Community of Practice KI*PerWeb*

Austausch zur Nutzung und Entwicklung KI-gestützter Webanwendungen

KIPERWEB

f-bb Forschungsinstitut Betriebliche Bildung

Agenda

- Update
  - News & Leaderboard-Update
- Input
  - „HuggingChat Tools und wie man sie in eigene Chatbots einbindet“
- Diskussion

Newsflash

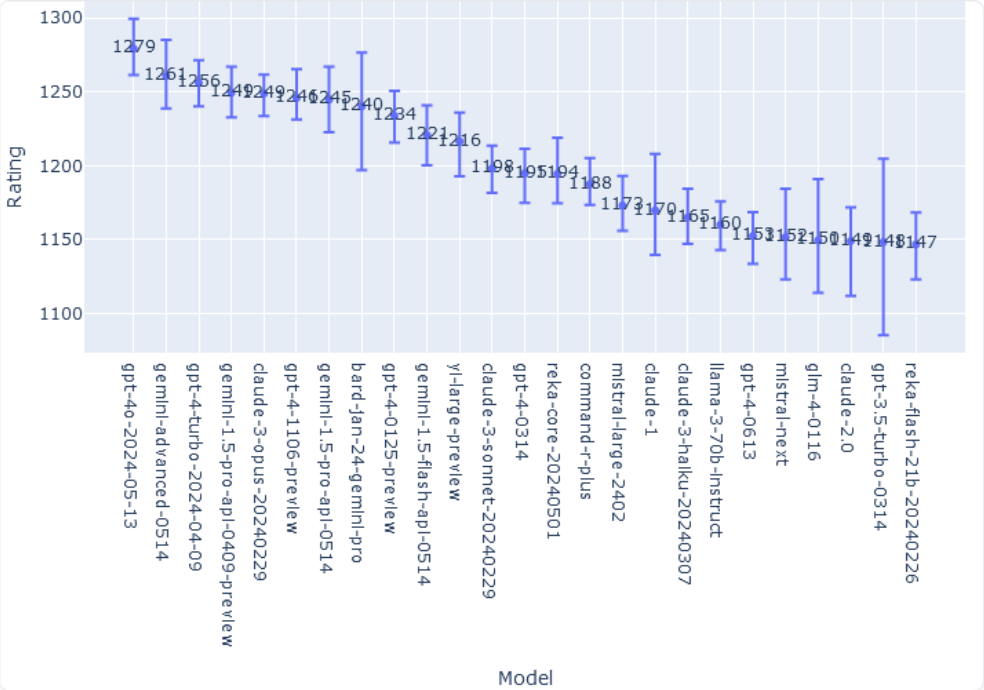
- **Stable Diffusion 3** ist seit 12.06. veröffentlicht (open-weights)
  - https://huggingface.co/collections/stabilityai/stable-diffusion-3-666992fb11f5d8d0ec8192e8
  - Direkt hinter Midjourney v6 (noch vor OpenAIs Dall-E3) auf dem text-to-image-Leaderboard von ArtificialAnalysis: https://artificialanalysis.ai/text-to-image/arena
- **Perplexica** als quelloffene Alternative zu perpplexity.ai, phind.com & Co.
  - LLM-Agenten (Ollama) & SearXNG für Websuche: https://the-decoder.de/perplexica-ist-eine-open-source-ki-suchmaschine-als-alternative-zu-perplexity/
  - Fokusmodi für unterschiedliche Fragetypen (All Mode, Writing Assistant, Academic Search, YouTube Search, Wolfram Alpha Search, Reddit Search)
- **Dream Machine** von Luma Labs ermöglicht seit 12.06. KI-Videogenerierung für alle
  - https://lumalabs.ai/dream-machine

Quelle: https://huggingface.co/stabilityai/stable-diffusion-3-medium

Leaderboard-Update (11.06.2024)

- LMSYS Chatbot Arena Leaderboard nun auch mit Filter für „German“
- Die besten freien offenen KI-Modelle sind auch hier Command R+ (CC-BY-NC-4.0), Llama-3-70B-Instruct (Llama 3 Community), Qwen2-72B-Instruct (Qianwen LICENCE), Mixtral-8x22b-Instruct-v0.1 (Apache 2.0) und Command R (CC-BY-NC-4.0) – allesamt vor GPT-3.5-Turbo-0613 – sowie Qwen1-5-110B-Chat (Qianwen LICENCE), Mixtral-8x7b-Instruct-v0.1 (Apache 2.0) und Llama-3-8b-Instruct) – vor GPT-3.5-Turbo-0125

Confidence Intervals on model strength (Arena Elo, German)



Quelle: https://chat.lmsys.org/?leaderboard

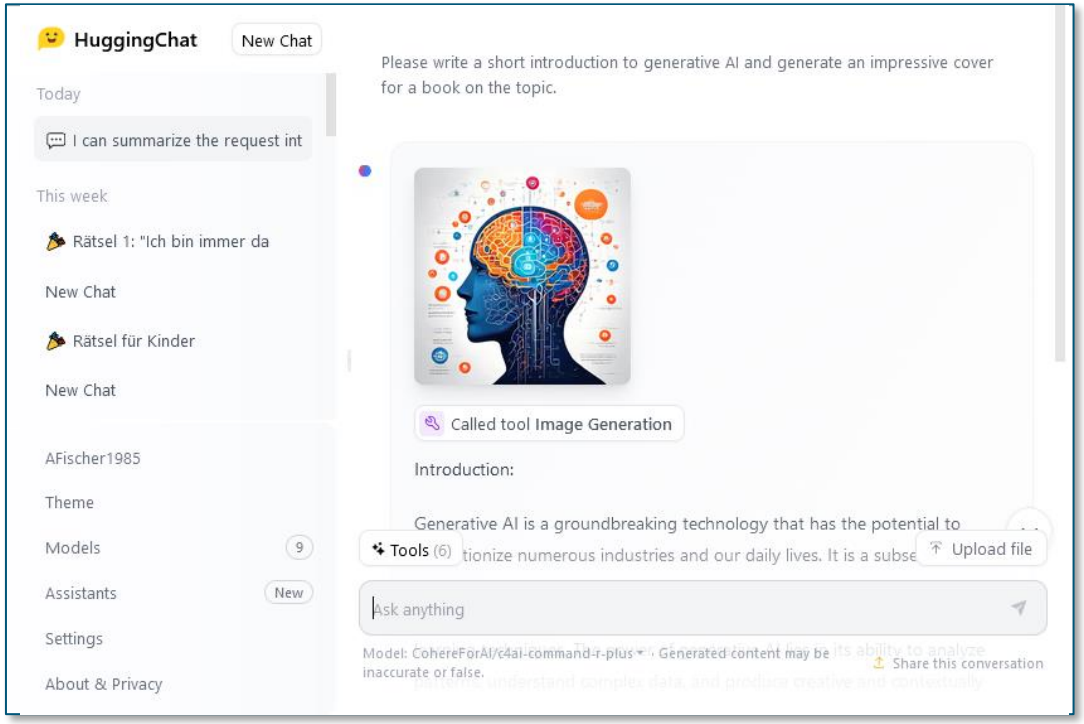
Tools on HuggingChat!

Quelle: https://huggingface.co/spaces/huggingchat/chat-ui/discussions/470

HuggingChat Tools

HuggingChat Tools (Beta):

- **Web Search:** Query the web and do some RAG on retrieved content against the user query
- **URL Fetcher:** Fetch text content from a given URL
- **Document Parser:** Parse content from PDF, text, csv, json and more
- **Image Generation:** Generate images based on a given text prompt
- **Image Editing:** Edit images based on a given text prompt
- **Calculator:** A simple calculator for evaluating mathematical expressions



Quellen: https://huggingface.co/chat
https://huggingface.co/spaces/huggingchat/chat-ui/discussions/470

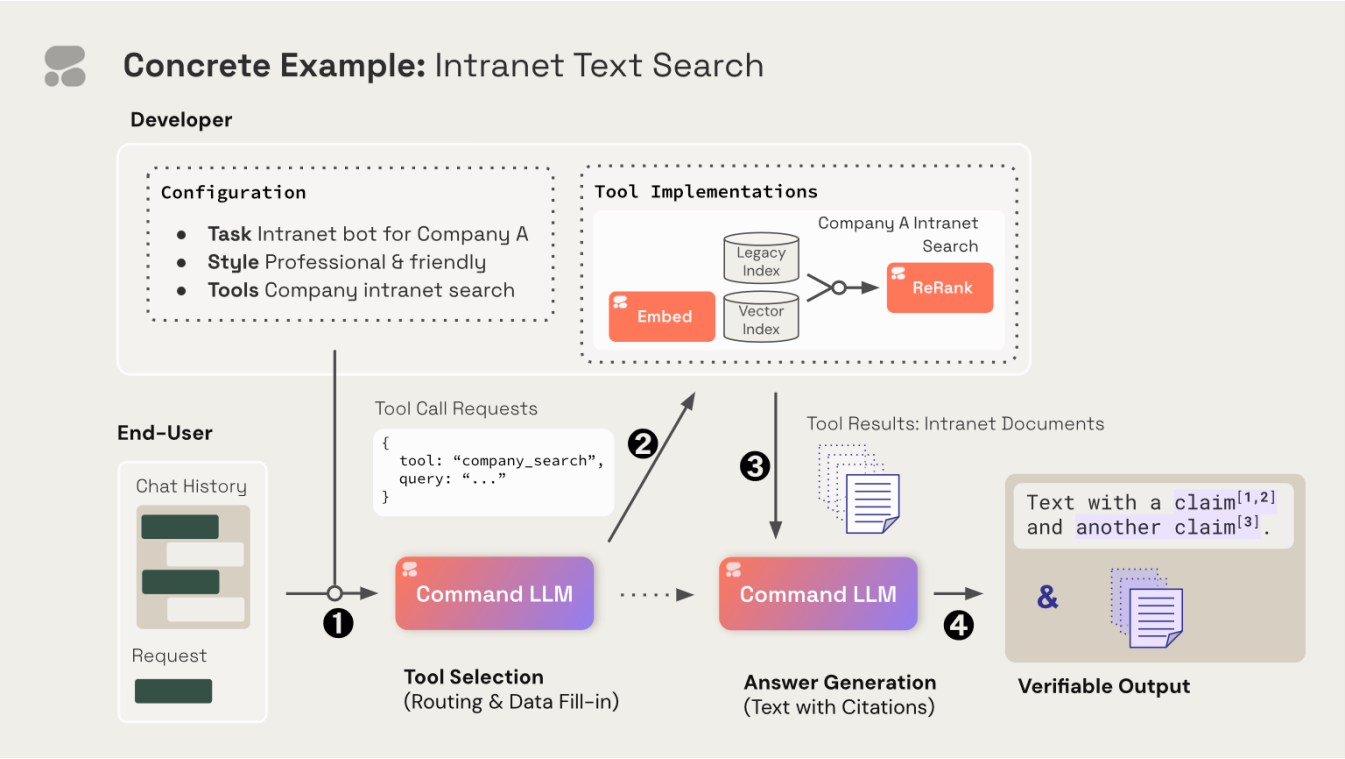
Example: Internal Tool „Calculator“

Code Blame 32 lines (29 loc) · 984 Bytes Raw

```
1   import type { BackendTool } from ".";
2   import vm from "node:vm";
3
4   const calculator: BackendTool = {
5           name: "query_calculator",
6           displayName: "Calculator",
7           description:
8                   "A simple calculator, takes a string containing a mathematical expression and returns the answer. Only supports +, -, *, ** (power) and /, as well as parenthes
9           isOnByDefault: true,
10          parameterDefinitions: {
11                  equation: {
12                          description:
13                                  "The formula to evaluate. EXACTLY as you would plug into a calculator. No words, no letters, only numbers and operators. Letters will make the
14                          type: "formula",
15                          required: true,
16                  },
17          },
18          async *call(params) {
19                  try {
20                          const blocks = String(params.equation).split("\n");
21                          const query = blocks[blocks.length - 1].replace(/[^-()\d/*+.]/g, "");
22
23                          return {
24                                  outputs: [{ calculator: `${query} = ${vm.runInNewContext(query)}` }],
25                          };
26                  } catch (cause) {
27                          throw Error("Invalid expression", { cause });
28                  }
29          },
30  };
31
32  export default calculator;
```

Quelle: https://github.com/huggingface/chat-ui/blob/main/src/lib/server/tools/calculator.ts

Single-Step Tool Use with Command-R (Theory)



Quelle: https://docs.cohere.com/docs/tool-use

Single-Step Tool Use with Command-R (Practice)

```
prompt="""<BOS_TOKEN><|START_OF_TURN_TOKEN|><|SYSTEM_TOKEN|># Safety Preamble
The instructions in this section override those in the task description and style guide sections. Don't answer questions that are harmful or immoral.

# System Preamble
## Basic Rules
You are a powerful conversational AI trained by Cohere to help people. You are augmented by a number of tools, and your job is to use and consume the output of these tools to best
help the user. You will see a conversation history between yourself and a user, ending with an utterance from the user. You will then see a specific instruction instructing you what
kind of response to generate. When you answer the user's requests, you cite your sources in your answers, according to those instructions.

# User Preamble
## Task and Context
You help people answer their questions and other requests interactively. You will be asked a very wide array of requests on all kinds of topics. You will be equipped with a wide
range of search engines or similar tools to help you, which you use to research your answer. You should focus on serving the user's needs as best you can, which will be wide-ranging.

## Style Guide
Unless the user asks for a different style of answer, you should answer in full sentences, using proper grammar and spelling.

## Available Tools
Here is a list of tools that you have available to you:

```python
def internet_search(query: str) -> List[Dict]:
    \"\"\"Returns a list of relevant document snippets for a textual query retrieved from the internet

    Args:
        query (str): Query to search the internet with
    \"\"\"
    pass
```

```python
def directly_answer() -> List[Dict]:
    \"\"\"Calls a standard (un-augmented) AI chatbot to generate a response given the conversation history
    \"\"\"
    pass
```<|END_OF_TURN_TOKEN|><|START_OF_TURN_TOKEN|><|USER_TOKEN|>Whats the biggest penguin in the world?<|END_OF_TURN_TOKEN|><|START_OF_TURN_TOKEN|><|SYSTEM_TOKEN|>Write 'Action:'
followed by a json-formatted list of actions that you want to perform in order to produce a good response to the user's last input. You can use any of the supplied tools any number
of times, but you should aim to execute the minimum number of necessary actions for the input. You should use the `directly-answer` tool if calling the other tools is unnecessary.
The list of actions you want to call should be formatted as a list of json objects, for example:
```json
[
    {
        "tool_name": title of the tool in the specification,
        "parameters": a dict of parameters to input into the tool as they are defined in the specs, or {} if it takes no parameters
    }
]```<|END_OF_TURN_TOKEN|><|START_OF_TURN_TOKEN|><|CHATBOT_TOKEN|>"""
```

Command-R antwortet mit JSON

```
Action: ```json
[
    {
        "tool_name": "internet_search",
        "parameters": {
            "query": "biggest penguin in the world"
        }
    }
]
```
```

JSON wird von eigenen Tools verarbeitet und neue Anfrage wird gestartet

Quelle: https://huggingface.co/CohereForAI/c4ai-command-r-v01

ZeroGPU Spaces

- Document Parser:
  - https://huggingface.co/spaces/huggingchat/pdf-to-markdown
  - https://github.com/huggingface/chat-ui/blob/main/src/lib/server/tools/documentParser.ts
- Image Generation
  - https://huggingface.co/spaces/ByteDance/Hyper-SDXL-1Step-T2I
  - https://github.com/huggingface/chat-ui/blob/main/src/lib/server/tools/images/generation.ts
- Image Editing
  - https://huggingface.co/spaces/multimodalart/cosxl
  - https://github.com/huggingface/chat-ui/blob/main/src/lib/server/tools/images/editing.ts

{outputs: [{imageGeneration: `*An image has been generated for the following prompt: "${prompt}". Answer as if the user can already see the image. Do not try to insert the image or to add space for it. The user can already see the image. Do not try to describe the image as you the model cannot see it. Be concise.*`, },],display: false,};

Example: „Image Generation“-Tool in Custom-Chatbots!

```python
# Integrate Image Generation in custom Chatbots 🔥
#-------------------------------------------------

import gradio as gr
import os, shutil, time
from gradio_client import Client
client = Client("ByteDance/Hyper-SDXL-1Step-T2I")
id=0
def multimodalResponse(message,history):
  global id
  id=id+1
  print(message)
  result = client.predict(
    num_images=1,
    height=1024,
    width=1024,
    prompt=message,
    seed=3413,
    api_name="/process_image")
  shutil.copy(result[0]['image'],os.getcwd())
  os.rename('image.webp', 'image'+str(id)+'.webp')
  return "Prompt '"+message+"': ![image](/file="+os.getcwd()+"/image"+str(id)+".webp)"
bot=gr.Chatbot(
    value=[[None,"I'm a simple image-generating chatbot. Please tell me what you would like to see."]],
    render_markdown=True)
interface=gr.ChatInterface(multimodalResponse,chatbot=bot, multimodal=False)
interface.launch(allowed_paths=["."])
```


Quelle: https://huggingface.co/spaces/AFischer1985/ImageGenerationIntegration/blob/main/app.py