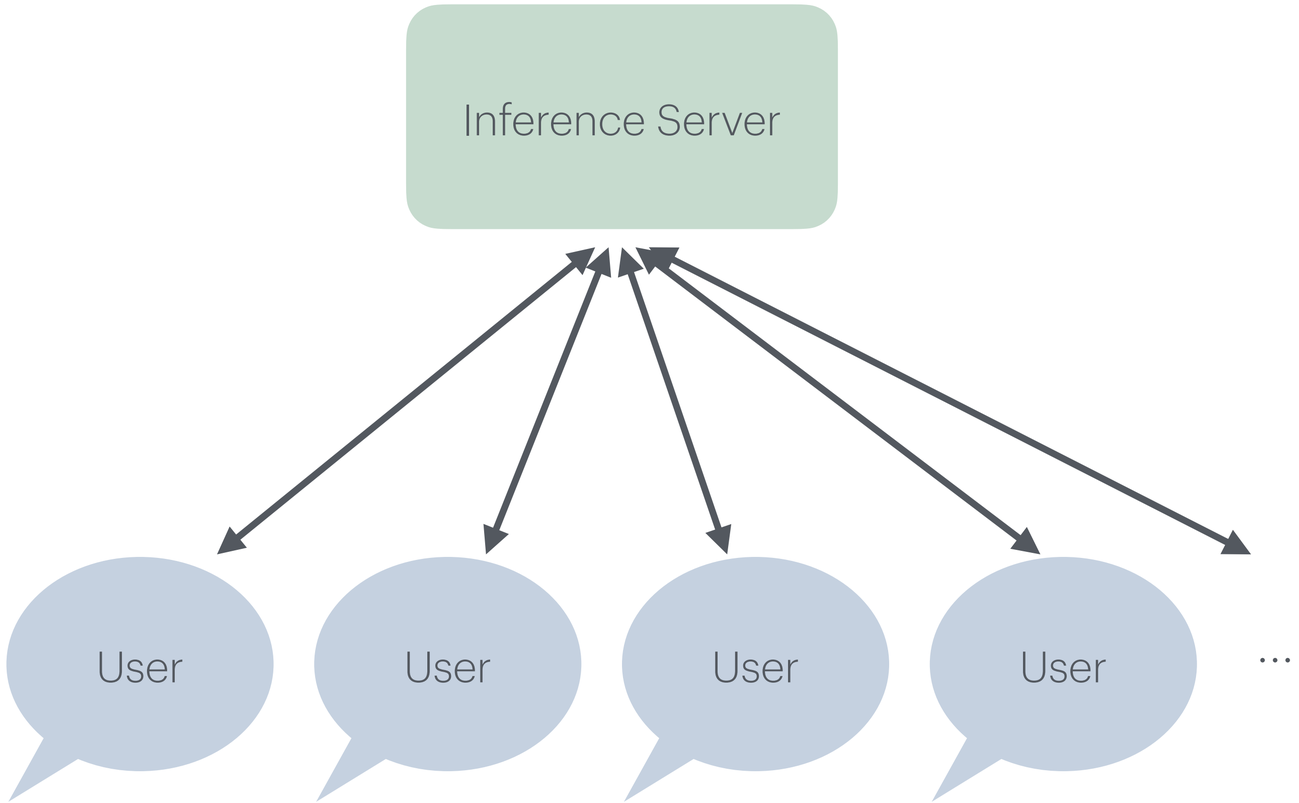
```
> curl http://localhost:8000/v1/chat/completions \
$     -H "Content-Type: application/json" \
$     -d '{
$         "model": "mistralai/Mixtral-8x7B-Instruct-v0.1",
$         "messages": [
$             {"role": "user", "content": "Tell me a joke"}
$         ]
$     }'
```

```
#!/usr/bin/python3
from openai import OpenAI

# Modify OpenAI's API key and API base to use vLLM's API server.
openai_api_key = "EMPTY"
openai_api_base = "http://localhost:8000/v1"
client = OpenAI(
    api_key=openai_api_key,
    base_url=openai_api_base,
)
completion = client.completions.create(
    model="mistralai/Mixtral-8x7B-Instruct-v0.1",
    messages=[{"role": "user", "content": "Tell me a joke."}],
)
print("Completion result:", completion)
```

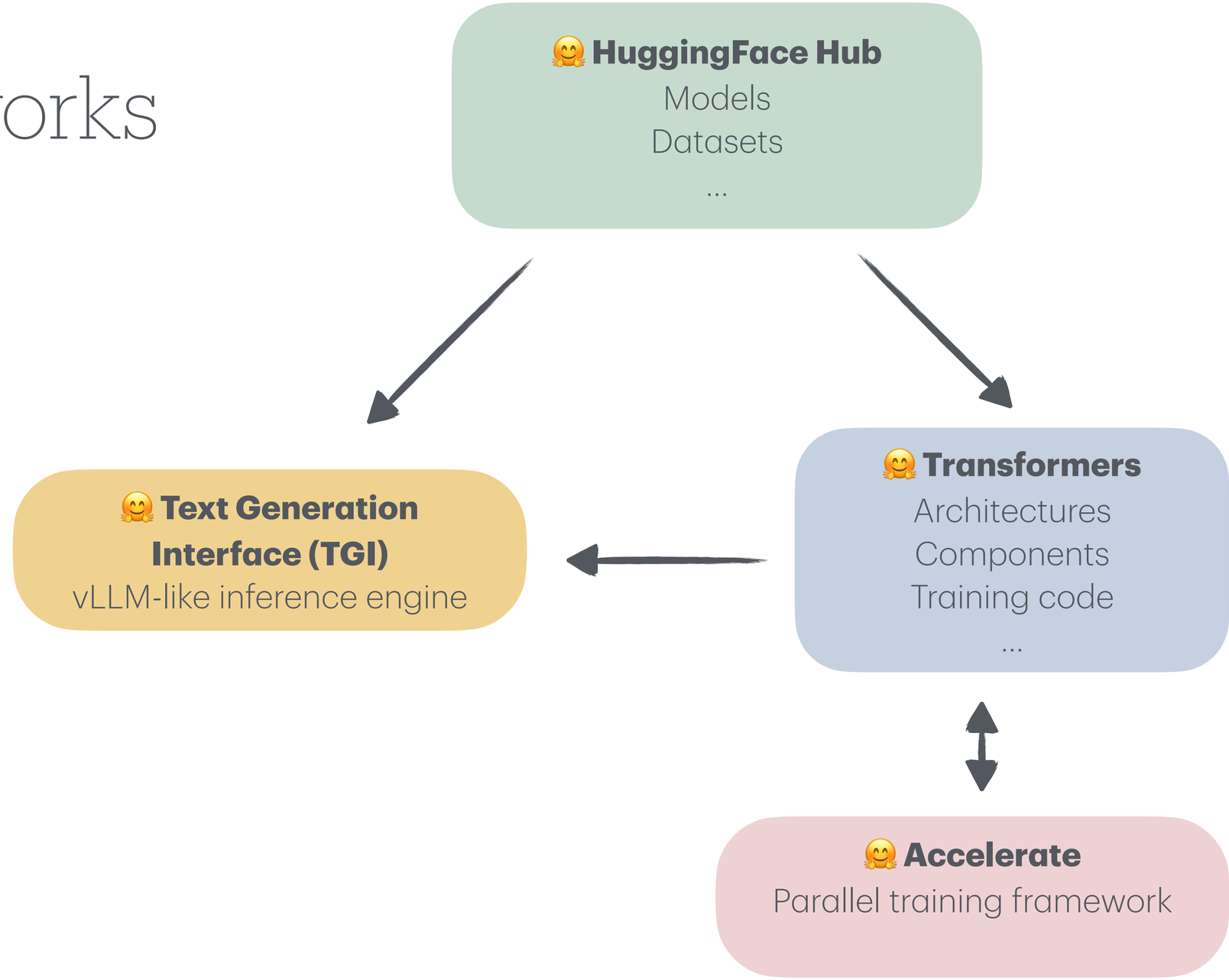
VLLM

- Fastest inference
  - Hides latency
- Private
- Requires (multiple) GPUs



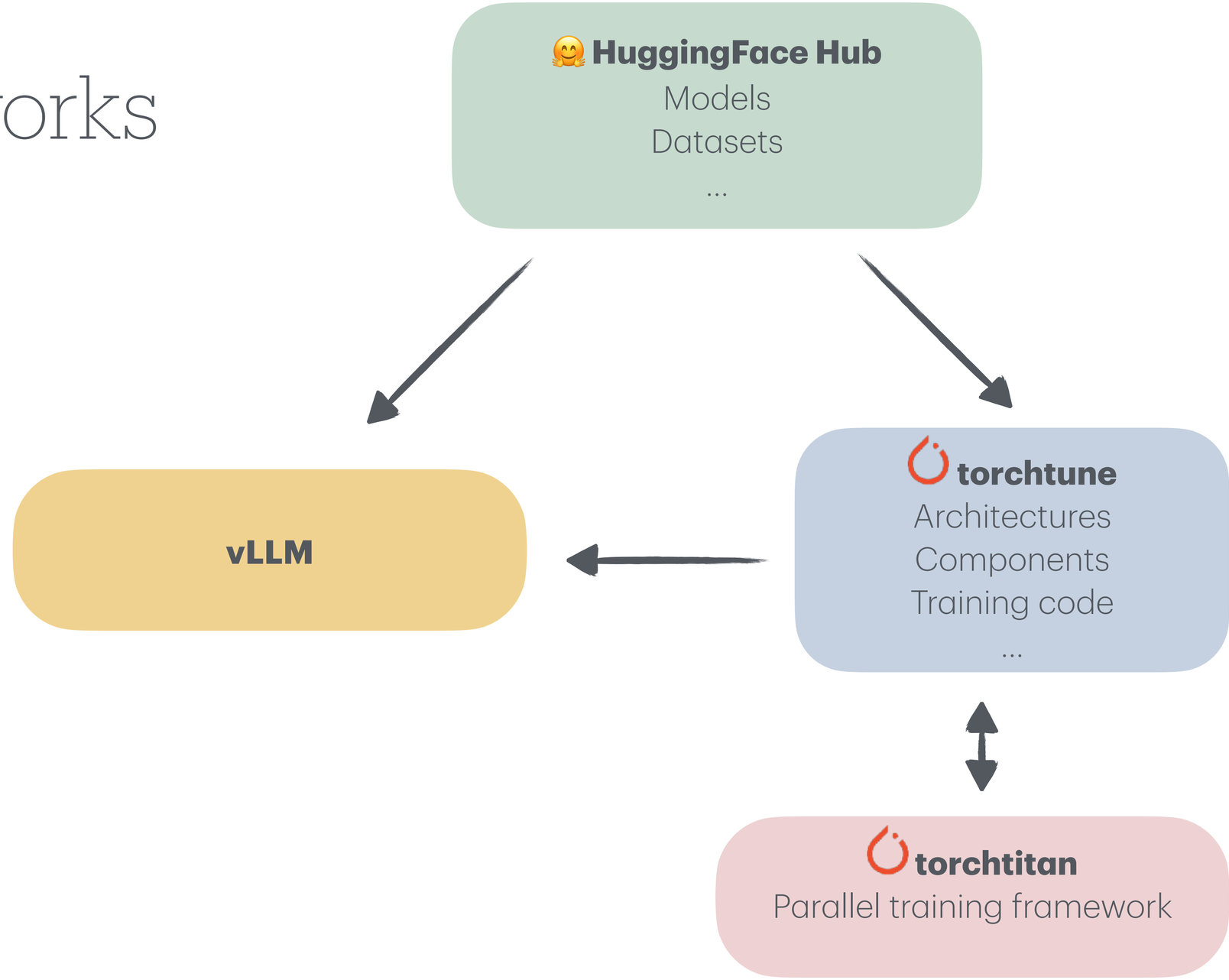
Training frameworks

- Huggingface ecosystem
  - Transformers
  - TGI
  - Accelerate
  - ...



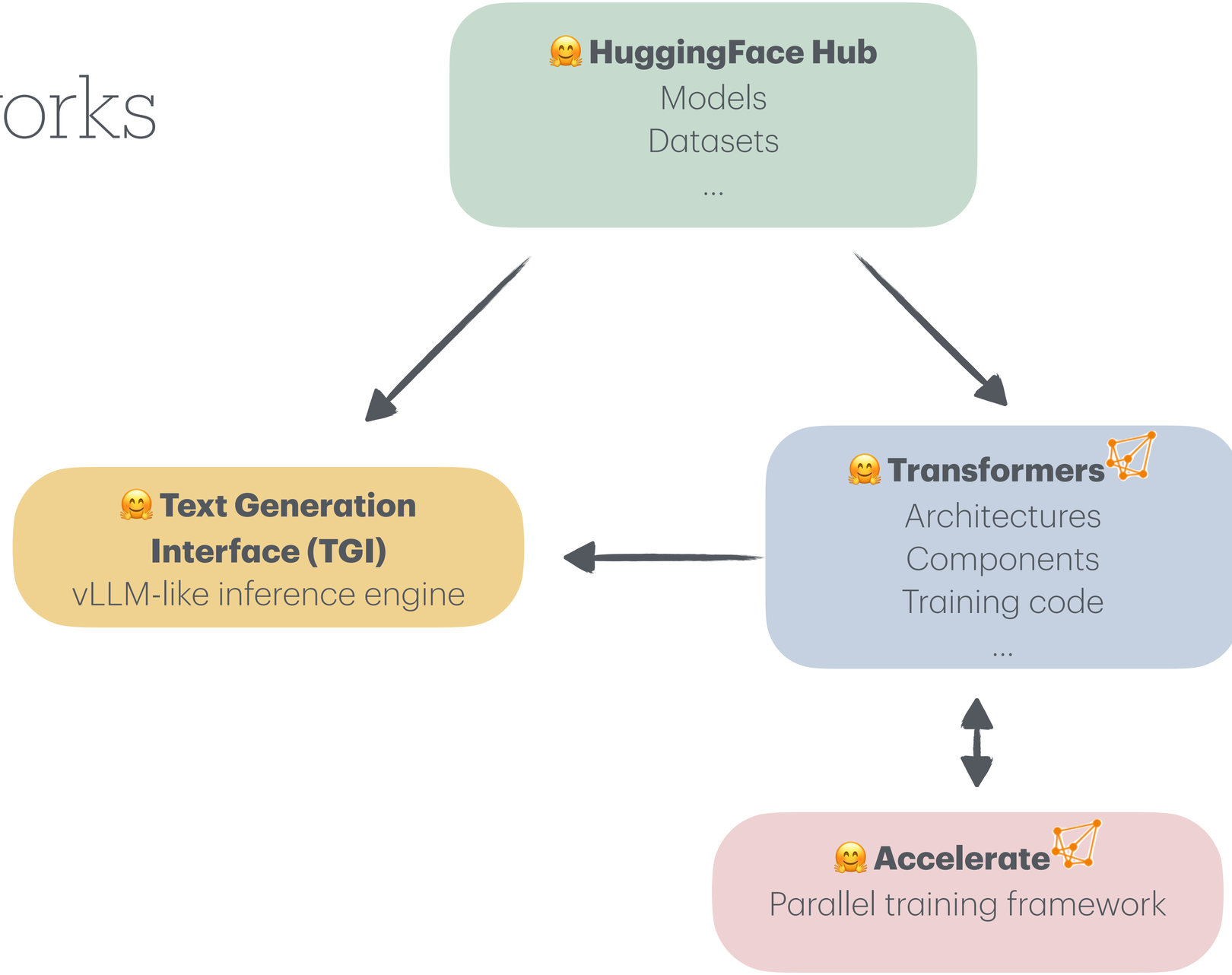
Training frameworks

- Torch ecosystem
  - Torchtune
  - Torchrun / FSDP / DDP
  - torchtitan
  - ...



Training frameworks

- DeepSpeed ecosystem
  - Based on HuggingFace
  - Integrated into
    - Transformers
    - Accelerate



References

- [1] HuggingFace Models (link)
- [2] LMSys
- [3] Ollama
- [4] llama_cpp