
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*

Eine Kurzfassung (wenn verlangt) in Deutsch und/oder in Englisch (*Abstract*) umfasst auf etwa 1/2 bis 1 Seite die Darstellung der Problemstellung, der angewandten Methode(n) und des wichtigsten Ergebnisses.

Wie man ein gelungenes Abstract verfasst, erfahren Sie in den Seminaren oder der Beratung des Schreibzentrums der Kompetenzwerkstatt¹.

Schlagwörter/Schlüsselwörter: gegebenenfalls Angabe von 3 bis 10 Schlagwörtern.

¹<https://www.th-koeln.de/schreibzentrum>

Inhaltsverzeichnis

Tabellenverzeichnis	IV
Abbildungsverzeichnis	V
1 Einleitung	1
1.1 Wie funktioniert ein RAG	2
1.1.1 Vorteile von RAGs	2
1.1.2 Kompetenz des Betreibers	3
1.1.3 Datenbasis	3
1.1.4 Budget	4
1.2 Objektive Beurteilung von RAGs	4
1.3 Darstellung des Themas und der Forschungsfragen	4
1.4 Praxistauglichkeit und Herausforderungen	4
1.5 Softwaretechnische Fragestellungen	5
1.6 Rechtliche Fragestellungen	6
2 Methoden und Materialien	7
2.0.1 Werkzeuge	7
2.0.2 Daten	8
2.0.3 Fragebögen	9
2.0.4 Evaluation	10
2.1 Metriken	11
2.1.1 Retrieval Augmented Generation	11
2.1.2 Nvidia Metrics	13
2.1.3 Natural Language Comparison	14
2.1.4 Non LLM String Similarity	15
2.1.5 General purpose	16
2.1.6 Andere Metriken	16
2.1.7 Irrelevante Metriken	16
3 Ähnliche Arbeiten	17
3.1 RAG Evaluation: Assessing the Usefulness of Ragas	17
3.2 RAG-Bewertungsprozess	18

4	Versuche	20
4.1	Versuchsplan	20
4.1.1	Forschungsfragen	20
4.1.2	Variablen in den Versuchen	20
4.1.3	Kosten- und Zeitanalyse	22
4.1.4	Versuchsprotokoll	22
4.1.5	Bewertungskriterien für die Geschäftstauglichkeit	23
4.2	Konkretisierung der Versuche	24
4.2.1	Dokumentenverarbeitung	24
4.2.2	Testset-Generierung	24
4.2.3	Bewertung	24
5	Ergebnisse und Diskussionen	26
5.1	Ergebnisse aus den Versuchen	26
5.1.1	Generierte Fragebögen	26
5.1.2	Manuelle Auswertung der Fragebögen	27
5.1.3	Auswertung der Reports	30
5.1.4	Unterschiede über mehrere Durchläufe	32
5.1.5	Zuverlässigkeit von Metriken	35
5.1.6	Kostenberechnung	37
5.2	Abhängigkeit der Metriken untereinander	38
5.3	Identifikation von Interessenten	38
6	Zusammenfassungen	39
6.1	Benutzung von RAGAS	39
6.2	Testsets	39
6.3	Bewertung	40
6.4	Fazit	40
6.5	Zukunftsausblick	41
6.6	Reflexion der Arbeit	41
	Literatur	42
	Anhang	44

Tabellenverzeichnis

4.1	Kombinationen aus Dokumentenanzahl und Embedding-Modell für die Versuche (X = Kombination wird getestet)	24
4.2	Kombinationen aus Dokumentenanzahl und Testset-Größe	24
4.3	Übersicht aller 24 zu generierenden Bewertungsberichte mit Abkürzungen und Wiederholungen	25
5.1	Übersicht der generierten Fragen und Fehlerquoten pro Testset für DeepSeek	27
5.2	Übersicht der generierten Fragen und Fehlerquoten pro Testset für OpenAI	27
5.3	Anzahl fraglicher Fragen pro Testset und Gesamtübersicht für Ollama . . .	28
5.4	Anzahl fraglicher Fragen pro Testset und Gesamtübersicht für OpenAI . .	29
5.5	Verteilung der Bewertungen für DeepSeek (mit Prozentangaben)	30
5.6	Verteilung der Bewertungen für OpenAI (gesamt) mit Prozentangaben . .	31
5.7	Durchschnittswerte und Standardabweichungen der Metriken über vier Durchläufe für DeepSeek	34
5.8	Durchschnittswerte und Standardabweichungen der Metriken über vier Durchläufe für GPT-4	35
5.9	Dauer der Evaluation pro Dokumentenanzahl mit DeepSeek	37
5.10	Dauer der Evaluation pro Dokumentenanzahl mit OpenAI	37

Abbildungsverzeichnis

1.1	Struktur eines RAGs, Quelle: [7]	2
3.1	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.	19
5.1	DeepSeek Ergebnis für 300 Fragen (mit Code-Dokumenten)	31
5.2	ChatGPT Ergebnis für 300 Fragen (mit Code-Dokumenten)	31
5.3	Abweichungen des Faithfulness Scores bei Code-Dokumenten.	32
5.4	DeepSeek Ergebnis für 100 Fragen (ohne Code-Dokumente)	33
5.5	ChatGPT Ergebnis für 100 Fragen (ohne Code-Dokumente)	33
5.6	Bewertung der vier Durchläufe mit DeepSeek	33
5.7	Bewertung der vier Durchläufe mit GPT-4	34
5.8	Abweichungen des Answer Relevancy Scores.	35
5.9	Abweichungen des Faithfulness Scores.	36
5.10	Abhängigkeit der Metriken voneinander (OpenAI, 300 Fragen, 400 Dokumente)	38

1 Einleitung

Im Jahr 2022 veränderte OpenAI mit ihrem browserbasierten ChatGPT (Generative Pre-trained Transformer) die Welt komplett. In nur fünf Tagen erreichte ChatGPT eine Million Nutzer*innen [20] und ist im Leben vieler Menschen Alltag und in manchen gar nicht mehr wegzudenken. Die GPT-KI (Künstliche Intelligenz) von OpenAI gehört zur Familie der Large Language Models (LLMs) oder auch Multimodal Large Language Models (MLLMs). MLLMs können neben Text weitere Datenmodalitäten wie zum Beispiel Bilder, Audio und Video verarbeiten. LLMs von anderen Anbietern haben mit der Qualität und den Fähigkeiten von OpenAIs GPT gleichgezogen. Mittlerweile gibt es viele Arten, LLMs zu bewerten, und ein reger Wettbewerb ist um die vielen Bewertungen entstanden.

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

Im Oktober 2023 kam der Verfasser das erste Mal mit Retrieval-Augmented Generation (RAGs) 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 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 konnte die funktionsfähigkeit überwacht und eventuelle schlechte Ergebnisse dokumentiert werden. Diese Zeitintensive Aufgabe kostete uns wertvolle Zeit, welche das Team lieber in die Entwicklung investiert hätte. Gerne hätten wir unterschiedliche Prompts innerhalb unseres Systems getestet oder eine automatische Überprüfung unserer neuesten Änderungen gehabt.

RAGAS wurde entwickelt, um diese Probleme 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 haben, nimmt RAGAS mithilfe von LLMs vor. Mit dieses Testsets und von RAGAS eigens entwickelten Metriken, welche die wichtigsten Funktionen eines RAGs abdecken, kann eine Bewertung des Systems vorgenommen werden.

Damit benutzt RAGAS die neue LLM-Technologie, um das durch LLMs entstandene System selbst zu testen. Dies spart viele menschliche Ressourcen, welche zeit- und kostenintensiv sind.

1.1 Wie funktioniert ein RAG

Bei Retrieval Augmented Generation (RAG) erweitert man den Prompt für das LLM um Suchergebnisse aus einer Dokumentensammlung, einer Datenbank, einem Wissensgraphen (Knowledge Graph) oder einer anderen Suche (z.B. Internetsuche). Das Wissen für die Antwort kommt also aus angebundenen Quellen und nicht aus dem LLM.

[7]

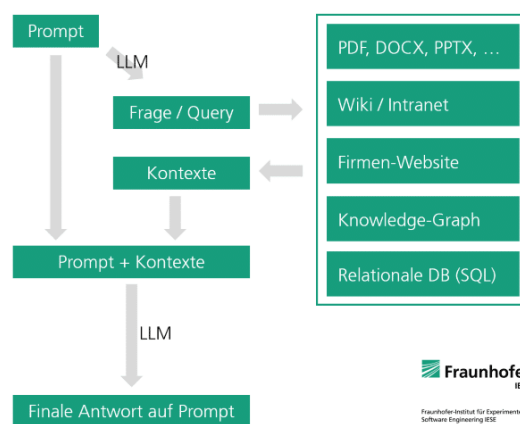


Abbildung 1.1: Struktur eines RAGs, Quelle: [7]

1.1.1 Vorteile von RAGs

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

1. Im Trainingsset für die LLMs selten vorkommendes Wissen können selbst LLMs schlecht lernen. [5] [10]
2. LLMs kennen nur die verwendeten Trainingsdaten und müssen weiter trainiert werden, um neue Informationen zu erlernen.

3. Firmeninterne Dokumente sind nicht im Trainingsset, und daher können LLMs keine Fragen zu firmeninternen Daten beantworten.

Die Nutzung eines RAGs ist eine der drei Möglichkeiten, um ein LLM zu verbessern. Die anderen beiden Möglichkeiten sind Finetuning und die Verwendung eines LLMs mit einem großen context window. RAGs haben neben dem Fine-Tuning und der Nutzung eines LLMs mit großem context window entscheidende Vorteile.

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

1.1.2 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 Sichten der neuen Trainingsdaten und ist zudem durch die benötigte Hardware teuer.

Das Benutzen eines LLMs benötigt die geringste Kompetenz des Betreibers, da hier das LLM unverändert bleibt. Hier werden einfach die Daten inklusive der Frage an das LLM gesendet.

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 suchenden (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.

1.1.3 Datenbasis

Sollten die Daten dynamisch sein, ist das RAG 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 den anderen Methoden.

1.1.4 Budget

Das Finetuning erfordert über einen langen Zeit teure Rechenzeit auf Hochleistungs-GPUs, was das Training eines Modells kostenintensiv macht.

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.

1.2 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 versuchen, automatisieren diesen Prozess unter anderem mithilfe von LLMs zu automatisieren. 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 die reine subjektive Bewertung durch LLMs sind jedoch nicht objektiv. Anhand mehrerer Techniken kann versucht werden, die Bewertung mithilfe von LLMs zu objektivieren.

1.3 Darstellung des Themas und der Forschungsfragen

In dieser Bachelorarbeit wird untersucht, wie gut diese Tools sowohl subjektive als auch objektive Bewertungen durchführen können. Im Mittelpunkt wird das Tool RAGAS stehen, welches die Bewertung durchführt.

1.4 Praxistauglichkeit und Herausforderungen

Es stellen sich mehrere Herausforderungen für die Bewertung von RAGs durch diese Tools.

- Die Kosten, die bei der Bewertung entstehen.

- Die Dauer für die Durchführung der Bewertung. Die Bewertung kann schneller durchgeführt werden, wenn mehr Ressourcen zur Verfügung stehen.
- Das Aufsetzen des zu testenden Systems. Dies beinhaltet eine eventuelle doppelte Speicherung der Daten und die für das Testen benötigten Aufrufe des LLMs.
- Das System muss auf dem neuesten Stand gehalten werden, da sich dieses aktuelle Thema schnell entwickelt.

1.5 Softwaretechnische Fragestellungen

In dem Artikel *RAG in der Praxis – Generierung synthetischer Testdatensätze* untersucht Luka Panic [9] 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 diesem Paper [22] wird gezeigt, dass, wenn ein LLM eine von zwei gegebenen Antworten aussuchen müsste, die erste bevorzugt wurde, selbst wenn die gleiche Frage mit einer anderer 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 [19] 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 dieser Arbeit kurz untersucht.

Wie in diesem Paper [6] 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 entwickelt um die Suche mit RAGs zu verbessern. [4]

1.6 Rechtliche Fragestellungen

Am 01.08.2024 trat die Verordnung über Künstliche Intelligenz der Europäischen Union (KI-VO) in Kraft. Die Verordnung setzt Regelungen und Maßstäbe für die Verwendung von KI. RAGs sind gemäß Artikel 3 Nr. 1 KI-VO KI-Systeme und fallen damit in den Anwendungsbereich der KI-VO. Bei der Nutzung oder Bereitstellung von LLMs muss sich an die KI-VO gehalten werden. Die Nutzer*innen der RAGs müssen sich der aus der KI-VO ergebenden Pflichten bewusst sein, wie bei der Verwendung von vertraulichen Daten.

2 Methoden und Materialien

2.0.1 Werkzeuge

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

Ollama

Ollama 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 [8].

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. Desto näher zwei Wörter im Raum beieinander sind desto ähnlicher sind sie.

Vektordatenbank

ChromaDB 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 [1].

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 den vorherigen Tools unterscheidet, ist, dass keine „reference answer“ benötigt wird. RAGAS ist beliebt, da es sich gut mit vielen Tools integriert [3].

Langchain

TODO

2.0.2 Daten

Da 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. Die Dokumente stammen aus Unterlagen eines Einzelunternehmens, welches vereinfachte CMS-Webseiten für Grundschulen entwickelt hat. Die Unternehmung wird nicht mehr aktiv verfolgt und die 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 eigenen Unternehmungen ließen sich 73 nutzbare Dokumente finden. Neben den Dokumenten, welche Businesspläne, Finanzpläne, aber auch Elternbriefe umfassen, gibt es den dazugehörigen Code. Für die zehn Dokumente wurden nur „originale“ Dokumente genutzt. Um von 73 gegebenen Dokumenten auf 100 Dokumente zu kommen, wurden mithilfe von LLMs 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. Für die 400 Dokumente wurde der Code des realen Produktes mit einbezogen. Dieser besteht aus drei Projekten: 1. die Webseite, die öffentlich zugänglich ist, 2. dem Admin-Bereich und 3. dem Backend.

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

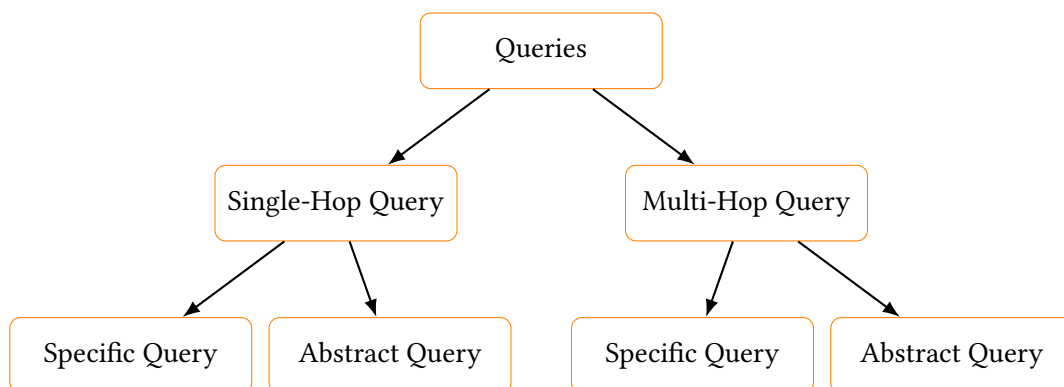
1. **10 Dokumente:** Ein einzelner Anwendungsfall – das RAG-System wird nur temporär genutzt und danach verworfen.

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.

2.0.3 Fragebögen

Fragearten

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



Diese verschiedenen Fragearten 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 [17] 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?“

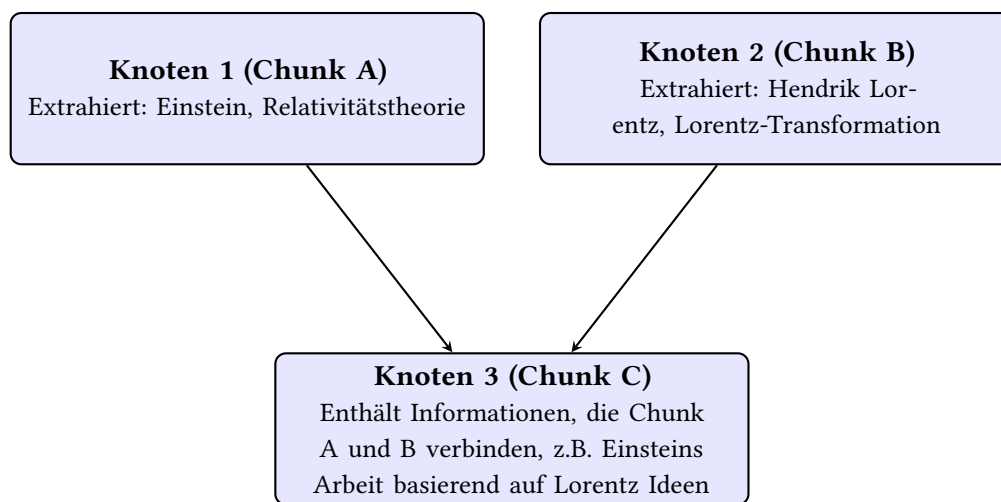
Bei Multi-Hop-Queries handelt es sich um Fragen, die mehr als eine Wissensabfrage benötigen. Die Frage „Welche Wissenschaftler haben Einsteins Relativitätstheorie beeinflusst und welche Theorie haben sie vorgeschlagen?“ benötigt erst eine Abfrage, um die Wissenschaftler herauszufinden, und dann weitere, 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.

Wissensgraph

Für die eben erwähnten Multi-Hop-Queries müssen aus den gegebenen Dokumenten Themen, welche 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 (Vergleich mit Wikipedia-Links in Artikeln).



Für die Daten aus den Versuchen mit 100 Dokumenten hat RAGAS 27 Themen identifiziert. Unter anderem waren dort folgende Themen dabei:

Finanzmanagement, Bildungsprojekt Digitalisierung, Projektmanagement und Planung, Zuwendungsverwaltung, Break-Even-Analyse, Finanzplanung und Investitionen, Finanzplanung und Liquidität.

2.0.4 Evaluation

2.1 Metriken

In diesem Kapitel geht es um die verschiedenen Metriken, die für die Bewertung von RAG Evaluationstools verwendet werden können. Metriken sind das Herzstück der Bewertung von RAGs, da sie die Qualität des RAGs bewerten und somit die Entwicklung und den Fortschritt des RAGs messen.

2.1.1 Retrieval Augmented Generation

Diese Metriken basieren auf Faktenextraktion, mithilfe welcher sich dann Bewertungen berechnen lassen. Für die Extraktion der Fakten wird häufig ein LLM verwendet welcher als Richter fungiert.

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. Die Präzision@k ist das Verhältnis der Anzahl relevanter Textabschnitte auf Rang k zur Gesamtanzahl der Textabschnitte auf Rang k. (eigene Übersetzung nach [12])

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

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 [13])

Wenn es eine gute 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.

Context Entities Recall

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

Die ContextEntityRecall-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 [11])

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

Noise Sensitivity

NoiseSensitivity 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 [15])

Diese Metrik ist eine der wichtigsten Metriken in dieser Arbeit da sie die Richtigkeit der Antworten und damit die Qualität des RAGs bewertet.

Response Relevancy

Die ResponseRelevancy-Metrik misst, wie relevant eine Antwort im 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 [18])

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 zu der Frage sind.

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 [14])

Multimodal Faithfulness/Multimodal Relevance

Da sich diese Metriken mit mehr als textuellen Daten befassen, werden diese nicht im Rahmen dieser Arbeit betrachtet.

2.1.2 Nvidia Metrics

Diese Metriken sind subjektiver Art und benutzen wieder eine LLM, um die Bewertung zu treffen. 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 "LLM-as-a-judge" Prompts, die jeweils eine Bewertung (0, 2 oder 4) zurückgeben. 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 [16])

Das LLM bewertet die Antwort mit der Referenz und auch die Referenz mit der Antwort. Hat Vorteile gegenüber der Answer Correctness, da es weniger Aufrufe mit weniger Tokens an LLM braucht. Es werden im Vergleich zur Answer Correctness auch robustere Bewertungen getroffen, bietet jedoch weniger Einblicke in die Bewertung. Diese Metrik wird im Rahmen dieser Arbeit betrachtet auch um einen Vergleich zu anderen Metriken zu haben.

Context Relevance

Diese Metrik ist sehr ähnlich zur Context Precision, als Alternative und um einen Vergleich zu haben wird diese im Rahmen dieser Arbeit betrachtet, auch wenn sie keine direkte Aussage über das zu bewertende LLM macht.

Response Groundedness

Wenn die Answer Accuracy eine gute Bewertung liefert, ist die Response Groundedness eine gute Bewertung für die Faktualität der Antwort. Diese Logik ist ähnlich zur Kombination von Context Relevancy und Context Precision. Hier wird es in den Versuchen interessant zu vergleichen wie diese Metriken zusammenhängen.

2.1.3 Natural Language Comparison

Factual Correctness

Diese Metriken basieren zu Teilen auf der Wahrheitsmatrix (Confusion matrix), welche die vier Kategorien True Positive, False Positive, False Negative und True Negative definiert.[21] Aus dieser Matrix lassen sich dann precision, recall und f1 score berechnen.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \quad (2.2)$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \quad (2.3)$$

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (2.4)$$

[21]

Semantic Similarity

This metric uses embeddings to calculate the semantic similarity between the answer and the reference. TODO: should this be used?

2.1.4 Non LLM String Similarity

Wie der Name schon sagt, wird die String Similarity ohne LLM berechnet. Diese Metriken sind relative einfache Metriken und werden im Rahmen dieser Arbeit keine große Rolle spielen, jedoch als Vergleich zu anderen Metriken dienen.

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.

ROUGE Score

Mithilfe von n-gram recall, precision, und dem F1 score wird die Ähnlichkeit zwischen der Antwort und der Referenz berechnet.

String Presence

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

Exact Match

Eine noch einfachere Metrik, die nur prüft ob die Antwort exakt der Referenz entspricht. Diese ist für einzelne Wörter sinnvoll.

2.1.5 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
- Simple Criteria Scoring
- Rubrics based Scoring
- Instance Specific Rubrics Scoring

2.1.6 Andere Metriken

Summarization

Anzahl der richtig beantworteten Fragen geteilt durch die Anzahl der Fragen. Dies ist eine sehr einfache und oberflächliche Metrik.

2.1.7 Irrelevante Metriken

SQL

SQL spezifische Metriken, welche nicht im Rahmen dieser Arbeit betrachtet werden.

Agents or Tool use cases

Metriken zum Bewerten des Einsatzes von Agenten oder Tools, dies liegt ebenso außerhalb des Themas dieser Arbeit. <https://docs.ragas.io/en/stable/concepts/metrics/>

Diese Metrik wird Teil dieser Arbeit sein, da sie in gewissen Nutzungsfällen, wie z.B. stark Fakten basierte Fragen, eine gute Bewertung liefern kann.

3 Ähnliche Arbeiten

3.1 RAG Evaluation: Assessing the Usefulness of Ragas

https://tech.beatrust.com/entry/2024/05/02/RAG_Evaluation%3A_Assessing_the_Usefulness_of_Ragas

Das Team von Beatrust hat im Februar 2024 eine Reihe zu RAGs veröffentlicht. Es werden unter anderem die Notwendigkeit und auch die einzelnen Metriken von RAGAS erklärt. Im dritten Artikel dieser Reihe machen sie ein Versuch um die Nützlichkeit von RAGAS zu untersuchen. Der Versuch besteht aus 50 Fragen aus einem Interessensfeld des Authors, diese wurden vom einem RAG mit GPT-4 und einem mit GPT-3.5-turbo beantwortet und dann sowohl von RAGAS als auch von ihm bewertet. Der Author kommt zu dem Ergebniss, dass RAGAS geeignet ist um RAGs zu bewerten und besser ist als die Bewertung von Langchain. Es wird jedoch angemerkt, dass der Author eine höhere Übereinstimmung mit seinen Ergebnissen erwartet hätte.

<https://www.qed42.com/insights/simplifying-rag-evaluation-with-ragas>

<https://medium.aiplanet.com/evaluate-rag-pipeline-using-ragas-fbdd8dd466c1>

<https://arxiv.org/pdf/2309.01431>

3.2 RAG-Bewertungsprozess

Das Flussdiagramm veranschaulicht die drei Hauptphasen des RAG-Bewertungsprozesses:

1. Dokumentenverarbeitung

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





2. Erstellung des Testsets

- Verwendet LLM zur Generierung von Fragen
- Erstellt Testsets mit Fragen und Referenzantworten

3. Bewertungsprozess

- Verwendet das generierte Testset
- Ruft Kontext aus ChromaDB ab
- Bewertet Modellantworten mit LLM als Richter
- Generiert umfassende 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 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.

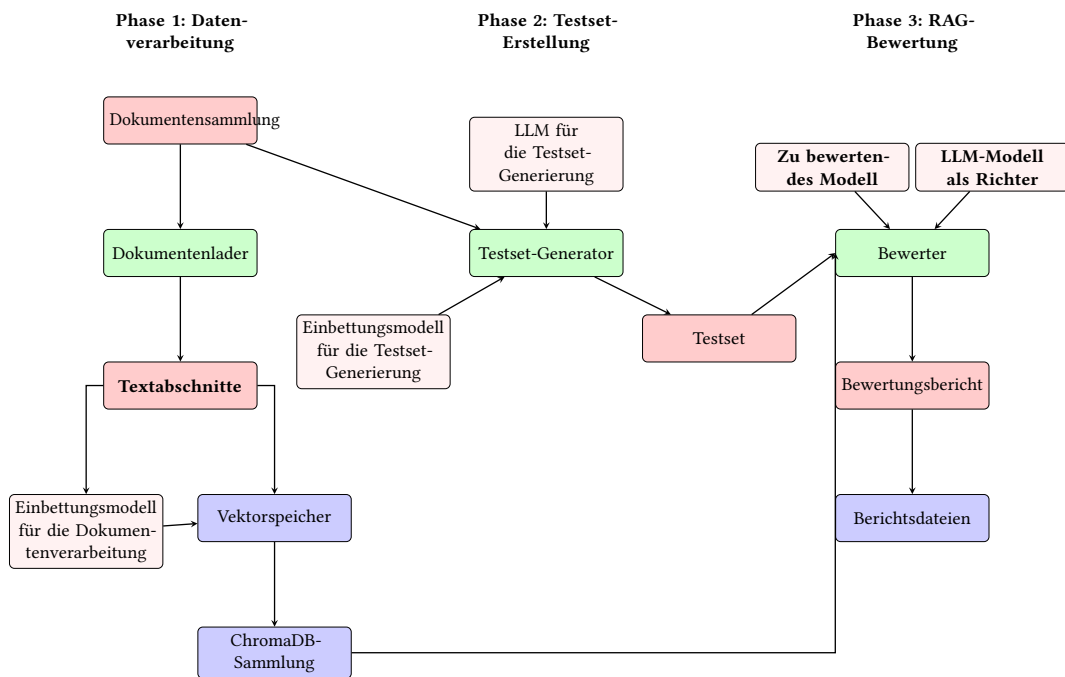


Abbildung 3.1: Flussdiagramm des RAG-Bewertungsprozesses, das die Interaktion zwischen verschiedenen Komponenten und Modellen zeigt. Spezifische Modellnamen (z.B. gpt-4-turbo, text-embedding-3-large) sind im Haupttext beschrieben.

4 Versuche

4.1 Versuchsplan

Um systematisch zu bewerten, ob RAG-Bewertungstools für den Einsatz in kleinen und mittleren Unternehmen (KMU)¹ bereit sind, sind umfassende Versuche erforderlich. Der folgende Versuchsplan skizziert die wichtigsten Variablen, die Methodik und die Bewertungskriterien.

4.1.1 Forschungsfragen

Der Versuch wird die folgenden zentralen Forschungsfragen behandeln:

1. Sind aktuelle RAG-Bewertungsframeworks in Bezug auf Kosten, Komplexität und Ressourcenanforderungen für den Einsatz in kleinen Unternehmen geeignet?
2. Wie beeinflussen verschiedene Dokumententypen und Datenvolumina die Qualität von Abruf und Generierung?
3. Wie zuverlässig und konsistent sind die verfügbaren Bewertungsmetriken zur Beurteilung der RAG-Leistung?
4. Was ist das optimale Gleichgewicht zwischen Kosten, Leistung und Implementierungskomplexität für jeden Anwendungsfall in kleinen Unternehmen?

—

4.1.2 Variablen in den Versuchen

Dokumententypen

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

- PDF (.pdf)

¹2.

- Klartext (.txt)
- Word-Dokumente (.docx, .doc)
- Excel-Tabellen (.xlsx, .xls)
- CSV-Dateien (.csv)
- E-Mails (.eml)
- PowerPoint-Präsentationen (.pptx, .ppt)

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

Modelle zur Bewertung

Mehrere Modelle werden bewertet, die verschiedene Kostenschichten und Fähigkeiten repräsentieren. Hierbei ist es wichtig zu überlegen, welche Optionen für kleine Unternehmen 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.

Bewertungsmetriken

Während des Versuchs werden neben der menschlichen Bewertung zwei Frameworks zur Bewertung verwendet. Giskard und RAGAS werden die später beschriebenen Metriken generieren, die später verglichen und bewertet werden können. Die menschliche Bewertung wird als subjektives Maß verwendet, um die Ergebnisse der anderen beiden 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 kleinen Unternehmen sind, mit klaren 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. Hier benutzt der Author Embeddings von OpenAI sowie Open-Source-Embeddings von nomic.ai.

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

Tabelle 4.1: Kombinationen aus Dokumentenanzahl und Embedding-Modell für die Versuche (X = Kombination wird getestet)

4.2.2 Testset-Generierung

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

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

Tabelle 4.2: Kombinationen aus Dokumentenanzahl und Testset-Größe

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

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

Tabelle 4.3: Übersicht aller 24 zu generierenden Bewertungsberichte mit Abkürzungen und Wiederholungen

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

5 Ergebnisse und Diskussionen

5.1 Ergebnisse aus den Versuchen

5.1.1 Generierte Fragebögen

Da insgesamt 1.290 Fragen generiert wurden, lassen sich diese aufgrund des zeitlichen Aufwandes nicht alle bewerten. Aus jedem werden stichprobenartig 10 Fragen ausgesucht und überprüft, wie sinnvoll diese sind.

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 ist aufgrund von technischen Problemen oder ungültigen Antworten seitens DeepSeek nicht generiert worden.
- 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 also 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 auch nicht generiert.
- Dieses Mal sind jedoch nur 4 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.

Angefragt	Generiert	Irrelevant	Verlust	Fehlerquote
15	11	0	4	27 %
30	27	7	3	33 %
50	40	0	10	20 %
100	88	4	12	16 %
150	137	49	13	41 %
300	267	101	33	45 %

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

OpenAI

Angefragt	Generiert	Irrelevant	Verlust	Fehlerquote
15	12	0	3	20%
30	30	1	0	3%
50	48	0	2	4%
100	95	1	5	6%
150	150	8	0	5%
300	300	16	0	5%

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

—

5.1.2 Manuelle Auswertung der Fragebögen

DeepSeek

Bei der manuellen Sichtung der Testsets wurden weitere Fehler entdeckt.

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.„

Bei dieser Frage hat das LLM verdreht, wer bezahlt, und fragt, wie viel die Schulen weniger **verdienen** und nicht, wie viel sie weniger **bezahlen**.

Frage: „Hallo! Ich bin Schulleiter/in und überlege, ob wir als Pilotenschule bei Develop 4 Future 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.,„

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, oder auch, wie viel Prozent Erreichbarkeit garantiert wird.

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

Es gab auch Fragen, welche sich vom Kontext verwirren ließen. 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.,„

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

Tabelle 5.3: Anzahl fraglicher Fragen pro Testset und Gesamtübersicht für Ollama

Die **Fehlerquote** von 45 % zeigt, dass die Fragen, die DeepSeek generiert, nicht einfach eingesetzt werden können. Es sind hier eindeutige Unterschiede im Vergleich zu von Menschen generierten Fragen erkennbar.

OpenAI

Bei ChatGPT wurden auch Mängel bei der manuellen Überprüfung festgestellt. 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.“

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.

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?“,

Testset	Fragliche Fragen
OpenAI – 10 Dok (15 Fragen)	0
OpenAI – 10 Dok (30 Fragen)	3
Summe 10 Dok: 3 / 20 = 15%	
OpenAI – 100 Dok (50 Fragen)	3
OpenAI – 100 Dok (100 Fragen)	1
Summe 100 Dok: 4 / 20 = 20%	
OpenAI – 400 Dok (150 Fragen)	5
OpenAI – 400 Dok (300 Fragen)	7
Summe 400 Dok: 12 / 20 = 60%	
Gesamt (OpenAI): 19 / 60 = 31,67%	

Tabelle 5.4: Anzahl fraglicher Fragen pro Testset und Gesamtübersicht für OpenAI

Wenn wir 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 beachtliche Menge!

Bei einem Vergleich mit einem von Menschen erstellten Fragebogen sind hier jedoch deutliche Unterschiede, was die Qualität der Fragen betrifft.

—

5.1.3 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.

Manuelle Auswertung DeepSeek

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

Auch die nicht bewerteten Metriken sind mit bis zu 83 % fast unbrauchbar. Dies liegt wieder daran, dass das LLM zu lange zum Antworten braucht oder eine ungültige Antwort geliefert hat.

Metrik	Richtig	Falsch	Nicht bewertet
Richtige Frage	34 (56.7%)	26 (43.3%)	–
Gültige Antwort	22 (36.7%)	38 (63.3%)	–
context_precision	10 (16.7%)	–	50 (83.3%)
faithfulness	11 (18.6%)	2 (3.4%)	47 (78.0%)
context_recall	60 (100%)	–	–
answer_relevancy	43 (71.7%)	15 (25.0%)	2 (3.3%)
answer_relevancy sollte höher sein	4 (100%)	–	–

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

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.

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%)

Tabelle 5.6: Verteilung der Bewertungen für OpenAI (gesamt) mit Prozentangaben

GPT-4 schneidet deutlich besser als DeepSeek ab, 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.

Probleme mit 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.

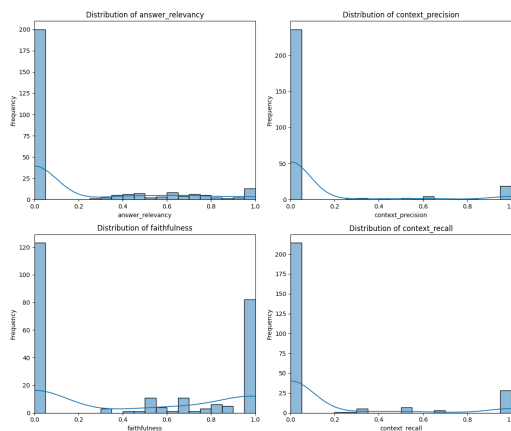


Abbildung 5.1: DeepSeek Ergebnis für 300 Fragen (mit Code-Dokumenten)

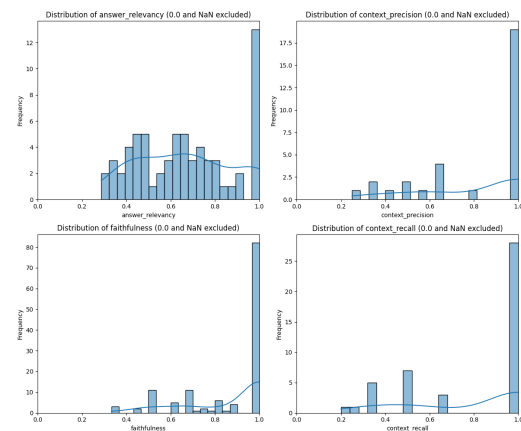


Abbildung 5.2: ChatGPT Ergebnis für 300 Fragen (mit Code-Dokumenten)

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.

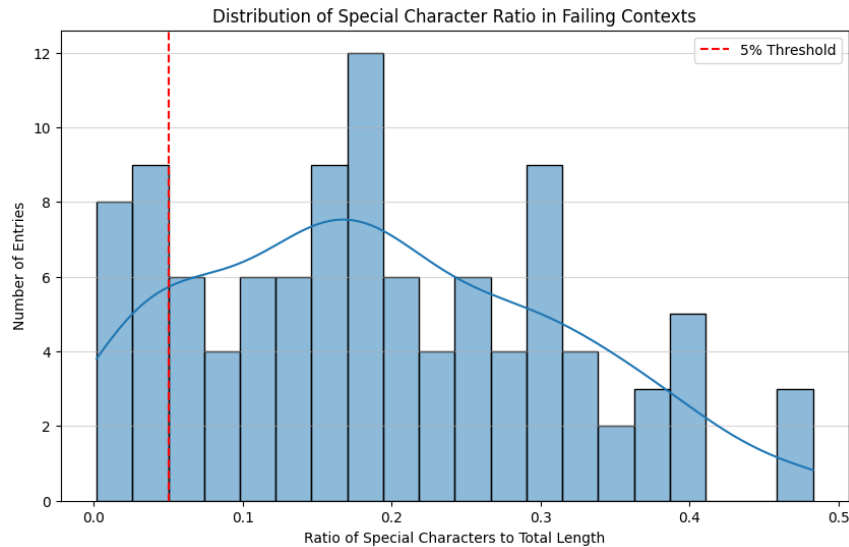


Abbildung 5.3: Abweichungen des Faithfulness Scores bei Code-Dokumenten.

Ein großer Teil der Fragen hat sich also durch Dokumente mit minderwertiger Qualität, bezogen auf mögliche Fragestellungen, verwirren lassen. Es zeigt sich wieder einmal, dass die Qualität der Daten eine entscheidende Rolle spielt! Wenn wir uns jetzt die Ergebnisse ohne Dokumente, welche Code enthalten, ansehen, sehen wir, dass die 0.0-Bewertungen bei DeepSeek deutlich zurückgehen, aber wie zu erwarten ist, ist OpenAI's ChatGPT-4 immer noch deutlich besser.

—

5.1.4 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 um die Unterschiede pro Durchlauf für das Modell festzustellen und nicht die Modelle miteinander zu vergleichen.

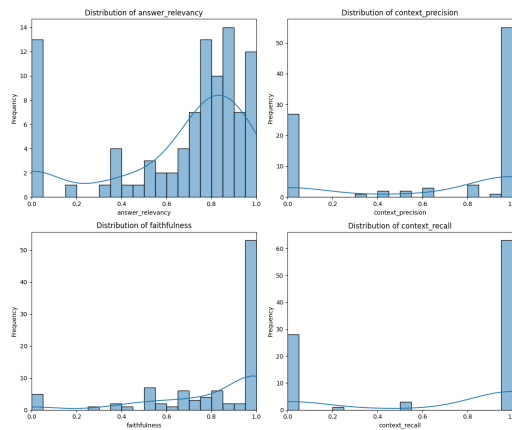


Abbildung 5.4: DeepSeek Ergebnis für 100 Fragen (ohne Code-Dokumente)

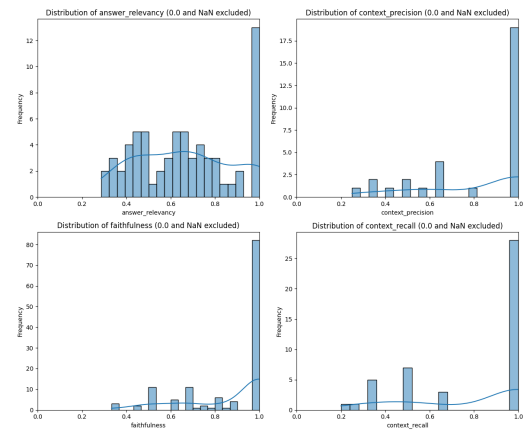


Abbildung 5.5: ChatGPT Ergebnis für 100 Fragen (ohne Code-Dokumente)



Abbildung 5.6: Bewertung der vier Durchläufe mit DeepSeek

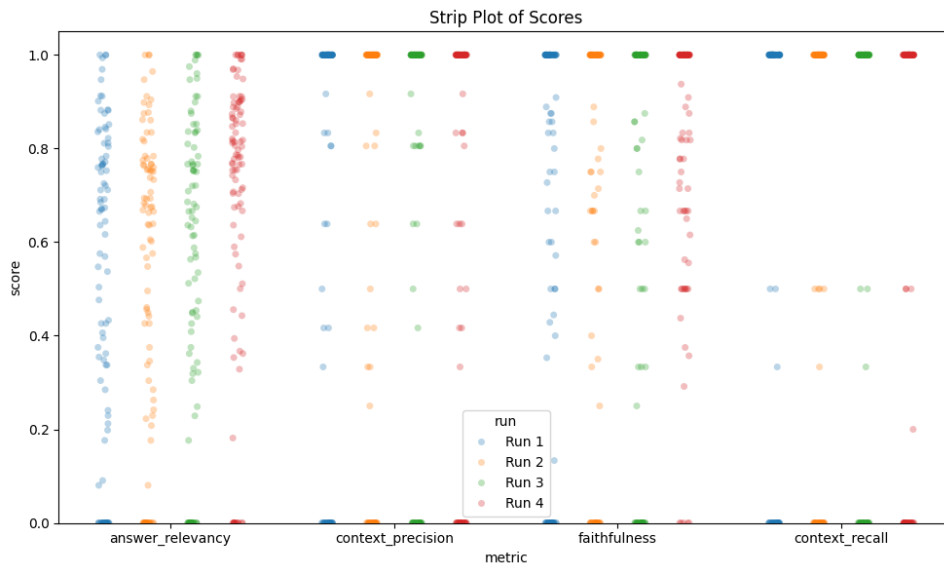


Abbildung 5.7: Bewertung der vier Durchläufe mit GPT-4

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

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

Beim Betrachten der Strip Plots lässt sich gut sehen, dass die Verteilung der Werte pro Durchlauf sehr ähnlich ist und keine großen Abweichungen erkennbar sind.

Beim Betrachten des Durchschnitts und der Standardabweichung 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** 3,1 %. Dies liegt für 4 Durchläufe im Rahmen, bedeutet aber auch, dass dies beim Einrichten einer automatischen Pipeline berücksichtigt werden sollte.

Metrik	Mean 1	Mean 2	Mean 3	Mean 4	Std 1	Std 2	Std 3	Std 4
Answer Relevancy	0.490	0.493	0.666	0.681	0.340	0.351	0.315	0.314
Faithfulness	0.632	0.613	0.815	0.800	0.434	0.444	0.273	0.295
Context Precision	0.669	0.684	0.666	0.670	0.445	0.445	0.444	0.445
Context Recall	0.663	0.667	0.681	0.671	0.461	0.466	0.458	0.456

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

5.1.5 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 GPT-4.1 verwendet.

Context Precision & Recall

Beide Metriken wurden 50 Mal bewertet und haben sich wie bei DeepSeek in der Gesamtbewertung als sehr stabil herausgestellt.

Answer Relevancy

Hier wurden minimale Abweichungen festgestellt; diese belaufen sich aber auf die zweite Nachkommastelle in der Prozentangabe und sind daher vernachlässigbar.

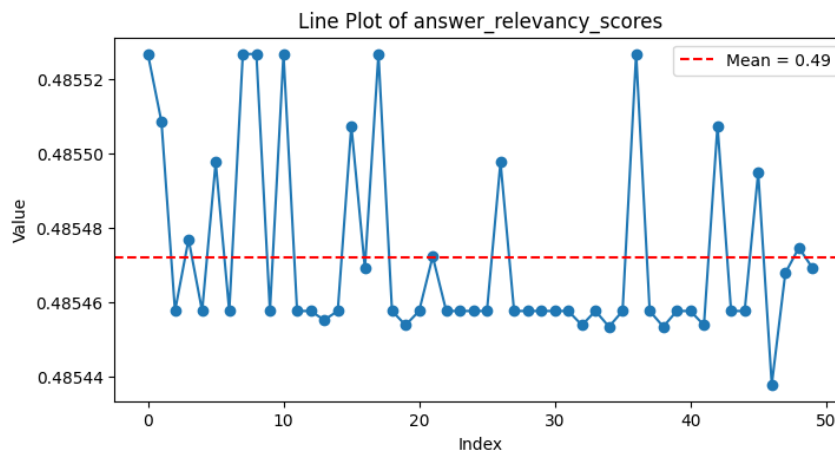


Abbildung 5.8: Abweichungen des Answer Relevancy Scores.

Faithfulness

Bei der Faithfulness sieht dies schon etwas anders aus. 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.

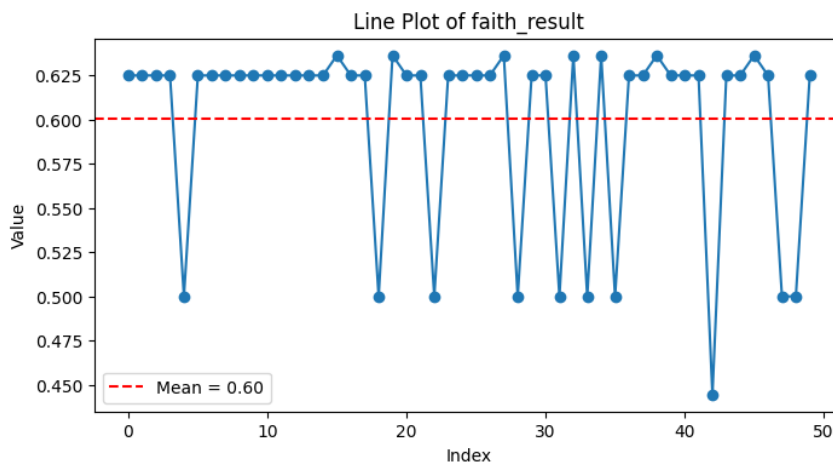


Abbildung 5.9: Abweichungen des Faithfulness Scores.

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

Ausführungszeiten DeepSeek

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. Bei dem Testset mit 15 Fragen sind wir von 2 Minuten mit maximal 16 gleichzeitigen Anfragen auf 7 Minuten bei maximal einer gleichzeitigen Anfrage. Mit diesen Zahlen lässt sich annehmen, dass der Versuch mit 300 Fragen ohne Rate Limit seitens OpenAI sicherlich unter einer Stunde geschafft werden könnte.

Ausführungszeiten OpenAI

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

Anzahl	Dauer (hh:mm)
15	00:02
30	00:03
50	00:04
100	00:07
150	01:37
300	02:30

Tabelle 5.9: Dauer der Evaluation pro Dokumentenanzahl mit DeepSeek

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 %.

Anzahl	Dauer (hh:mm)
15	00:41
30	01:03
50	01:35
100	03:41
150	05:20
300	17:29

Tabelle 5.10: Dauer der Evaluation pro Dokumentenanzahl mit OpenAI

5.1.6 Kostenberechnung

Die Bewertung des RAGs mit den 300 Fragen 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: 17 h
- Stromkosten: $0,12\text{€}/\text{h} \Rightarrow 17\text{ h} \times 0,12\text{€}/\text{h} = \mathbf{2,04\text{€}}$
- 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€(Strom) + 10,00€(Abschreibung) = **12,04€**
- Gesamtlaufzeit (720 Runs): $720 \times 17 \text{ h} = 12.240 \text{ h} \approx \mathbf{1 \text{ Jahr, 4 Monate, 10 Tage}}$

5.2 Abhängigkeit der Metriken untereinander

Dadurch, dass die Metriken für den Kontext einen früheren Schritt in der Abfrage an ein RAG bewerten als die für die Antwort, ergeben sich gewisse Abhängigkeiten.

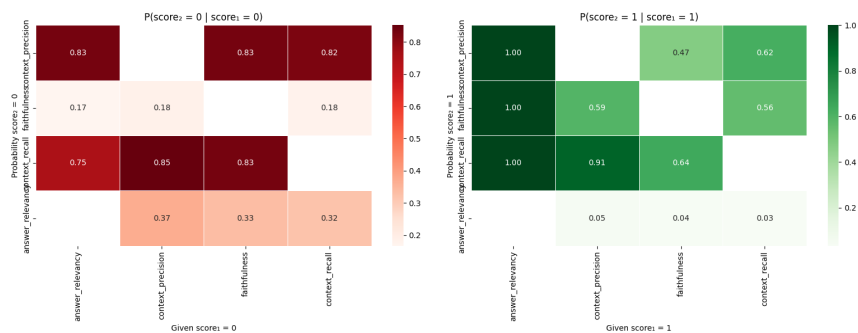


Abbildung 5.10: Abhängigkeit der Metriken voneinander (OpenAI, 300 Fragen, 400 Dokumente)

Wenn die Metriken für den Kontext (context_precision und context_recall) eine Bewertung von 0 haben, ist die Wahrscheinlichkeit, dass die anderen Metriken auch 0 sind, relativ hoch. In diesem konkreten Beispiel werden die Abhängigkeiten des OpenAI RAGs für die 300 Fragen gezeigt. Es lässt sich sehen, dass die faithfulness deutlich weniger von den Kontext-Metriken abhängt.

5.3 Identifikation von Interessenten

6 Zusammenfassungen

6.1 Benutzung von RAGAS

Dank der vielen Integrationen hat sich die Verwendung von RAGAS als einfach herausgestellt.

6.2 Testsets

Die Generierung von Testsets ist eines der Alleinstellungsmerkmale von RAGAS. Die Generierung der Testsets hat sich aus softwaretechnischer Sicht als unkompliziert erwiesen. Es gab zu den wichtigen Themen ausreichend Dokumentation und Beispiele.

Es ist mit RAGAS möglich, Testsets 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 Testsets mit DeepSeek kam es alleine 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 Testsets 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.

Die Qualität des Testsets ist entscheidend, da sich hier entstandene Fehler weiter bis in die Bewertung durchziehen und eine korrekte Bewertung des eigentlichen RAGs verzerren!

Die bei den Versuchen generierten Testsets lassen Zweifel an einer zuverlässigen und hochwertigen Generierung von Fragen aufkommen.

6.3 Bewertung

Auch das Generieren von Bewertungen hat sich mithilfe von RAGAS als einfach umzusetzen 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 Testsets zurückführen.

Bei den Metriken lässt sich sagen, dass die Metriken zum Kontext (`recall` und `precision`) gut abschneiden. Die `faithfulness` zeigt eine erhöhte Abweichung zu der 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.

6.4 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 Testsets 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 ließen sich sehr gut sehen.

Insgesamt 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 genauer der Anwendungsfall betrachtet werden.

6.5 Zukunftsausblick

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.

6.6 Reflexion der Arbeit

Literatur

- [1] Try Chroma. *Try Chroma*. Accessed: 2025. 2025. URL: <https://www.trychroma.com/>.
- [2] 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.
- [3] ExplodingGradients. *Integrations – How-to guide*. Zugegriffen am 18. Juni 2025. Mai 2025. URL: <https://docs.ragas.io/en/latest/howtos/integrations/>.
- [4] Hugging Face. *Pleias-RAG-1B*. Accessed: 2025. 2025. URL: <https://huggingface.co/PleIAS/Pleias-RAG-1B>.
- [5] 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/>.
- [6] 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. Google DeepMind Technical Report. 2024.
- [7] Thorsten Honroth, Julien Siebert und Patricia Kelbert. *Retrieval Augmented Generation (RAG): Chatten mit den eigenen Daten*. Zugriff am 7. Februar 2025. Mai 2024. URL: <https://www.iese.fraunhofer.de/blog/retrieval-augmented-generation-rag/>.
- [8] Ollama. *Ollama*. Accessed: 2025. 2025. URL: <https://ollama.com/>.
- [9] 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>.
- [10] Ofir Press u. a. „Measuring Faithfulness in Chain-of-Thought Reasoning“. In: *arXiv preprint arXiv:2211.08411* (2022). URL: <https://arxiv.org/abs/2211.08411>.

-
- [11] Ragas. *Context Entities Recall*. Accessed: 2024. 2024. URL: https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/context_entities_recall/.
 - [12] Ragas. *Context Precision*. Accessed: 2024. 2024. URL: https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/context_precision/.
 - [13] Ragas. *Context Recall*. Accessed: 2024. 2024. URL: https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/context_recall/.
 - [14] Ragas. *Faithfulness*. Accessed: 2024. 2024. URL: https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/faithfulness/.
 - [15] Ragas. *Noise Sensitivity*. Accessed: 2024. 2024. URL: https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/noise_sensitivity/.
 - [16] Ragas. *Nvidia Metrics*. Accessed: 2024. 2024. URL: https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/nvidia_metrics/.
 - [17] Ragas. *Query types in RAG*. Accessed: 2024. 2024. URL: https://docs.ragas.io/en/stable/concepts/test_data_generation/rag/#query-types-in-rag.
 - [18] Ragas. *Response Relevancy*. Accessed: 2024. 2024. URL: https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/answer_relevance/.
 - [19] Ammar Shaikh u. a. *CBEval: A framework for evaluating and interpreting cognitive biases in LLMs*. 2024. DOI: 10.48550/arXiv.2412.03605. arXiv: 2412.03605 [cs.CL]. URL: <https://arxiv.org/abs/2412.03605>.
 - [20] Doit Software. *ChatGPT Statistiken*. Accessed: 2025. 2025. URL: <https://doit.software/de/blog/chatgpt-statistiken#screen5>.
 - [21] Wikipedia. *Confusion matrix*. Accessed: 2024. 2024. URL: https://en.wikipedia.org/wiki/Confusion_matrix.
 - [22] Jingfeng Yang u. a. *Large Language Models are not Fair Evaluators*. 2023. arXiv: 2305.17926 [cs.CL]. URL: <https://arxiv.org/abs/2305.17926>.

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