

---

# Empirische Analyse von RAG Evaluation Tools für betriebliche Abläufe

Bachelorarbeit zur Erlangung des akademischen Grades  
*Bachelor of Science*  
im Studiengang Allgemeine Informatik  
an der Fakultät für Informatik und Ingenieurwissenschaften  
der Technischen Hochschule Köln

Vorgelegt von: Leon Alexander Bartz  
Matrikel-Nr.: 1114236017  
Adresse: Richard-Wagner-Straße 47  
50679 Köln  
leon\_alexander.bartz@smail.th-koeln.de

Eingereicht bei: Prof. Dr. Boris Naujoks  
Zweitgutachterin: Prof. Dr. Dietlind Zühlke

Köln, 30.06.2025

## Kurzfassung/Abstract

Heutzutage gewinnen Large Language Models (LLMs) wie ChatGPT und Gemini immer weiter an Beliebtheit. In Kombination mit relevanten, häufig nicht öffentlichen Dokumenten können Sie noch hilfreicher sein.

Wenn man mithilfe von LLMs jetzt relevante Daten in einer Datenbank sucht, um damit kontextabhängige Fragen zu beantworten hat man ein Retrieval-Augmented Generation (RAG). Wie gut RAGs jedoch funktioniert, hängt von vielen unterschiedlichen Faktoren ab. RAGs manuell zu bewerten ist ein zeit- und kostenintensiver Prozess.

Retrieval Augmented Generation Assessment (RAGAS) hat ein System entwickelt, um RAGs automatisch zu bewerten. Wie gut diese automatisierte Bewertung von RAGs mithilfe von RAGAS funktioniert ist der Hauptfokus dieser Arbeit. Dabei werden besonders drei Bereiche untersucht, die generierten Fragebögen, die Bewertung und die Zuverlässigkeit bei mehreren Wiederholungen. Wir haben beobachtet, dass das genutzte LLM eine große Rolle spielt. Bessere LLMs haben sowohl weniger Fehler während der Generierung der Fragebögen gemacht als auch allgemein bessere Fragen generiert. Es hat sich außerdem gezeigt, dass Fehler aus den Fragebögen sich durch die Bewertung ziehen und dadurch die Bewertung negativ beeinflussen. Die Metriken und Bewertungen waren konstant über mehrere Bewertungen hinweg und es waren nur minimale Schwankungen festzustellen.

Zusammenfassen lässt sich sagen, dass Ragas nicht als alleiniger Faktor eingesetzt werden kann und mindestens bei der Testset-Generierung eine menschliche Überprüfung stattfinden sollte. Für die meisten Fälle ist zudem ein LLM mit vielen Parametern zu empfehlen, da dieses bessere Ergebnisse liefert.

Zukünftig lässt sich Ragas mit weiterentwickelten LLMs und einer verbesserten Fragebogengenerierung vielleicht komplett automatisieren !TODO!

Schlagwörter/Schlüsselwörter: gegebenenfalls Angabe von 3 bis 10 Schlagwörtern. LLM, KI, RAG, Ragas, Automatisierung

# Inhaltsverzeichnis

<b>Tabellenverzeichnis</b>	<b>V</b>
<b>Abbildungsverzeichnis</b>	<b>VI</b>
<b>1 Einleitung</b>	<b>1</b>
1.1 Wie funktioniert ein LLM . . . . .	2
1.2 Funktionsweise von RAGs . . . . .	3
1.2.1 Vorteile von RAGs . . . . .	4
1.2.2 Faktoren für den Einsatz von RAGs . . . . .	5
1.3 Objektive Beurteilung von RAGs . . . . .	6
1.4 Softwaretechnische Fragestellungen . . . . .	7
1.5 Rechtliche Fragestellungen . . . . .	8
<b>2 Methoden und Materialien</b>	<b>9</b>
2.1 Software/Programme . . . . .	9
2.1.1 Ollama . . . . .	9
2.1.2 Embeddings . . . . .	9
2.1.3 Vektordatenbank ChromaDB . . . . .	9
2.1.4 RAGAS . . . . .	10
2.1.5 Langchain . . . . .	10
2.2 Daten . . . . .	10
2.2.1 Originale Dokumente . . . . .	10
2.2.2 Synthetische Daten . . . . .	11
2.2.3 Anwendungsszenarien . . . . .	12
2.3 Fragebögen . . . . .	12
2.3.1 Fragetypen . . . . .	12
2.3.2 Wissensgraph . . . . .	13
2.4 RAG-Bewertungsprozess . . . . .	15
2.5 Metriken . . . . .	17
2.5.1 Verwendete Metriken . . . . .	17
2.5.2 Weitere RAGAS Metriken . . . . .	19
2.5.3 Spezielle Metriken . . . . .	22

<b>3</b>	<b>Ähnliche Arbeiten</b>	<b>23</b>
3.1	RAG Evaluation: Assessing the Usefulness of Ragas . . . . .	23
3.2	Benchmarking Large Language Models in Retrieval-Augmented Generation	23
<b>4</b>	<b>Versuche</b>	<b>25</b>
4.1	Versuchsplan . . . . .	25
4.1.1	Forschungsfragen . . . . .	25
4.1.2	Variablen in den Versuchen . . . . .	26
4.1.3	Kosten- und Zeitanalyse . . . . .	27
4.1.4	Versuchsprotokoll . . . . .	27
4.1.5	Bewertungskriterien für die Geschäftstauglichkeit . . . . .	28
4.2	Konkretisierung der Versuche . . . . .	29
4.2.1	Dokumentenverarbeitung . . . . .	29
4.2.2	Testset-Generierung . . . . .	29
4.2.3	Bewertung . . . . .	29
<b>5</b>	<b>Ergebnisse und Diskussionen aus den Versuchen</b>	<b>31</b>
5.1	Generierte Fragebögen . . . . .	31
5.1.1	Deepseek/Nomic . . . . .	31
5.1.2	OpenAI . . . . .	32
5.1.3	Manuelle Auswertung der Fragebögen . . . . .	32
5.2	Auswertung der Reports . . . . .	35
5.2.1	Manuelle Auswertung DeepSeek . . . . .	36
5.2.2	Manuelle Auswertung OpenAI . . . . .	36
5.2.3	Probleme mit Programm-Code . . . . .	37
5.3	Unterschiede über mehrere Durchläufe . . . . .	37
5.4	Zuverlässigkeit von Metriken . . . . .	40
5.4.1	Context Precision & Recall . . . . .	41
5.4.2	Answer Relevancy . . . . .	41
5.4.3	Faithfulness . . . . .	42
5.5	Ausführungszeiten . . . . .	42
5.6	Kostenberechnung . . . . .	44
<b>6</b>	<b>Zusammenfassende Betrachtungen</b>	<b>45</b>
6.1	Forschungsfragen . . . . .	45
6.1.1	Relevanz . . . . .	45
6.1.2	Zuverlässigkeit . . . . .	46
6.1.3	Varianz . . . . .	47
6.1.4	Effizienz . . . . .	47
6.2	Fazit . . . . .	47
6.3	Reflexion der Arbeit . . . . .	49

<b>Literatur</b>	<b>51</b>
<b>Anhang</b>	<b>55</b>

## Tabellenverzeichnis

1.1	Vergleich der RAG-Evaluierungswerkzeuge . . . . .	1
4.1	Verteilung der Dateitypen und generierten Inhalte . . . . .	26
4.2	Kombinationen aus Dokumentenanzahl und Embedding-Modell für die Ver- suche . . . . .	29
4.3	Kombinationen aus Dokumentenanzahl und Testset-Größe . . . . .	29
4.4	Übersicht der Versuche . . . . .	30
5.1	DeepSeek Übersicht Fehlerquote für Fragen . . . . .	32
5.2	OpenAI Übersicht Fehlerquote für Fragen . . . . .	32
5.3	DeepSeek Fehlerhafte Fragen . . . . .	34
5.4	OpenAI Fehlerhafte Fragen . . . . .	35
5.5	DeepSeek Verteilung der Bewertungen . . . . .	36
5.6	OpenAI Verteilung der Bewertungen . . . . .	37
5.7	DeepSeek Abweichungen mehrere Durchläufe . . . . .	38
5.8	OpenAI Abweichungen mehrere Durchläufe . . . . .	39
5.9	OpenAI vs. DeepSeek Evaluationszeiten . . . . .	43

## Abbildungsverzeichnis

1.1	Embedding Beispiel . . . . .	3
1.2	Struktur eines RAGs . . . . .	4
2.1	Fragetypen in RAGAS . . . . .	12
2.2	Beziehungen zwischen Dokumenten-Chunks . . . . .	14
2.3	Flussdiagramm des RAG-Bewertungsprozesses . . . . .	16
5.1	DeepSeek Metriken für 300 Fragen . . . . .	38
5.2	ChatGPT Metriken für 300 Fragen . . . . .	39
5.3	DeepSeek Score Verteilung 100 Fragen . . . . .	40
5.4	ChatGPT Score Verteilung 100 Fragen . . . . .	41
5.5	Abweichungen des Answer Relevancy Scores . . . . .	42
5.6	Abweichungen des Faithfulness Scores . . . . .	43
6.1	Pisikopyramide KI-Systme . . . . .	48

# 1 Einleitung

Im Jahr 2022 erregte OpenAI Aufsehen mit ihrem browserbasierten ChatGPT (Generative Pre-trained Transformer) [1]. In nur fünf Tagen erreichte ChatGPT eine Million Nutzer\*innen [2] und ist aus dem Alltag vieler Menschen nicht mehr wegzudenken. Die GPT-KI (Künstliche Intelligenz) von OpenAI gehört zur Familie der LLMs oder auch Multimodal Large Language Models (MLLMs). MLLMs können neben Text weitere Medien wie zum Beispiel Bilder, Audio und Video verarbeiten. LLMs von anderen Anbietern wie Googles Gemini [3] und DeepSeek [4] haben mit vergleichbare Qualität und ähnliche Fähigkeiten wie OpenAIs GPT-Modelle.

Die GPT-Modelle von OpenAI und anderen Anbietern wie Googles Gemini [5] sind meistens nur über eine API (Programmierschnittstelle) gegen Entgelt verfügbar. Open-Source-Modelle wie DeepSeek [4] oder Metas LLAMA [6] erfreuen sich immer größerer Beliebtheit, da sie gratis auf der eigenen Hardware ausgeführt werden können.

Im Oktober 2023 kam der Verfasser dieser Arbeit erstmalig mit RAG in Kontakt; damals war die Idee, mit Hilfe eines LLMs Fragen über mehrere firmeninterne Informationsquellen zu beantworten. Bei einem Hackathon gelang es dem Team des Verfassers, einen Prototypen (im Folgenden „KID-System“ genannt) zu entwickeln, der mit einem gewissen Erfolg Fragen zu firmeninternen Themen beantworten konnte.

Einer der Schritte während der Entwicklung war das ständige Testen der neuesten Änderungen. Dadurch konnten die Funktionsfähigkeit überwacht und eventuell schlechte Ergebnisse dokumentiert werden. Diese zeitintensive Aufgabe kostete wertvolle Zeit, die das Team lieber in die Entwicklung investiert hätte. Gerne hätten wir unterschiedliche Prompts innerhalb unseres KID-Systems getestet oder eine automatische Überprüfung unserer neuesten Änderungen genutzt.

Tabelle 1.1: Vergleich der Open Source RAG-Evaluierungswerkzeuge

Werkzeug	GitHub Sterne	Schlüsselwörter
Giskard	4.7k	KI/LLM/RAG-Evaluierung, Performance, Bias, Sicherheit, Sicherheitslücken-Identifizierung
Opik	10.4k	LLM/RAG/Agenten-Workflows, Debugging, Evaluierung, Monitoring, Tracing
Ragas	9.7k	RAG-spezifische Evaluierung, <b>Automatische Testdaten</b> , Objektive Metriken



Mittlerweile gibt es viele Arten, LLMs zu bewerten und ein reger Wettbewerb ist um die vielen Bewertungen entstanden. RAGAS [7] wurden entwickelt, um das oben geschilderte Probleme des ständigen Testesns zu lösen. Es hat zudem das Alleinstellungsmerkmal, dass man weder eigene Fragen noch die generierten Fragen selbst beantworten muss. Sowohl die Generierung eines Fragenkatalogs (Testsets) als auch die Beantwortung der Fragen, um eine Musterlösung zu erstellen, nimmt RAGAS mithilfe von LLMs vor. Mithilfe dieses Testsets und von RAGAS eigens entwickelter Metriken, welche die wichtigsten Funktionen eines RAGs abdecken, kann eine Bewertung des Systems vorgenommen werden.

Damit benutzt RAGAS selber LLMs, um das durch LLMs entstandene System zu testen. Dies spart menschliche Ressourcen, welche zeit- und kostenintensiv sind. Wie gut LLMs für diese Aufgabe geeignet sind, ist jedoch, auch aufgrund ihrer Halluzinationen, fraglich. Ragas wurde für diese Arbeit ausgewählt, da es das einzige Tool ist, das alle Komponenten hat, um ein RAG spezifisches System zu testen. Die anderen Tools, wie Giskard und Opik bieten auch die Möglichkeit, LLMs zu bewerten, der Schritt der Fragen Generierung und der Musterlösung fehlt jedoch. Um zu verstehen, wie RAGAS arbeitet, werden zunächst die grundlegenden Funktionsweisen von Sprachmodellen erläutert.

## 1.1 Wie funktioniert ein LLM

Die meisten LLMs heutzutage sind sogenannte Transformer, daher kommt auch das T in GPT. Transformer sind Neuronale-Netzwerk-Modelle, die auf dem Aufmerksamkeits-Mechanismus basieren. Das Konzept der Aufmerksamkeit wurde im Paper „Attention Is All You Need“ [8] vorgestellt; es ist einer der fundamentalen Bausteine für die heutigen LLMs. LLMs setzen sich aus vier Schritten zusammen:

1. Tokenisierung (Tokenizer)
2. Einbettung (Embedding)
3. Berechnung der Wahrscheinlichkeit des nächsten Tokens (Vorhersage)
4. Strategien zur Auswahl der Ausgabe (manchmal auch Dekodierung genannt).

„Token sind Wörter, Zeichensätze oder Kombinationen aus Wörtern und Interpunktionszeichen, die von großen Sprachmodellen generiert werden, wenn sie Text zerlegen.“ [9] Die Aufgabe des Verwandeln der Eingabe in Tokens übernimmt der Tokenizer.

Damit ein Neuronales Netzwerk mit Tokens rechnen kann, müssen diese in Vektoren umgewandelt werden. Ein Umwandeln in einfache Zahlen reicht hier nicht, da die Fähigkeit benötigt wird, semantische Ähnlichkeiten zu modellieren. Das Umwandeln ist mithilfe von Embeddings möglich. Häufig sind Embeddings neuronale Netze, die mit großen Mengen an

Texten trainiert wurden, um Wörtern eine Position im höherdimensionalen Raum zu geben. Diese Position soll die semantische Bedeutung der Wörter beibehalten.

In Abbildung 1.1 ist ein Beispiel für die Umwandlung eines Satzes in Vektoren zu sehen. Die Abbildung vereinfacht den Tokenisierungs- und Embedding-Prozess, da Tokens in der Realität nicht immer so klar voneinander getrennt sind und die Vektoren meistens mehrere tausend Dimensionen haben.

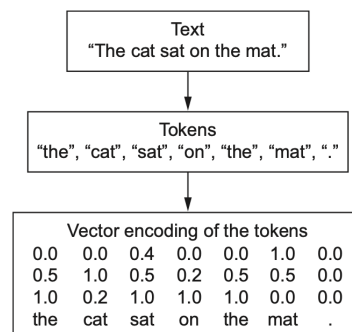


Abbildung 1.1: Beispiel für ein Embedding von einem Satz zu Vektoren.

Die Vektoren können jetzt in ein Neuronales Netzwerk, das meistens die Transformer-Architektur verwendet, gefüttert werden. Das Ergebnis besteht aus den Wahrscheinlichkeiten für alle der möglichen nächsten Token.

Im letzten Schritt muss ausgewählt werden, welcher Token genutzt werden soll. Würde hier immer nur der wahrscheinlichste Token gewählt, würde man nur Texte generieren, die das LLM während des Trainings bekommen hat. Um einen neuen Text zu generieren werden also zufällig weniger wahrscheinliche Tokens ausgewählt. Dieser Prozess führt zu dem, was als Halluzinationen bekannt ist. Es ist ein fester Bestandteil der LLM-Architektur [10].

Mit dem Wissen, wie LLMs funktionieren, kann nun betrachtet werden, wie RAGs funktionieren, wie LLMs mit RAGs kombiniert werden und welche Vorteile RAGs haben.

## 1.2 Funktionsweise von RAGs

Bei einem Rag kommt das Wissen für die Antwort nicht nur aus dem LLM selbst sondern auch aus angebundenen Quellen. Diese angebundenen Quellen können Dokumentensammlungen, Datenbanken, Wissensgraphen (Knowledge Graph) oder andere Suchquellen wie eine Internetsuche sein [11].

Die Abbildung 1.2 zeigt den Ablauf für eine Anfrage an ein RAG.

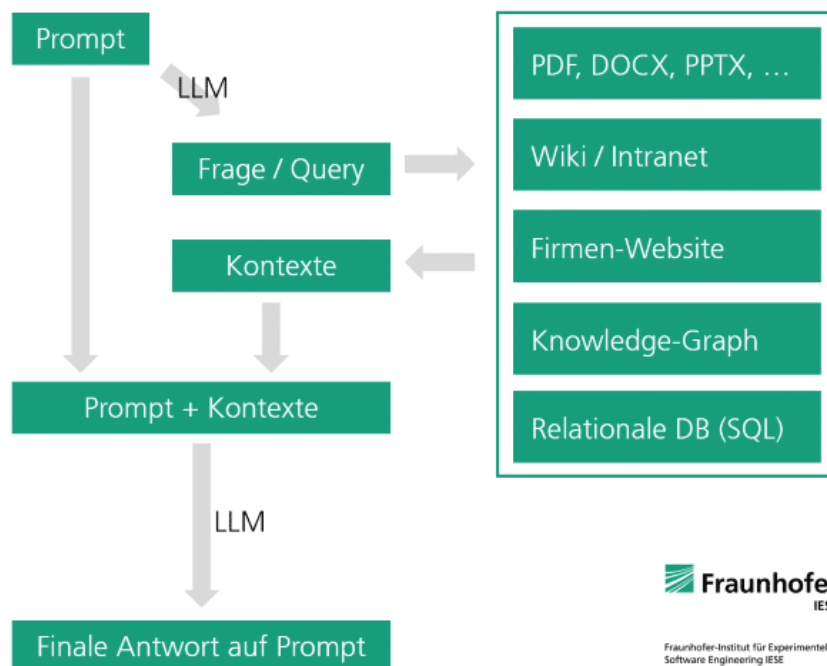


Abbildung 1.2: Struktur eines RAGs, Quelle: [11]

1. Im ersten Schritt wird ein Prompt an das LLM gesendet. Dieser Prompt besteht aus der Frage des Nutzers, aber auch weiteren Details, wie einer Persona, deren Perspektive das LLM einnehmen soll.
2. Das LLM kann dann mittels dieses Prompts eine Anfrage generieren, um für diesen Prompt relevante Dokumente zu suchen.
3. Diese Anfrage gibt dann im optimalen Fall die benötigten Dokumente für die Beantwortung des Prompts zurück. Der Schritt des Abfragens relevanter Dokumente wird Retrieval genannt. Die gefundenen Dokumente werden Kontext genannt.
4. Der Original Prompt zusammen mit dem abgefragten Kontext wird nun einem LLM zur Verfügung gestellt, um den kontextabhängigen Prompt zu beantworten.

### 1.2.1 Vorteile von RAGs

Bei der Wissensabfrage durch LLMs zeigen sich unter anderem folgende Schwachstellen:

**Aktualität:** LLMs kennen nur die verwendeten Trainingsdaten. Es fehlt also häufig Wissen der letzten Jahre.

**Unbalancierte Trainingsdaten:** Im Trainingsset für die LLMs selten vorkommendes Wissen können LLMs schlecht lernen. [12, 13]

**Vertraulichkeit:** Nicht öffentliche Dokumente sind nicht im Trainingsset und daher können LLMs keine Fragen zu nicht öffentlichen Daten beantworten.

Die Nutzung eines RAGs ist eine von drei häufig verwendeten Möglichkeiten, um ein LLM zu verbessern. Die anderen beiden Möglichkeiten sind Finetuning und die Verwendung eines LLMs mit einem großen Kontextfenster. Das Finetuning beschreibt das weitere Trainieren eines Modells mit weiteren Trainingsdaten. Die Größe des Kontextfensters beschreibt die Menge an Text, gemessen in Tokens, die das LLM gleichzeitig berücksichtigen kann. RAGs haben gegenüber dem Finetuning und der Nutzung eines LLMs mit großem Kontextfenster entscheidende Vorteile.

ChatGPT-4.1 erreicht inzwischen ein Kontext Window Größe von 1,047,576 Tokens, bei GPT-3.5-Turbo sind es 16,385 [14]. Gemini unterstützt ebenfalls bis zu 1 Million Tokens, wobei Google, der Besitzer von Gemini, angibt, dass sie erfolgreich 10 Millionen Token Context Windows getestet haben [15].

## 1.2.2 Faktoren für den Einsatz von RAGs

Es gibt einige Faktoren, welche die Entscheidung beeinflussen können, ob ein RAG besser für den betrieblichen Ablauf geeignet ist. Dazu gehören z. B. die Kompetenz der Betreiber des RAGs, die Art der Daten und die finanziellen Möglichkeiten des Unternehmens.

### 1.2.2.1 Kompetenz des Betreibers

Für das Finetuning von LLMs ist technisches Wissen notwendig, um die Themen Natural Language Processing (NLP), Deep Learning, Modellkonfiguration, Datenaufbereitung und Evaluierung anzuwenden. Der gesamte Prozess des Finetunings ist technisch anspruchsvoll, erfordert das Kuratieren der neuen Trainingsdaten und ist zudem durch die benötigte Hardware teuer. Zudem ist das betreiben der Hardware teuer da sie einen hohen Stromverbrauch hat.

Das Benutzen eines LLMs benötigt die geringste Informatische Kompetenz des Anwenders, da hier das LLM unverändert bleibt. Hier werden einfach die Daten inklusive der Frage an das LLM gesendet. Entscheidend ist allerdings das Erstellen zielführender Prompts, dies benötigt Erfahrung.

Während das LLM in einem RAG auch unverändert bleibt, wird es in ein System mit mehreren Komponenten eingebunden. Hier ist ein allgemeines Verständnis von LLMs und effektiven Methoden für den Retrieval Teil des RAGs notwendig. Zudem müssen hier manuell für

jedes Dateiformat (E-Mail, PDF etc.) Anbindungen erstellt werden. Sollte ein seltener oder proprietärer Datentyp verwendet werden, muss hier eventuell eigens eine Anbindung entwickelt werden.

#### **1.2.2.2 Aktualisierung der Datenbasis**

Sollten die Daten dynamisch sein, wie E-Mails, ist das RAG vermutlich die vorzuziehende Lösung. Dies liegt an den Eigenschaften der schnellen und kontinuierlichen Aktualisierung der Daten. Wie oben erläutert, kann es jedoch sein, dass es schlechte oder keine Unterstützung von selten verwendeten Dateiformaten gibt.

Der Prozess des Finetunings erstellt hingegen eine Momentaufnahme, die ein erneutes Training erfordert. Beim Finetuning ist es möglich, dass das Modell Muster erkennt und firmeneigene Begriffe verstehen kann. Dies ist ein deutlicher Vorteil gegenüber dem RAG und dem Nutzen eines LLMs mit großem Context Window.

#### **1.2.2.3 Budget**

Das Finetuning erfordert über einen langen Zeitraum teure Rechenzeit auf Hochleistungs-Graphical-Processing-Units (Hochleistungs-GPUs). Dabei handelt es sich um spezialisierte Grafikprozessoren, deren Architektur auf die massive parallele Verarbeitung von Daten optimiert ist, was sie für die rechenintensiven Operationen des Machine Learnings, insbesondere neuronale Netze, unerlässlich macht. Zudem ist die Qualität der Daten entscheidend; ohne das vorherige Filtern der Daten durch Menschen ist dieser Prozess aktuell nicht möglich. Sollten sich die Daten als unzureichend erweisen, ist die gesamte Rechenzeit verschwendet. Das RAG verursacht dagegen zusätzliche Kosten durch das Speichern der Daten in einer Vektordatenbank. Die wohl kostenintensivste Methode ist die Nutzung eines LLMs mit einem großen Kontextfenster.

### **1.3 Objektive Beurteilung von RAGs**

Je mehr Daten einem RAG zur Verfügung stehen, desto aufwendiger ist es, die Qualität des RAGs zu beurteilen. Eine Beurteilung durch Menschen müsste bei Anpassungen am RAG oder Änderungen an den Daten neu durchgeführt werden.

Tools wie RAGAS, die bereits eine automatisierte Bewertung implementieren, nutzen bei diesem Prozess unter anderem LLMs. Diese Tools generieren aus den ihnen gegebenen Daten Fragebögen, die zu einer Frage eine beispielhafte Antwort und die genutzten Stellen aus den vorher gegebenen Dokumenten beinhalten. Sollten nach diesem automatisierten Test die

gewünschten Ergebnisse nicht erreicht werden, kann beispielsweise die Veröffentlichung blockiert werden.

Sowohl menschliche Bewertungen als auch je nach Vorgaben die eventuell subjektive Bewertung durch LLMs sind nicht objektiv. Anhand mehrerer Techniken kann versucht werden, die Bewertung mithilfe von LLMs zu objektivieren. Die Metriken, welche RAGAS nutzt, versuchen z.B. die Anzahl der genannten Fakten aus den Antworten zu extrahieren, um so mit Zahlen arbeiten zu können.

## 1.4 Softwaretechnische Fragestellungen

In dem Artikel *RAG in der Praxis – Generierung synthetischer Testdatensätze* untersucht Panic [16] die Testset-Generierung mithilfe von RAGAS. Es treten bei 17 % der generierten Fragen Fehler beim Generieren der Testfragen auf. Dies hat vielfältige Gründe, die von nicht verwertbaren Antworten des LLMs bis zu Verbindungsproblemen oder dem Erreichen des Limits der maximalen Anfragen an APIs reichen.

Auch bei der Bewertung von Antworten können sich ungewollte und bisher noch ungeahnte Biases einschleichen. In dem Paper „Large Language Models are not Fair Evaluators“ [17] von Wang et al. wird gezeigt, dass, wenn ein LLM eine von zwei gegebenen Antworten aussuchen müsste, die erste bevorzugt wurde, selbst wenn die gleiche Frage mit einer anderen Reihenfolge gestellt wurde. RAGAS vergleicht keine Antworten miteinander, und daher ist dieser Bias kein direktes Problem für die verwendeten Metriken. Was jedoch einen Einfluss auf die Bewertung von Antworten haben kann, ist der Bias zu gewissen Nummern. Wie in [18] von Shaikh et al. beschrieben, bevorzugen LLMs bei der Bewertung lieber Zahlen, welche Vielfache von 5 und 10 sind.

Auch die allgemeine stochastische Natur von LLMs spielt eine Rolle, da bei der gleichen Anfrage unterschiedliche Antworten und somit auch Bewertungen zurückgegeben werden. Wie groß diese Abweichungen sind, wird in Abschnitt 5.4 kurz untersucht.

Wie in dem Paper „Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context“ vom Gemini Team [19] beschrieben, stellt Gemini 1.5 einen bedeutenden Fortschritt in der multimodalen Verarbeitung großer Kontextfenster dar. Das wirft auch die Frage auf, ob RAGs nicht irrelevant sein könnten und durch LLMs mit großen Kontextfenstern abgelöst werden. Es gibt einige Gründe, die dagegen sprechen: LLMs mit größeren Kontextfenstern werden immer langsamer und teurer; die genauen Kosten sind abzuwarten. Jedes Mal alle Daten in den Kontext zu laden, besonders wenn dies über das Internet geschieht, ist eine weitere Hürde. LLMs fällt es auch schwer, bei zu vielen Informationen noch die relevanten zu finden, was zu schlechteren Antworten führen kann. Diese Faktoren

lassen darauf schließen, dass RAGs, die nicht nur eine einfache Suche nutzen, noch länger relevant bleiben.

Inzwischen werden spezielle LLMs wie Pleias-RAG [20] entwickelt. Diese wurde mittels Finetuning verbessert, um Speziell Aufgaben wie das Suchen und das Zusammenfassen von Dokumenten selbst mit weniger Parametern gut zu können.

## 1.5 Rechtliche Fragestellungen

Am 01.08.2024 trat die

„*VERORDNUNG (EU) 2024/1689 DES EUROPÄISCHEN PARLAMENTS UND DES RATES vom 13. Juni 2024 zur Festlegung harmonisierter Vorschriften für künstliche Intelligenz und zur Änderung der Verordnungen (EG) Nr. 300/2008, (EU) Nr. 167/2013, (EU) Nr. 168/2013, (EU) 2018/858, (EU) 2018/1139 und (EU) 2019/2144 sowie der Richtlinien 2014/90/EU, (EU) 2016/797 und (EU) 2020/1828 (Verordnung über künstliche Intelligenz)*“ (KI-VO) [21]

in Kraft. Die Verordnung setzt Regelungen und Maßstäbe für die Verwendung von KI. RAGs sind gemäß Artikel 3 Nr. 1 in Verbindung mit Artikel 51 KI-VO KI-Systeme [22] und fallen damit in den Anwendungsbereich der KI-VO. Bei der Nutzung oder Bereitstellung von LLMs muss die KI-VO berücksichtigt werden. Die Nutzer\*innen der RAGs müssen sich der aus der KI-VO ergebenden Pflichten bewusst sein.

Ebenso gilt es, bei Anbindung firmeninterner Dokumente die

„*Verordnung (EU) 2016/679 des Europäischen Parlaments und des Rates vom 27. April 2016 zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener Daten, zum freien Datenverkehr und zur Aufhebung der Richtlinie 95/46/EG (Datenschutz-Grundverordnung)*“ (DS-GVO) zu beachten. Die DS-GVO regelt den Umgang mit personenbezogenen Daten, die in solchen firmeninternen Dokumenten enthalten sein könnten.

## **2 Methoden und Materialien**

### **2.1 Software/Programme**

Für die RAGs und die Bewertung der RAGs werden Tools benötigt. Im Nachfolgenden werden diese Tools genauer erklärt.

#### **2.1.1 Ollama**

Ollama [23] ist ein Open-Source LLM-Server, der auf einem eigenen Computer oder in der Cloud ausgeführt wird. Es können verschiedene Open-Source LLMs und Embedding-Modelle ausgeführt werden. In der vorliegenden Arbeit werden die Modelle ollama/nomic-embed-text und ollama/deepseek-r1:32b verwendet.

#### **2.1.2 Embeddings**

Vektoren sind die Beschreibung einer Position im mehrdimensionalen Raum. Es handelt sich hier meist um Positionen, welche mehrere Tausende Dimensionen haben. Embeddings ermöglichen die Darstellung von z. B. Wörtern im mehrdimensionalen Raum. Je näher zwei Wörter im Raum beieinander sind, desto ähnlicher sind sie. Welche Embeddings für welche Versuche genutzt werden, wird in Kapitel 4.2.3 „Bewertung“ beschrieben.

#### **2.1.3 Vektordatenbank ChromaDB**

ChromaDB [24] ist eine Open-Source-Vektordatenbank, die zur persistenten Speicherung und effizienten Abfrage von hochdimensionalen Embeddings eingesetzt wird.

Die Vektordatenbank ist also ein fester Bestandteil des RAGs und wird sowohl für die Open-Source-Modelle als auch für die Closed-Source-Modelle von z.B. OpenAI benutzt. Dies schafft eine einheitlichere Basis zum Vergleichen der LLMs.



### 2.1.4 RAGAS

RAGAS ist ein Open-Source-Tool und liefert neben dem Tool selbst hilfreiche Dokumentation für die Metriken und die Bewertung von RAGs. Es werden Funktionen wie die automatische Generierung von Interessengruppen, die Testset-Generierung und die Bewertung von RAGs anhand von Testsets bereitgestellt. Für diese Arbeit sind die Funktionen der Testset-Generierung und die damit ermöglichte Bewertung der RAGs relevant.

Was RAGAS von Giskard und Comet unterscheidet, ist, dass keine „reference answer“ benötigt wird. RAGAS ist mit 9.4 tausend Sternen bei Github beliebt, dazu trägt sicherlich auch die gute integration mit vielen anderen Werkzeugen bei.

### 2.1.5 Langchain

Langchain [25] ist ein Framework, das bei der Entwicklung von Anwendungen, die LLMs nutzen, eine erhebliche Hilfe ist. Durch die vielen Komponenten und Integrationen wichtiger Tools bietet es die Grundlage für die Verbindung von Daten und komplexen Workflows. In dieser Arbeit wird Langchain genutzt, um die Vektordatenbank und Ollama mit RAGAS zu verbinden. Genutzt wurde die Version 0.3.25.

## 2.2 Daten

Da in dieser Arbeit die Nutzung von RAG-Evaluation-Tools für betriebliche Abläufe untersucht werden soll, werden zum Teil echte, nicht generierte Dokumente – im Folgenden „Originale Dokumente“ genannt – verwendet. Da die Menge an originalen Dokumenten begrenzt ist, werden diese mit synthetischen Daten ergänzt.

### 2.2.1 Originale Dokumente

Die Dokumente stammen aus Unterlagen des Einzelunternehmens „Develop 4 Future (D4F)“, das vereinfachte CMS-Webseiten für Grundschulen entwickelt hat. Der Einzelunternehmer hinter D4F war der Autor. Die Unternehmung wird nicht mehr aktiv verfolgt und die anonymisierten Daten können ohne Bedenken für diese Arbeit genutzt werden. In den Versuchen wurden drei unterschiedliche Anzahlen an Dokumenten getestet: 10, 100 und 400. Aus den Unterlagen von D4F ließen sich 73 nutzbare Dokumente finden. Neben den Dokumenten, welche Businesspläne, Finanzpläne, aber auch Elternbriefe umfassen, gibt es den dazugehörigen Code. Dieser besteht aus drei Projekten:

1. Die Webseite, die öffentlich zugänglich ist

2. dem Admin-Bereich, welcher nur für das Personal der Grundschule zugänglich ist
3. dem Backend, welches die Daten verwaltet.

Für die Versuche mit zehn Dokumente wurden nur „originale“ Dokumente genutzt.

**Anfang: „SmartKlassenzimmer\_Risikoanalyse.txt“**

Risikoanalyse für das Projekt „SmartKlassenzimmer“

Risiko: Verzögerungen bei der Hardware-Lieferung

Maßnahme: Verträge mit mehreren Lieferanten, Pufferzeiten im Zeitplan

Risiko: Technische Probleme bei der Installation

Maßnahme: Vorab-Tests, erfahrenes Technik-Team, Notfallplan

Risiko: Akzeptanzprobleme bei Lehrkräften

Maßnahme: Umfassende Schulungen, Einbindung in die Entwicklung

Risiko: Datenschutzverletzungen

Maßnahme: Strenge Zugriffskontrollen, regelmäßige Audits, DSGVO-konforme Prozesse

Risiko: Budgetüberschreitung

Maßnahme: Monatliche Kostenkontrolle, Reserven im Budget

Risiko: Geringe Nutzung der Plattform durch Schüler:innen

Maßnahme: Motivierende Inhalte, Feedbackrunden, Anpassung der Funktionen

**Ende**

### 2.2.2 Synthetische Daten

Um von 73 gegebenen Dokumenten auf 100 Dokumente zu kommen, wurden mithilfe von LLMs synthetisch weitere Dokumente generiert. Beim Generieren der Dokumente wurden dem LLM die bisherigen Dokumente zur Verfügung gestellt und komplett neue Bereiche/Projekte erfunden. Diese bestehen dann aus Kostenplanungen, Zeitplänen und Elternbriefen für Datenschutzinformationen.

### 2.2.3 Anwendungsszenarien

Die folgenden drei Stufen wurden gewählt, um typische Anwendungsszenarien realistisch abzubilden:

1. **10 Dokumente:** Ein einzelner Anwendungsfall – das RAG-System wird nur temporär genutzt und danach verworfen.
2. **100 Dokumente:** Kontinuierliche Nutzung durch eine Einzelperson – das System wächst schrittweise über die Zeit hinweg.
3. **400 Dokumente:** Gemeinsame Nutzung durch mehrere Personen – das RAG muss verschiedene Themenbereiche abdecken und eine breitere Wissensbasis verwalten.

## 2.3 Fragebögen

Die Generierung von Fragebögen ist eines der Alleinstellungsmerkmale von RAGAS. Sie sind die Grundlage für die Bewertung der RAGs. Es gibt unterschiedliche Fragetypen, die unterschiedliche Aspekte der RAG-Performance evaluieren.

### 2.3.1 Fragetypen

RAGAS unterstützt verschiedene Fragetypen für die Testset-Generierung, die unterschiedliche Aspekte der RAG-Performance evaluieren. Die Abbildung 2.1 zeigt die verschiedenen Fragetypen, die RAGAS für die Evaluation von RAG-Systemen verwendet:

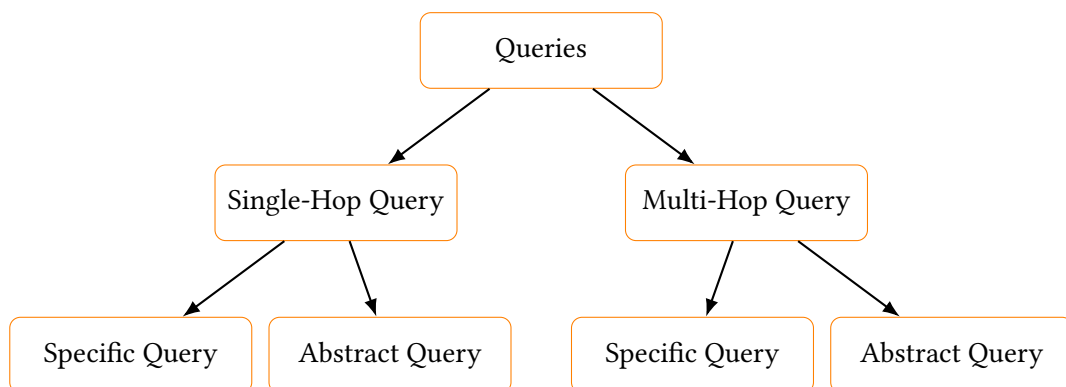


Abbildung 2.1: Fragetypen in RAGAS [26]

Diese in Abbildung 2.1 gezeigten verschiedenen Fragetypen ermöglichen es, unterschiedliche Aspekte der RAG-Performance zu testen. Während spezifische Fragen häufig mit einer einzigen Anfrage an die Wissensdatenbank beantwortet werden können, benötigen abstrakte Fragen eine Erklärung. In der RAGAS-Dokumentation [27] wird für konkrete Fragen das Beispiel gegeben: „Wann hat Einstein die Relativitätstheorie veröffentlicht?“, während eine abstraktere Frage wäre: „Wie hat Einsteins Relativitätstheorie unser Verständnis der Welt verändert?“

Bei Multi-Hop-Queries handelt es sich um Fragen, die mehr als eine Wissensabfrage benötigen. Die Frage „Welche Wissenschaftler haben Einsteins Relativitätstheorie beeinflusst und welche Theorie haben sie vorgeschlagen?“ benötigt erst eine Abfrage, um die Wissenschaftler herauszufinden, und dann weitere, um die jeweils vorgeschlagene Theorie abzufragen. Für die abstrakte Multi-Hop-Query können wir wieder nach einer Erklärung für den Inhalt und wie sich dieser über die Zeit verändert hat, fragen.

Diese Unterscheidung wird getroffen, um sowohl sehr gezielte Wissensabfragen als auch abstraktere Abfragen über mehrere Dokumente zu testen.

### 2.3.2 Wissensgraph

Für die eben erwähnten Multi-Hop-Queries müssen aus den gegebenen Dokumenten Themen, die zusammenhängen, aber nicht direkt im gleichen Dokument vorkommen, gefunden werden. Da dies bei großen Datensätzen manuell oder selbst mit einem LLM schwierig ist, wird ein Wissensgraph erstellt.

Dies geschieht in drei Schritten. Zuerst werden die Dokumente beim sogenannten Chunking in kleinere Einheiten (Knoten) unterteilt. Aus diesen Einheiten können dann Entitäten wie z.B. Namen (Einstein) oder Schlüsselbegriffe (Relativitätstheorie) extrahiert werden. Im letzten Schritt werden dann Verbindungen zwischen Knoten hergestellt, dies wird in der Abbildung 2.2 dargestellt.

Für die Daten aus den Versuchen mit 100 Dokumenten hat RAGAS 27 Themen identifiziert. Unter anderem wurden folgende Themen gefunden: Finanzmanagement, Bildungsprojekt Digitalisierung, Projektmanagement und Planung, Zuwendungsverwaltung, Break-Even-Analyse, Finanzplanung und Investitionen, Finanzplanung und Liquidität.

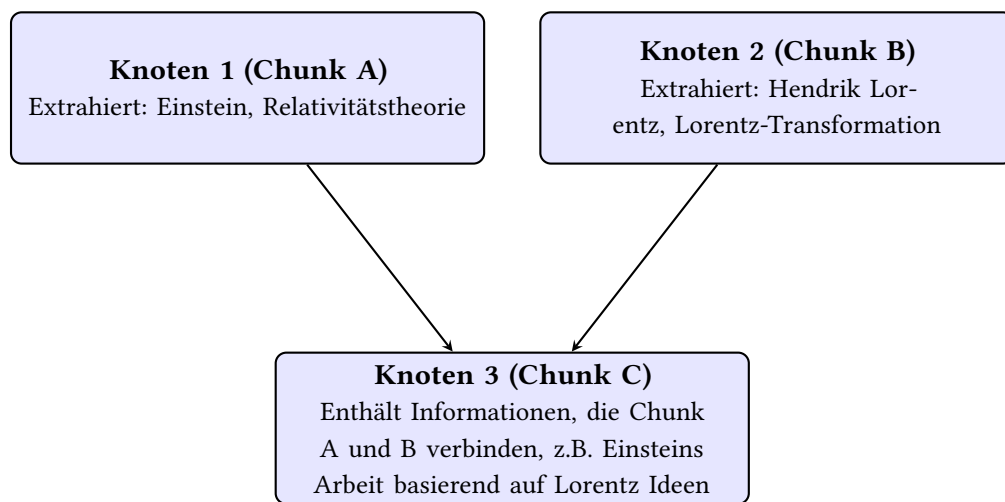


Abbildung 2.2: Beziehungen zwischen Dokumenten-Chunks: Chunk C verbindet die Inhalte von Chunk A und B.

## 2.4 RAG-Bewertungsprozess

Das Flussdiagramm 2.3 veranschaulicht die drei Hauptphasen des RAG-Bewertungsprozesses:

**1. Dokumentenverarbeitung:**

- 1.1 Dokumente werden geladen und in Abschnitte unterteilt
- 1.2 Textabschnitte werden eingebettet
- 1.3 Eingebettete Vektoren werden in ChromaDB gespeichert





**2. Fragebogengenerierung:**

- 2.1 Generierung von Fragen mittels LLM
- 2.2 Erstellung von Testsets mit Fragen und Referenzantworten

**3. Bewertungsprozess:**

- 3.1 Verwendung des generierte Testsets
- 3.2 Abruf des Kontexts aus der ChromaDB
- 3.3 Bewertung der Modellantworten mittels LLM als Richter
- 3.4 Generierung umfassender Bewertungsberichte

**Legende für Flussdiagrammfarben:**

-  **Modell:** (z.B. LLMs, Einbettungsmodelle)
-  **Speicher:** (z.B. Vektorspeicher, ChromaDB)
-  **Prozess:** (z.B. Dokumentenlader, Bewertung)
-  **Daten:** (z.B. Dokumentensammlung, Testset, Bericht)

Das in Abbildung 2.3 dargestellte Diagramm hebt hervor, wie bestimmte Komponenten, wie LLM, für verschiedene Zwecke wiederverwendet werden, während separate Einbettungsmodelle für spezifische Aufgaben beibehalten werden. Dieser modulare Ansatz ermöglicht flexible Versuche mit verschiedenen Modellen und Konfigurationen, während ein konsistentes Bewertungsframework beibehalten wird.

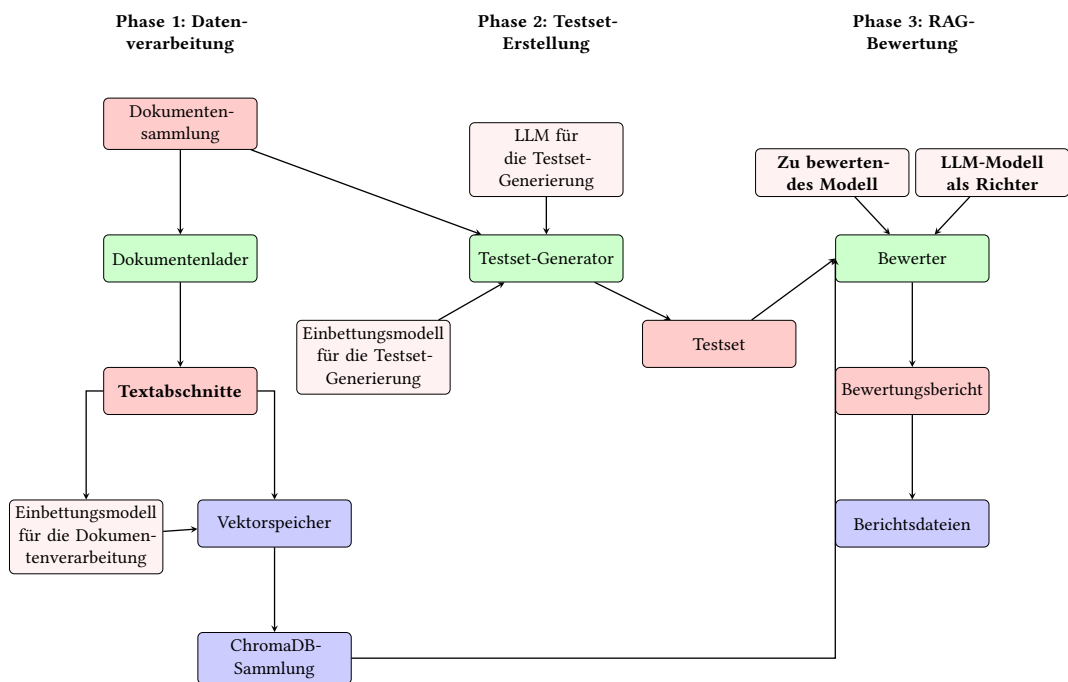


Abbildung 2.3: Flussdiagramm des RAG-Bewertungsprozesses, das die Interaktion zwischen verschiedenen Komponenten und Modellen zeigt. Spezifische Modellnamen (z.B. gpt-4-turbo, text-embedding-3-large) sind im Haupttext beschrieben.

## 2.5 Metriken

Diese Kapitel beschreibt verschiedenen Metriken, die für die Bewertung von RAG-Evaluierungstools verwendet werden können. Metriken sind das Herzstück der Bewertung von RAGs, da sie die Qualität der RAGs bewerten und somit die Entwicklung und den Fortschritt der RAGs messen. Weitere Informationen können in der Dokumentation von RAGAS gefunden werden [28].

In dieser Arbeit werden die vier Metriken „Context Precision“, „Context Recall“, „Response Relevancy“ und „Faithfulness“ für die Bewertung genutzt. Die Kombination dieser Metriken deckt die wichtigen Funktionen eines RAGs ab. Sie werden von RAGAS standardmäßig genutzt und wurden zusammen mit der Idee der Fragengenerierung durch LLMs in dem Paper von RAGAS vorgestellt [7].

### 2.5.1 Verwendete Metriken

Diese Metriken basieren auf Faktenextraktion, mithilfe derer sich dann Bewertungen berechnen lassen. Für die Extraktion der Fakten wird häufig ein LLM verwendet, das als Richter fungiert. Diese sind alle in Ragas implementiert und können direkt genutzt werden.

#### 2.5.1.1 Context Precision

Die Kontextpräzision ist eine Metrik, die den Anteil relevanter Textabschnitte in den abgerufenen Kontexten misst. Sie wird als Mittelwert der Präzision  $k$  für jeden Textabschnitt im Kontext berechnet.

$$\text{Kontext-Präzision } K = \frac{\sum_{k=1}^K (\text{Präzision } k \times v_k)}{\text{Gesamtzahl der relevanten Elemente in den Top } K \text{ Ergebnissen}}$$

$$\text{Präzision } k = \frac{\text{richtig positive } k}{(\text{richtig positive } k + \text{falsch positive } k)}$$

(eigene Übersetzung nach [29])

Diese Metrik ist für uns als Qualitätskontrolle wichtig, da sie uns sagt, ob es Probleme beim Testen mit dem Vector Store gibt.

Wenn es einen guten Context Precision Score gibt, dann lässt sich hier gut bewerten, ob das LLM in der Lage ist, die relevanten Informationen in dem Kontext zu finden. Da dies ein wichtiger Aspekt eines guten RAGs ist, wird diese Metrik im Rahmen dieser Arbeit betrachtet.



### 2.5.1.2 Context Recall

Context Recall misst, wie viele der relevanten Dokumente (oder Informationsstücke) erfolgreich abgerufen wurden. Es konzentriert sich darauf, keine wichtigen Ergebnisse zu verpassen. Ein höherer Recall bedeutet, dass weniger relevante Dokumente ausgelassen wurden. Kurz gesagt geht es beim Recall darum, nichts Wichtiges zu übersehen. (eigene Übersetzung nach [30])

Wenn es einen guten Context Recall Score gibt, dann lässt sich hier gut bewerten, ob das LLM in der Lage ist, die relevanten Informationen in dem Kontext zu finden. Da dies ein wichtiger Aspekt eines guten RAGs ist, wird diese Metrik im Rahmen dieser Arbeit betrachtet.

### 2.5.1.3 Response Relevancy

Die Response Relevancy-Metrik misst, wie relevant eine Antwort in Bezug auf die Nutzereingabe ist. Höhere Werte zeigen eine bessere Übereinstimmung mit der Nutzereingabe an, während niedrigere Werte vergeben werden, wenn die Antwort unvollständig ist oder redundante Informationen enthält. (eigene Übersetzung nach [31])

Diese Metrik bildet mit der Noise Sensitivity eine wichtige Grundlage für die Bewertung des RAGs. Denn selbst wenn die Antworten richtig sind, ist die Bewertung des RAGs nicht gut, wenn die Antworten nicht relevant für die Frage sind.

### 2.5.1.4 Faithfulness

Die Faithfulness-Metrik misst, wie faktentreu eine Antwort im Vergleich zum abgerufenen Kontext ist.

Eine Antwort gilt als faktentreu, wenn alle ihre Aussagen durch den abgerufenen Kontext gestützt werden können.

Die Berechnung erfolgt nach folgender Formel:

$$\text{Faithfulness Score} = \frac{\text{Anzahl der durch den Kontext gestützten Aussagen in der Antwort}}{\text{Gesamtanzahl der Aussagen in der Antwort}} \quad (2.1)$$

(eigene Übersetzung nach [32])

## 2.5.2 Weitere RAGAS Metriken

Die folgenden Metriken sind ebenfalls in RAGAS implementiert, werden jedoch nicht in dieser Arbeit betrachtet.

### 2.5.2.1 Retrieval Augmented Generation

#### Context Entities Recall

In diesem Kontext ist eine Entität eine Informationseinheit, die im Kontext vorkommt. Dies könnte z.B. ein Name, ein Ort, ein Datum oder eine andere Informationseinheit sein.

Die Context Entity Recall-Metrik misst den Recall des abgerufenen Kontexts, basierend auf der Anzahl der Entitäten, die sowohl in der Referenz als auch im abgerufenen Kontext vorkommen, relativ zur Gesamtanzahl der Entitäten in der Referenz.

Einfach ausgedrückt misst sie, welcher Anteil der Entitäten aus der Referenz im abgerufenen Kontext wiedergefunden wird.

(eigene Übersetzung nach [33])

Diese Metrik ist als Qualitätskontrolle wichtig, da sie aussagt, ob es Probleme beim Testen mit dem Vector Store gibt.

#### Noise Sensitivity

Noise Sensitivity misst, wie häufig ein System Fehler macht, indem es falsche Antworten gibt, wenn entweder relevante oder irrelevante abgerufene Dokumente verwendet werden. Um die Noise Sensitivity zu bestimmen, wird jede Aussage in der generierten Antwort daraufhin überprüft, ob sie auf der Grundlage der Referenz korrekt ist und ob sie dem relevanten (oder irrelevanten) abgerufenen Kontext zugeordnet werden kann. (eigene Übersetzung nach [34])

Diese Metrik ist wichtig, sie ist jedoch durch die Context Precision und Context Recall abgedeckt.

#### Multimodal Faithfulness/Multimodal Relevance

RAGAS bietet auch Metriken für MLLMs. Da es in dieser Arbeit nur um LLMs geht, sind diese nicht für diese Arbeit relevant.

### 2.5.2.2 Nvidia Metrics

Die „Nvidia Metrics“ erheben keinen Anspruch auf Objektivität, sie fragen das LLM direkter nach einer Bewertung und nutzen es nicht zur Extraktion von Daten für weitere Berechnungen. Hier werden einzelne Bewertungen generiert, welche keinen tieferen Einblick in die Bewertung gewähren.

#### **Answer Accuracy**

Answer Accuracy misst die Übereinstimmung zwischen der Antwort eines Modells und einer Referenz (Ground Truth) für eine gegebene Frage. Dies geschieht über zwei verschiedene LLMs, diese geben jeweils eine Bewertung (0, 2 oder 4) zurück. Die Metrik wandelt diese Bewertungen in eine Skala von [0,1] um und nimmt dann den Durchschnitt der beiden Bewertungen der Richter.

(eigene Übersetzung nach [35])

Die „Answer Accuracy“ Metrik nutzt ein LLM. Die Antwort und die Musterlösung werden dem LLM zur Bewertung vorgelegt. Da es hier zu einem positional Bias kommen kann, wird das LLM zweimal nach einer Bewertung gefragt, jeweils mit einer anderen Reihenfolge. Dies hat Vorteile gegenüber der Answer Correctness, da es weniger Aufrufe mit weniger Tokens an das LLM braucht. Es werden im Vergleich zur Answer Correctness auch robustere Bewertungen getroffen, jedoch werden weniger Einblicke in die Bewertung ermöglicht.

### 2.5.2.3 Natural Language Comparison

#### **Factual Correctness**

Diese Metriken basieren zum Teil auf der Wahrheitsmatrix (Confusion Matrix), welche die vier Kategorien True Positive, False Positive, False Negative und True Negative definiert [36]. Aus dieser Matrix lassen sich dann Precision, Recall und F1 Score berechnen.

True Positive (TP) = Anzahl der Aussagen in der Antwort, die auch in der Referenz enthalten sind  
False Positive (FP) = Anzahl der Aussagen in der Antwort, die nicht in der Referenz enthalten sind  
False Negative (FN) = Anzahl der Aussagen in der Referenz, die nicht in der Antwort enthalten sind

$$\begin{aligned}\text{Precision} &= \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \\ \text{Recall} &= \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \\ \text{F1 Score} &= 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}\end{aligned}$$

[36]

### **Semantic Similarity**

Diese Metrik nutzt Embeddings, um zu messen, wie ähnlich die Antwort der Musterlösung ist.

#### **2.5.2.4 Non LLM String Similarity**

Die String-Ähnlichkeitsmetrik wird ohne LLM berechnet. Das bietet Vorteile hinsichtlich Geschwindigkeit und Kosten im Vergleich zur LLM-Variante.

##### **BLEU Score**

Der BLEU Score misst die Ähnlichkeit zwischen der Antwort und der Referenz. Dabei wird die Wortanzahl der Referenz berücksichtigt und eine entsprechende Bestrafung für zu kurze Antworten eingeführt [37].

##### **ROUGE Score**

Mithilfe von n-gram Recall, Precision und dem F1 Score wird die Ähnlichkeit zwischen der Antwort und der Referenz berechnet[37].

##### **String Presence**

Eine einfache Metrik, um zu sehen, ob die Referenz in der Antwort enthalten ist [37].

**Frage** „Was ist die Hauptstadt von Deutschland?“

**Antwort** „Berlin ist die Hauptstadt von Deutschland.“

**Referenz** „Berlin“

##### **Exact Match**

Eine noch einfachere Metrik, die nur prüft, ob die Antwort exakt der Referenz entspricht [37].

**Frage** „Was ist die Hauptstadt von Deutschland?“

**Antwort** „Berlin“

**Referenz** „Berlin“

### 2.5.2.5 General Purpose

Dies sind Metriken, welche manuell konfiguriert werden müssen, aber eine gute Bewertung der Qualität eines RAGs liefern können. Die Metriken reichen von einfachen Fragen, wie „Ist die Antwort schädlich“ oder „Hat die Intention des Users verletzt“, bis hin zu komplexeren, einleitend definierten Bewertungen.

**Aspect Critic:** Dem LLM können eigene Kriterien vorgegeben werden, z.B. „Ist die Antwort schädlich?“, hier gibt es nur Ja oder Nein als Antwort.

**Simple Criteria Scoring:** Ähnlich wie der Aspect Critic, jedoch mit einer Zahl als Antwort.

**Rubrics based Scoring:** Erlaubt eine Bewertung mit Vorgaben, welche Kriterien für welchen Score erfüllt sein müssen.

**Instance Specific Rubrics Scoring:** Ist sehr ähnlich zu Rubrics based Scoring, erlaubt jedoch noch genauere Definition von Bewertungskriterien pro Frage.

Der Vollständigkeit halber wird angemerkt, dass RAGAS noch die Summarization-Metrik anbietet, die die Anzahl der richtig beantworteten Fragen geteilt durch die Anzahl aller Fragen misst.

### 2.5.3 Spezielle Metriken

Diese Metriken sind nicht Teil von RAGAS und werden in in einer ähnlichen Arbeit verwendet die in Abschnitt 3.2 beschrieben wird.

- **Noise Robustness (Rauschrobustheit)**, die untersucht das Verhalten, wenn mehr Informationen gegeben sind als notwendig wären, um die Frage zu beantworten. Dies könnte die Frage nach einem bestimmten Ereignis sein und das Rauschen würde ein Dokument zu einem anderen Ereignis sein.
- **Negative Rejection (Negative Ablehnung)**, werden dem LLM nur irrelevante Dokumente zur Verfügung gestellt. Das LLM sollte in diesem Fall antworten, dass es die Frage nicht beantworten kann.
- **Information Integration (Informationsintegration)**, untersucht wie gut ein LLM zwei Fragen in einem aus mehreren Dokumenten beantworten kann.
- **Counterfactual Robustness (Kontrafaktische Robustheit)**, die dem LLM zwei Dokumente mit widersprüchlichen Informationen gibt.

## 3 Ähnliche Arbeiten

### 3.1 RAG Evaluation: Assessing the Usefulness of Ragas

Das Team von Beatrust hat im Februar 2024 eine Reihe von Artikeln zu RAGs veröffentlicht [38]. Es werden unter anderem die Notwendigkeit und auch die einzelnen Metriken von RAGAS erklärt.

Im dritten Artikel dieser Reihe unternehmen sie einen Versuch, die Nützlichkeit von RAGAS zu untersuchen. Der Versuch besteht aus 50 Fragen aus einem Interessensfeld des Autors. Diese wurden von einem RAG mit GPT-4 und einem mit GPT-3.5-turbo beantwortet und dann sowohl von RAGAS als auch von ihm bewertet.

Der Artikel kommt zu dem Ergebnis, dass RAGAS geeignet ist, um RAGs zu bewerten und besser ist, als die Bewertung von Langchain. Es wird jedoch angemerkt, dass der Autor eine höhere Übereinstimmung mit seinen Ergebnissen erwartet hätte.

### 3.2 Benchmarking Large Language Models in Retrieval-Augmented Generation

In ihrem Paper „Benchmarking Large Language Models in Retrieval-Augmented Generation“ [39] haben Jiawei Chen, Hongyu Lin, Xianpei Han and Le Sun ihre eigenen Metriken entwickelt, um die Fähigkeit von RAGs zu bewerten.

Die vier Metrike „Noise Robustness“, „Negative Rejection“, „Information Integration“ und „Counterfactual Robustness“ aus dem Paper wurden in 2.5.3 erklärt.

Diese Metriken werden von den Autoren Retrieval-Augmented Generation Benchmark (RGB) genannt. Mithilfe von RGB bewerten sie in englischer und chinesischer Sprache sechs damals state of the art Modelle ChatGPT (OpenAI 2022), ChatGLM-6B (THUDM 2023a), ChatGLM2-6B (THUDM 2023b), Vicuna-7B-v1.3 (Chiang et al. 2023), Qwen-7B-Chat (QwenLM 2023), BELLE-7B-2M (Yunjie Ji 2023).

Es ließ sich zeigen, dass die LLMs generell gut darin sind, einfache Fragen auch bei irrelevanten Dokumenten zu finden. Wenn sich die Information jedoch über einen größeren Text verteilt, dann haben die LLMs Schwierigkeiten. Auch eine hohe Rauschrate von über 80 % führt zu einer deutlichen Verschlechterung.

Bei der „Negative Ablehnung“ war das beste Ergebnis für Englisch bei 45 % und 43,33 % für Chinesisch. Dies bedeutet, dass die LLMs Anweisungen missachten und sich verwirren lassen, wenn es keinen relevanten Inhalt gibt.

Bei der Informationsintegration, also dem Beantworten einer komplexeren Frage über mehrere Dokumente, sank die Genauigkeit der Antworten deutlich. Es wurde ein maximaler Score von 60 % für englische und 67 % für chinesische Fragen erreicht. Wenn auch Rauschen mit hinzugefügt wird, sinkt die Genauigkeit weiter auf 43 % und 55 %.

Die letzte „Metrik Kontrafaktische“ Robustheit zeigte, dass, selbst wenn das LLM selber die richtige Antwort kennt, es den falschen Informationen vertraut. Das ist ein großes Problem für RAGs und deren Zuverlässigkeit.

Insgesamt zeigt das Paper auf, dass die Untersuchten LLMs 2023 Probleme in wichtigen Bereichen haben, die für RAGs wichtig sind. Besonders die Robustheit gegen Rauschen und die Informationsintegration sind große Probleme, die es zu lösen gilt.

## 4 Versuche

Um systematisch zu bewerten, ob RAG-Bewertungstools für den Einsatz in kleinen und mittleren Unternehmen (KMU) [40] bereit sind, sind umfassende Versuche erforderlich. Es würden unter anderem die Zeitaufwände für die Untersuchungen bestimmt und relevante Datensätze ausgewählt werden.

### 4.1 Versuchsplan

Der folgende Versuchsplan skizziert die wichtigsten Variablen, die Methodik und die Bewertungskriterien.

#### 4.1.1 Forschungsfragen

Der Versuch wird die folgenden zentralen Forschungsfragen behandeln:

- **Relevanz:** Können RAG-Evaluierungstools aussagekräftige und kontextrelevante Fragebögen im Kontext der betrieblichen Abläufe besonders für KMUs generieren?
- **Zuverlässigkeit:** Ermöglichen die durch das Evaluierungstool RAGAS bereitgestellten Metriken und Bewertungen eine valide und zuverlässige Einschätzung der Leistungsfähigkeit von RAG-Systemen?
- **Varianz:** Zeigen die mit RAGAS durchgeführten Bewertungen signifikante Schwankungen, und welche Implikationen ergeben sich daraus für die Verlässlichkeit der Evaluation?
- **Effizienz:** Was ist das optimale Gleichgewicht zwischen Kosten und Leistung für jeden Anwendungsfall in einem KMU?



## 4.1.2 Variablen in den Versuchen

### 4.1.2.1 Dokumententypen

Verschiedene Dokumentenformate werden getestet, um die Vielseitigkeit des Systems zu bewerten:

Tabelle 4.1: Verteilung der Dateitypen und generierten Inhalte

Dateityp	Erweiterungen	Gesamt	Synthetisch	% Synthetisch
Dokumente	.docx, .doc	22	0	0,0%
Tabellen	.xlsx, .csv	17	6	35,3%
Präsentationen	.pptx	2	0	0,0%
Bilder	.jpg, .jpeg, .png	6	0	0,0%
Textdateien	.txt	21	21	100,0%
PDFs	.pdf	5	0	0,0%
<b>Summe</b>		<b>73</b>	<b>27</b>	<b>36,5%</b>

### 4.1.2.2 Datenvolumen

Die Skalierbarkeit des Systems wird, wie bereits beschrieben, mit unterschiedlichen Datenmengen getestet: 10, 100 und 400 Dokumente.

- Für die Versuche mit **10 Dokumenten** werden originale Dokumente ausgewählt.
- Für die Versuche mit **100 Dokumenten** müssen synthetische Dokumente generiert werden, vorzugsweise mit einem LLM.
- Für die Versuche mit **400 Dokumenten** wird zusätzlich Code verwendet.

### 4.1.2.3 Modelle zur Bewertung

Mehrere LLM-Modelle werden bewertet, die verschiedene Kostenschichten und Fähigkeiten repräsentieren. Hierbei ist es wichtig zu überlegen, welche Optionen für KMU gültige Anwendungsfälle sind.

**Open-Source-Modelle** (z.B. Llama 2, Mistral 7B, Deepseek R1) bieten eine Vielzahl von Vorteilen, wie die Möglichkeit, sie zu modifizieren und mehr Kontrolle über die Daten zu haben. Entscheidend ist zudem die technische Kompetenz, welche benötigt wird, um diese Modelle selbst zu hosten.

**Mittelklasse-API-Modelle** (z.B. Claude Haiku, GPT-3.5 Turbo) sind günstiger als die Hochleistungsmodelle und bieten dennoch eine gute Leistung. Da sie nicht Open Source sind, bieten sie weniger Kontrolle über die Daten und das Modell selbst. Manchmal muss man mehr für private Instanzen zahlen.

**Hochleistungsmodelle** (z.B. GPT-4, Claude 3 Opus) sind die teuerste Option, bieten aber auch die beste Leistung, sowohl in Bezug auf Geschwindigkeit als auch auf die Qualität der generierten Antworten.

#### 4.1.2.4 Bewertungsmetriken

Während des Versuchs wird neben der menschlichen Bewertung RAGAS zur Bewertung verwendet. RAGAS wird die beschriebenen Metriken generieren, die später verglichen und bewertet werden können. Die menschliche Bewertung wird als subjektives Maß verwendet, um die Ergebnisse von RAGAS zu vergleichen.

#### 4.1.3 Kosten- und Zeitanalyse

Die Kosten für die Bewertung eines RAGs werden mithilfe der in RAGAS eingebauten Funktionen berechnet. Die Zeit, welche die Ausführung braucht, wird ebenfalls für die Bewertungen gemessen.

#### 4.1.4 Versuchsprotokoll

1. **Dokumentensammlung und -vorbereitung:** Die Dokumente werden in allen oben genannten Zielformaten gesammelt.
2. **Testset-Generierung:** Verschiedene Fragetypen (faktisch, inferentiell, vergleichend) werden generiert und Referenzantworten zur Bewertung erstellt. Dies geschieht automatisch durch das RAGAS-Framework. Das Testset wird manuell auf Qualität und Abdeckung validiert, wobei dies anhand einer Reihe zufälliger Proben erfolgt.
3. **Systemkonfiguration:** Die Einbettungsmodelle und Parameter werden konfiguriert, Vektorspeicher mit konsistenten Einstellungen eingerichtet und die Bewertungsframeworks implementiert.
4. **Durchführung der Bewertung:** Die hochgeladenen Dateien, generierten Dokumente und das Testset werden wiederverwendet. Im ersten Schritt werden diese Daten erstellt. Anschließend wird die Bewertungspipeline ausgeführt und die Ergebnisse werden aufgezeichnet.

5. **Analyse und Berichterstattung:** Eine vergleichende Analyse über alle Variablen hinweg wird durchgeführt, einschließlich einer Kosten-Nutzen-Analyse für die geschäftliche Entscheidungsfindung und Empfehlungen für optimale Konfigurationen.

#### 4.1.5 Bewertungskriterien für die Geschäftstauglichkeit

Die endgültige Bewertung wird RAG-Systeme in diesen Dimensionen bewerten:

- **Implementierungskomplexität:** Wie schwierig ist die Einrichtung und Wartung?
- **Kostenvorhersehbarkeit:** Sind die Kosten stabil und vorhersehbar?
- **Leistungszuverlässigkeit:** Sind die Ergebnisse konsistent und nicht komplett anders bei jeder Bewertung?
- **Skalierbarkeit:** Wie gut bewältigt das System wachsende Datenanforderungen?

Dieser Ansatz mit Versuchen bietet einen umfassenden Rahmen, um zu bewerten, ob aktuelle RAG-Bewertungstools ausreichend ausgereift für die Einführung in einem KMU sind. Sie bieten klare Anleitungen zu optimalen Konfigurationen und Implementierungsstrategien.

## 4.2 Konkretisierung der Versuche

### 4.2.1 Dokumentenverarbeitung

Damit die Dokumente in der Vektordatenbank gesichert werden können, müssen sie erst in Vektoren konvertiert werden. Hierfür wurden Embeddings von OpenAI sowie Open-Source-Embeddings von nomic.ai [41] verwendet.

Tabelle 4.2: Kombinationen aus Dokumentenanzahl und Embedding-Modell für die Versuche (X = Kombination wird getestet)

Embedding-Modell	10	100	400
openai/text-embedding-3-large	X	X	X
ollama/nomic-embed-text	X	X	X

### 4.2.2 Testset-Generierung

Um die optimale Anzahl an Fragen pro Testset zu untersuchen, werden folgende Kombinationen generiert:

Tabelle 4.3: Kombinationen aus Dokumentenanzahl und Testset-Größe

Dokumentenanzahl	Anzahl Fragen pro Testset	Anzahl Testsets pro Modell
10	15, 30	2
100	50, 100	2
400	150, 300	2
Summe Testsets pro Modell		6

### 4.2.3 Bewertung

Um die Robustheit und Übertragbarkeit der Bewertungsergebnisse zu erhöhen, werden alle Kombinationen aus Embedding-Modell und Bewertungsmodell getestet. Das bedeutet, dass für jedes Testset sowohl **openai/text-embedding-3-large** als auch **ollama/nomic-embed-text** als Embedding-Modell verwendet werden und die Bewertung jeweils mit **GPT-4** sowie **Deepseek-R1 (ollama/deepseek-r1:7b)** erfolgt. Insgesamt ergeben sich so 24 Versuche (2 Embeddings  $\times$  2 Bewerter  $\times$  6 Testset-Varianten).

Verwendete Abkürzungen in der Tabelle:

- OAI-E = openai/text-embedding-3-large

Tabelle 4.4: Übersicht aller 12 zu generierenden Bewertungsberichte mit Abkürzungen und Wiederholungen

Versuch	Embedding	Dokumente	Fragen	Bewerter	Richter	Wdh.
1	OAI-E	10	15	GPT-4	GPT-4	1
2	OAI-E	10	30	GPT-4	GPT-4	4
3	OAI-E	100	50	GPT-4	GPT-4	1
4	OAI-E	100	100	GPT-4	GPT-4	4
5	OAI-E	400	150	GPT-4	GPT-4	1
6	OAI-E	400	300	GPT-4	GPT-4	1
7	OLL-E	10	15	DSK-R	DSK-R	1
8	OLL-E	10	30	DSK-R	DSK-R	1
9	OLL-E	100	50	DSK-R	DSK-R	1
10	OLL-E	100	100	DSK-R	DSK-R	1
11	OLL-E	400	150	DSK-R	DSK-R	1
12	OLL-E	400	300	DSK-R	DSK-R	1

- OLL-E = ollama/nomic-embed-text
- GPT-4 = openai/gpt-4
- DSK-R = ollama/deepseek-r1:7b

## 5 Ergebnisse und Diskussionen aus den Versuchen

### 5.1 Generierte Fragebögen

Da insgesamt 1.290 Fragen generiert wurden, lassen sich diese aufgrund des zeitlichen Aufwandes nicht alle manuell bewerten. Aus jedem der sechs generierten Fragebögen werden stichprobenartig jeweils zehn Fragen ausgesucht und überprüft, wie sinnvoll diese sind.

#### 5.1.1 Deepseek/Nomic

Bei der Generierung von Fragen mit DeepSeek kam es zu mehreren Problemen. Bei dem Fragenset, welches 300 Fragen umfassen sollte, traten folgende Probleme auf:

- Von den angeforderten 300 Fragen wurden nur 267 Fragen (89 %) überhaupt generiert; der Rest wurde aufgrund von technischen Problemen oder ungültigen Antworten seitens DeepSeek nicht generiert.
- Von diesen sind 101 zu Themen rund um Bezahlmethoden, Versand und Ähnlichem. In den Dokumenten, welche DeepSeek zur Verfügung gestellt wurden, traten diese Themen nicht auf. Diese Fragen sind daher als ungültig bewertet worden.

Am Ende bleiben 166 von 300 Fragen übrig. **Ca. 45 %** der angeforderten Fragen sind irrelevant.

Beim Generieren des Testsets mit 100 Fragen zeigte sich eine leichte Verbesserung:

- Von den 100 angeforderten Fragen wurden 88 Fragen generiert. Ganze 12 % wurden hier nicht generiert.
- Dieses Mal sind jedoch nur vier Fragen zu irrelevanten Themen wie Bezahlmethoden, Versand etc.

Am Ende hat das Testset mit 100 Fragen eine **Fehlerquote** von **16 %**.

Aus der Tabelle 5.1 wird ersichtlich, dass die **Fehlerquote** in den Testsets für die 400 Dokumente deutlich größer ist. Der Grund dafür wird noch untersucht.

Tabelle 5.1: Übersicht der generierten Fragen und Fehlerquoten pro Testset für DeepSeek

Angefragt	Generiert	Irrelevant	Fehlerquote
15	11	0	27 %
30	27	7	33 %
50	40	0	20 %
100	88	4	16 %
150	137	49	41 %
300	267	101	45 %

### 5.1.2 OpenAI

Tabelle 5.2: Übersicht der generierten Fragen und Fehlerquoten pro Testset für OpenAI

Angefragt	Generiert	Irrelevant	Fehlerquote
15	12	0	20%
30	30	1	3%
50	48	0	4%
100	95	1	6%
150	150	8	5%
300	300	16	5%

Die Tabelle 5.2 zeigt die Übersicht der generierten Fragen und Fehlerquoten pro Testset für OpenAI. Es ist klar zu sehen, dass OpenAI deutlich weniger Probleme mit der Generierung von Fragen hat. Abgesehen von dem Ausreißer mit 15 Fragen, wo 20 % der Fragen irrelevant sind, liegt die Fehlerquote bei den anderen Testsets bei maximal 6 %.

### 5.1.3 Manuelle Auswertung der Fragebögen

Bei der manuellen Auswertung der Fragebögen gucken wir uns einzelne Fragen genauer an und zeigen auf, weshalb diese problematisch sind.

#### 5.1.3.1 DeepSeek

Bei der manuellen Sichtung der Testsets wurden weitere Fehler entdeckt.

##### Beispiel 1:

Neben dem vorhin angesprochenen Problem mit den irrelevanten Themen hat DeepSeek auch zwischendurch Fragen und beispielhafte Antworten auf Englisch generiert.

**Frage:** „How much does it cost?“

**Antwort:** „For orders under \$50, shipping costs \$5.99.“

Diese Frage ist auf Englisch, obwohl explizit Deutsch als Sprache angegeben wurde. Zudem ist diese Frage nicht sinnvoll, da es nie um Lieferungen oder Lieferkosten in den Daten ging.

**Beispiel 2:**

Bei dieser Frage hat das LLM verdreht, wer bezahlt, und fragt, wie viel die Schulen weniger **verdienen** und nicht, wie viel sie weniger **bezahlen**. D4F wird von den Schulen bezahlt und bietet das Produkt den Pilotschulen zu einem günstigeren Preis an, dadurch entstehen die niedrigeren Einnahmen seitens D4F, nicht seitens der Grundschulen.

**Frage:** „Hallo! Ich bin Schulleiter/in und überlege, ob wir als Pilotenschule bei D4F teilnehmen sollen. Könnt ihr mir sagen, wie viel weniger die beiden Pilotenschulen im ersten Jahr verdienen verglichen mit anderen Schulen?“

**Antwort:** „Die beiden Pilotenschulen verdienen im ersten Jahr 2.000 € weniger als die anderen Schulen.“

Diese Frage ist also, auch wenn sie erst gut aussieht, inhaltlich falsch.

**Beispiel 3:**

Auch gab es Probleme mit Fragen, die zu allgemein gefasst waren.

„Ich möchte wissen, wie die Verfügbarkeit der Webseite für Schulen ist.“

Hier ist nicht geklärt, worauf sich die Verfügbarkeit bezieht. Es könnte sich hier sowohl um die Frage handeln, ob aktuell eine Webseite gekauft werden kann, als auch, wie viel Prozent Erreichbarkeit garantiert wird.

**Beispiel 4:**

Ebenso ist „Wie hoch ist die Gesamtsumme der Passiva?“ eine Frage, welche nicht spezifiziert, um welches Jahr es sich handelt und ist daher fehleranfällig.

**Beispiel 5:**

Es gab auch Fragen, die aufgrund des Kontextes verwirrend waren.

**Frage:** „Hallo, ich bin ein kanadischer Student, der sich für die Schulsysteme in Deutschland interessiert. Könntest du mir erklären, warum sich die meisten Grundschulen in NRW befinden?“



**Antwort:** „Die meisten Grundschulen befinden sich in NRW, damit sie das vom Bundesland zur Verfügung gestellte System Logineo einbinden können, das Lehrer- und Schülerverwaltung bietet.“

Dies ist nicht korrekt, die Anzahl der Grundschulen richtet sich nach der demografischen Verteilung der Bevölkerung.

Tabelle 5.3: Anzahl von fehlerhaften Fragen, aus den 10 zufällig ausgewählten Fragen pro Testset und Gesamtübersicht für DeepSeek

Testset	Fehlerhafte Fragen
Ollama – 10 Dok (15 Fragen)	2/10
Ollama – 10 Dok (30 Fragen)	6/10
<b>Summe 10 Dok:</b>	<b>8 / 20 = 40%</b>
Ollama – 100 Dok (50 Fragen)	2/10
Ollama – 100 Dok (100 Fragen)	4/10
<b>Summe 100 Dok:</b>	<b>6 / 20 = 30%</b>
Ollama – 400 Dok (150 Fragen)	5/10
Ollama – 400 Dok (300 Fragen)	8/10
<b>Summe 400 Dok:</b>	<b>13 / 20 = 65%</b>
<b>Gesamt (Ollama):</b>	<b>27 / 60 = 45%</b>

Die in 5.3 berechnete Fehlerquote von 45 % zeigt, dass die Fragen, die DeepSeek generiert, nicht einfach eingesetzt werden können. Es wird deutlich, dass diese Fragen weit entfernt davon sind, die Qualität von menschlich erstellten Fragen zu erreichen.

### 5.1.3.2 OpenAI

Auch bei ChatGPT wurden Mängel bei der manuellen Überprüfung festgestellt.

#### Beispiel 1:

Die Fragen werden so gestellt, dass sie den „gegebenen Kontext“ bewerten sollen. Es fehlen dadurch wichtige Informationen, welche zum Finden der relevanten Dokumente notwendig sind. „Analysieren Sie den bereitgestellten Kontext und erläutern Sie unter der Voraussetzung, dass Sie keine externen Quellen verwenden dürfen, welches zentrale Thema oder welcher Hauptzweck in dem Textabschnitt behandelt wird. Begründen Sie Ihre Antwort anhand spezifischer Textstellen.“

#### Beispiel 2:

Bei einer Frage war das vorliegende Dokument ein Fragebogen. Fehlerhafterweise wurde

die erste Option als die richtige Antwort verstanden, da der Fragebogen nicht ausgefüllt ist, ergibt dies keinen Sinn.

### Beispiel 3:

Auch eine Frage, welche die Antwort schon beinhaltet, wurde generiert: „Kannst du mir erklären, was das besondere Merkmal des neuen Schulwebseiten-Systems ist, das ich als Lehrer verwenden werde, um Abwesenheitsmeldungen schnell und einfach zu veröffentlichen?“

Tabelle 5.4: Anzahl von fehlerhaften Fragen, aus den 10 zufällig ausgewählten Fragen pro Testset und Gesamtübersicht für OpenAI

Testset	Fehlerhafte Fragen
OpenAI – 10 Dok (15 Fragen)	0/10
OpenAI – 10 Dok (30 Fragen)	3/10
<b>Summe 10 Dok: 3 / 20 = 15%</b>	
OpenAI – 100 Dok (50 Fragen)	3/10
OpenAI – 100 Dok (100 Fragen)	1/10
<b>Summe 100 Dok: 4 / 20 = 20%</b>	
OpenAI – 400 Dok (150 Fragen)	5/10
OpenAI – 400 Dok (300 Fragen)	7/10
<b>Summe 400 Dok: 12 / 20 = 60%</b>	
<b>Gesamt (OpenAI): 19 / 60 = 31.67%</b>	

Wenn wir in der Tabelle 5.4 die Testsets mit Code (150/300 Fragen) ignorieren, kommen wir auf eine **Fehlerquote** von 17.5 %. Dies ist die Hälfte von Ollamas 35 %, also eine deutliche Verbesserung, jedoch immer noch eine relevante Menge.

Dennoch offenbaren sich im Vergleich zu einem von Menschen erstellten Fragebogen deutliche qualitative Unterschiede bezüglich der Fragenformulierung.

## 5.2 Auswertung der Reports

Für die Auswertung der Reports werden wieder dieselben Fragen wie vorher aus den Testsets verwendet. Dabei wird geprüft:

- Ist die Frage an sich richtig? Das heißt, ergibt es Sinn, mit dem ursprünglich gegebenen Kontext diese Frage zu stellen?
- Wurde die Frage vom RAG richtig beantwortet?
- Ist die Bewertung der vier Metriken richtig?

Auffällig war, dass Answer Relevancy am häufigsten abweichend war, deswegen wurde hier zusätzlich bewertet, ob die Bewertung besser oder schlechter sein sollte.

### 5.2.1 Manuelle Auswertung DeepSeek

Bei der manuellen Bewertung fällt in Tabelle 5.5 auf, dass ganze 64 % nicht richtig beantwortet wurden. Dabei muss jedoch beachtet werden, dass 43 % der Fragen erst gar nicht sinnvoll sind.

Auch die nicht bewerteten Metriken sind mit bis zu 48 % zu einem großen Teil unbrauchbar. Dies liegt wieder daran, dass das LLM zu lange zum Antworten braucht oder eine ungültige Antwort geliefert hat.

Tabelle 5.5: Verteilung der Bewertungen für DeepSeek (mit Prozentangaben)

Metrik	Richtig	Falsch	Nicht bewertet
Richtige Frage	31 (51.7%)	29 (48.3%)	–
Gültige Antwort	22 (36.7%)	38 (63.3%)	–
context_precision	39 (65.0%)	1 (1.7%)	20 (33.3%)
faithfulness	29 (48.3%)	2 (3.3%)	29 (48.3%)
context_recall	60 (100.0%)	–	–
answer_relevancy	44 (73.3%)	15 (25.0%)	1 (1.7%)
answer_relevancy sollte höher sein	4 (100.0%)	–	–

### 5.2.2 Manuelle Auswertung OpenAI

Bei der Nutzung der OpenAI API für GPT-4 kam es zu keinen Timeouts oder Ähnlichem, welche zu ungültigen Werten führen würden. Es kam jedoch zu zwischenzeitlichen Rate Limits. Diese könnten von einer Firma jedoch bei einem Vertragsschluss mit OpenAI erhöht werden.

Die Tabelle 5.6 zeigt, dass GPT-4 deutlich besser abschneidet als DeepSeek, 27 % an nicht sinnvollen Fragen ist jedoch immer noch ein hoher Wert. Die Hälfte der ungültigen Antworten ist durch sinnlose Fragen bedingt; hier ziehen sich also die schlecht generierten Fragen durch.

Tabelle 5.6: Verteilung der Bewertungen für OpenAI mit Prozentangaben

Metrik	Richtig	Falsch
Richtige Frage	43 (72.9%)	16 (27.1%)
Gültige Antwort	42 (71.2%)	17 (28.8%)
context_precision	56 (94.9%)	3 (5.1%)
faithfulness	51 (86.4%)	8 (13.6%)
context_recall	57 (96.6%)	2 (3.4%)
answer_relevancy	43 (72.9%)	16 (27.1%)
answer_relevancy sollte höher sein	7 (53.8%)	6 (46.2%)

### 5.2.3 Probleme mit Programm-Code

Da sowohl bei der Testset-Generierung als auch bei der Bewertung der Testsets, die 400 Dokumente nutzten, höhere **Fehlerquoten** zu beobachten sind, wird dies genauer untersucht.

Der Vergleich der Anzahl der 0.0-Bewertungen mit DeepSeek (Abbildung 5.1) im Vergleich zu OpenAI (Abbildung 5.2) ist eindeutig. 106 ( 40 %) der insgesamt 276 zu bewertenden Fragen waren mit komplett 0.0 bewertet worden. Bei der Analyse der speziellen Zeichen im Kontext fällt auf, dass 89 Bewertungen ( 83 %) zu mehr als 5 % nur aus diesen bestehen. Dies deutet darauf hin, dass DeepSeek starke Probleme hat, Fragen mit Code zu generieren und/oder zu finden.

Ein großer Teil der Fragen hat sich also auf Dokumente mit minderwertiger Qualität bezogen. Durch diese minderwertigen Dokumente hat sich das LLM dann verwirren lassen. Es zeigt sich wieder einmal, dass die Qualität der Daten eine entscheidende Rolle spielt. Bei Betrachtung der Ergebnisse ohne Dokumente in 5.3 und 5.4, welche Code enthalten, sehen wir, dass die 0.0-Bewertungen bei DeepSeek deutlich zurückgehen. Wie zu erwarten war, ist OpenAI's ChatGPT-4 immer noch deutlich besser.

## 5.3 Unterschiede über mehrere Durchläufe

Um zu prüfen, wie sich die Ergebnisse von Durchlauf zu Durchlauf unterscheiden, wurden für das Testset mit 100 Fragen für 100 Dokumente mit beiden Modellen vier Durchläufe vorgenommen. In diesem Versuch geht es dann um die Unterschiede pro Durchlauf für das Modell festzustellen und nicht die Modelle miteinander zu vergleichen. Der Mean und Std wurde über alle Fragen hinweg berechnet.

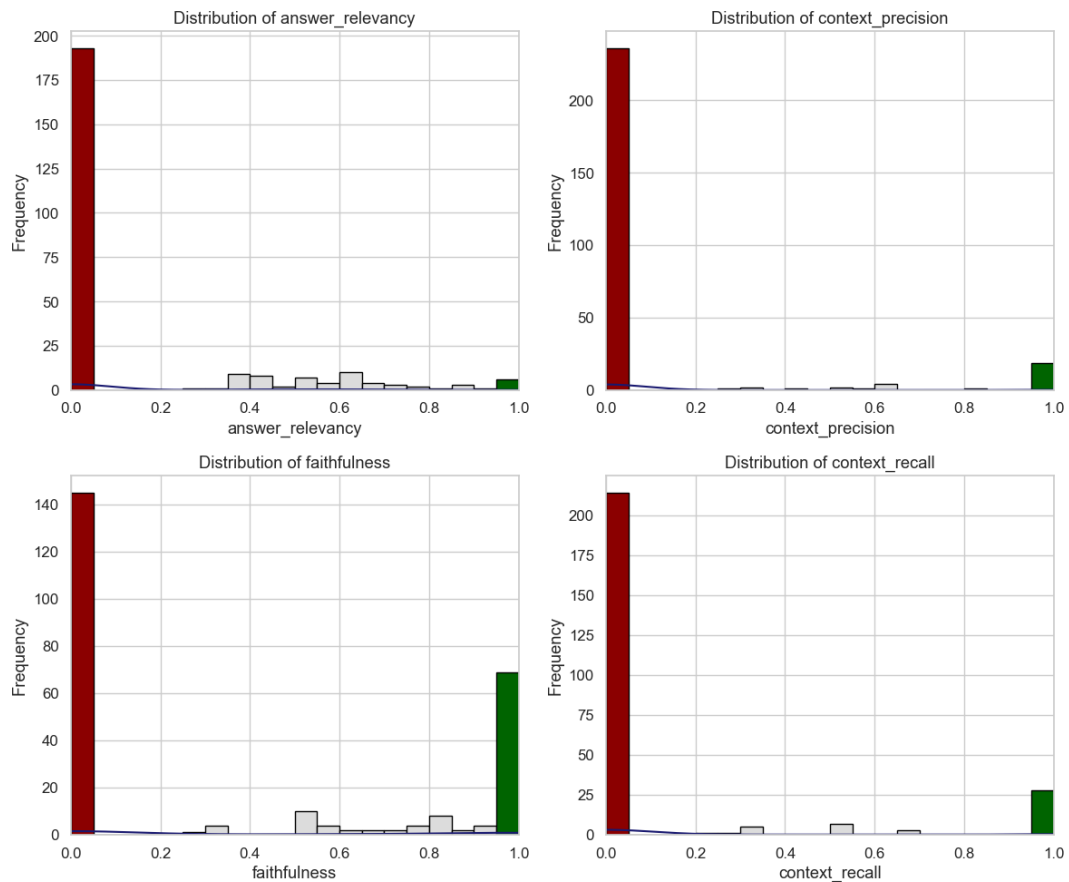


Abbildung 5.1: DeepSeek Ergebnis für 300 Fragen (mit Code-Dokumenten)

Tabelle 5.7: Durchschnittswerte und Standardabweichungen der Metriken über vier Durchläufe für DeepSeek

Metrik	Mean 1	Mean 2	Mean 3	Mean 4	Std 1	Std 2	Std 3	Std 4
Answer Relevancy	0.336	0.366	0.329	0.358	0.346	0.340	0.347	0.361
Faithfulness	0.624	0.593	0.627	0.623	0.442	0.429	0.436	0.453
Context Precision	0.344	0.344	0.344	0.344	0.449	0.449	0.449	0.449
Context Recall	0.415	0.415	0.415	0.415	0.484	0.484	0.484	0.484

Beim Betrachten des Durchschnitts und der Standardabweichung in Tabelle 5.7 lässt sich für die Answer Relevancy und die Faithfulness sehen, dass eine gewisse Schwankung vorhanden ist. Die Metriken für den Kontext sind jedoch sehr konstant!

Bei der Answer Relevancy lässt sich ein Unterschied von 3.7 % feststellen, bei der Faithfulness

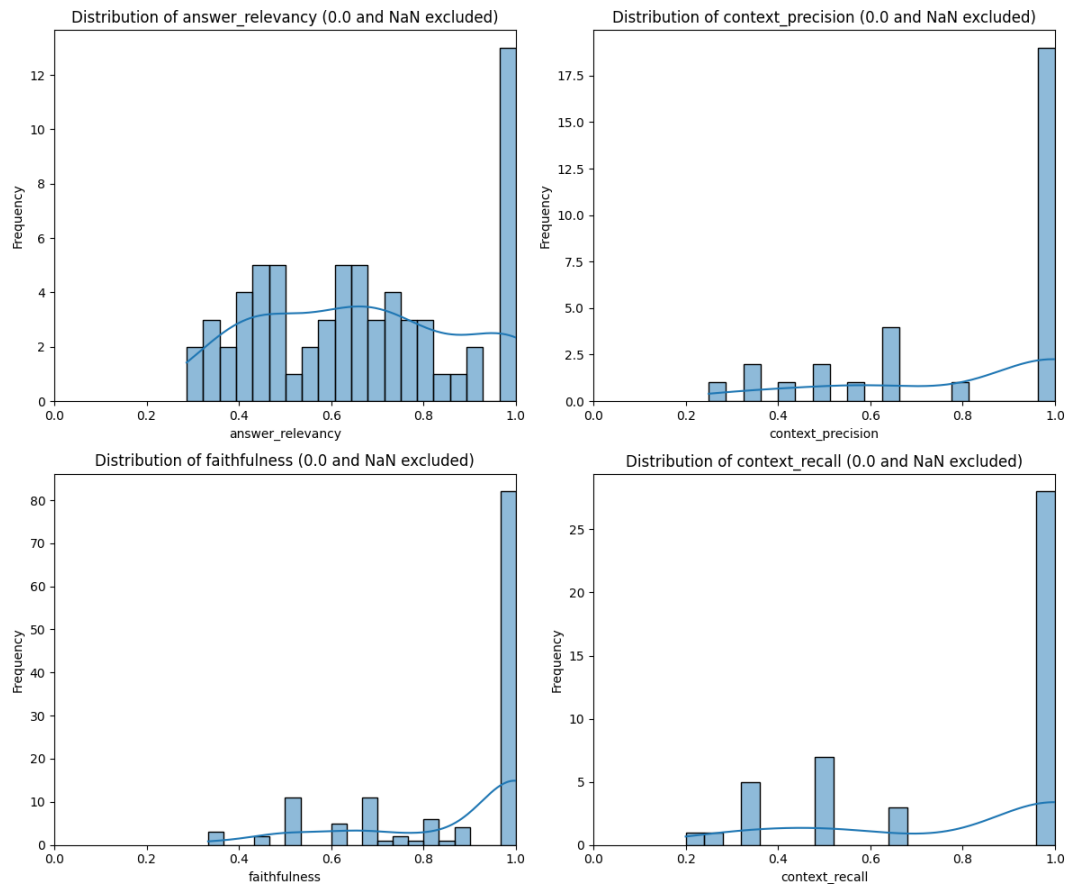


Abbildung 5.2: ChatGPT Ergebnis für 300 Fragen (mit Code-Dokumenten)

3.1 %. Dies liegt für vier Durchläufe im Rahmen, bedeutet aber auch, dass dies beim Einrichten einer automatischen Pipeline berücksichtigt werden sollte.

Tabelle 5.8: Durchschnittswerte und Standardabweichungen der Metriken über vier Durchläufe für GPT-4

Metrik	Mean 1	Mean 2	Mean 3	Mean 4	Std 1	Std 2	Std 3	Std 4
Answer Relevancy	0.680	0.681	0.666	0.667	0.320	0.314	0.315	0.313
Context Precision	0.671	0.670	0.666	0.667	0.443	0.445	0.444	0.445
Context Recall	0.647	0.671	0.681	0.665	0.472	0.456	0.458	0.461
Faithfulness	0.846	0.800	0.815	0.823	0.247	0.295	0.273	0.295

Beim Betrachten des Durchschnitts und der Standardabweichung in Tabelle 5.8 zeigt sich, dass GPT-4 insgesamt eine konstante Leistung liefert. Die Werte für Answer Relevancy be-

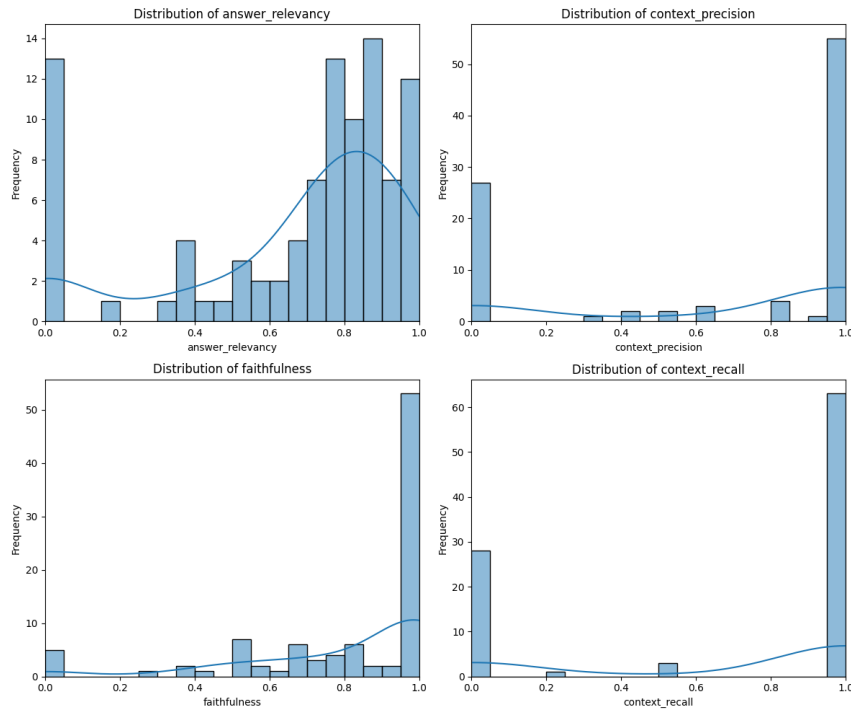


Abbildung 5.3: DeepSeek Ergebnis für 100 Fragen (ohne Code-Dokumente)

wegen sich zwischen 66.6 % und 68.1 %, was einer Abweichung von lediglich 1.5 % entspricht. Auch die Standardabweichung ist mit etwa 0,31 gleichmäßig verteilt.

Bei der Faithfulness sind die Mittelwerte etwas variabler und reichen von 80.0 % bis 84.6 %. Dies ergibt eine Differenz von 4.6 %, die im Rahmen liegt, aber auf gewisse inhaltliche Schwankungen hinweist.

Die Metriken für den Kontext (Context Precision und Context Recall) zeigen ebenfalls eine stabile Entwicklung mit geringen Unterschieden in den Mittelwerten. Die Standardabweichungen sind bei diesen Metriken nahezu konstant.

Insgesamt zeigt GPT-4 verlässliche Ergebnisse mit nur geringen Varianzen über mehrere Durchläufe hinweg.

## 5.4 Zuverlässigkeit von Metriken

Um genauer zu untersuchen, wie sich die Metriken bei mehrfacher Ausführung verhalten, wurden die vier Metriken jeweils 50 Mal ausgeführt. Bei der Bewertung wurde immer

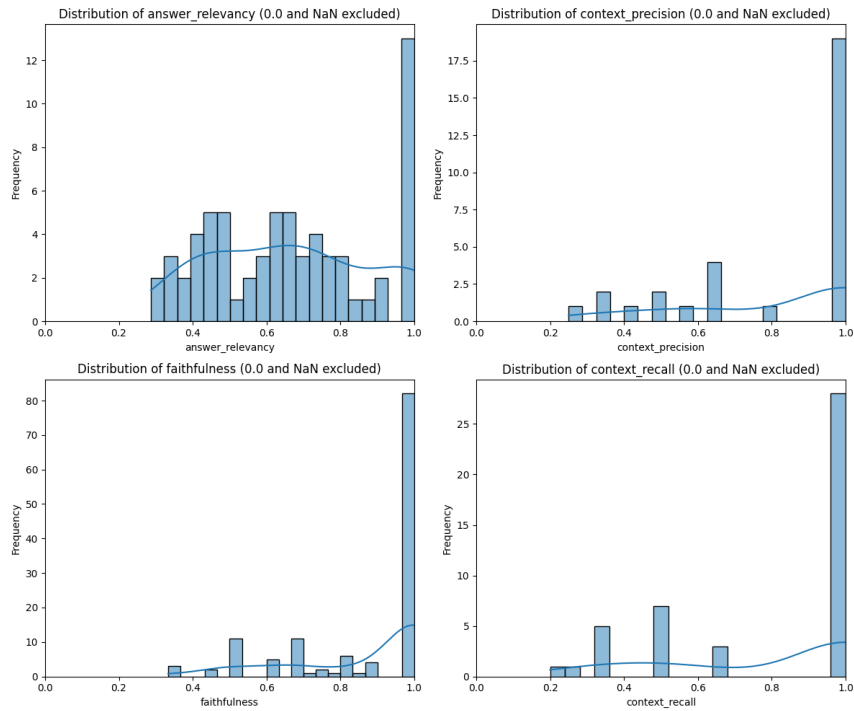


Abbildung 5.4: ChatGPT Ergebnis für 100 Fragen (ohne Code-Dokumente)

GPT-4.1 verwendet.

### 5.4.1 Context Precision & Recall

Beide Metriken wurden 50 Mal bewertet und haben sich wie bei DeepSeek in der Gesamtbeurteilung als sehr stabil herausgestellt. Es gab in keinem der Durchläufe eine Abweichung.

### 5.4.2 Answer Relevancy

Wie in Abbildung 5.5 zu sehen ist, sind minimale Abweichung festgestellt worden. Diese belaufen sich aber auf die zweite Nachkommastelle in der Prozentangabe und sind daher vernachlässigbar.



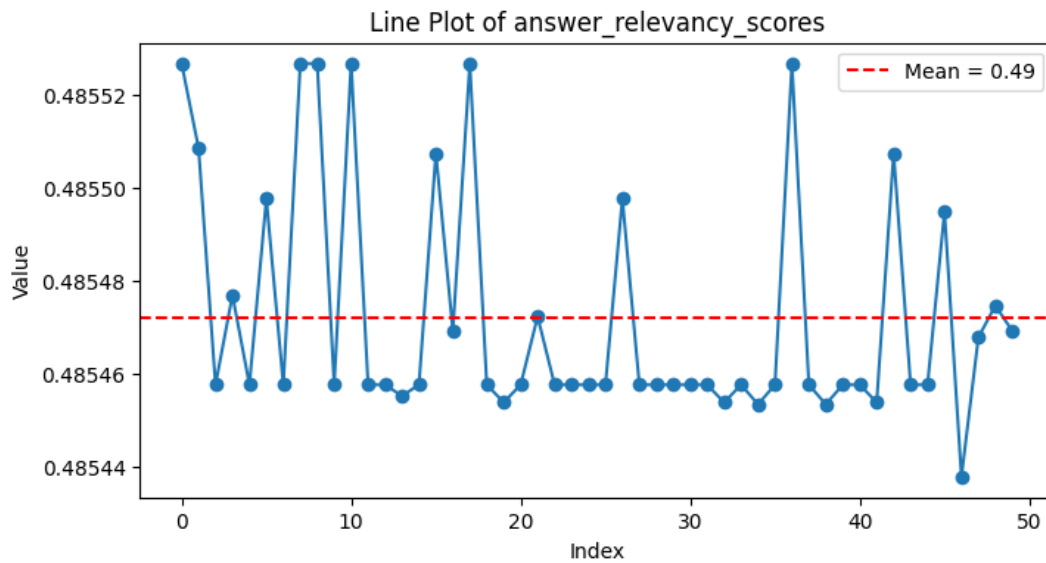


Abbildung 5.5: Abweichungen des Answer Relevancy Scores

### 5.4.3 Faithfulness

In Abbildung 5.6 lässt sich eine deutliche Abweichung feststellen. Die richtige Bewertung wäre 62.5 %. In 66 % der Fälle war dem auch so, es ist jedoch ersichtlich, dass der Wert teilweise bis zu 12.5 % abweichen kann.

Die Faithfulness-Metrik hat die größten Schwankungen; dies war auch schon bei dem Versuch mit mehreren Durchläufen ersichtlich.

## 5.5 Ausführungszeiten

Da es bei OpenAI zu den Rate Limits kommen kann, wurde die Anzahl an gleichzeitigen Abfragen von 16 auf 1 reduziert, da es sonst besonders bei längeren Durchläufen zu Problemen kommt. Das führt zu einer deutlichen Verschlechterung der Ausführungszeit. Das Generieren des Testsets mit 15 Fragen dauerte bei 16 gleichzeitigen Anfragen 2 Minuten, bei nur einer maximalen Anfrage wurden es 7 Minuten. Mit diesen Zahlen lässt sich annehmen, dass der Versuch mit 300 Fragen ohne Rate Limit seitens OpenAI sicherlich unter einer Stunde geschafft werden könnte.

Für die Bewertung des Testsets mit 300 Fragen (400 Dokumente) wurde Tracing genutzt. Dies lässt uns genauer prüfen, warum gewisse Bewertungen fehlgeschlagen sind. Es kam

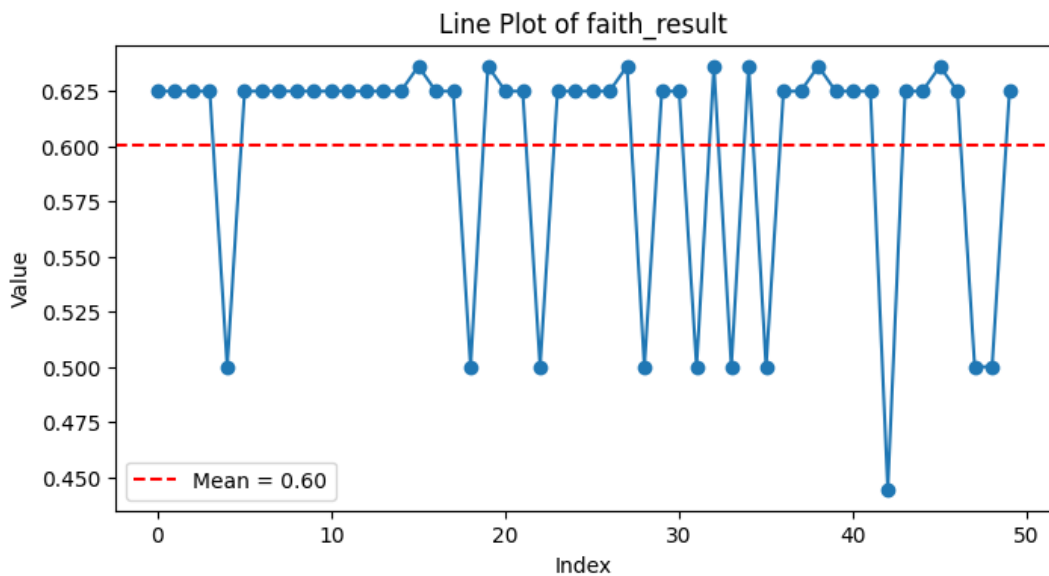


Abbildung 5.6: Abweichungen des Faithfulness Scores

insgesamt zu 20 Fehlern: 12 Zeitüberschreitungen, weil das LLM nicht innerhalb von 10 Minuten geantwortet hat, acht Antworten waren in einem ungültigen Format, sieben davon für context\_recall und eine für faithfulness. Mit den 20 fehlgeschlagenen Metriken kommen wir auf eine **Fehlerquote** von 1.9 %.

In der Tabelle 5.9 lassen sich die deutlichen Unterschiede in der Ausführungszeit zwischen OpenAI und DeepSeek erkennen. Mit Rate Limit ist OpenAI immer noch siebenmal schneller als DeepSeek, sollten hier höhere Limits gelten, könnte OpenAI sogar bis zu 20-mal schneller sein.

Tabelle 5.9: Dauer der Evaluation pro Dokumentenanzahl im Vergleich mit Differenz, Zeitangaben in Stunden:Minuten

Anzahl Dokumente	Dauer DeepSeek	Dauer OpenAI	Differenz
15	00:41	*00:02	00:39
30	01:03	*00:03	01:00
50	01:35	*00:04	01:31
100	03:41	*00:07	03:34
150	05:20	01:37	03:43
300	17:29	02:30	14:59

\*Ohne Rate-Limit ausgeführt

## 5.6 Kostenberechnung

Die Bewertung des RAGs mit den 300 Fragen und OpenAI hat 2 Stunden und 30 Minuten gedauert, dabei sind Kosten in Höhe von 12 Euro entstanden.

Dies kann man mit einer Bewertung, wie in den Versuchen, auf einem Mac Studio (M2 Ultra) vergleichen.

- Laufzeit pro Bewertung DeepSeek: 17 h
- Stromkosten:  $0,12 \text{ €/h} \Rightarrow 17 \text{ h} \times 0,12 \text{ €/h} = \mathbf{2,04 \text{ €}}$
- OpenAI API-Kosten pro Bewertung: 12,00 €  
Davon sollen 2,00 € lokal durch eigene Ausführung ersetzt werden  $\Rightarrow$  verbleibende Abschreibung: **10,00 €/Run**
- Geräteanschaffung: 7.200 €  $\Rightarrow$  amortisiert über 720 Runs à 10,00 €
- Gesamtkosten pro Run:  $2,04 \text{ € (Strom)} + 10,00 \text{ € (Abschreibung)} = \mathbf{12,04 \text{ €}}$
- Gesamtlaufzeit (720 Runs):  $720 \times 17 \text{ h} = 12.240 \text{ h} \approx \mathbf{1 \text{ Jahr, 4 Monate, 10 Tage}}$

Die Preise für die Nutzung eines LLMs über eine Schnittstelle zu entscheiden liegt hier komplett beim Anbieter. Auch die Preise für Strom oder Hochleistungs-GPUs können in Zukunft schwanken. Sollte es also in Zukunft Änderungen an diesen variablen Preisen geben, könnte diese Entscheidung anders ausfallen.

## 6 Zusammenfassende Betrachtungen

### 6.1 Forschungsfragen

#### 6.1.1 Relevanz

Die Generierung von Fragebögen ist eines der Alleinstellungsmerkmale von RAGAS. Die Generierung der Fragebögen hat sich aus softwaretechnischer Sicht als unkompliziert erwiesen.

Es ist mit RAGAS möglich, Fragebögen zu generieren, jedoch gibt es mehrere Faktoren, die Qualität und Praxistauglichkeit beeinflussen:

- Die Fähigkeit des LLMs, zuverlässig hochwertige Antworten zu generieren.
- Die Dokumente: Je komplexer und zusammenhangsloser die Dokumente sind, desto schlechter lassen sich Fragen generieren.

Bei der Generierung von Fragebögen mit DeepSeek kam es allein durch die nicht generierten oder zu irrelevanten Themen generierten Fragen zu einer Fehlerquote von bis zu 45 %. Selbst bei händisch ausgewählten Dokumenten lag die Fehlerquote bei mindestens 16 %.

Die händische Überprüfung hat dann weiter gezeigt, dass DeepSeek Probleme mit der konstanten Generierung von sinnvollen Fragen hat. Hier wiesen bis zu 65 % der Fragen für ungefilterte Dokumente Mängel auf! Selbst bei den gefilterten Dokumenten waren mindestens 30 % mangelbehaftet.

Die Generierung von Fragebögen mit OpenAIs GPT-4 hatte in Bezug auf nicht generierte oder Fragen zu irrelevanten Themen eine deutlich niedrigere Fehlerquote. Es gibt einen Ausreißer mit 20 %, der Rest bleibt jedoch deutlich unter 10 %. Die manuelle Auswertung hat hier aber auch gezeigt, dass viele Fragen, bis zu 60 % bei ungefilterten Dokumenten, Mängel aufweisen. Bei gefilterten Dokumenten kommt GPT-4 auf durchschnittlich 17,5 % und halbiert damit die Fehlerquote im Vergleich mit DeepSeek. Es ist also stark zu empfehlen ein LLM zu wählen, welches eine hohe Qualität bei der Generierung von Fragen bietet. Dies wird sich durch die bei der manuellen Überprüfung gesparten Zeit rentieren.

Die bei den Versuchen generierten Fragebögen lassen Zweifel an einer automatischen, zuverlässigen und hochwertigen Generierung von Fragen aufkommen. Eine händische

Überprüfung der generierten Fragen ist notwendig, um die Qualität der Fragebögen zu gewährleisten.

Im Kontext eines KMU ist die Generierung von Fragebögen mit RAGAS jedoch ein großer Vorteil. In einem KMU ist es oft nicht möglich alle Fragen händisch zu generieren, da entweder die Zeit oder die Ressourcen fehlen, die manuelle Überprüfung ist jedoch noch im Rahmen des Möglichen.

### 6.1.2 Zuverlässigkeit

Die Qualität der Fragebögen ist entscheidend, da sich hier entstandene Fehler weiter bis in die Bewertung durchziehen und eine korrekte Bewertung des eigentlichen RAGs verzerren! Auch das Generieren von Bewertungen hat sich mithilfe von RAGAS als einfach umsetzbar erwiesen. Sowohl das Tracing als auch die Kostenberechnung waren für die unterstützten Modelle problemlos zu benutzen. Das Tracing erlaubt außerdem einen tieferen Blick in die Berechnung der Metriken und macht das ganze System transparenter.

Es hat sich jedoch bei der manuellen Durchsicht gezeigt, dass hier bei DeepSeek 60 % und bei GPT-4 30 % der Fragen nicht richtig beantwortet wurden. Dies lässt sich teils auf die ungültigen Fragen in den Fragebögen zurückführen.

Bei den Metriken lässt sich sagen, dass die Metriken zum Kontext (recall und precision) gut abschneiden. Die faithfulness zeigt eine erhöhte Abweichung zur menschlichen Einschätzung und sollte mit einer Toleranz von 10 % beachtet werden.

Die Answer Relevancy hat die größte Abweichung; hier fällt auf, dass sowohl höhere als auch niedrigere Werte erwartet wurden.

Um die Aussagekraft der Bewertungen zu erhöhen, ist es jedoch zu empfehlen die Metriken einer genaueren Auswertung zu unterziehen. Beispielsweise sollten die Metriken Answer Relevancy und Faithfulness mehr gewichtet werden, wenn die Kontext-Metriken gut abschneiden.

Zusammenfassend lässt sich sagen, dass RAGAS durch die bereitgestellten Metriken eine gute Grundlage bietet um die einzelnen Komponenten eines RAGs zu bewerten. Die Metriken lassen sich mittels der Tracing-Funktion gut nachvollziehen und die Kosten für die Bewertung sind transparent. Wie auch bei der Generierung der Fragebögen hat das gewählte LLM einen großen Einfluss auf die Qualität der Bewertung.

### 6.1.3 Varianz

Sowohl die mehrfache gesamte Bewertung als auch das mehrfache Ausführen einzelner Metriken haben gezeigt, dass die Bewertung einer Schwankung von wenigen Prozentpunkten unterliegt. Es war zu sehen, dass LLMs mit mehr Parametern geringere Schwankungen aufwiesen. Hier war der Unterschied zwischen DeepSeek und OpenAI jedoch deutlich geringer als bei der Fragebogengenerierung oder der Bewertung. Insgesamt lässt sich sagen, dass die minimalen Schwankungen die Praxistauglichkeit nicht beeinflussen und sich auf die Ergebnisse verlassen werden kann.

### 6.1.4 Effizienz

In einem KMU gibt es mehrer Anwendungsfälle für RAGs. Sollte das KMU aktiv an der Entwicklung eines RAGs arbeiten, so ist eine Bewertung innerhalb von 17 Stunden nicht praxistauglich. Hier ist die Verwundung eines LLMs, das innerhalb einer Stunde eine Bewertung generieren kann, notwendig.

Wird jedoch getestet, wie gut ein RAG mit neuen Dokumenten umgeht, so ist eine Bewertung innerhalb von 17 Stunden durchaus praxistauglich. Die hohen Anschaffungskosten, die nicht parallele Ausführung und die lange Dauer der Bewertungen reduziert die Anwendungsfälle jedoch stark.

## 6.2 Fazit

Insgesamt ist RAGAS kein kompletter Ersatz für die menschliche Bewertung von RAGs. Die Idee hinter RAGAS, Fragen ohne menschliches Zutun zu generieren, um Zeit zu sparen, ist mit besseren LLMs teilweise gelungen. Um jedoch ein aussagekräftiges und zuverlässiges Ergebnis zu generieren, ist eine menschliche Kontrolle an mehreren Stellen notwendig. Zuerst bei der Auswahl der Dokumente: Hier muss sowohl ein Verständnis vorhanden sein, wie gut LLMs mit welchen Daten umgehen können, als auch welche Daten für das KMU relevant sind. Nach der Generierung der Fragebögen sollte erneut ein Mensch die Fragen überprüfen, um grob falsche Fragen zumindest zu löschen.

Da die Berichte relativ konstante Bewertungen abgeben, lassen sich dann durchaus Verschlechterungen oder Verbesserungen am RAG messen. Die Metriken geben Aufschluss darüber, welcher Teil des Systems nicht funktioniert; diese Zusammenhänge lassen sich sehr gut sehen.

Es muss jedoch auch der Zeit- und Kostenaufwand für eine solche Bewertung in Betracht gezogen werden. Für eine aktive Entwicklung ist das Abwarten von 17 Stunden für eine Bewertung eines RAGs nicht praxistauglich und ein Hindernis. Eine Bewertung innerhalb von einer Stunde ist praxistauglich, ist jedoch ein Kostenfaktor. Hier muss der Anwendungsfall genauer betrachtet werden.

Für Unternehmen bieten LLMs und RAGs großes Potenzial für Kosteneinsparungen; die Qualitätskontrolle spielt dabei eine immer größere Rolle. RAGAS bietet gute Ansätze, um die Qualitätskontrolle zu automatisieren. Dass RAGAS in Zukunft in die Prozesse zur Bewertung solcher Systeme einfließt, ist daher sehr wahrscheinlich.

Wie in Kapitel 1.5 beschreiben, muss die KI-VO beachtet werden. Die KI-VO verfolgt einen risikobasierten Ansatz [42]. Dessen müssen sich die Anwender bewusst sein. In Abbildung 6.1 werden die unterschiedlichen Risikoklassen ersichtlich. Die sich daraus ergebenden Einschränkungen und Verpflichtungen müssen für die jeweiligen Einsatzgebiete des RAGs beachtet werden.

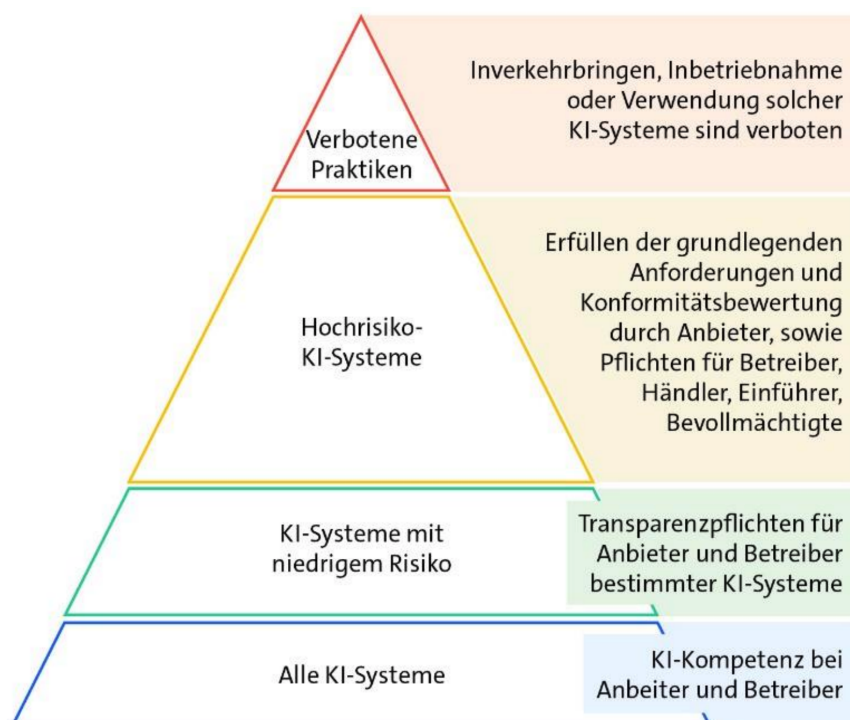


Abbildung 6.1: Risikopyramide KI-Systeme  
Quelle: [43]

Hinzukommen eventuell Verpflichtungen nach der DS-GVO, wenn in den Dokumenten

personenbezogene Daten nach Artikel 4 DSGVO, enthalten sein sollten. Ebenso wie bei der Verletzung der KI-VO können auch hier hohe Bußgelder drohen.

### 6.3 Reflexion der Arbeit

Vor Beginn dieser Arbeit antizipierte wir bereits Schwierigkeiten bei der Bewertung von RAG-Systemen mittels RAGAS:

**Kosten** Die Kosten, die bei der Bewertung entstehen.

**Zeitbedarf** Die Dauer für die Durchführung der Bewertung. Die Bewertung kann schneller durchgeführt werden, wenn mehr Ressourcen zur Verfügung stehen.

**Implementierung** Das Aufsetzen des zu testenden Systems. Dies beinhaltet eine eventuelle doppelte Speicherung der Daten und die für das Testen benötigten Aufrufe des LLMs.

**Aktualisierung** Das System muss auf dem neuesten Stand gehalten werden, da sich dieses aktuelle Thema schnell entwickelt.

Während der Arbeit an dieser Bachelorarbeit gab es einige Herausforderungen und Erkenntnisse, die den Ablauf und die Ergebnisse beeinflusst haben. Hier sind einige Überlegungen zur Reflexion der Arbeit:

1. Dokumente

Im ersten Schritt, dem Sammeln der Dokumente und deren Aufbereitung, gab es einige Herausforderungen. Es standen nicht genügend originale Dokumente zur Verfügung, sodass synthetische Dokumente generiert werden mussten. Damit das LLM realistische Dokumente generieren kann, wurden die originalen Dokumente als Grundlage genutzt und neue Bereiche und Projekte erfunden.

2. RAGs und RAGAS verstehen

Da es bereits Erfahrungen mit RAGs im betrieblichen Umfeld gab, war es einfach, die neuesten Entwicklungen zu verstehen. Die Einarbeitung in RAGAS und die Metriken war aufgrund der guten Dokumentation und der vorangegangenen Erfahrungen mit RAGs ebenfalls relativ einfach. Nicht nur die Verwendung der einzelnen Komponenten, sondern auch das Nutzen der einzelnen Metriken war einfach und gut umsetzbar.

3. Versuchsplan

Die Erstellung des Versuchsplans war eine große Herausforderung. Es mussten verschiedene Aspekte berücksichtigt werden, wie z. B. die Anzahl der Dokumente, die Art der Dokumente und die Modelle, die bewertet werden sollten. Zudem gab es eine Unvorhersehbarkeit der Dauer und Kosten der Versuche. Es konnte also erst nach



den ersten Versuchen eine realistische Einschätzung der Dauer und Kosten gegeben werden.

4. Implementierung

Die grundlegende Implementierung von RAGAS war dank Langchain und der guten Dokumentation einfach. Durch die vielen Abhängigkeiten und die schnelle Weiterentwicklung von Python-Bibliotheken gab es jedoch einige Probleme, die die Implementierung erschwerten. So gab es ein Problem mit dem Paket LiteLLM; einen Fix gab es schon, aber dieser war noch nicht in Version 1.67.5 übernommen worden, weshalb dieser Fix eigens implementiert werden musste. Das System, welches implementiert wurde, war jedoch sehr hilfreich, da hier die Generierung der Testsets und die Bewertung der RAGs automatisiert im Hintergrund abliefen. Dies war besonders bei langlaufenden Versuchen wichtig, damit die Versuche nicht manuell gestartet werden mussten.

5. Versuche

Will man verhindern, dass eine definierte Grenze durch die Iteration überschritten wird, sollte man eine Durchsatzbegrenzung implementieren. Eine solche Durchsatzbegrenzung soll dann die zugrundeliegende Dienste und Ressourcen schützen [44]. Der implementierte Durchsatzbegrenzer war nicht sehr ausgereift und war einer der Punkte, welche die Dauer der Versuche verlängert haben. Sowohl bei OpenAI als auch bei DeepSeek gab es Probleme mit der Durchsatzbegrenzungs-Implementierung, welche zu ungünstigen Ergebnissen führten und das erneute Starten der Versuche notwendig machten. Auch die Auswahl der Hardware wurde während der Arbeit gewechselt, da zuerst auf einem Laptop gearbeitet wurde, welcher nicht die Leistung hatte, um die Versuche in einem angemessenen Zeitrahmen durchzuführen. Die 32 GB RAM wurden komplett ausgeschöpft und führten zum Absturz des Systems. Das finale System, ein M2 Ultra mit 192 GB RAM, konnte die Versuche in einem angemessenen Zeitrahmen durchführen und war auch für die zukünftige Nutzung praxistauglich.

6. Auswertung

Die Auswertung war aufgrund der vielen Daten und der manuellen Auswertung aufwendig; außerdem fielen Fehler während der manuellen Auswertung auf. Diese führten dazu, dass die Auswertung der Fragebögen und der Bewertungen länger dauerte und erneut ausgeführt werden musste. Einer der Fehler wurde auch durch die Intransparenz der Modellversionierung seitens OpenAI verursacht. Während des Versuchs für die Zuverlässigkeit der Bewertung wurde die Modellversion von GPT-4 geändert, was zu einer Abweichung der Ergebnisse führte und das erneute Durchführen der Versuche erforderte.

## Literatur

- [1] OpenAI u. a. „GPT-4 Technical Report“. In: *arXiv e-prints*, arXiv:2303.08774 (März 2023), arXiv:2303.08774. DOI: 10.48550/arXiv.2303.08774. arXiv: 2303.08774 [cs.CL].
- [2] Doit Software. *ChatGPT Statistiken*. Accessed: 2025. 2025. URL: <https://doit.software/de/blog/chatgpt-statistiken#screen5>.
- [3] Gemini Team u. a. „Gemini: A Family of Highly Capable Multimodal Models“. In: *arXiv e-prints*, arXiv:2312.11805 (Dez. 2023), arXiv:2312.11805. DOI: 10.48550/arXiv.2312.11805. arXiv: 2312.11805 [cs.CL].
- [4] DeepSeek-AI u. a. „DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning“. In: *arXiv e-prints*, arXiv:2501.12948 (Jan. 2025), arXiv:2501.12948. DOI: 10.48550/arXiv.2501.12948. arXiv: 2501.12948 [cs.CL].
- [5] Gemini Team u. a. „Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context“. In: *arXiv e-prints*, arXiv:2403.05530 (März 2024), arXiv:2403.05530. DOI: 10.48550/arXiv.2403.05530. arXiv: 2403.05530 [cs.CL].
- [6] Hugo Touvron u. a. „Llama 2: Open Foundation and Fine-Tuned Chat Models“. In: *arXiv e-prints*, arXiv:2307.09288 (Juli 2023), arXiv:2307.09288. DOI: 10.48550/arXiv.2307.09288. arXiv: 2307.09288 [cs.CL].
- [7] Shahul Es u. a. „RAGAs: Automated Evaluation of Retrieval Augmented Generation“. In: *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*. Hrsg. von Nikolaos Aletras und Orphee De Clercq. St. Julians, Malta: Association for Computational Linguistics, März 2024, S. 150–158. URL: <https://aclanthology.org/2024.eacl-demo.16/>.
- [8] Ashish Vaswani u. a. „Attention Is All You Need“. In: *arXiv e-prints*, arXiv:1706.03762 (Juni 2017), arXiv:1706.03762. DOI: 10.48550/arXiv.1706.03762. arXiv: 1706.03762 [cs.CL].
- [9] Microsoft Learn. *Understand tokens*. Webseite auf Microsoft Learn. Abgerufen am 28. Juni 2025. Mai 2025. URL: <https://learn.microsoft.com/de-de/dotnet/ai/conceptual/understanding-tokens>.

- 
- [10] Fraunhofer IESE. *Wie funktionieren LLMs?* Blogbeitrag auf der Webseite des Fraunhofer IESE. Abgerufen am 29. Juni 2025. Juni 2025. URL: <https://www.iese.fraunhofer.de/blog/wie-funktionieren-llms/>.
- [11] Thorsten Honroth, Julien Siebert und Patricia Kelbert. *Retrieval Augmented Generation (RAG): Chatten mit den eigenen Daten*. Zugriff am 7. Juni 2025. Mai 2024. URL: <https://www.iese.fraunhofer.de/blog/retrieval-augmented-generation-rag/>.
- [12] Luyu Gao u. a. „RT-RAG: Leveraging Retrieval-Generated Chains for Open-Domain Question Answering“. In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2023, S. 9784–9800. URL: <https://aclanthology.org/2023.acl-long.546/>.
- [13] Nikhil Kandpal u. a. „Large Language Models Struggle to Learn Long-Tail Knowledge“. In: *arXiv e-prints*, arXiv:2211.08411 (Nov. 2022), arXiv:2211.08411. DOI: 10.48550/arXiv.2211.08411. arXiv: 2211.08411 [cs.CL].
- [14] OpenAI. *GPT-4 Vision*. Accessed: 2025-06-29. 2025. URL: <https://platform.openai.com/docs/models/gpt-4-vision>.
- [15] Google AI. *Google Gemini: Next-generation multimodal AI model*. Accessed: 2025-06-29. Feb. 2024. URL: <https://blog.google/technology/ai/google-gemini-next-generation-model-february-2024/#build-experiment>.
- [16] Luka Panic. *RAG in der Praxis – Generierung synthetischer Testdatensätze*. Abgerufen am 30. Mai 2025. 2024. URL: <https://pixion.co/blog/rag-in-practice-test-set-generation>.
- [17] Peiyi Wang u. a. „Large Language Models are not Fair Evaluators“. In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Hrsg. von Lun-Wei Ku, Andre Martins und Vivek Srikumar. Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, S. 9440–9450. DOI: 10.18653/v1/2024.acl-long.511. URL: <https://aclanthology.org/2024.acl-long.511/>.
- [18] Ammar Shaikh u. a. „CBEval: A framework for evaluating and interpreting cognitive biases in LLMs“. In: *arXiv e-prints*, arXiv:2412.03605 (Dez. 2024), arXiv:2412.03605. DOI: 10.48550/arXiv.2412.03605. arXiv: 2412.03605 [cs.CL].
- [19] Gemini Team. *Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context*. [https://storage.googleapis.com/deepmind-media/gemini/gemini\\_v1\\_5\\_report.pdf](https://storage.googleapis.com/deepmind-media/gemini/gemini_v1_5_report.pdf). Google DeepMind Technical Report. 2024.

- 
- [20] Hugging Face. *Pleias-RAG-1B*. Accessed: 2025. 2025. URL: <https://huggingface.co/PleIAS/Pleias-RAG-1B>.
- [21] EU-Kommission. *KI-Verordnung*. Accessed: 2025-06-29. 2025. URL: <https://ai-act-law.eu/de/>.
- [22] J. Martini und C. Wendehorst. *KI-VO: Verordnung über Künstliche Intelligenz*. C.H.BECK, 2024. ISBN: 978-3-406-81136-4.
- [23] Ollama. *Ollama*. Accessed: 2025. 2025. URL: <https://ollama.com/>.
- [24] Try Chroma. *Try Chroma*. Accessed: 2025. 2025. URL: <https://www.trychroma.com/>.
- [25] Harrison Chase and LangChain contributors. *LangChain: Language model application development framework*. Erste Veröffentlichung Oktober 2022; Version aktuell am 26. Juni 2025. 2022. URL: <https://www.langchain.com/>.
- [26] Ragas Team. *Testset Generation for RAG*. Accessed: 2025-06-29. 2025. URL: [https://docs.ragas.io/en/latest/concepts/test\\_data\\_generation/rag/](https://docs.ragas.io/en/latest/concepts/test_data_generation/rag/).
- [27] Ragas. *Query types in RAG*. Accessed: 2025. 2024. URL: [https://docs.ragas.io/en/stable/concepts/test\\_data\\_generation/rag/#query-types-in-rag](https://docs.ragas.io/en/stable/concepts/test_data_generation/rag/#query-types-in-rag).
- [28] Ragas. *Metrics*. Accessed: 2025. 2025. URL: <https://docs.ragas.io/en/stable/concepts/metrics/>.
- [29] Ragas. *Context Precision*. Accessed: 2025. 2024. URL: [https://docs.ragas.io/en/stable/concepts/metrics/available\\_metrics/context\\_precision/](https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/context_precision/).
- [30] Ragas. *Context Recall*. Accessed: 2025. 2024. URL: [https://docs.ragas.io/en/stable/concepts/metrics/available\\_metrics/context\\_recall/](https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/context_recall/).
- [31] Ragas. *Response Relevancy*. Accessed: 2025. 2024. URL: [https://docs.ragas.io/en/stable/concepts/metrics/available\\_metrics/answer\\_relevance/](https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/answer_relevance/).
- [32] Ragas. *Faithfulness*. Accessed: 2025. 2024. URL: [https://docs.ragas.io/en/stable/concepts/metrics/available\\_metrics/faithfulness/](https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/faithfulness/).
- [33] Ragas. *Context Entities Recall*. Accessed: 2025. 2024. URL: [https://docs.ragas.io/en/stable/concepts/metrics/available\\_metrics/context\\_entities\\_recall/](https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/context_entities_recall/).
- [34] Ragas. *Noise Sensitivity*. Accessed: 2025. 2024. URL: [https://docs.ragas.io/en/stable/concepts/metrics/available\\_metrics/noise\\_sensitivity/](https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/noise_sensitivity/).

- 
- [35] Ragas. *Nvidia Metrics*. Accessed: 2025. 2024. URL: [https://docs.ragas.io/en/stable/concepts/metrics/available\\_metrics/nvidia\\_metrics/](https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/nvidia_metrics/).
- [36] Wikipedia. *Confusion matrix*. Accessed: 2025. 2024. URL: [https://en.wikipedia.org/wiki/Confusion\\_matrix](https://en.wikipedia.org/wiki/Confusion_matrix).
- [37] Ragas Documentation. *Traditional NLP Metrics*. [https://docs.ragas.io/en/stable/concepts/metrics/available\\_metrics/traditional/](https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/traditional/). Accessed: 2025-06-28. Juni 2025.
- [38] Beatrust. *RAG Evaluation: Assessing the Usefulness of Ragas*. Accessed: 2025. 2024. URL: [https://tech.beatrust.com/entry/2024/05/02/RAG\\_Evaluation%3A\\_Assessing\\_the\\_Usefulness\\_of\\_Ragas](https://tech.beatrust.com/entry/2024/05/02/RAG_Evaluation%3A_Assessing_the_Usefulness_of_Ragas).
- [39] Jiawei Chen u. a. „Benchmarking Large Language Models in Retrieval-Augmented Generation“. In: *arXiv e-prints*, arXiv:2309.01431 (Sep. 2023), arXiv:2309.01431. doi: 10.48550/arXiv.2309.01431. arXiv: 2309.01431 [cs.CL].
- [40] ESF. *Glossar*. Accessed: 2025. 2025. URL: [https://www.esf.de/portal/DE/Service/Glossar/Functions/glossar.html?cms\\_lv3=f748ebe5-3f04-4af0-ae3f-64f888942114&cms\\_lv2=3943ee31-db5a-48c8-96e9-c69287930b3e](https://www.esf.de/portal/DE/Service/Glossar/Functions/glossar.html?cms_lv3=f748ebe5-3f04-4af0-ae3f-64f888942114&cms_lv2=3943ee31-db5a-48c8-96e9-c69287930b3e).
- [41] Nomic Team. *Introducing Nomic Embed: A Truly Open Embedding Model*. <https://www.nomic.ai/blog/posts/nomic-embed-text-v1>. Online; veröffentlicht auf dem Nomic AI Blog.
- [42] Martin Ebers und Benedikt M. Quarch. *Rechtshandbuch ChatGPT*. Nomos, 2024. ISBN: 978-3-7560-1285-5.
- [43] Bitkom e.V. *Umsetzungsleitfaden zur KI-Verordnung (EU) 2024/1689*. Zuletzt abgerufen am 29. Juni 2025. 2024. URL: <https://www.bitkom.org/Bitkom/Publikationen/Umsetzungsleitfaden-zur-KI-Verordnung-EU-2024-1689>.
- [44] Built In. *Rate Limiting Explained: Top Strategies for System Design*. 2023. URL: <https://builtin.com/software-engineering-perspectives/rate-limiter> (besucht am 28.06.2025).

## Anhang

## Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer oder der Verfasserin/des Verfassers selbst entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Anmerkung: In einigen Studiengängen findet sich die Erklärung unmittelbar hinter dem Deckblatt der Arbeit.

---

Ort, Datum

---

Unterschrift