Nutzung der KI-Toolbox über API-Calls

Die KI-Toolbox kann man auch über eine API-Schnittstelle erreichen. Die OAS3 Dokumentation dazu kann man hier sehen:

https://ki-toolbox.tu-braunschweig.de/swagger-ui/index.html

Beispiele wie man die Verbindung aufbauen kann, kann man in den folgenden Beispielen sehen.

- API-Verbindungen (Shell/Curl)
- API-Verbindungen Interface (Gradio)

API-Verbindungen (Shell/Curl)

Es ist möglich sehr leicht mit Curl Kommandos die API-Schnittstelle bei der KI-Toolbox zu nutzen.

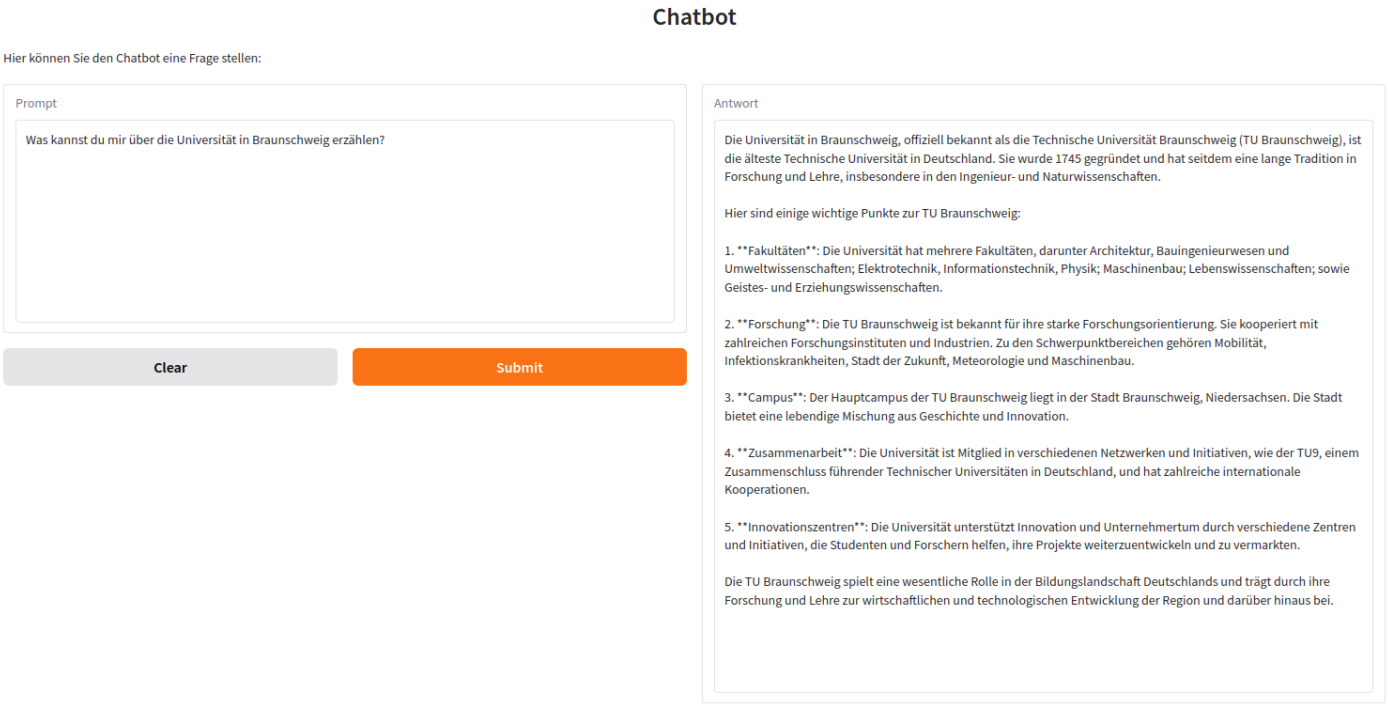
Beispiel Shell/Curl Kommando:

```
curl -X 'POST' \
  'https://ki-toolbox.tu-braunschweig.de/api/v1/chat/send' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer <Token>' \
  -H 'Content-Type: application/json' \
  -d '{
  "thread": null,
  "prompt": "<Prompt>",
  "model": "<Model>",
  "customInstructions": "<Benutzerdefinierte Anweisungen>",
  "hideCustomInstructions": <true oder false>,
}'
```

Zeile 2: API-Call Adresse anpassen wir nötig
Zeile 4: <Token> mit der API-Schlüssel (Token) ersetzen oder aus den Umgebungsvariablen laden
Zeilen 7 - 11: "Message Body" - anpassen wir nötig, z.B. "<Prompt>" mit dem Prompt ersetzen und "<Model>" mit dem Modelnamen
Zeile 10: hier können benutzerdefinierte Anweisungen eingegeben werden
Zeile 11: das Feld hier kann als true gesetzt werden falls die benutzerdefinierte Anweisungen auf dem Frontend nicht angezeigt werden sollen

API-Verbindungen Interface (Gradio)

Es ist möglich schnell mit Python und Gradio einen Interface zu bauen um die API-Schnittstelle zu der KI-Toolbox zu nutzen.



Beispiel Code:

```
import gradio as gr
import os
import requests
import json

def generate_chat_response(user_message):
    url = "https://ki-toolbox.tu-braunschweig.de/api/v1/chat/send"
    api_key = os.getenv("API_KEY")

    headers = {
        "Accept": "application/json",
        "Authorization": f"Bearer {api_key}",
        "Content-Type": "application/json"
```

```
    }

    data = {
        "prompt": user_message,
        "model": "gpt-4o"
    }

    # Sending the POST request
    response = requests.post(url, headers=headers, data=json.dumps(data))

    if response.status_code == 200:
        # If the request was successful, return the reply from the model

        final_response = {}
        for line in response.iter_lines(decode_unicode=True):
            chunk = json.loads(line)
            if chunk.get("type") == "done":
                final_response = chunk
                break

        return final_response.get("response", "")
    else:
        # If there was an error, return the status code and error message
        return f"Error: {response.status_code}, {response.text}"

demo = gr.Interface(
    fn=generate_chat_response,
    inputs=gr.Textbox(label="Prompt", lines=10),
    outputs=gr.Textbox(label="Antwort", lines=30),
    title="Chatbot",
    description="Hier können Sie den Chatbot eine Frage stellen:",
)

demo.launch()
```

Zeilen 1 - 4: nötige Imports
Zeile 7: hier kann die API-Adresse eingegeben werden
Zeile 8: der API-Schlüssel wird aus den environment Variablen geladen (vorher muss der Schlüssel exportiert werden, z.B. mit dem Kommando "export API_KEY=<API-Schlüssel>"
Zeilen 10 - 14: Headers (nach Bedarf anpassen)
Zeilen 16 - 19: Request Body (nach Bedarf anpassen)
Zeilen 21 - 37: Den Response einlesen und bearbeiten (nach Bedarf anpassen)

Zeilen 39 - 45: Interface (nach Bedarf anpassen und erweitern, z.B. mehr Felder hinzufügen)