

Deep-Learning-Modell als Microservice

Entwicklung einer RESTful Web-API für ein Deep-Learning-Modell

Über mich



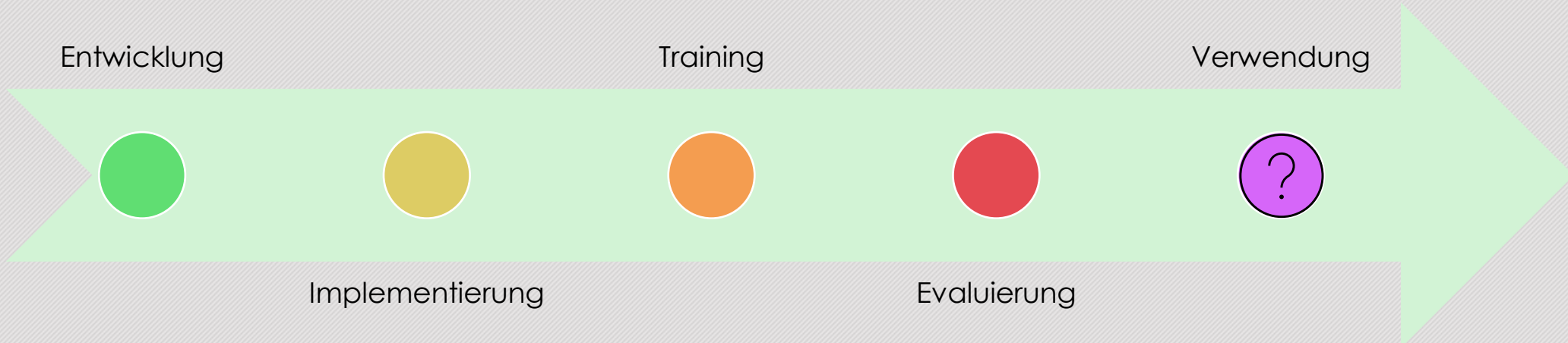
- ITS Master absolviert (2020)
- Software Entwicklung
- Data Scientist bei Symptoma GmbH
 - Fokus im Bereich NLP
 - Gerne mal vorbeischauen unter <https://www.symptoma.at/>



Agenda

- Problem „Deployment“
- REST API
- Praktisches Beispiel
 1. Unser Modell
 2. API mit Flask
 3. API Dokumentation
 4. Unser Modell als API
 5. Docker
 6. Google Cloud

Problem “Deployment”



Problem “Deployment”

Verwendung

- Download

```
git clone https://github.com/abc/dl-model.git
```

EXAMPLE

- „Predict“



train.py

Problem “Deployment”

Verwendung

- Download

```
git clone https://github.com/abc/dl-model.git
```

EXAMPLE

- „Predict“

```
python train.py -task abs -mode validate -encoder bert -batch_size  
3000 -test_batch_size 500 -data_path ./data -log_file  
./logs/model2 -model_path ./model/checkpoint_21_2021-01-  
30T10:47:24.404933.pt -sep_optim true -use_interval true -  
visible_gpus 1 -max_pos 512 -min_length 20 -max_length 100 -alpha  
0.9 -result_path ./logs/result -temp_dir ./temp -large true -  
use_bert_emb true -block_trigram true
```

EXAMPLE

Was ist eine REST-API?

- **RE**presentational **S**tate **T**ransfer
 - ...ist ein Architekturstil für die Kommunikation verteilter Systeme
 - ...wird meist per HTTP(S) realisiert
- **A**pplication **P**rogramming **I**nterface
 - ...ist eine Programmierschnittstelle

[1]

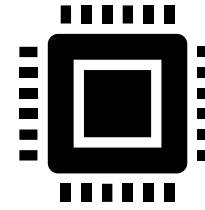
Was ist eine REST-API?




Praktisches Beispiel

NLP-Modell als Microservice mit REST API

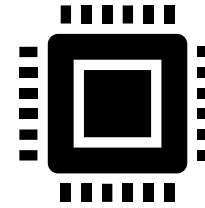
NLP-Modell



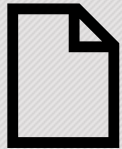
1

- Unser Modell: **GPT-2** [2]
- NLP Modell für Text-Generierung
- Framework  [Transformers](#)

NLP-Modell



1



train.py

```
from transformers import pipeline, AutoTokenizer, AutoModelForCausalLM, AutoConfig
import torch

model_str = "gpt2"
device = 0 if torch.cuda.is_available() else -1
generate_pipe = pipeline("text-generation", model=model_str, tokenizer=model_str,
device=device)

if __name__ == '__main__':
    # Call this file with: python train.py "This is the text I want to be continued"
    from codecs import decode
    import sys

    input_text = decode(sys.argv[1], 'unicode_escape')
    out = generate_pipe(input_text)
    print(out[0]['generated_text'])
```

EXAMPLE

API mit Flask



2

- [Flask](#) ist ein Web-Framework für Python
- Beispiel für eine minimale Applikation:

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'
```

EXAMPLE

[3]

API mit Flask



2

```
$ export FLASK_APP=src/service.py
$ flask run
* Serving Flask app "src/service.py"
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

EXAMPLE

```
$ curl -s http://localhost:5000
Hello, World!
```

EXAMPLE

API Dokumentation



3

- [Flask-RESTX](#) ist eine Erweiterung für Flask
- Hilft bei der Erstellung einer RESTful API
- Stellt Werkzeuge zu Verfügung, um eine API zu beschreiben/dokumentieren
- (Ist ein Community “Fork” von [Flask-RESTPlus](#))

[4]

API Dokumentation



3

```
from flask import Flask
from flask_restx import Resource, Api

app = Flask(__name__)
api = Api(app)

@api.route('/hello')
class HelloWorld(Resource):
    def get(self):
        return 'Hello, World!'
```

EXAMPLE

API Dokumentation



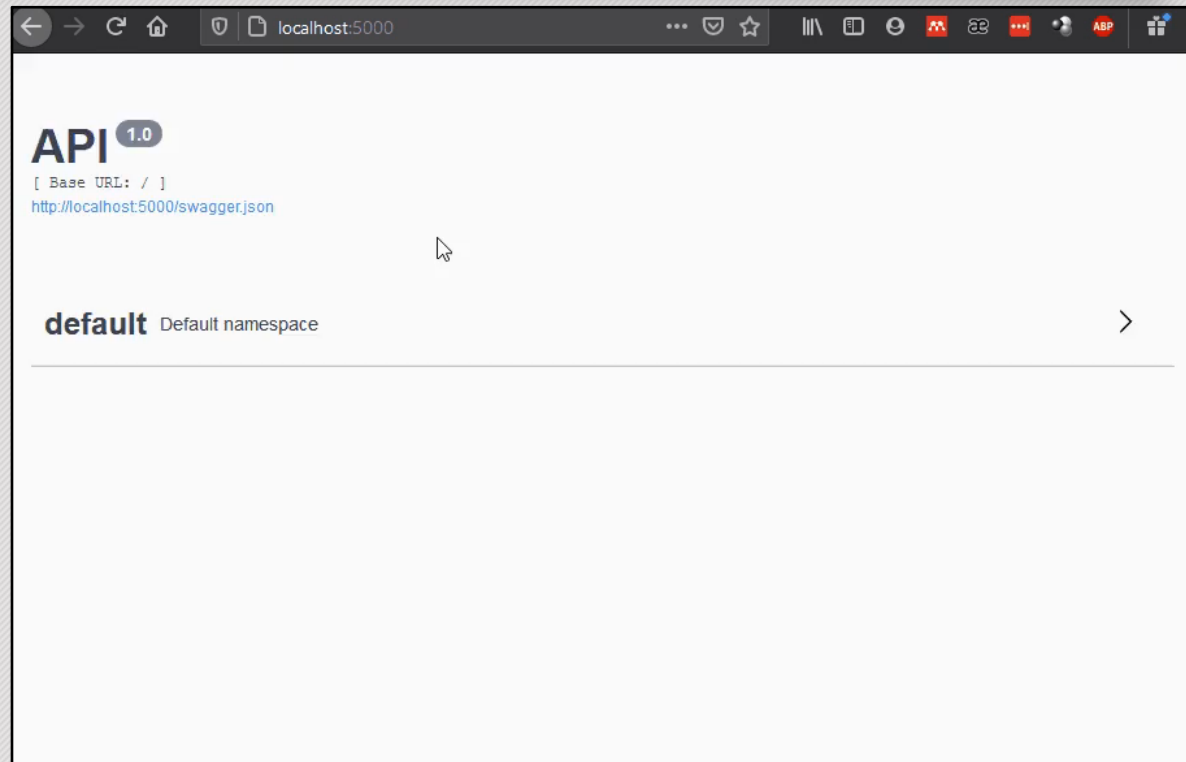
3

```
from flask import Flask
from flask_restx import Resource, Api

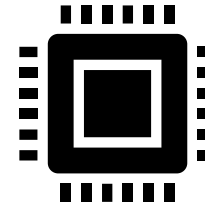
app = Flask(__name__)
api = Api(app)

@api.route('/hello')
class HelloWorld(Resource):
    def get(self):
        return 'Hello, World!'
```

EXAMPLE



Unser Modell als API

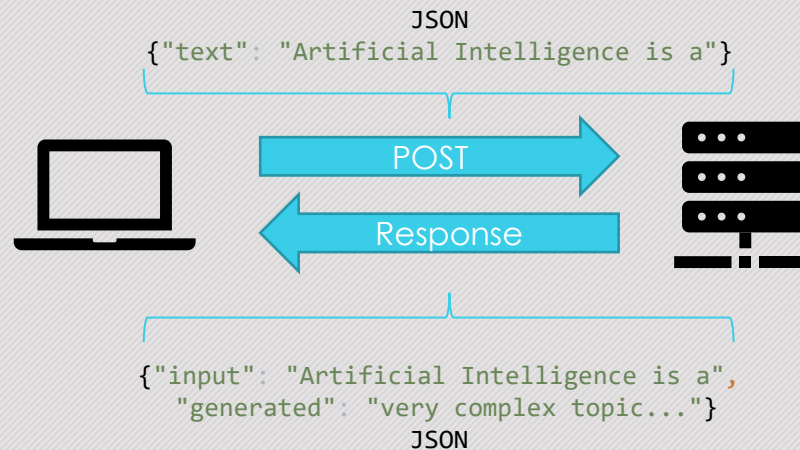


4

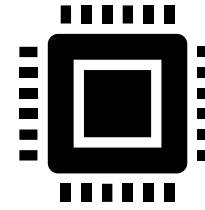
- „Hello World“ API:



- Text-Generation API:



Unser Modell als API



4

```
from flask import Flask, request
from flask_restx import Resource, Api, fields

app = Flask(__name__)
api = Api(app, version="0.1", title="AI Text-Generation")
ns = api.namespace("text-generation")

# Swagger defs
generation_input_def = api.model("Text Generation Input", {
    'text': fields.String(required=True, description="Input prompt",
        help="Text cannot be blank.", example="Artificial Intelligence is a")
})

... # Code for loading model

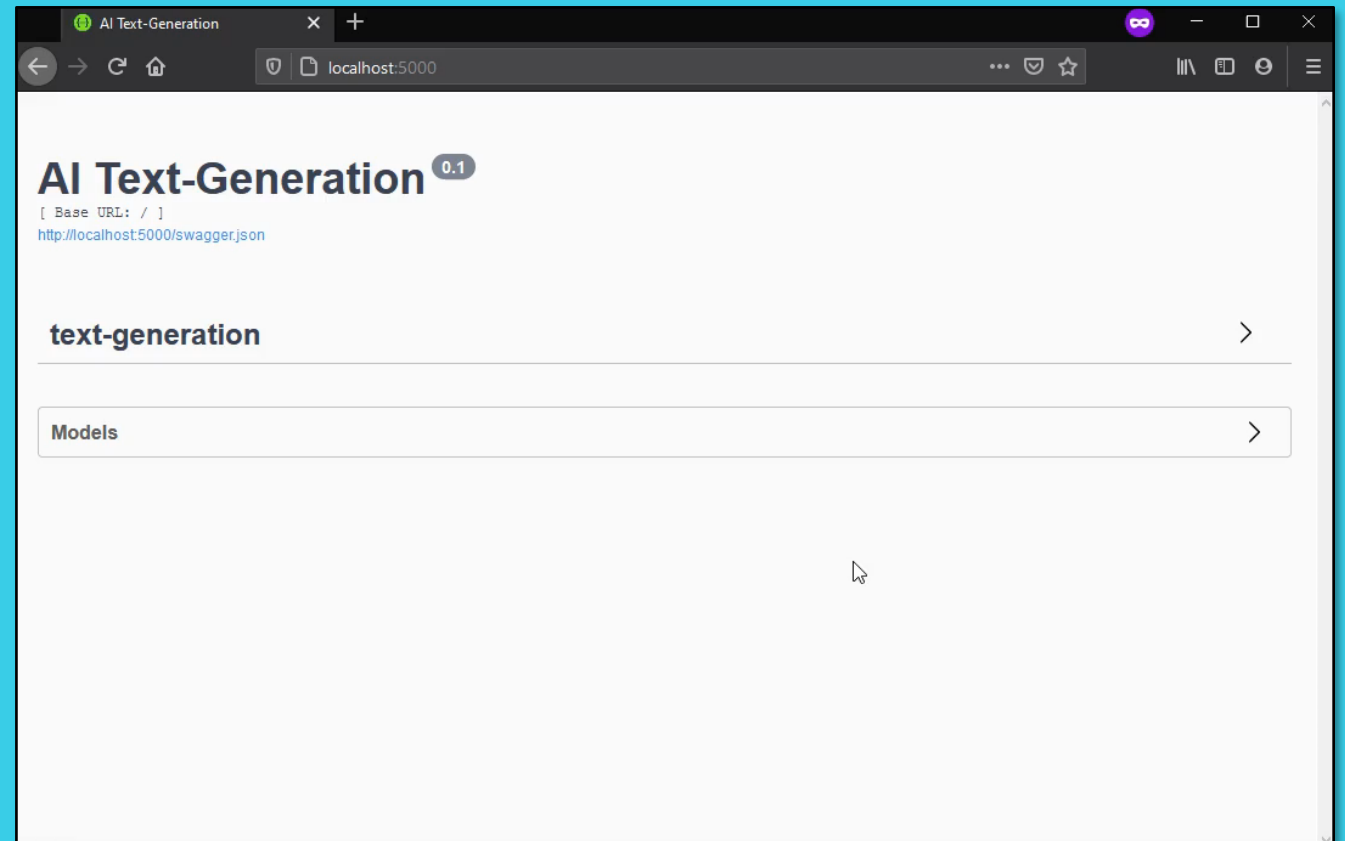
@ns.route('/generate')
class Generation(Resource):
    @ns.expect(generation_input_def)
    def post(self):
        input_text = request.json['text']

        ... # model predict code

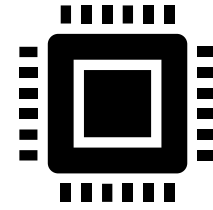
        return {'input': input_text, 'generated': generated_text}
```

EXAMPLE

Unser Modell als API



Unser Modell als API



4

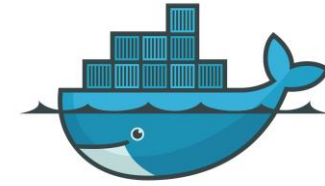
“

[Artificial Intelligence is a] popular technology that enables robots to predict what human beings can do by analyzing their body language and facial expressions.

GPT-2

”

Docker



5

- Virtualisierungssoftware
- Anwendungen können in Container isoliert werden (Image)
- Docker Hub ermöglicht die Verteilung von Images

[5]

Docker und GPUs



5

- [NVIDIA Container Toolkit](#)
 - ermöglicht die Ausführung von GPU-beschleunigten Containern

```
docker run --gpus all <your-container>
```

```
docker run --gpus device=1 <your-container>
```

API ➡ Docker



5



Dockerfile

```
FROM pytorch/pytorch ①
WORKDIR /usr/src/app

COPY . . ②

RUN pip install -r ./requirements.txt ③
RUN python ./src/download.py ④

ENTRYPOINT [ "python" ]
CMD [ "src/service.py" ] ⑤
```

File: requirements.txt

```
flask~=1.1.2
flask-restx~=0.2.0
transformers~=4.1.1
```

File: download.py

```
from transformers import AutoTokenizer, AutoModelForCausalLM

model_str = "gpt2"
AutoTokenizer.from_pretrained(model_str).save_pretrained(f"my-path/tokenizer")
AutoModelForCausalLM.from_pretrained(model_str).save_pretrained(f"my-path/model")
```

File: service.py

```
(...)

if __name__ == '__main__':
    app.run(host="0.0.0.0", port=5000)
```

API ➡ Docker



5

1. Image bauen

```
docker build -t <your_username>/my-repo .
```

2. Image verteilen

```
docker push <your_username>/my-repo
```

3. Auf gewünschtem System

```
docker pull <your_username>/my-repo  
docker run -p 80:5000 <your_username>/my-repo
```


Google Cloud Platform (GCP)



6

- Infrastructure as a Service (IaaS)
- Compute Engine
 - VM Instanzen erstellen
 - Selbst konfigurieren (installieren aller Treiber etc.)
 - Vorgefertigte Deep Learning Instanzen
- AI Platform
 - Tools für vollständigen ML Lebenszyklus
- 300\$ Startguthaben

[6]

API ➡ GCP



6

1. GCP Instanz erstellen
2. Über SSH einloggen
3. Docker Image pullen und starten



API Live-Demo

GitHub Projekt



- Source Code:

<https://github.com/RafaelWO/NLP-Microservice>

- Docker Image:

<https://hub.docker.com/repository/docker/rafaelwo/nlp-ms>

Weiterführende Themen

- Flask - Production Deployment:
<https://flask.palletsprojects.com/en/1.1.x/deploying/#deployment>
- Flask - Einfache Authentication via API-Key:
<https://blog.ruanbekker.com/blog/2018/06/01/add-a-authentication-header-to-your-python-flask-app/>
- PyTorch Modell mit Flask:
https://pytorch.org/tutorials/intermediate/flask_rest_api_tutorial.html

Weiterführende Themen

- Keras Modell mit Flask:
<https://blog.keras.io/building-a-simple-keras-deep-learning-rest-api.html>

Referenzen

- [1] <https://www.cloudcomputing-insider.de/was-ist-eine-rest-api-a-611116/>
- [2] Radford, Alec, et al. "Language models are unsupervised multitask learners." OpenAI blog 1.8 (2019): 9. ([link](#))
- [3] <https://flask.palletsprojects.com/en/1.1.x/>
- [4] <https://flask-restx.readthedocs.io/en/latest/>
- [5] <https://docs.docker.com/>
- [6] <https://cloud.google.com/>, <https://cloud.google.com/ai-platform>