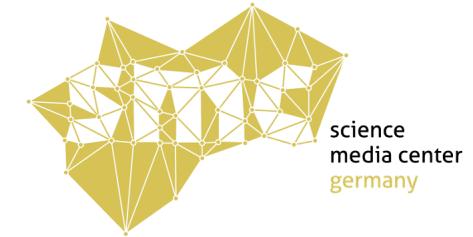




VON DATEN ZUR RAG

EIN PRAXISEINBLICK IN DIE ENTWICKLUNG RAG-BASIERTER CHATBOTS

SciCAR · 2025 · Mani Erfanian Abdoust



THEMEN FÜR HEUTE

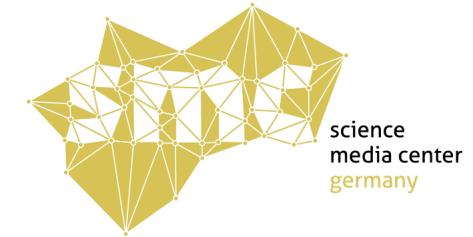
Was ist RAG?

Wie ist ein RAG-System aufgebaut?

Wie lässt sich RAG einsetzen und wie kann man es nutzen?

Schritt für Schritt: Ein eigenes RAG-System aufsetzen (und evaluieren) – Hands-on mit Code-Beispiel

WAS IST DAS PROBLEM MIT "KLASSISCHEN" LLMS?

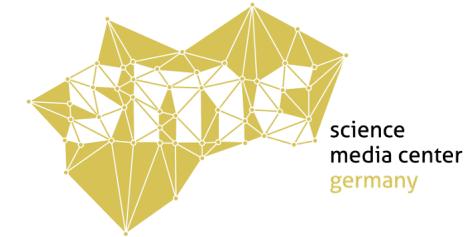


Zeitliche Begrenzung: Wissen nur bis zu einem Stichtag

Thematische Lücken: Bestimmte Bereiche unvollständig abgedeckt



Halluzinationen: Fehlendes Wissen wird plausibel „erfunden“



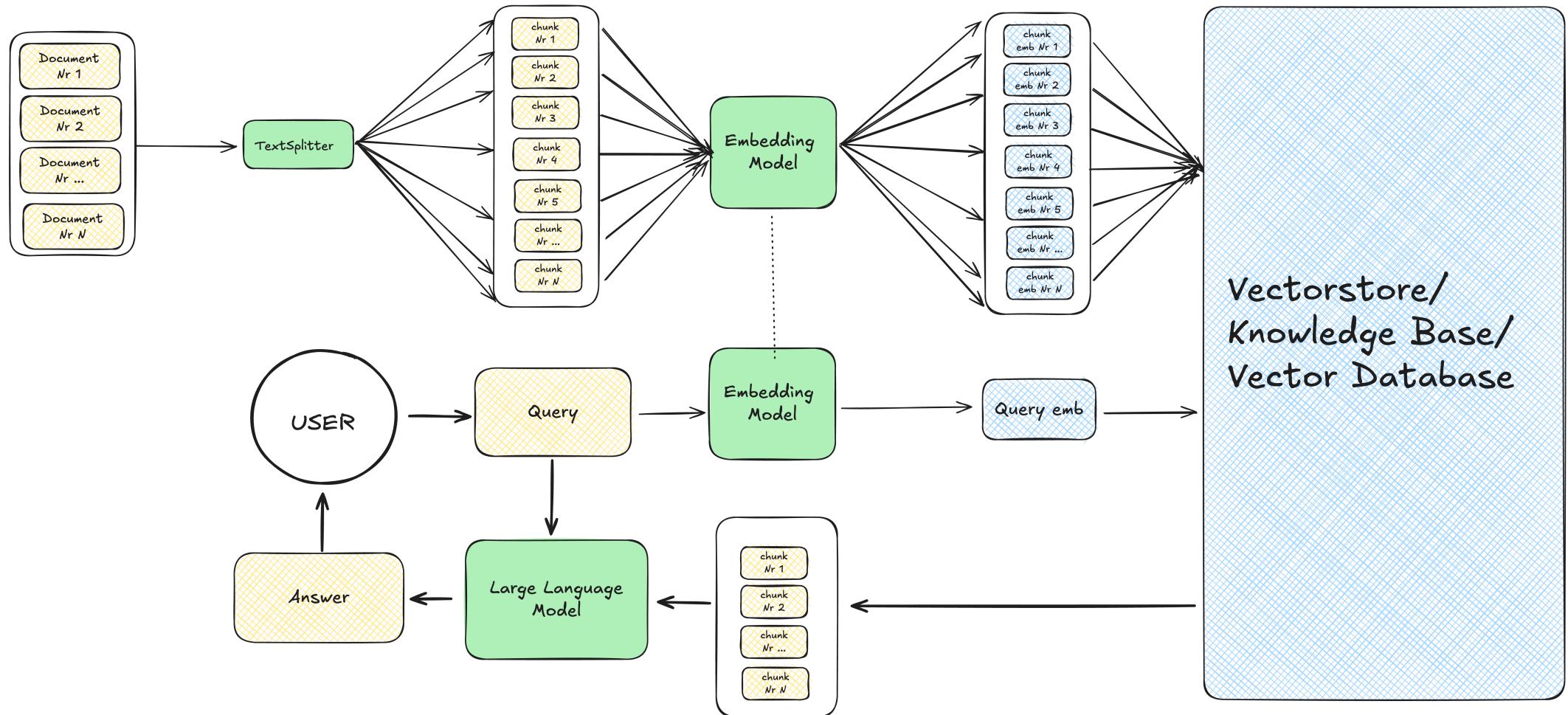
WIE HILFT RAG?

RAG steht für Retrieval-augmented generation

Retrieval: Relevante Dokumentpassagen aus einer vordefinierten Wissensbasis abrufen

Augmentation: Gefundene Passagen als Kontext zum LLM hinzufügen

Generation: LLM generiert die Antwort auf Basis dieses Kontexts





Wir nutzen eine eigene, vordefinierte Wissensbasis mit
domänenspezifische bzw. aktuellen Dokumenten



Vorteil: Die generierten Antworten basieren auf diesen
Quellen und sind dadurch prüfbar

RAG IM JOURNALISTISCHEN ALLTAG



Archivbasierte Chatbots: Chatbot beantwortet Fragen mit referenzierten Archivmaterial (z. B. Zeit, FT, WP)

Suche & Zusammenfassungen: Textpassagen schnell und semantisch passend durchsuchen und zusammenfassen lassen

Fact-Checking: Dokumentensammlungen zur Überprüfung von Aussagen nutzen

Vergleichsanalysen: Mehrere Versionen eines Dokuments miteinander vergleichen



KOMMERZIELLE RAG-ANGEBOTE

Chatbots mit Retrieval (z. B. NotebookLM)

Cloud & Managed Services

Kosten: Laufende Gebühren, Größenbeschränkungen, Abfrage-Limits

Datenhoheit: Daten liegen bei den Unternehmen

Abhängigkeit: Langfristige Abhängigkeit von den US Unternehmen

Prüfbarkeit: Wie verlässlich sind die Antworten?

VORTEILE EINES EIGENEN RAG-SYSTEMS



Datenhoheit: Kontrolle über die Weitergabe von Daten
(Dokumente, Embeddings und Abfragen)

Modular & austauschbar: Einzelne Komponenten
(Embedding Modell, Vectorstore, Top k, LLM...) lassen
sich flexibel anpassen und ersetzen

Messbar: Eigene Metriken definieren und die Leistung
kontinuierlich überwachen.

AUFBAU EINES EIGENEN RAG-SYSTEMS



Use Case definieren

Wissensquelle indexieren

Testset erstellen

Naives RAG-System erstellen

Metriken festlegen

System evaluieren und optimieren

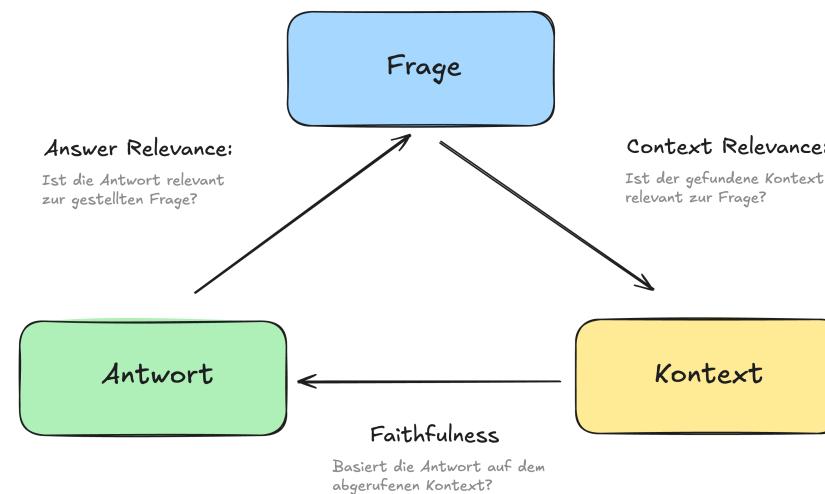
UNSER RAG BEISPIEL

Use-case: Interessent*innen informieren sich über ein Thema mithilfe des SMC Archivs



Wissensdatenbank: Mehrere Artikel aus dem SMC Archiv

Metriken: Generische RAG Triad Metriken



Tipp: Eigene Metriken definieren!



TESTSET ERSTELLEN

Wir generieren synthetische QA Paare mit DeepEval



Wir definieren die Styling-Configs rein natursprachlich

```
from deepeval.synthesizer.config import StylingConfig

styling_config = StylingConfig(
    input_format=(
        "Natural, conversational questions in German that use everyday language "
        "rather than technical terminology. Questions should use paraphrases and "
        "pronouns, and occasionally contain mild irrelevant details. "
        "Each question addresses exactly one scientific aspect."
    ),
    expected_output_format=(
        "Concise answers in German, clearly structured citing evidence."
    ),
    task=(
        "Answering focused scientific, non-trivial queries that address specific "
        "aspects of medical, climate, environment, or technology topics."
    ),
    scenario=(
        "Science journalists, scientists, or curious non-experts seeking precise "
        "information on a specific scientific topic they've encountered."
    )
)
```

Wir nehmen ein günstiges und schnelles LLM und verknüpfen es mit den **Styling-Configs**

```
from deepeval.synthesizer import Synthesizer

# günstiges und schnelles LLM
synthesizer = Synthesizer(
    model="gpt-50-mini", # generatives Sprachmodell
    styling_config=styling_config # Styling Config
)
```



Wir definieren die Dokumentensammlung und teilen die Texte in Chunks

```
import glob
from deepeval.synthesizer.config import ContextConstructionConfig

# Ordner mit Textdokumenten
data_dir = "data/example_data"
story_files = glob.glob(f"{data_dir}/*.txt")

# Chunking-Konfiguration
context_construction_config = ContextConstructionConfig(
    chunk_size=512,
    chunk_overlap=20
)
```



Jetzt werden automatisch **Frage/Antwort-Paare** generiert

```
# Synthetische Frage/Antwort-Paare erzeugen
synthesizer.generate_goldens_from_docs(
    document_paths=story_files, # unsere Wissensdatenbank
    include_expected_output=True, # für die Gold Antwort
    context_construction_config=context_construction_config # Chunking
)
```



Testset zusammengefasst:

```
[Confident AI Synthesizer Log] SUCCESS: Context Construction: Utilizing 40 out of 92 chunks.  
---  
#1  
Q: Wie wirken sich die steigenden Strompreise auf das Verbraucherverhalten aus, wenn die Erzeu  
A: Die steigenden Strompreise beeinflussen das Verbraucherverhalten nur geringfügig, da die ho  
---  
#2  
Q: Wird die Dunkelflaute in den nächsten Tagen die Windstromerzeugung stark beeinflussen?  
A: Ja, die Dunkelflaute in den kommenden Tagen könnte die Windstromerzeugung erheblich...  
---  
#3  
Q: Wie beeinflussen verschiedene Arten von Stromspeichern die Preise auf dem Markt? Gibt es da  
A: Stromspeicher beeinflussen die Marktpreise erheblich. Sie erhöhen die Preise in Niedrigprei  
---  
...
```



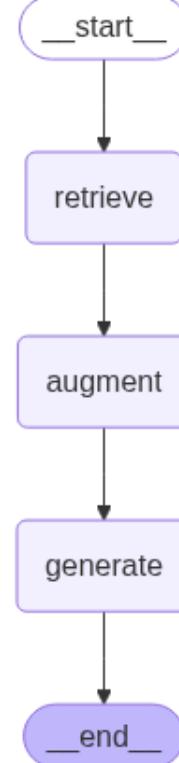
NAIVE RAG PIPELINE

Wir nutzen das Testset um eine naive RAG-Pipeline zu evaluieren



```
import src.graph.naive_rag as naive_rag_graph  
  
naive_rag_graph.graph
```

```
# run a test question  
query =  
"Welche neuen Medikamente zur Behandlung von Psychosen gibt  
  
result = naive_rag_graph.graph.invoke(  
    {"question": query},  
    config=run_cfg  
)
```



[query]

Welche neuen Medikamente zur Behandlung von Psychosen gibt es und wie wirken diese?

[context]

- 1) "Zudem ist es prinzipiell gut, wenn Behandelnde aus einem breite
- 2) "Obwohl die Schizophrenie eine der sozioökonomisch teuersten Erk
- 3) "folgt. Ich sehe aufgrund des beschriebenen Wirkmechanismus insb

...

[answer_no_1]

Basierend auf den vorliegenden Informationen gibt es zwei vielversprechende neue Medikamente zur Behandlung von Psychosen:...



Wir iterieren durch jede **Frage** und speichern
den retrieved **Kontext** und die generierten **Antworten**

```
import pandas as pd
import time
from tqdm import tqdm

results = []

# iterate through each questions and generate answers
for idx in tqdm(range(len(synth_data)), desc=f"Generating Answer for Question..."):
    question = synth_data.iloc[idx]['input']

    try:
        # Run RAG pipeline
        start_time = time.time()
        result = naive_rag_graph.graph.invoke(
            {"question": question},
            config=run_cfg
        )
        process_time = time.time() - start_time

        # Create base result dictionary
        result_dict = {
            'query_id': idx,
            'question': question,
```



... und evaluieren den Output mit den 3 RAG Metriken via DeepEval

```
import pandas as pd
# evaluation
from src.utils.evals import rate_dataset
rated_results = asyncio.run(rate_dataset(results, model="gpt-4o-mini", max_concurrency=5))

# save evaluated results
output_path = Path('results/simple_rag_res_rated.json')
with open(output_path, 'w', encoding='utf-8') as f:
    json.dump(results, f, ensure_ascii=False, indent=2)
```

[query]

Wie wirken sich Sprachmodelle auf die Forschung im Bereich der Genome von Bakterien und Pflanzen aus? Gibt es Unterschiede in den Ansätzen

[context]

- 1) "• großes Sprachmodell wurde mit Erbgut von Bakterien und Viren..."
- 2) "Auf die Frage, ob „Sprach“-Modelle wie Evo für die Computational Biology ge
- 3) "Sprachmodelle in der Computational Biology:..."

[answer_no_1]

"Sprachmodelle wie Evo eröffnen der Genomforschung bei Bakterien und Pflanzen v

[contextual_relevancy_no_1]

0.8387096774193549

[faithfulness_no_1]



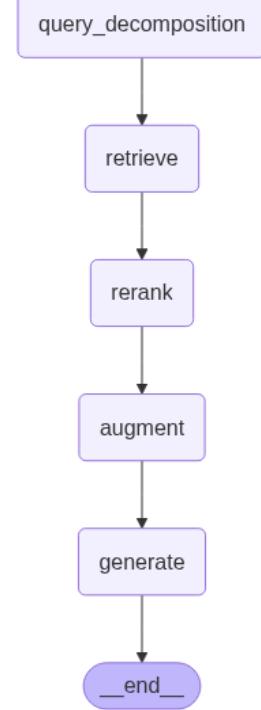
ADVANCED RAG PIPELINE

Wir erweitern die naive Pipeline mit zusätzlichen Komponenten und evaluieren den Effekt



```
import src.graph.advanced_rag as advanced_rag_graph  
  
advanced_rag_graph.graph
```

```
# run a test question  
query =  
"Welche neuen Medikamente zur Behandlung von Psychosen gi  
  
result = advanced_rag_graph.graph.invoke(  
    {"question": query},  
    config=run_cfg  
)
```



[query]

Welche neuen Medikamente zur Behandlung von Psychosen gibt es und wie wirken diese?

[query decomposition]

True

[sub queries]

Sub Query no 0:

Welche neuen Medikamente zur Behandlung von Psychosen sind in d

Sub Query no 1:

Wie wirken die neuen Medikamente zur Behandlung von Psychosen a

Sub Query no 2:

Welche Nebenwirkungen sind mit den neuen Medikamenten zur Behan

Erneut generieren wir **Antworten** für jede **Frage**, speichern diese zusammen mit dem retrieved **Kontext**

```
import pandas as pd
import time
from tqdm import tqdm

results = []

# iterate through each questions and generate answers
for idx in tqdm(range(len(synth_data)), desc=f"Generating Answer for Question..."):
    question = synth_data.iloc[idx]['input']

    try:
        # Run RAG pipeline
        start_time = time.time()
        result = advanced_rag_graph.graph.invoke(
            {"question": question},
            config=run_cfg
        )
        process_time = time.time() - start_time

        # Create base result dictionary
        result_dict = {
            'query_id': idx,
            'question': question,
```

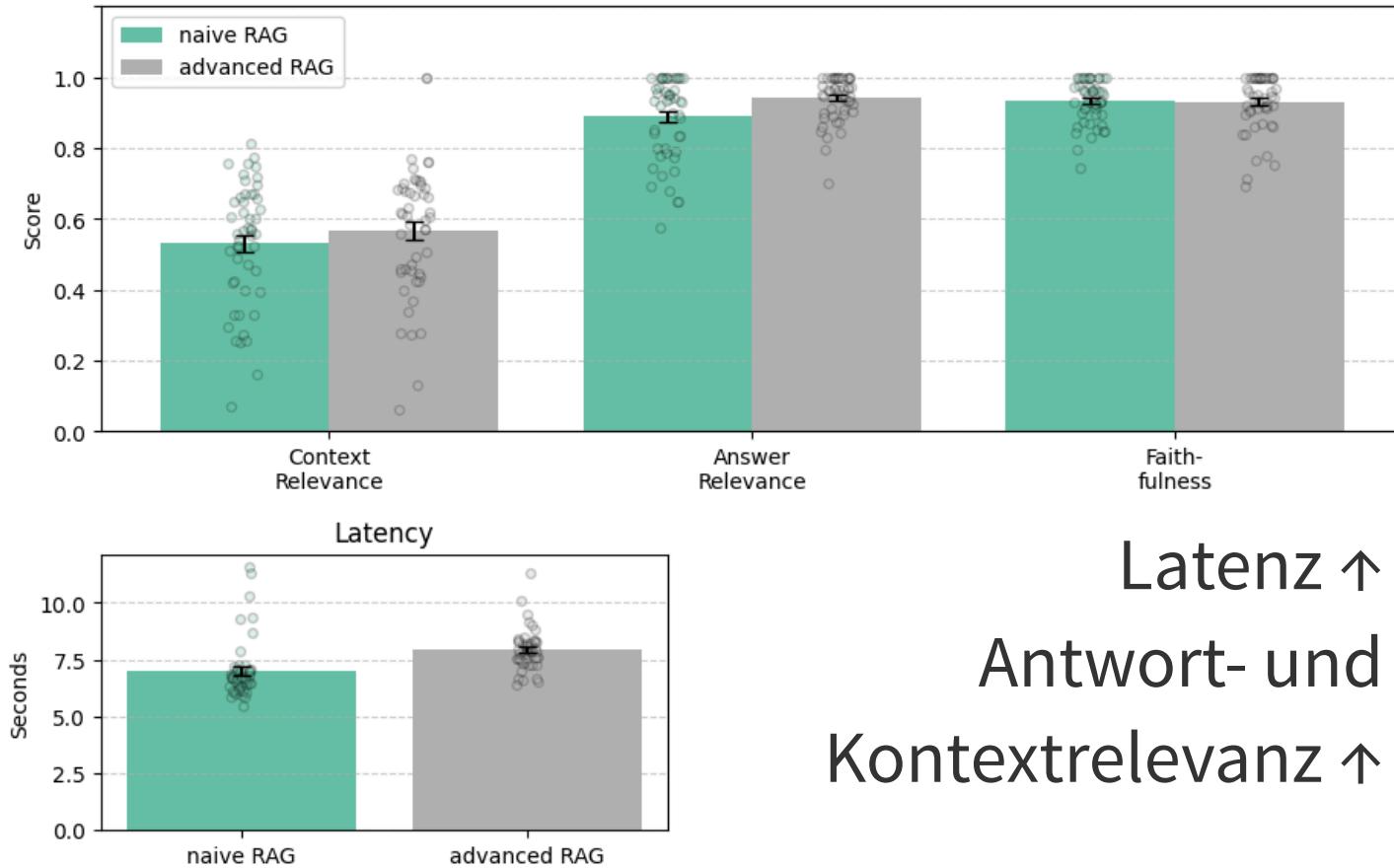


... und evaluieren dann das Ergebnis

```
# evaluation
from src.utils.evals import rate_dataset
rated_results = asyncio.run(rate_dataset(results, model="gpt-4o-mini", max_concurrency=5))

# save evaluated results
output_path = Path('results/advanced_rag_res Rated.json')
with open(output_path, 'w', encoding='utf-8') as f:
    json.dump(results, f, ensure_ascii=False, indent=2)
```

VERGLEICHSSANALYSE



Latenz ↑
Antwort- und
Kontextrelevanz ↑



Was passiert "under the hood" wenn ich eine RAG Metrik benutze?

```
from deepeval.metrics import AnswerRelevancyMetric
from deepeval.test_case import LLMTTestCase

metric = AnswerRelevancyMetric(model="gpt-5-mini")
test_case = LLMTTestCase(
    input="welche neuen medikamente zur behandlung von psychosen gibt es und wie wirken diese?",
    actual_output="Basierend auf den bereitgestellten Informationen gibt es zwei vielversprechende
    neue Medikamente zur Behandlung von Psychosen:..."
)

metric.measure(test_case)
print(metric.score)
```

Für Use Cases wie Gesetzestexte zusammenfassen
reichen die Standardmetriken nicht aus...



→ Wir definieren eine eigene Metrik!

```
from deepeval.metrics import GEval
from deepeval.test_case import LLMTestCase, LLMTTestCaseParams

summarization_correctness_metric = GEval(
    name="Summarization_Correctness",
    evaluation_steps=[
        "Prüfe, ob die wesentlichen rechtlichen Inhalte korrekt wiedergegeben werden.",
        "Überprüfe, ob keine zusätzlichen Informationen erfunden wurden.",
        "Stelle sicher, dass die Zusammenfassung den Sinn und die rechtliche Bedeutung nicht verfälscht.",
        "Achte darauf, dass die Sprache klar und für juristische Laien nachvollziehbar bleibt."
    ],
    evaluation_params=[LLMTTestCaseParams.ACTUAL_OUTPUT, LLMTTestCaseParams.INPUT],
)

llm_test_case = LLMTTestCase(
    input=document.FULL_TEXT,
    actual_output=document.SUMMARY,
)

summarization_correctness_metric.measure(llm_test_case)
print(summarization_correctness_metric.score)
```



Beispiel: RAG QA-Bot für Schüler:innen mit verständlichen Antworten

```
from deepeval.metrics import GEval
from deepeval.test_case import LLMTestCase, LLMTTestCaseParams

student_comprehensibility = GEval(
    name="Student_Comprehensibility",
    evaluation_steps=[
        "Bewerte, ob einfache, altersgerechte Sprache verwendet wird (kurze Sätze, aktive Form).",
        "Kennzeichne Fachbegriffe/Jargon und prüfe, ob sie vermieden oder kurz erklärt werden.",
        "Achte auf klare Struktur (1–3 Kernpunkte, ggf. kurzes Beispiel).",
        "Überprüfe, ob keine überflüssigen Details die Verständlichkeit mindern."
    ],
    evaluation_params=[LLMTTestCaseParams.ACTUAL_OUTPUT, LLMTTestCaseParams.INPUT],
    strict_mode=True, # strict mode -> binärer output (0 oder 1)
)

llm_test_case = LLMTTestCase(
    input=INPUT_QUESTION,
    actual_output=OUTPUT_ANSWER,
)

student_comprehensibility.measure(llm_test_case)
print(student_comprehensibility.score)
```



Zu kompliziert formuliert

```
from deepeval.test_case import LLMTestCase
INPUT_QUESTION = "Was genau bedeutet Klimawandel?"

OUTPUT_ANSWER = (
    "Der Klimawandel bezeichnet signifikante statistische Verschiebungen in der "
    "Energie- und Strahlungsbilanz des Klimasystems, induziert durch anthropogene "
    "Forcings und nichtlineare Rückkopplungsmechanismen."
)

llm_test_case = LLMTestCase(
    input=INPUT_QUESTION,
    actual_output=OUTPUT_ANSWER,
)

student_comprehensibility.measure(llm_test_case)
print(student_comprehensibility.score) # -> 0
```

Output:
FAIL



Einfach & altersgerecht

```
from deepeval.test_case import LLMTTestCase
INPUT_QUESTION = "Was genau bedeutet Klimawandel?"

OUTPUT_ANSWER = (
    "Klimawandel heißt, dass es auf der Erde nach und nach wärmer wird, "
    "weil wir Menschen viele Abgase in die Luft pusten. Dadurch ändert sich das Wetter."
)

llm_test_case = LLMTTestCase(
    input=INPUT_QUESTION,
    actual_output=OUTPUT_ANSWER,
)

student_comprehensibility.measure(llm_test_case)
print(student_comprehensibility.score) # -> 1
```

Output
PASS



Beispiel: RAG QA-Bot, der kurze und prägnante Antworten liefern soll

```
def length_metric(test_case: LLMTestCase, max_len=500):
    return 1 if len(test_case.actual_output) <= max_len else 0

llm_test_case = LLMTestCase(
    input=INPUT_QUESTION,
    actual_output=OUTPUT_ANSWER,
)

print(length_metric(llm_test_case))
```

Github Repo:



Linkedin:



Gesellschafter:

**Klaus Tschira
Stiftung**

