LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Vigilanz Kulturen
SFB 1369

vigil.AI

Teilprojekt „Digitale Infrastruktur“

Dr. Markus Frank M.P.A.
Prof. Dr. Erika Thomalla

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Vortragsinhalte

Teilprojekt „Digitale Infrastruktur – vigil.AI“

1. Projekthintergrund
2. KI/ML-Einsatz
   a. LLM-Einbettung
   b. RAG optimierte Datenstruktur
   c. Stable Diffusion Generierung
3. Die Infrastruktur
   a. Technische Spezifikation
   b. Einzelkomponenten: vigil.AI, Augmented Reading, Knowledge Base
4. Roadmap
   a. Antragsprototyp
   b. SFB Förderphase 3
   c. Projektübergreifende Integration

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

1. Projekthintergrund

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

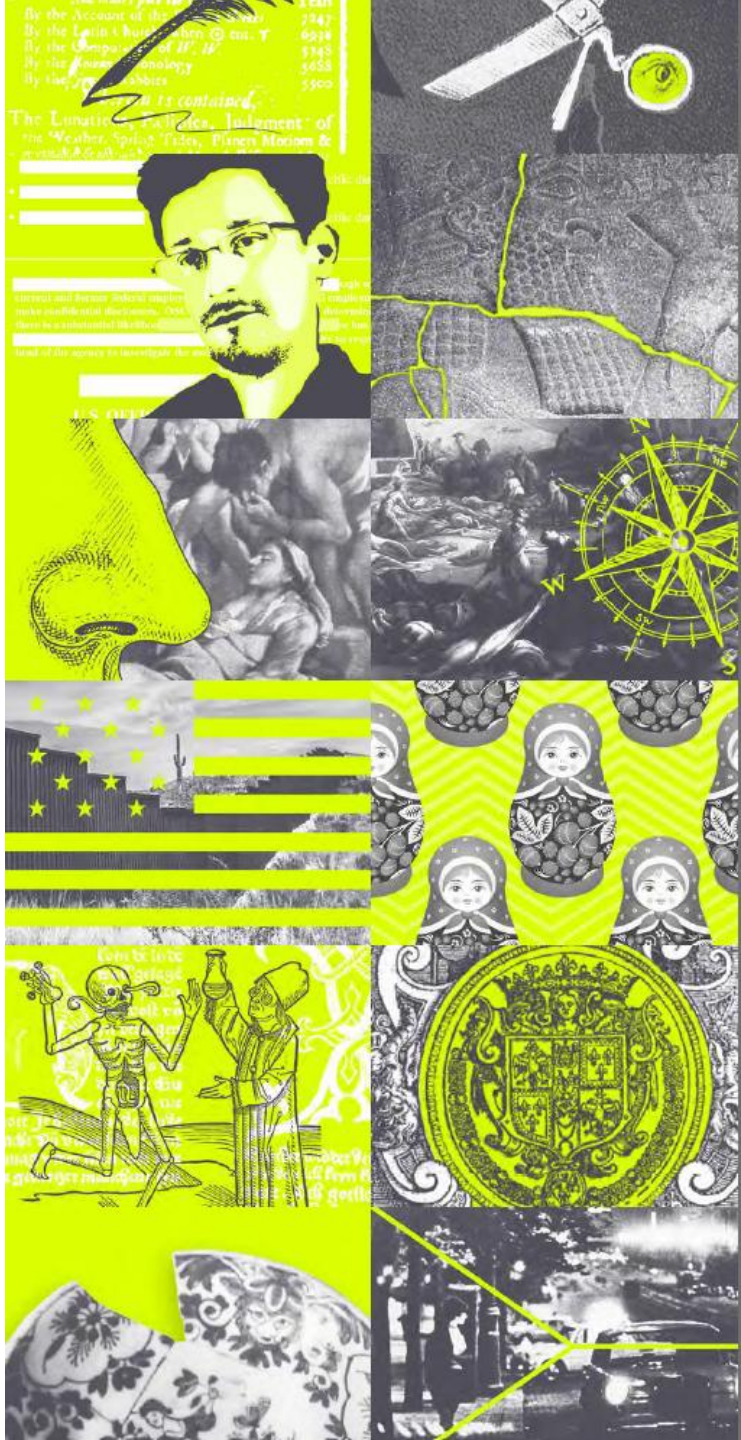
Projekthintergrund

SFB 1369 Vigilanzkulturen

- ➔ DFG Sonderforschungsbereich, aktuell 2. Förderphase
- ➔ https://www.sfb1369.uni-muenchen.de/index.html

- ➔ Umfang (aktuell)
  - Vorstand (Sprecher): Arndt Brendecke, Julia Burkhardt
  - 18 Teilprojekte
  - 24 Teilprojektleiter, ca. 36 wiss.MA, ca. 4 Management, 22 stud.HK

- ➔ Veröffentlichungen
  - Große Publikationsreihe bei De Gruyter (16 Bände)
  - Mehrere kleinere Reihen & Working Papers
  - Seit 2018 mehrere hundert Veröffentlichungen



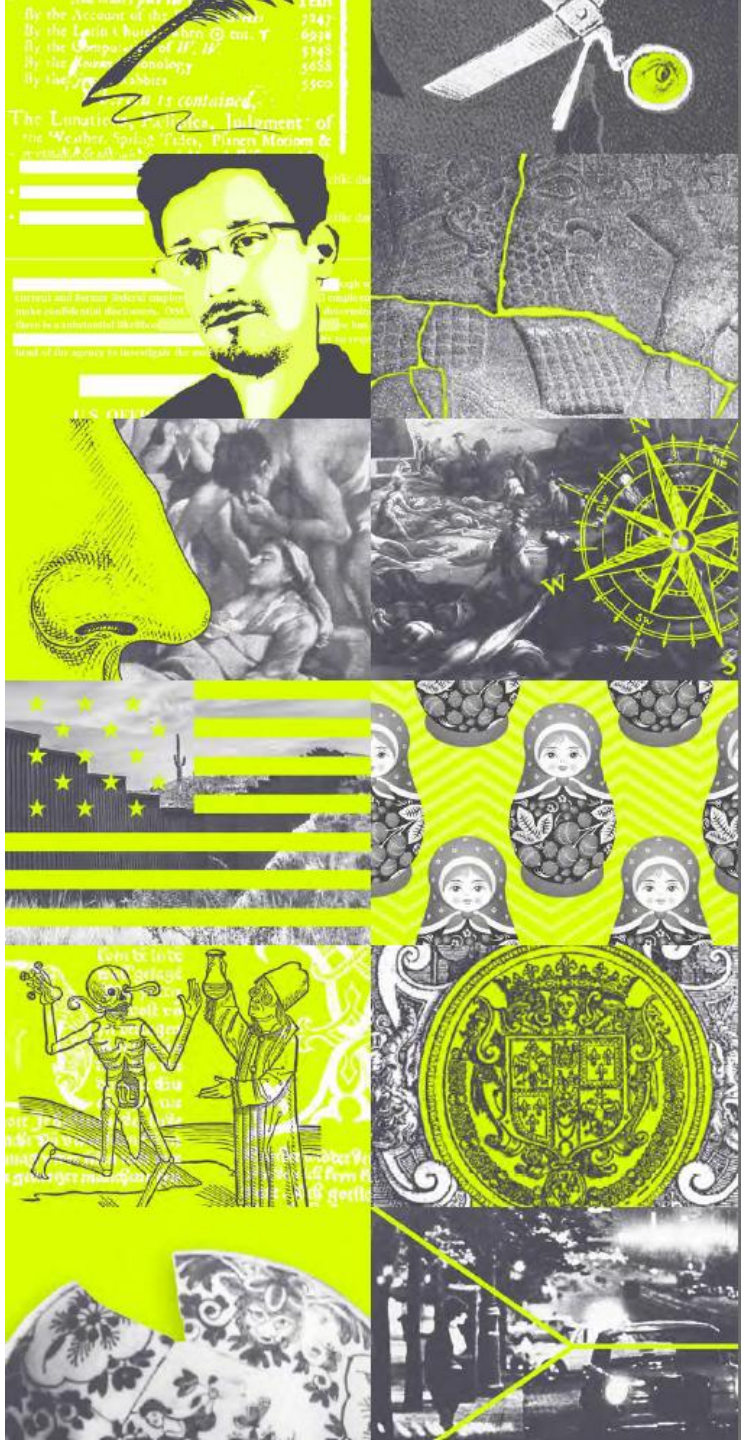
LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Projekthintergrund

Teilprojekt „Digitale Infrastruktur – vigil.AI“

- ➔ Anfrage vom SFB-Vorstand: Erstellung einer digitalen Infrastruktur für den gesellschaftlichen Transfer der SFB-Forschungsergebnisse, Infrastruktur soll in der 3. Förderphase Status eines eigenen Teilprojektes erhalten

- ➔ Vorschlag der **vigil.AI**-Infrastruktur
- ➔ Teilprojektleitung: Markus Frank, Erika Thomalla
- ➔ Roadmap:
  - 2026: Erstellung Antagsprototyp
  - Ende 2026: Begehung der 2. Förderphase
  - Anfang 2027: Antrag auf Förderung einer 3. Phase
  - Herbst 2027: potenzieller Förderbeginn



Projekthintergrund

Teilprojekt „Digitale Infrastruktur – vigil.AI“

➔ Ressourcen (genehmigt & beantragt)

Antragsprototyp

65% wiss.MA (Alisa Fenske)
Interface Entwicklung/Adaption, KI-Integration

Markus Frank
KI-Implementierung (u.A. RAG, LoRA Tuning), Usability Testung

3. Förderphase

100% wiss.MA Postdoc
KI-Implementierung, Aufbau Instruction-Korpus, Usability Testung

75% wiss.MA
Interface Entwicklung, Usability Testung

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

2. KI/ML-Einsatz

LLM Einbettung

Wie werden LLMs in das Projekt eingebunden?

OpenAI Modelle

Zugriff über die OpenAI-API

Alle LLM-Funktionen der Infrastruktur in maximaler Modellqualität (z.B. GPT 5.1) + Embeddings

Einsatz im Prototyp
Öffentliche Demonstrationen, Gutachterzugang

ITG RTX 4000 Server

Gehostet als Docker vllm Image

Alle LLM-Funktionen der Infrastruktur mit OA-Modellen + Embeddings

Einsatz im Prototyp
Interne Entwicklung & Evaluation

In der 3. Förderphase:
Ablösung der OpenAI-Modelle

LLM Einbettung

OpenAI Modelle

GPT-5.1	GPT-5 mini	GPT-5 nano	GPT-5 pro
The best model for coding and agentic tasks across industries	A faster, cheaper version of GPT-5 for well-defined tasks	The fastest, cheapest version of GPT-5—great for summarization and classification tasks	The smartest and most precise model
Price	**Price**	**Price**	**Price**
Input: $1.250 / 1M tokens	Input: $0.250 / 1M tokens	Input: $0.050 / 1M tokens	Input: $15.00 / 1M tokens
Cached input: $0.125 / 1M tokens	Cached input: $0.025 / 1M tokens	Cached input: $0.005 / 1M tokens	Cached input: -
Output: $10.000 / 1M tokens	Output: $2.000 / 1M tokens	Output: $0.400 / 1M tokens	Output: $120.00 / 1M tokens

- ➔ Nutzung der OpenAI API über Python FastAPI-Serverbackend
- ➔ Budget: vorläufig 100 $/mo
- ➔ Kapazitätsmaximum für die Modi
  - **Augmented Reading**: ca. 7.500 komplexe Aufrufe /mo
  - **vigil.AI**: ca. 25.000 Konversation Turns /mo

```python
import os
from openai import OpenAI

client = OpenAI(
    api_key=os.environ.get("OPENAI_API_KEY"),
)

response = client.responses.create(
    model="gpt-5.1",
    instructions="Systemprompt",
    input="Nutzereingabe",
)

print(response.output_text)
```

LLM Einbettung

ITG RTX 4000 PRO Blackwell Server

- ➔ Ebenfalls Zugriff über die OpenAI-API, lokale Hostung der KI-Funktionalitäten in LM Studio.

- ➔ Neuer ITG-Grafikkartenserver: 4x RTX 4000 PRO Blackwell, je
  - Architektur: Blackwell
  - CUDA: 8960 Kerne
  - NVIDIA TENSOR: 280 Kerne
  - Speicher: 24 GB GDDR7 (672 GB/s)
- ➔ Optimal für: Betrieb kleinerer Sprachmodelle, Bildgenerierung, Berechnung Embeddings
- ➔ Nicht gut geeignet: Betrieb großer Sprachmodelle, Fine-Tuning von größeren Modellen

```python
import os
from openai import OpenAI

client = OpenAI(
    base_url="http://127.0.0.1:1234/v1/",
    api_key="lm-studio")
)

response = client.responses.create(
    model="gpt-oss-20B",
    instructions="Systemprompt",
    input="Nutzereingabe",
)

print(response.output_text)
```

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

LLM Einbettung

ITG RTX 4000 PRO Blackwell Server

Modellbetrieb:

➔ Echtzeit-Kommunikation (VRAM)
- 1x RTX: **Modelle bis 32B Q4** (aktuell: GPT-OSS-20B, QWEN3 32B, Mistral Small)
- 4x RTX: **Modelle bis 80B Q4** (aktuell: Qwen3-next mit Tensor-Parallelization)

➔ Gehostet werden z.B.
- 2 RTX: ein LLM (Tensor-Parallelization)
- 1 RTX: Embedding-Modell, Stable Diffusion
- 1 RTX: dynamischer Testbetrieb

1x RTX
docker/vllm
GPT-OSS-20B
Kontext: 8.192
--max-num-seqs 4

```
# Setup für den normalen, internen Modellbetrieb, bei dem 3 N
# und relativ einfache Modellaufgaben, die nur 8192 Tokens Ko
docker run -d \
  --name ITG-GPT-20B-1X \
  --gpus '"device=0"' \
  --ipc=host \
  -p 8001:8000 \
  -v /srv/hf-cache:/root/.cache/huggingface \
  vllm/vllm-openai:latest \
  --model openai/gpt-oss-20b \
  --kv-cache-dtype fp8 \
  --async-scheduling \
  --no-enable-prefix-caching \
  --max-model-len 8192 \
  --gpu-memory-utilization 0.85 \
  --max-num-seqs 4 \
  --max-num-batched-tokens 8192 \
  --api-key ITG-OSS
  # --seed 42 \ # optionaler fixer Seed-Wert für Reproduktion
```

2x RTX
docker/vllm
GPT-OSS-20B
Kontext: 16.384
--max-num-seqs 6

```
# GPT OSS-Konfiguration für 2 x RTX 4000 (Tensor Parallelism)
# Setup für den öffentlichen (zugangsbeschränkten) Modellbetr
# alternativ schwierige Aufgaben mit weniger Nutzern und sehr
docker run -d \
  --name ITG-GPT-20B-2X \
  --gpus '"device=0,1"' \
  --ipc=host \
  -p 8001:8000 \
  -e CUDA_VISIBLE_DEVICES=0,1 \
  -v /srv/hf-cache:/root/.cache/huggingface \
  vllm/vllm-openai:latest \
  --model openai/gpt-oss-20b \
  --kv-cache-dtype fp8 \
  --async-scheduling \
  --no-enable-prefix-caching \
  --tensor-parallel-size 2 \
  --max-model-len 16384 \
  --gpu-memory-utilization 0.85 \
  --max-num-seqs 6 \
  --max-num-batched-tokens 16384 \
  --swap-space 4 \
  --disable-log-requests \
  --api-key ITG-OSS
  # --seed 42
```

Zielperformance
Augmented Reading: 50 tps
vigil.AI: 20 tps

LLM Einbettung

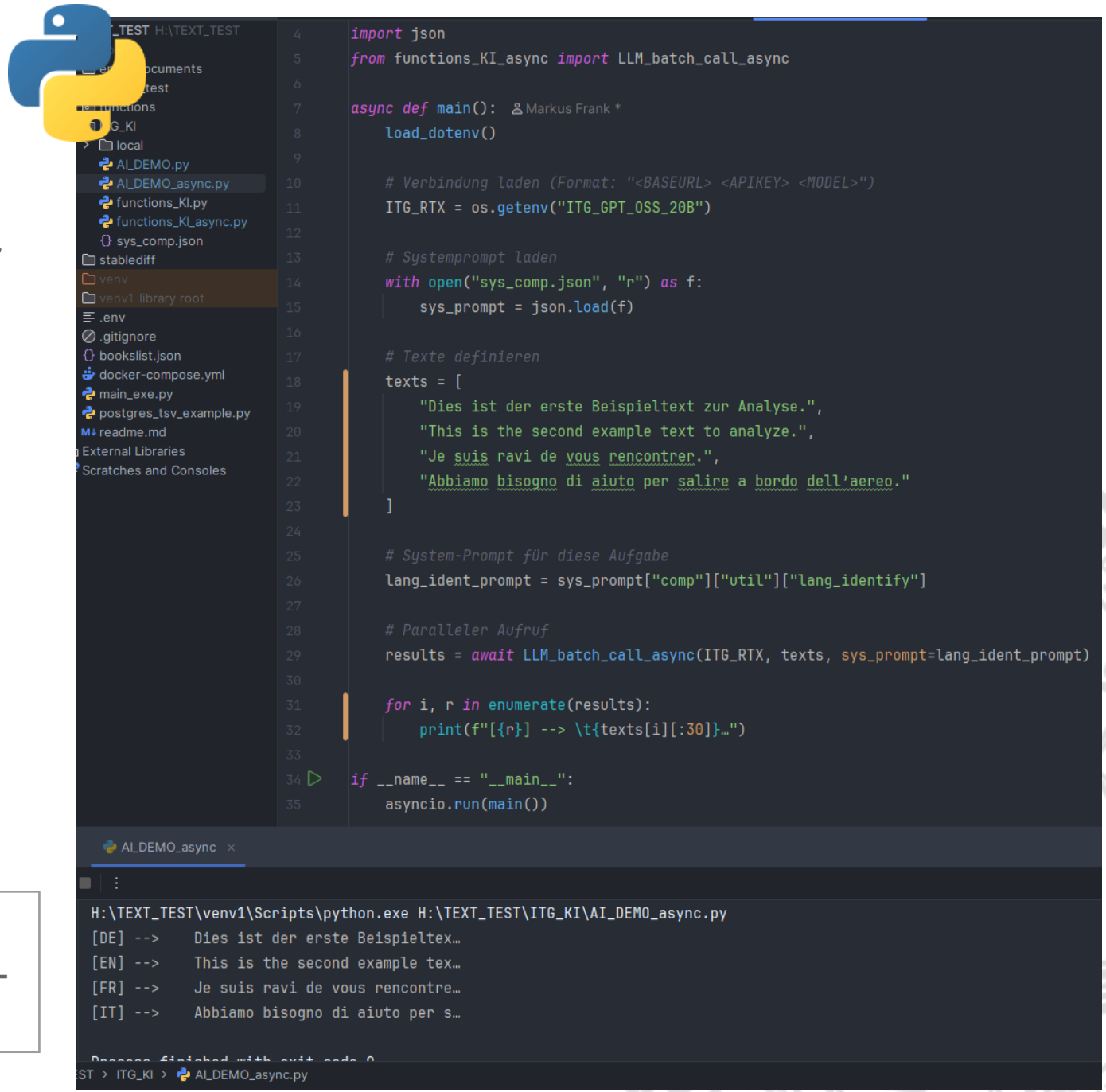
ITG RTX 4000 PRO Blackwell Server

Programmierparadigma

- ➔ Setup kann mehrere Anfragen parallel als Batch verarbeiten
- ➔ Anfragen sollten auf *--max-num-seqs* abgestimmt sein
- ➔ Asynchrone Ansprache der API bringt starken Leistungszuwachs
- ➔ Python: **asyncio** verwenden und je nach Szenario gather() oder as_completed()

Beispiel

Test eines Utility Systemprompts zur Sprach-Identifikation



RAG optimierte Datenstruktur

Zwei Ansätze für den Modelleinsatz

Retrieval Augmented Generation (RAG)

Ansatz des Prompt/Kontext-Engineering

Prompt wird vor der LLM-Anfrage mit allen Informationen angereichert, die für eine gute Antwort benötigt werden.

Ziel: Minimierung von Halluzinationen

Vorteile: LLM beliebig austauschbar, Datenstruktur ohne LLM nutzbar, kein Training

Low-Rank Adaptation Fine-Tuning

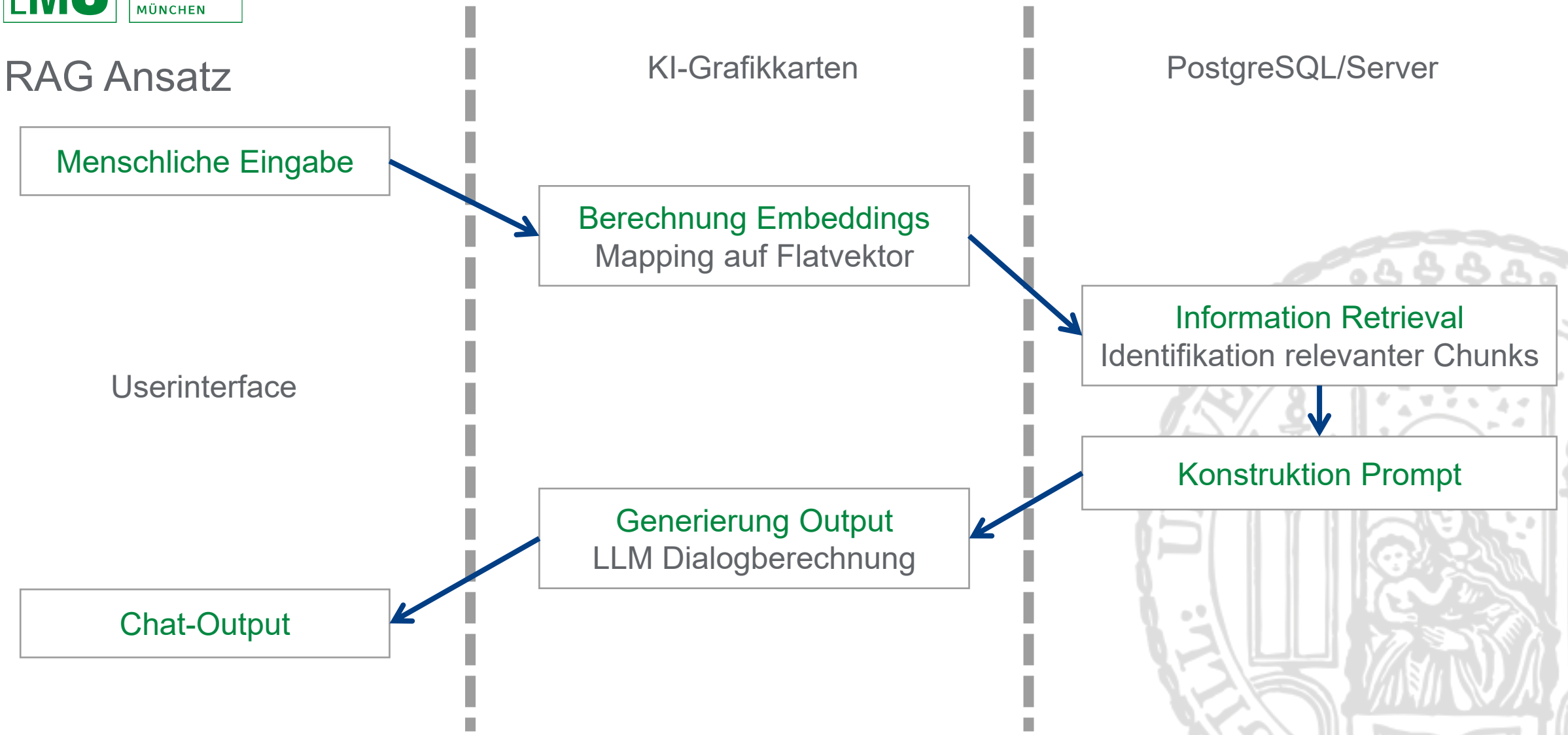
Ansatz des Modell-Finetunings

Ausgewählte Schichten eines LLMs werden mithilfe von Instructions neu trainiert.

Ziel: Verbesserung des Antwortverhaltens

Nachteile: Auf getuntes LLM festgelegt, Eigenbetrieb erforderlich, großes Instructions-Korpus erforderlich, Training hardware- & zeitaufwändig (trotz QLoRA)

RAG optimierte Datenstruktur

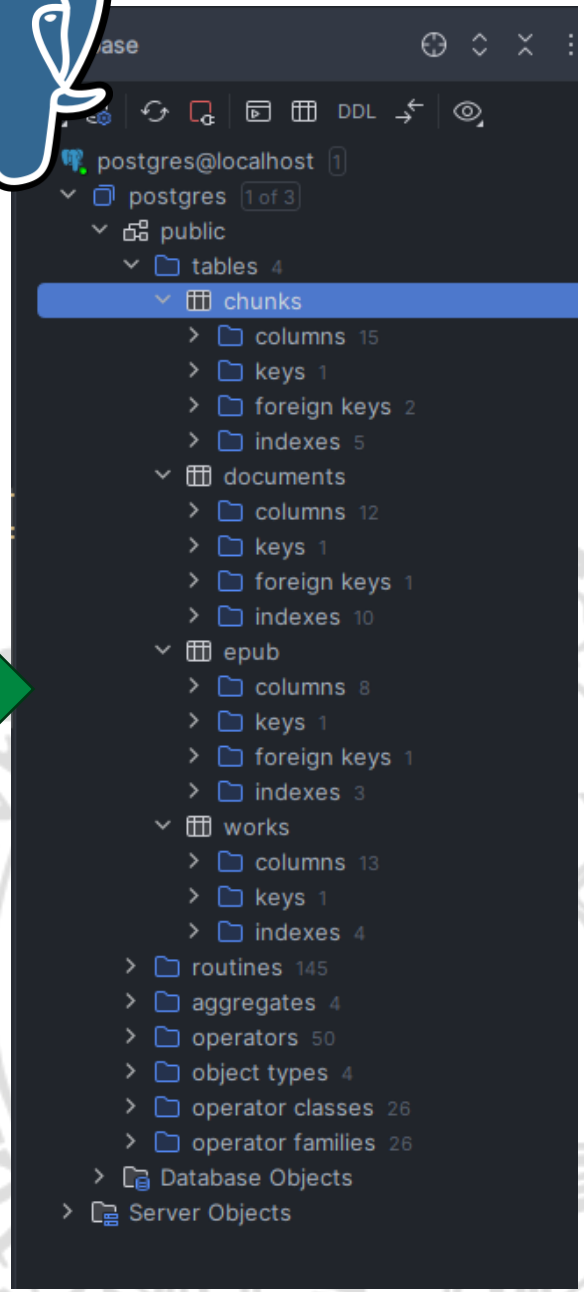


RAG optimierte Datenstruktur

Dokument-Aufbereitung

1. Quelle: epub-Version der Veröffentlichungen
2. Python-Extraktion aller Informationen
3. Chunking der Fließtexte
4. Erzeugung eines TSV-Index auf Fließtexte
5. Berechnung Embeddings für Fließtexte


```
[FO]: Docker-Container bereit.

[INFO]  pub-Files extrahieren...
--> Extrahiert: 14 (xml & png)
[INFO]: epub-Dateien erfolgreich extrahiert.

[INFO]  Metadaten Band extrahieren...
--> Extrahiert: 14 Metadatensätze
[INFO]: Metadaten erfolgreich extrahiert.

[INFO]  Metadaten Kapitel extrahieren...
--> Extrahiert: 233 Metadatensätze
[INFO]: Kapitel erfolgreich extrahiert.

[INFO]  Kapitel Chunks generieren...
--> Generiert: 1362 Chunks
[INFO]: Chunks erfolgreich extrahiert.

[INFO]  JSON aus epub erstellen...
--> Erstellt: 236 epub-Datensätze
[INFO]: epub-XML erfolgreich extrahiert.

[INFO] Relationen erstellt bzw. geleert
[INFO]: Works importiert.
[INFO]: Documents importiert.
[INFO]: Chunks importiert.
[INFO]: EPUB importiert.
--> TSV indiziert: 542 Chunks (deutsch)
--> TSV indiziert: 570 Chunks (englisch)
--> TSV indiziert: 101 Chunks (italienisch)
--> TSV indiziert: 149 Chunks (französisch)

[INFO]: Datenbank erfolgreich erstellt und Daten eingetragen.
```



RAG optimierte Datenstruktur

Chunking von Information

Chunk-Relation enthält den gesamten Fließtext aller Veröffentlichungen in überlappenden Chunks zu je ca. 800 Wörtern

Band 1 Kapitel 1: 3223 Wörter

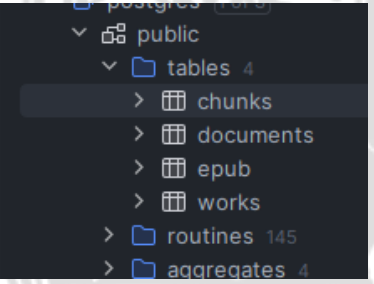
Chunk 1 W 0-847

Chunk 2 W 762-1.604

Chunk 3 W 1.526-2.384

Chunk 4 W 2.293-3.223

	index_span	text	tc
1	0:847	Zeiten der Wachsamkeit, der Titel dieses Bandes ruft eine paränetische Figur auf. Sie sug…	
2	762:1604	Dabei ist zu berücksichtigen, dass die verschiedenen temporalen Schichten unterschiedlich…	
3	1526:2384	Dabei ist hervorzuheben, dass die Zeitlichkeit des Wartens und Wachens eine solche Klimax…	
4	2293:3223	Es gehe darum, den Körper und die kognitiven Fähigkeiten zu erschöpfen. Daran anschließen…	



RAG optimierte Datenstruktur

Chunking von Information

Problem: Feste Chunkgrößen (z.B. 800 W) erzeugen suboptimale Chunkgrenzen und oft zu kleine Chunks am Kapitelende.

Lösung: Definition spezifischer Chunking-Funktion

- ➔ Chunkgröße hat eine Toleranz (z.B. 15%)
- ➔ Chunk-Balance: Vor Chunking Berechnung der optimalen Chunkanzahl nach Größe und Überlappung (z.B. 4 bei 3.200 W)
- ➔ Sentence Awareness: Gechunkt wird primär an Satzgrenzen. (Verbesserung der Embedding-Berechnungen)

```python
def sentence_aware_chunks_balanced(  3 usages  Frank, Markus
    text: str,
    target_words: int = 800,
    overlap_words: int = 100,
    tolerance: float = 0.15  # ±15 % um Satzgrenzen zu respektieren
) -> List[Tuple[int, int, str]]:
    """..."""
    if target_words <= 0:
        raise ValueError("target_words muss > 0 sein.")
    if not (0 <= overlap_words < target_words):
        raise ValueError("0 <= overlap_words < target_words muss gelten.")

    sentences = split_sentences(text)
    if not sentences:
        return []

    # Wortlängen je Satz & kumulative Wortindizes: cum[i] = Wörter bis Satz
    sent_words = [len(s.split()) for s in sentences]
    cum = [0]
    for w in sent_words:
        cum.append(cum[-1] + w)
    W = cum[-1]
    if W <= target_words:
        return [(0, W, " ".join(sentences))]
```

RAG optimierte Datenstruktur

Information Retrieval

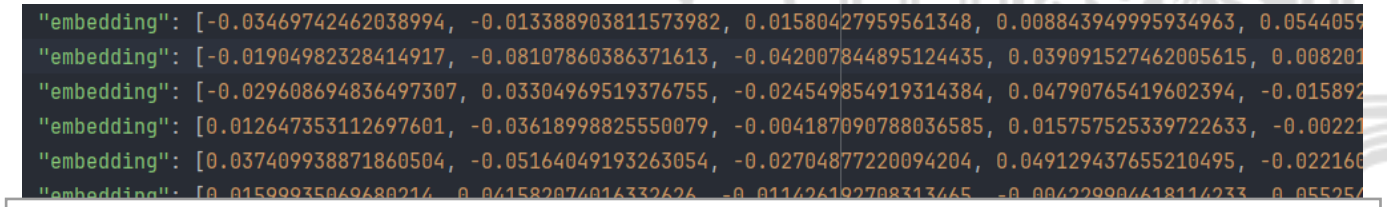
Prompt als String

Prompt als Vektor

```
WITH q AS (
  SELECT
    to_tsquery('german', 'Diskursrelationen')     AS tsq,
    ARRAY[ ... ]::vector   AS query_vec
)
SELECT
    c.chunk_id,
    c.text,
    -- Lexikalisches Ranking
    ts_rank_cd(c.tsv, q.tsq, 32)  AS lex_score,
    -- Semantische Ähnlichkeit (Cosine als 1 - cosine_distance)
    1 - (c.embedding <=> q.query_vec) AS sem_score,
    -- Kombinationsscore (gewichtetes Mittel)
    (
      0.6 * ts_rank_cd(c.tsv, q.tsq, 32) +
      0.4 * (1 - (c.embedding <=> q.query_vec))
    ) AS hybrid_score
FROM chunks c, q
WHERE c.tsv @@ q.tsq
  AND c.lang = 'german'
ORDER BY hybrid_score DESC
LIMIT 10;
```

Resultat: Sehr mächtiges und durch Indizierung sehr performantes datenbankgetriebenes Volltext-Retrievalsystem

- ➔ TSV: positionale, wortlemmatisierte Indizierung des Chunks
- ➔ Embedding-Vektor: Semantische Vektorraumrepräsentation des Chunks (Flat-Vektor: 1536 oder 786 DIM)
- ➔ **hybride lexikalische und semantische Identifikation relevanter Chunks**



Emb.Mod. (786 DIM): jina-embeddings-v2-base-de

RAG optimierte Datenstruktur

Prompt-Konstruktion

Das System identifiziert die lexikalisch und semantisch relevantesten Chunks aus dem Textbestand des SFB. Sie werden vollständig inklusive weiterer Metadaten zur Anreicherung des Prompts verwendet.

- ➔ Chunks: 5-10 Stück, sortiert nach Relevanz (8-15.000 Tokens)
- ➔ Metainformation: Autoren, Kapitel, Werk, Keywords etc.

Die Menge der gesendeten Chunks ist limitiert durch den Modellkontext, die gewünschte Ausgabepräzision und die „Kosten“ ($ oder Rechenleistung)

Systemprompt vigil.AI

```
sysp = f"""
You are a chatbot for communicating with the knowledge base of a
research project. You receive inputs [Frage] that have been enriched
with context [Kontext] by a RAG system. Strictly observe the context.
Strictly observe the desired response behavior [Antwortverhalten].
Speak in a friendly but neutral manner.
"""
```

Angereicherter Prompt

```
prompt = f"""
Beantworte die folgende Frage basierend auf dem bereitgestellten Kontext.
Wenn die Antwort nicht im Kontext enthalten ist, sage ausdrücklich, dass du
über keine gesicherten Informationen verfügst. Liefere darüber hinaus jedoch
naheliegende Antworten, die nicht absolut gesichert sind.

Frage: {query}

Kontext: {context}

Antwortverhalten:
- Antworte in einem wissenschaftlichen Stil.
- Orientiere dich stilistisch an der Sprache des Kontextes,
antworte aber grundsätzlich auf deutsch.
- Antworte in ganzen Sätzen, strukturiere Sinnabschnitte in Paragraphen.
- Überschreite eine Antwortlänge von 300 Wörtern nicht.
- Vermeide wörtliche Zitate. Wenn Du den Kontext indirekt zitierst,
gib in Klammern dahinter die Autoren, Kapiteltext und Buchtext des Kontextes an.
- Erzeuge Argumentationen unmittelbar aus dem Fließtext des Kontextes.
- Halluziniere keine Textstellen abseits des Kontextes.
"""
```

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

RAG optimierte Datenstruktur

Exkurs: QLoRA Instruction Tuning

Ansatz: Quantized Low-Rank Adaptation Fine-Tuning

- ➔ Ein Open-Weight Modell mit 20-32B Parametern wird mit **Instructions** fine-getuned.
- ➔ Partielles Tuning: nur **low-rank Layer** des Modells
- ➔ Quantisiertes Tuning: Weights der getunten Layer sind mit **4 Bits quantisiert** (Q4)
- ➔ BEISPIEL: 4x RTX 4000 PRO: 5000 Instructions, 3 Iterationen, 20B-Foundationmodell: < 1 Tag Trainingsdauer

- ➔ Tuning speziell auf die Beantwortung von Fragen auf Basis von Kontext: Optimierung auf die angereicherten Prompts der projektspezifischen Inputs.
- ➔ RAG + Tuning arbeiten in Kombination

Instruction-Set (5000 Instanzen)


```
{"instances":
    {
    "instruction":"Erkläre wie Vigilanz im Mittelalter aussieht.",
    "input":"{context}",
    "output":"Im Mittelalter nahm Vigilanz unterschiedliche...",
    },
    {
    "instruction":"Was versteht man allgemein unter Vigilanz?",
    "input":"{context}",
    "output":"Unter Vigilanz verstehen wir im SFB...",
    },
    ...
    4998
    ...
}
```

Stable Diffusion Generierung

Automatische „künstlerische“ Illustration der Daten

- ➔ Inhalte sollen automatisch mit künstlerischen Illustrationen angereichert werden im Stile des SFB-Designs
- ➔ Illustrationen sollen in der aktuell aufgerufenen Information verankert werden.
- ➔ Stil: **Ölgemälde, Neonfarben, dicke Pinselstriche, vibrierende Farben**
- ➔ Stilistik verschleiert Bildfehler im Bereich Gesichter, Hände etc.

Stable Diffusion Generierung

Automatische „künstlerische“ Illustration der Daten

- ➔ Berechnung auf **RTX 4000 PRO 24GB**
- ➔ KI-Modell: **Stable Diffusion 3 Medium** (SD-3-M)
- ➔ Prompt & Negativ-Prompt sind **teils statisch** und werden **teilweise aus dem Kontext** generiert → Stil bleibt gleich, nur Szenario ändert sich
- ➔ Seed-Rotation erzeugt einzigartige Bilder
- ➔ Jeweils Dark- und Light-Versionen
- ➔ Parameter:
  - SIZE: 1024x1024px; STEPS: 28; CFG: 6.0
  - Generierungsdauer: 4 sec

Prompt

oil painting **of people watching cautiously from a house onto the street**, intense neon yellow light, thick impasto brushstrokes, glowing atmosphere, dramatic contrast, rich texture on canvas, warm highlights, expressive painterly realism

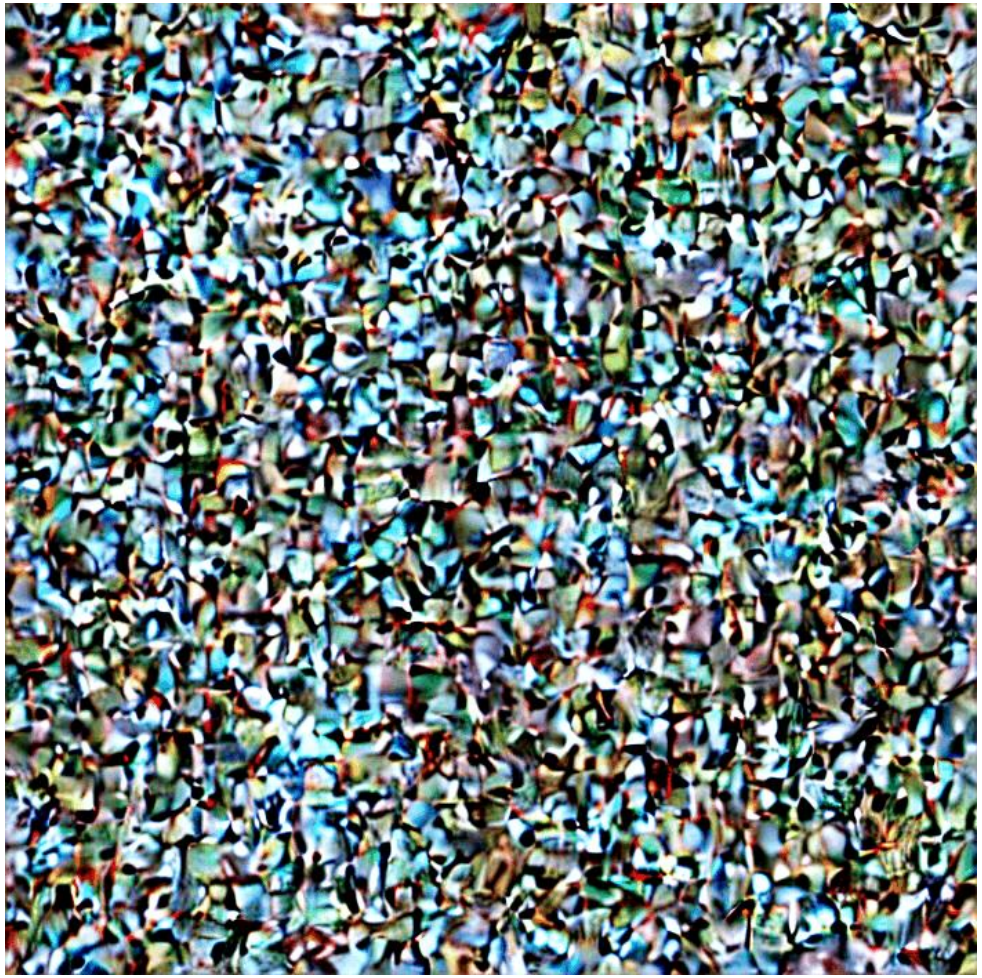
Negativ-Prompt

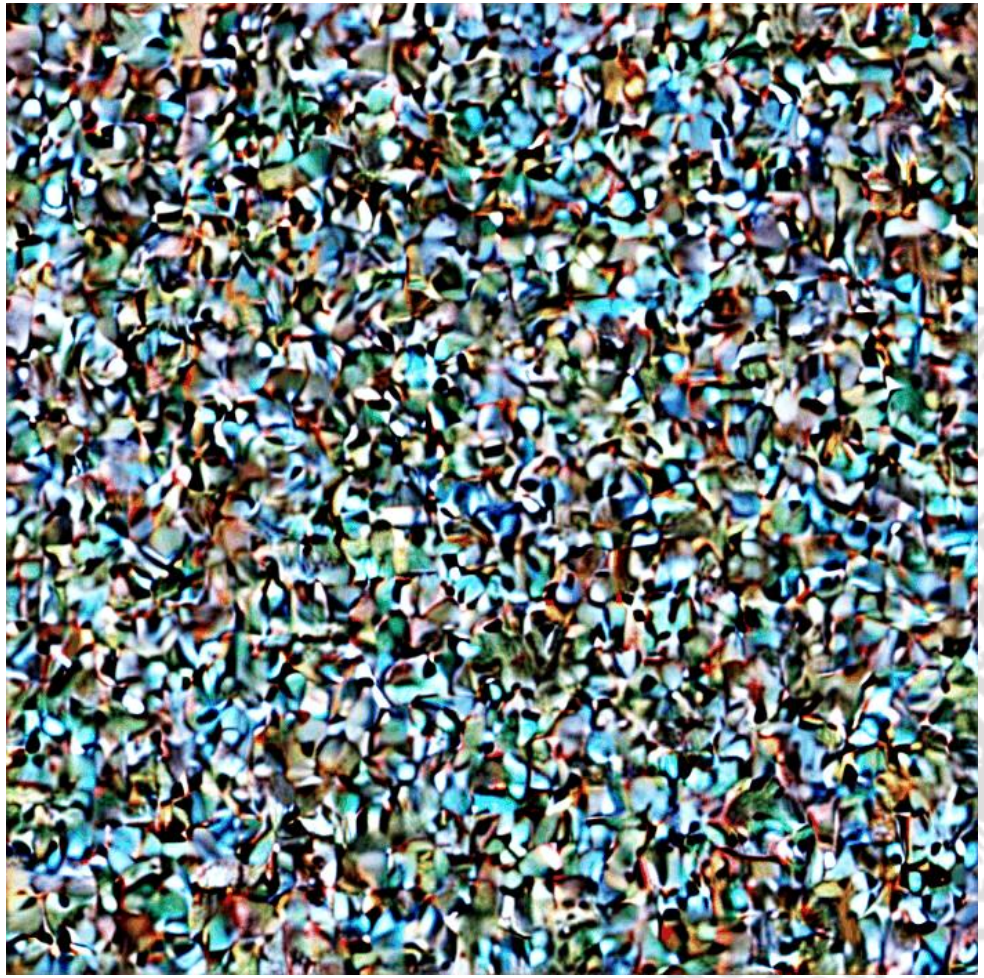
dull colors, low contrast, abstract, watercolor, digital art, cartoon, smooth surface, blurry, text, watermark, signature, plastic look, 3D render, distorted faces, extra limbs, flat lighting

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Stable Diffusion Generierung

32 Steps, SD-1.5 (aus Callback extrahiert)





LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Stable Diffusion Generierung

Automatische „künstlerische“ Illustration der Daten

- **Testlauf** mit Geforce GTX 3070 8GB
- Modell: SD-1.5
- SIZE: 704x704, STEPS: 32, CFG: 6.5
- 100 Bilder, identischer Prompt, **inkrementeller Seed**
- Generierungszeit: ca. 4 sec

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

DH.blue

3. Die Infrastruktur

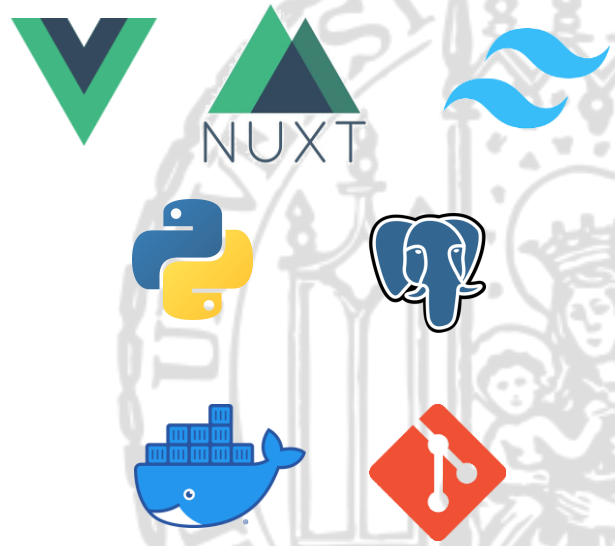
Technische Spezifikation

Techstack

➔ Hardware (ITG)
- Web- & Datenbankserver mit Load-Balancer
- Geforce RTX 4000 PRO Blackwell Grafikkartenserver

➔ Techstack (**DH.blue** Projektinstanz)
- **vue**/nuxt (gebuildetes JS/TS-Frontend)
- **tailwind** (CSS Framework)
- **fastAPI** (schlankes Python-Backend)
- **PostgreSQL** (Projektdatenbank mit tsv- & embedding-Indizes)
- **Docker** (Containerisierung der Komponenten)
- **gitlab** (Versionierung und Continuous Integration)

➔ Erwarteter Umfang: 5.000-10.000 Chunks



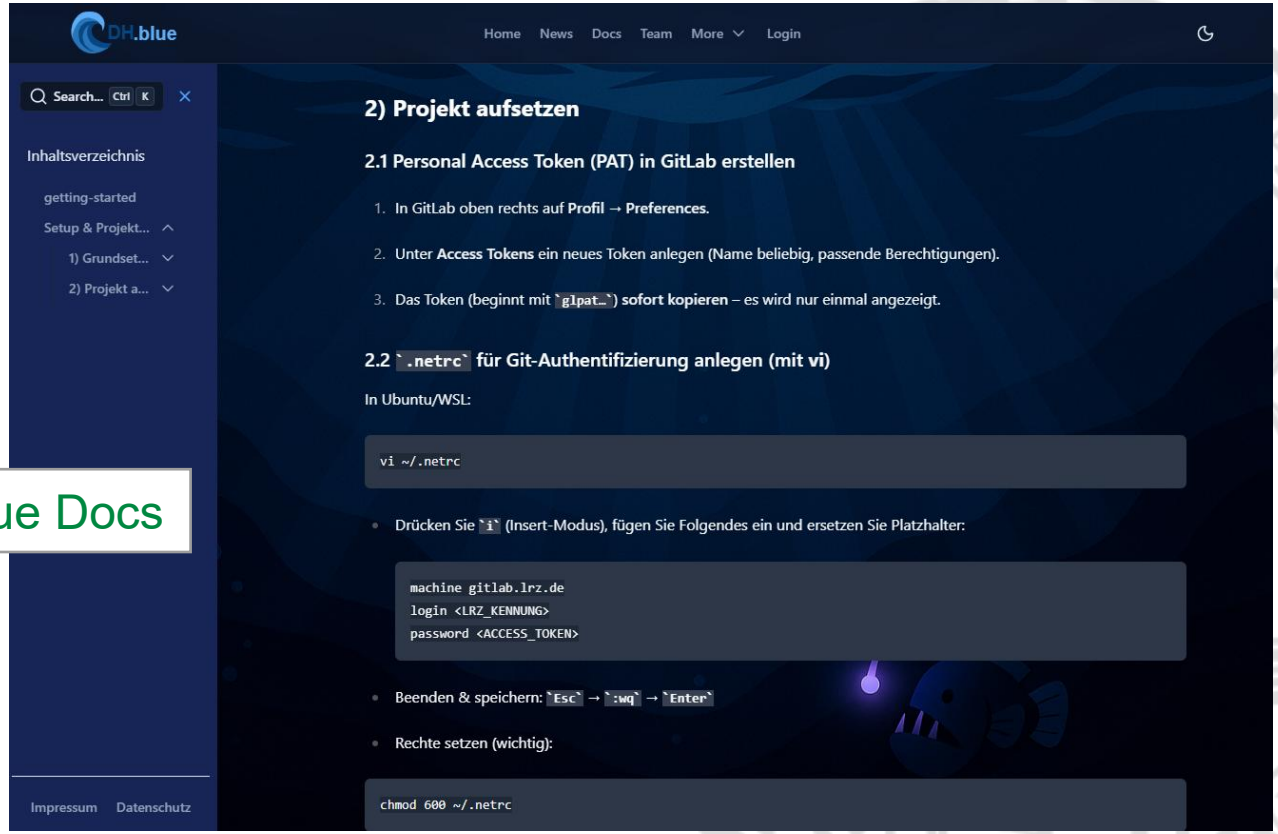
Einzelkomponenten

Knowledge Base

Exploration der gesamten Wissensbasis des SFB
Kein Echtzeit-KI-Einsatz

- ➔ „Klassisches“ Interface mit Schwerpunkt auf:
  - Suche: Metadaten- & Volltextsuche
  - Ausgabe: Werke, Kapitel, Paragraphen (vollständig, mit Abbildungen, Referenzen etc.)
- ➔ Repositoriumsfunktion
- ➔ Rückfalloption bei LLM-Ausfall

DH.blue Docs



Einzelkomponenten

Knowledge Base

GIN = Generalized Inverted Index
BTREE = Balanced Tree
GIN TRGM = GIN Trigram Index
IVFFLAT = Nearest Neighbour Index
GPT VB = LLM vorberechnet
ITG EMB = live berechnet

Suchfunktionen

- GIN TRGM ➔ Autoren & Editoren
- BTREE ➔ Jahr
- GIN TRGM ➔ ISBN / DOI
- GIN TRGM ➔ Sammlungen
- GIN TRGM ➔ Titel [Werk & Kapitelebene] / Kapitelnummern
- GPT VB + GIN TRGM ➔ Keywords [Kapitelebene; Typen: Generell, Personen, Orte]

- GIN TRGM ➔ Volltexte exakt [kapitelbasiert, orthographische Oberfläche]
- GIN ➔ Lexikalische Volltextsuche [chunkbasiert, TSV; lemmatisiert + positional]
- ITG EMB + IVFFLAT ➔ Semantische Volltextsuche [chunkbasiert, emb. Vektorraum; z.B. cos_dist]

Einzelkomponenten

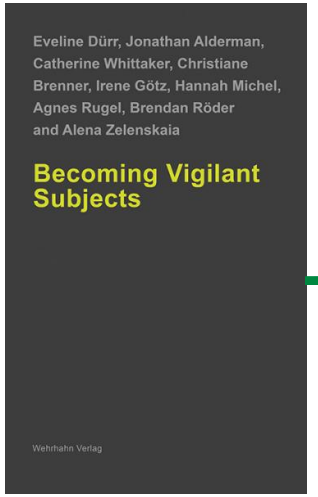
Augmented Reading

Exploration einer multimodal & modifizierten angereicherten Wissensbasis
LLM-Textmodifikation & SD3 Visualisierung

➔ Suchfunktionen entsprechen denjenigen der Knowledge Base
➔ Augmenting KI-Features „**fluide Textransformation**“ (einzeln und in Kombination):
- **Translation**: Übersetzung aller Passagen (DE/EN/FR/IT) in eine gewünschte Zielsprache
- **Stylistic Adaptation**: Texte unterschiedlicher Autoren und Veröffentlichungen werden stilistisch vereinheitlicht
- **Summarization**: Texte werden automatisch zusammengefasst
- **Bridging**: Es werden „naturalistische“ Überleitungen zwischen Texten generiert, die aus unterschiedlichen Quellen zusammengestellt wurden
- **Art Illustration**: Textpassagen werden automatisch „künstlerisch illustriert“

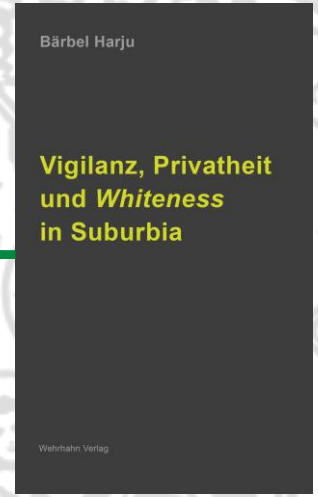
➔ Weiteres: **Georeferenzierung** von Orten (Leaflet), **Graphenvisualisierung** von Netzwerken (Personen, Themen, andere Veröffentlichungen etc.)

Bridging Beispiel (Kleine Reihe, BD 3 → 1, ChatGPT 5.1, 200 W)



Nachdem der vorausgehende Band gezeigt hat, wie Subjektivierungsprozesse in sehr unterschiedlichen historischen, sozialen und kulturellen Kontexten durch Praktiken der Wachsamkeit geprägt werden, eröffnet sich nun die Möglichkeit, diese Einsichten auf ein vermeintlich vertrautes, zugleich aber hochgradig normiertes gesellschaftliches Feld zu übertragen. Die zuvor betrachteten Formen der Aufmerksamkeit – ob an religiösen Schwellen, in rechtlichen Auseinandersetzungen, in Grenzräumen oder in alltäglichen Situationen sozialer Bewertung – verdeutlichten, wie eng Selbst- und Fremdbeobachtung miteinander verwoben sind und wie stark Subjekte durch räumliche Ordnungen, Diskurse und Machtkonstellationen geformt werden.

Der anschließende Text führt diese Perspektive weiter, indem er die US-amerikanische Vorstadt der Nachkriegszeit als einen Raum untersucht, in dem Vigilanz, Privatheit und Identität auf besondere Weise verschränkt sind. Die dortige Betonung von Sicherheit, Sichtbarkeit und Konformität lässt sich als moderne Ausprägung jener Mechanismen lesen, die in den vorherigen Fallstudien bereits in historischen und globalen Variationen sichtbar wurden. So wird deutlich, dass auch in Suburbia Subjektivität nicht einfach gegeben ist, sondern in einem Gefüge aus sozialer Kontrolle, architektonisch erzeugter Transparenz und kulturell codierten Erwartungen entsteht. Die folgende Analyse knüpft damit unmittelbar an die zuvor entwickelten Konzepte an und zeigt, wie Vigilanzkulturen in einem spezifischen gesellschaftlichen Milieu der Moderne weiterwirken und neue Formen des „wachsamen“ Subjekts hervorbringen.



Einzelkomponenten

vigil.AI

Echtzeit Chat mit der Wissensbasis des SFB
Einsatz RAG, getuntes LLM, SD3 Visualisierung

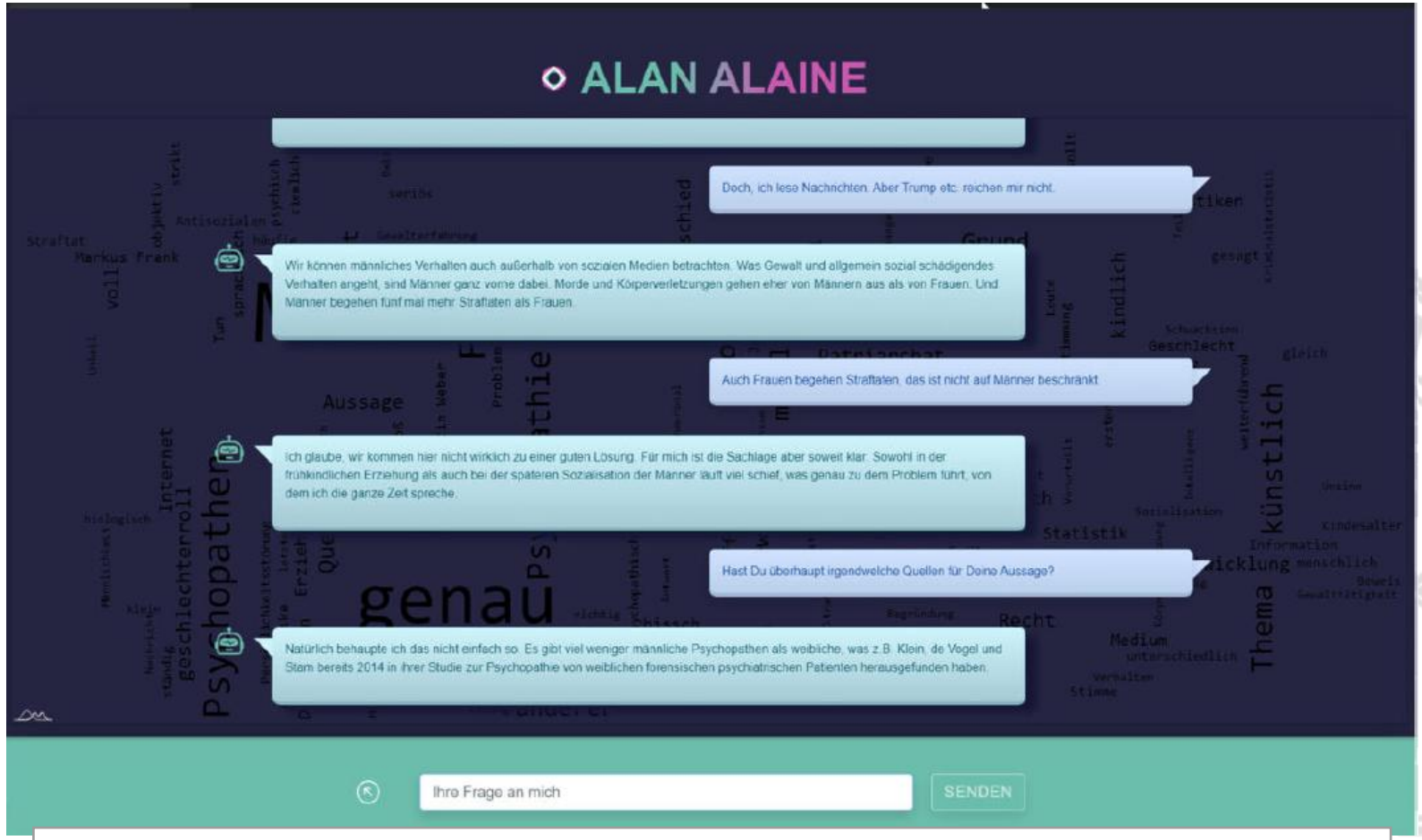
- ➔ Chat auf RAG-Basis zur Minimierung von Halluzinationen
- ➔ LLMs wählbar (extern gehostet & ITG gehostet), RAG ist modellunabhängig
- ➔ Implementierung mündliche Kommunikation denkbar (speech-to-text, text-to-speech)
- ➔ Upload von Texten oder Bildern nicht vorgesehen
- ➔ Interaktionen werden protokolliert für Usability-Tests und spätere Instructions im Rahmen des Finetunings

Einzelkomponenten

vigil.AI

Visuelle Begleitung des Gesprächs durch designmäßige

- ➔ Einbindung der aktuellen Prompt-Embeddings als Designelemente
- ➔ Generierung von passenden Bildern zum aktuellen Konversationskomplex z.B. rechts neben dem Dialogfenster
- ➔ graphbasierte Visualisierung des RAG-Prozesses im Hintergrund z.b. links neben dem Dialogfenster



AlanAlaine Chatinfrastruktur (Frank, Fenske)

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

4. Roadmap

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Antragsprototyp

Projektplan (grob)

Zeit	AI-Development (Markus)	Interface-Entwicklung (Alisa)
Q 1	Datenstrukturen & LLM-Betriebstestung (Testung RAG, ITG RTX, OpenAI)	DH.blue Adaption auf vigil.AI (Anpassungen Frontend & Backend, FastAPI)
Q 2	LLM-Betriebstestung & Finetuning (ITG RTX, Usability-Testung, FT-Pipeline)	Implementierung **Knowledge Base** (Suchfunktionen, Textoberfläche, ITG RTX)
Q 3	Finetuning & Usability (ITG RTX, Usability-Testung, Instructions)	Implementierung **Agumented Reading** (Textoberfläche, ITG RTX, Geomapping, Grafenvisualisierung)
Q 4	Antrag, Finetuning & Usability (Antragsvorbereitung, ITG RTX, Usability-Testung, Instructions)	Implementierung **AI Chatbot** (Chatoberfläche, ITG RTX)

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

SFB Förderphase 3

Projektplan (grob)

- ➔ Antragsprototyp wird iterativ ausgebaut (iteratives Wasserfallmodell, jährliche Iteration), Codebasis dokumentiert und auf github veröffentlicht
- ➔ Öffentlichkeit soll in die Entwicklung der Plattform mit einbezogen werden: kollaborative Workshops
  - Einblicke in die Blackbox „KI“
  - Erarbeitung von Instructions für das Finetuning
- ➔ 3. Förderphase eines SFB steht im Zeigen des Transfers: vigil.AI als Werkzeug der Öffentlichkeitsarbeit und der Erschließung der Ergebnisse für eine breitere Öffentlichkeit. Eine LLM-basierte Infrastruktur kann Ergebnisse fluid und damit zielgruppengerecht erschließen.

Projektübergreifende Integration

Zukünftige Anwendung

- ➔ Ansatz ist bereits generalisiert und kann auf andere Projekte (alle DH.blue-Projekte) übertragen werden.
- ➔ Projektspezifisch sind nur **finegetunte LLMs** (für den Betrieb auch normale LLMs funktional)

EC Cross-Cultural Philology (3061)

Infrastrukturansatz vollständig kompatibel für die gesamte Veröffentlichungsbasis

Übertragung auf DH.blue Instanz des Clusters

Generelle Synergien

auf jedes DH.blue Projekt übertragbar

generell zur Volltextexploration von Veröffentlichungen anwendbar

DH.blue

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Kontaktdaten

Dr. Markus Frank M.P.A.
Markus.Frank@itg.uni-muenchen.de

Ludwig-Maximilians-Universität
IT-Gruppe Geisteswissenschaften
Geschwister-Scholl-Platz 1
80539 München

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

ITG Betrieb

4x RTX 4000 PRO Betrieb

➔ **2 RTX**: 1 GPT-OSS-20B mit 16K Kontext und 6 parallelen Anfragen
- Hauptmodell
- Gute Kontextgröße und Parallelisierung

➔ **1 RTX**: 4 Embedding-Modelle & Stable Diffusion 3.5 Medium
- Embedding-Modelle für DE, EN, FR, IT
- Modell zur Bildgenerierung

➔ **1 RTX**: Testbetrieb
- Karte für experimentelle Tests von Modellen & Konfigurationen
- Nicht vorbelegt, reines Testsetup

```
docker run -d \
  --name ITG-GPT-20B-2X \
  --gpus '"device=0,1"' \
  --ipc=host \
  -p 8001:8000 \
  -e CUDA_VISIBLE_DEVICES=0,1 \
  -v /srv/hf-cache:/root/.cache/huggingface \
  vllm/vllm-openai:latest \
  --model openai/gpt-oss-20b \
  --kv-cache-dtype fp8 \
  --async-scheduling \
  --no-enable-prefix-caching \
  --tensor-parallel-size 2 \
  --max-model-len 16384 \
  --gpu-memory-utilization 0.85 \
  --max-num-seqs 6 \
  --max-num-batched-tokens 16384 \
  --swap-space 4 \
  --disable-log-requests \
  --api-key ITG-OSS
  # --seed 42
```

AI Setupmöglichkeiten

KI-Grafikkarten

RTX PRO 4000 Blackwell

CUDA: 8.960
24 GB GDDR7
192-bit / 672 GB/s
AI TOPS: ~ 800

140 W
Breite: 1
NVLink

ab ca. 1.300 €

RTX PRO 5000 Blackwell

CUDA: 14.080
48 GB GDDR7
512-bit / 1.344 GB/s
AI TOPS: ~ 2.050

300 W
Breite: 2
NVLink

ab ca. 4.200 €

RTX PRO 6000 Blackwell MAX-Q

CUDA: 24.064
96 GB GDDR7
512-bit / 1.792 GB/s
AI TOPS: ~ 3.500

300 W
Breite: 2
NVLink

ab ca. 7.600 €

AI Setupmöglichkeiten

„sichere“ Szenarien für GPT-OSS-20B

1x RTX PRO 4000 Blackwell	1x RTX PRO 5000 Blackwell	1x RTX PRO 6000 Blackwell
Normalbetrieb Kontext: 8.192 4 parallele Anfragen	Normalbetrieb Kontext: 8.192 14 parallele Anfragen	16K RAG-Betrieb Kontext: 16.384 16 parallele Anfragen
16K RAG-Betrieb Kontext: 16.384 2 parallele Anfragen	16K RAG-Betrieb Kontext: 16.384 7 parallele Anfragen	24K RAG Betrieb Kontext: 24.575 10 parallele Anfrage
24K RAG-Betrieb Kontext: 24.575 1 parallele Anfrage	24K RAG-Betrieb Kontext: 24.575 4 parallele Anfrage	32K RAG Betrieb Kontext: 32.786 6 parallele Anfrage