
Empirische Analyse von RAG Evaluation Tools für Betriebliche Abläufe

Bachelorarbeit zur Erlangung des akademischen Grades
Bachelor of Arts/Engineering/Science
im Studiengang <Name des Studiengangs>
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
Zweitgutachter*in: Prof. Dr. Dietlind Zühlke

Ort, TT.MM.JJJJ

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	V
Abbildungsverzeichnis	VI
1 Einleitung	1
1.1 Was ist ein RAG	1
1.1.1 Warum ein RAG	1
1.1.2 Kompetenz	2
1.1.3 Art der Daten	2
1.1.4 Budget	2
1.2 Objektive Beurteilung von RAGs	2
1.3 Darstellung des Themas und der Forschungsfragen	3
1.4 Praxistauglichkeit und Herausforderungen	3
1.5 Softwaretechnische Fragestellungen	3
2 Methoden und Materialien	5
2.0.1 Tools	5
2.0.2 Daten	6
2.0.3 Evaluation	6
2.0.4 Datenbank	6
3 Ähnliche Arbeiten	7
3.0.1 RAG Evaluation: Assessing the Usefulness of Ragas	7
3.1 RAG-Bewertungsprozess	8
4 Experimente	10
4.1 Experiment plan	10
4.1.1 Forschungsfragen	10
4.1.2 Experimentelle Variablen	10
4.1.3 Kosten- und Zeitanalyse	12
4.1.4 Experimentelles Protokoll	12
4.1.5 Bewertungskriterien für die Geschäftstauglichkeit	13
4.2 Konkretisierung der Experimente	14
4.2.1 Dokumentenverarbeitung	14

4.2.2	Testset-Generierung	14
4.2.3	Bewertung	15
5	Ergebnisse und Diskussionen	17
5.1	Antworten aus den Experimenten	17
5.2	Identifikation von Interessenten	17
6	Zusammenfassungen	18
6.1	Zukunftsausblick	18
6.2	Diskussion der Ergebnisse mit potenziellen Anwendern	18
6.3	Reflektieren der Arbeit	18
7	Metriken	19
7.1	Retrieval Augmented Generation	19
7.1.1	Context Precision	19
7.1.2	Context Recall	20
7.1.3	Context Entities Recall	20
7.1.4	Noise Sensitivity	20
7.1.5	Response Relevancy	21
7.1.6	Faithfulness	21
7.1.7	Multimodal Faithfulness/Multimodal Relevance	22
7.2	Nvidia Metrics	22
7.2.1	Answer Accuracy	22
7.2.2	Context Relevance	22
7.2.3	Response Groundedness	23
7.3	Natural Language Comparison	23
7.3.1	Factual Correctness	23
7.3.2	Semantic Similarity	23
7.4	Non LLM String Similarity	23
7.4.1	BLEU Score	24
7.4.2	ROUGE Score	24
7.4.3	String Presence	24
7.4.4	Exact Match	24
7.5	General purpose	24
7.6	Andere Metriken	25
7.6.1	Summarization	25
7.7	Irrelevante Metriken	25
7.7.1	SQL	25
7.7.2	Agents or Tool use cases	25
	Literatur	26

Tabellenverzeichnis

4.1	Kombinationen aus Dokumentanzahl und Embedding-Modell für die Experimente (X = Kombination wird getestet)	14
4.2	Kombinationen aus Dokumentanzahl und Testset-Größe	15
4.3	Übersicht aller 24 zu generierenden Bewertungsberichte mit Abkürzungen und Wiederholungen	16

Abbildungsverzeichnis

1.1	Struktur eines RAGs, Quelle: [2]	1
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.	9

1 Einleitung

1.1 Was ist ein RAG

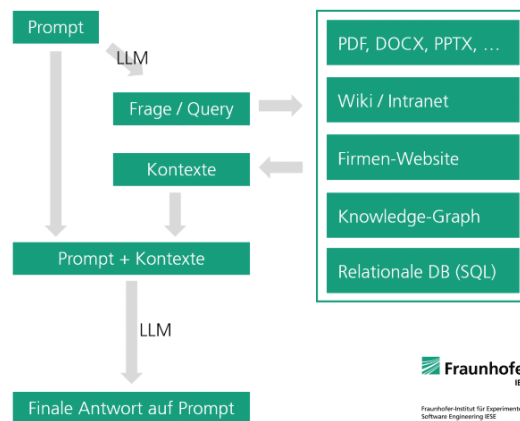


Abbildung 1.1: Struktur eines RAGs, Quelle: [2]

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

[2]

1.1.1 Warum ein RAG

Wenn es darum geht, Wissen abzurufen, haben LLMs zwei Probleme.

Wissen, welches nur selten im Training erwähnt wurde, können selbst die größten LLMs nicht lernen. Das zweite Problem ist, dass LLMs einen gewissen Wissensstand haben und weiter trainiert werden müssen, um die neuesten Informationen zu kennen.

Die Nutzung eines RAGs ist eine der beiden Möglichkeiten, um ein LLM zu verbessern. Die andere Möglichkeit ist das Finetuning des LLMs. Es gibt jedoch wichtige Unterschiede zwischen diesen beiden Methoden.

Es gibt einige Faktoren, welche die Entscheidung beeinflussen können, ob ein RAG oder ein Finetuning besser für den betrieblichen Ablauf geeignet ist.

1.1.2 Kompetenz

Beim Finetuning ist ein gewisses technisches Wissen notwendig, um die Themen Natural Language Processing (NLP), Deep Learning, Modellkonfiguration, Datenaufbereitung und Evaluierung anzuwenden. Der gesamte Prozess des Finetunings ist sehr komplex und erfordert viel Zeit und Ressourcen.

1.1.3 Art der Daten

Sollten die Daten dynamisch sein, ist das RAG die bessere Lösung, da es die Daten schneller und kontinuierlich aktualisieren kann. Der Prozess des Finetunings erstellt immer eine Momentaufnahme, die ein erneutes Training erfordert. Beim Finetuning ist es möglich, dass das Modell Muster erkennt und firmeneigene Begriffe verstehen kann.

1.1.4 Budget

Das Finetuning erfordert teure Rechenzeit auf Hochleistung-GPUs, was das Training eines Modells sehr teuer macht. Das RAG hat dagegen zusätzliche Kosten, die durch das Speichern der Daten in einer Vektordatenbank entstehen.

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 immer wieder neu durchgeführt werden. Menschliche Beurteilungen sind teurer, daher ergibt es Sinn, mithilfe von LLMs die

Beurteilung zu automatisieren. Es gibt bereits Tools wie RAGAS, die versuchen, diesen Prozess unter anderem mithilfe von LLMs zu automatisieren. Diese Tools generieren aus den ihnen gegebenen Daten Fragebögen, die auf eine 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 zum Beispiel die Veröffentlichung blockiert werden.

Sowohl menschliche Bewertungen als auch die reine subjektive Bewertung durch LLMs sind nicht objektiv. Mithilfe von mehreren Techniken kann versucht werden, die Bewertung mithilfe von LLMs objektiv zu machen.

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 werden die beiden Tools RAGAS und Giskard stehen, welche die Bewertung durchführen.

1.4 Praxistauglichkeit und Herausforderungen

Es stellen sich mehrere Herausforderungen für die Bewertung von RAGs durch diese Tools. Die erste sind die Kosten, die bei der Bewertung entstehen. Für die Bewertung muss das neue System, welches getestet wird, aufgesetzt werden. Dies beinhaltet eine eventuelle doppelte Speicherung der Daten und die für das Testen benötigten Aufrufe des LLMs. Neben den Kosten ist auch die Zeit, welche es dauert, die Bewertung durchzuführen, relevant, da die Bewertung schneller durchgeführt werden kann, wenn mehr Ressourcen zur Verfügung stehen. Das System muss auch auf dem neuesten Stand gehalten werden, da sich dieses noch relativ junge Thema schnell entwickelt.

Content filtering

1.5 Softwaretechnische Fragestellungen

In dem Artikel *RAG in der Praxis – Generierung synthetischer Testdatensätze* von Luka Panic [3] treten Fehler beim Generieren der Testfragen auf. 17% ist hier die Fehlerquote. 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 [13] wird gezeigt, dass, wenn ein LLM eine von zwei gegebenen Antworten aussuchen müsste, die erste bevorzugt wurde, selbst wenn die gleiche Frage mit anderer Reihenfolge gestellt wurde. RAGs vergleicht keine Antworten miteinander und daher ist dieser Bias kein direktes Problem für uns. Was jedoch einen Einfluss auf die Bewertung von Antworten haben kann, ist der Bias zu gewissen Nummern. Wie in [11] beschrieben, bevorzugen LLMs bei der Bewertung lieber Zahlen, welche Mehrfache von 5 und 10 sind.

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

Wie in diesem Paper [1] 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 sind 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.

Kann ich hier wirklich Diskussionen von Twitter linken oder mache ich mich dann lächerlich? <https://x.com/agishaun/status/1758561862764122191> <https://x.com/ptsi/status/1758511315646320>

Pitfalls in LLM Assisted Evaluation <https://medium.aiplanet.com/evaluate-rag-pipeline-using-ragas-fbdd8dd466c1>

2 Methoden und Materialien

2.0.1 Tools

Für das RAG selbst und die Bewertung der RAGs werden gewisse Tools benötigt. Diese Tools werden im Folgenden 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 unserem Fall werden die Modelle ollama/nomic-embed-text und ollama/deepseek-r1:32b verwendet.

ChromaDB

ChromaDB ist eine Open-Source Vektordatenbank, die für die Speicherung von Vektordatenbanken verwendet wird. Diese Datenbank wird in unseren Experimenten sowohl für die Open-Source Modelle verwendet, als auch für die Closed-Source Modelle von OpenAI. Damit soll eine einheitliche Grundlage für die Experimente geschaffen werden.

RAGAS

RAGAS ist eine Bibliothek, die Werkzeuge bereitstellt, um die Evaluation von Large Language Model (LLM) Anwendungen zu verbessern. Sie wurde entwickelt, um die Bewertung von LLM-Anwendungen einfach und zuverlässig zu gestalten.¹

RAGAS ist ein Open-Source-Tool und liefert neben dem Tool selber hilfreiche Dokumentation für die Metriken und die Bewertung von RAGs. Für diese Arbeit sind die Funktionen der Testset-Generierung und die damit ermöglichte Bewertung der RAGs relevant. Es gibt noch weitere Funktionen wie die automatische Generation von Interessengruppen, diese sind für diese Arbeit jedoch nicht relevant.

¹<https://docs.ragas.io/en/stable/#frequently-asked-questions>

Was RAGAS besonders macht zu vorherigen Tools ist, dass keine "reference answer" benötigt wird. RAGAS ist beliebt, da es sich gut mit vielen Tools integriert.

Giskard

Giskard ist ein teils Open-Source-Tool, welches die Bewertung von RAGs unterstützt. Der Schwerpunkt von Giskard liegt eher auf der generellen Bewertung von LLMs. Dazu gehören unter anderem Prompt-Injectionen, Halluzinationen und andere Fehler, welche durch die Verwendung von LLMs entstehen können.

2.0.2 Daten

2.0.3 Evaluation

2.0.4 Datenbank

3 Ähnliche Arbeiten

3.0.1 RAG Evaluation: Assessing the Usefulness of Ragas

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

Dieser Blog Post vom 05.02.2024 ist der dritte Beitrag in dieser Reihe. In den vorherigen Beiträgen wurde die Notwendigkeit von RAG Evaluation beschrieben und die Metriken. Dieser Beitrag untersucht mithilfe eines Experiments die Nützlichkeit von RAGAS. Bei diesem Experiment wurde ein eigenes Datenset erstellt.

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

3.1 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 Experimente mit verschiedenen Modellen und Konfigurationen, während ein konsistentes Bewertungsframework beibehalten wird.

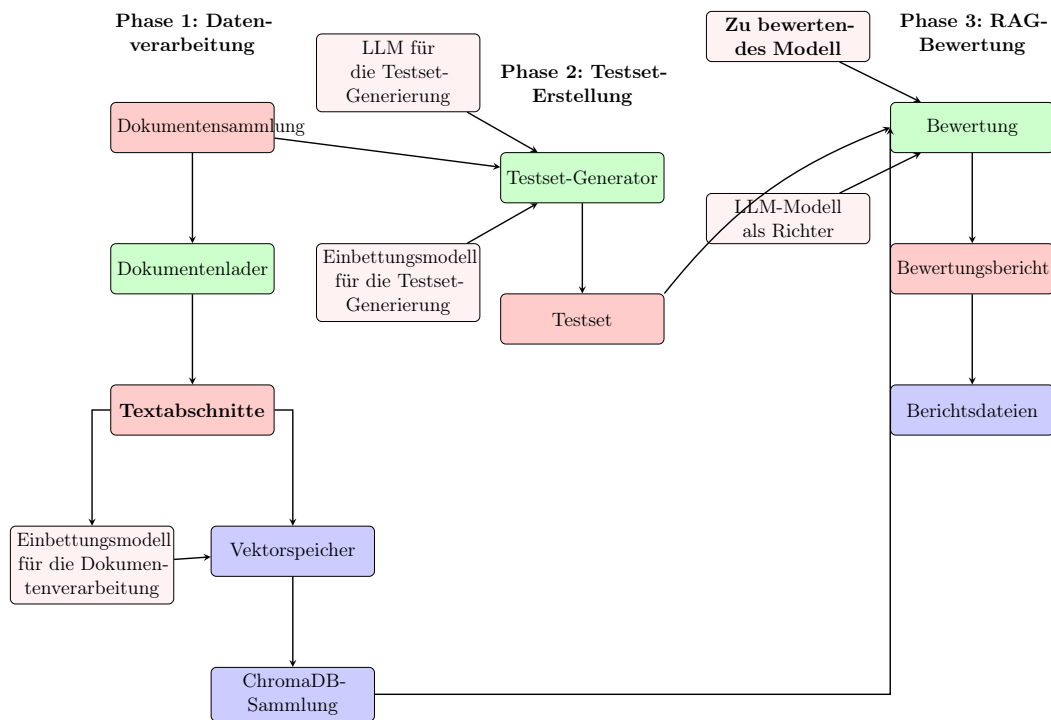


Abbildung 3.1: Flussdiagramm des RAG-Bewertungsprozesses, das die Interaktion zwischen verschiedenen Komponenten und Modellen zeigt. Spezifische Modellnamen (z.B. gpt-4-turbo, text-embedding-3-large) sind im Haupttext beschrieben.

4 Experimente

4.1 Experiment plan

Um systematisch zu bewerten, ob RAG-Bewertungstools für den Einsatz in kleineren Unternehmen bereit sind, ist ein umfassender experimenteller Ansatz erforderlich. Der folgende Experiment plan skizziert die wichtigsten Variablen, die Methodik und die Bewertungskriterien.

4.1.1 Forschungsfragen

Das Experiment 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 Dokumenttypen 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 Experimentelle Variablen

Dokumenttypen

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

- Klartext (.txt)
- PDF-Dokumente mit Text, Tabellen und Bildern
- HTML-Inhalte von Webseiten

- Microsoft Office-Dokumente (.docx, .xlsx)
- JSON und strukturierte Datenformate

Datenvolumen

Die Skalierbarkeit des Systems wird mit unterschiedlichen Datenmengen getestet:

- Klein (10-50 Dokumente, 100 Seiten): Dies könnte pro Anwendungsfall eingerichtet und nach dem Experiment verworfen werden
- Mittel (100-500 Dokumente, 1.000 Seiten): Dies könnte pro Benutzer eingerichtet werden und im Laufe der Zeit wachsen
- Groß (1.000+ Dokumente, 10.000 Seiten): Dies könnte unternehmensweit eingerichtet werden und im Laufe der Zeit wachsen

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, was rechtliche Vorteile bietet, aber auch Nachteile. Die Fähigkeit, sie selbst zu hosten, kann ein Plus, aber auch ein Minus sein, je nach Art des Unternehmens. 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. Sie haben ähnliche Vor- und Nachteile wie die Mittelklasse-API-Modelle. Aus rechtlicher Sicht sind die Hochleistungsmodelle die sicherste Option, da sie bestimmte Dokumentationen veröffentlichen und rechtliche Garantien bieten müssen.

Bewertungsmetriken

Während des Experiments 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

Ob wir dies tun wollen, ist noch nicht klar. RAGAS bietet Kostenberechnung an, aber ich habe es mir noch nicht angesehen.

4.1.4 Experimentelles Protokoll

1. **Dokumentensammlung und -vorbereitung** Sammeln Sie Dokumente in allen oben genannten Zielformaten.
2. **Testset-Generierung** Generieren Sie verschiedene Fragetypen (faktisch, inferentiell, vergleichend) und erstellen Sie Referenzantworten zur Bewertung. Dies geschieht automatisch durch das RAGAS-Framework. Validieren Sie das Testset manuell auf Qualität und Abdeckung. Dies wird an einer Reihe zufälliger Proben durchgeführt.
3. **Systemkonfiguration** Konfigurieren Sie Einbettungsmodelle und Parameter, richten Sie Vektorspeicher mit konsistenten Einstellungen ein und setzen Sie Bewertungsframeworks ein.
4. **Durchführung der Bewertung** Da wir die hochgeladenen Dateien, die generierten Dokumente und das Testset wiederverwenden können, werden diese zuerst erstellt. Dann wird die Bewertungspipeline ausgeführt und die Ergebnisse werden aufgezeichnet.
5. **Analyse und Berichterstattung** Vergleichende Analyse über alle Variablen hinweg, Kosten-Nutzen-Analyse für 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:** Leistet das System konsistent?
- **Skalierbarkeit:** Wie gut bewältigt das System wachsende Datenanforderungen?

Dieser experimentelle Ansatz 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 Experimente

4.2.1 Dokumentenverarbeitung

Embedding-Modell	10	100	1000
openai/text-embedding-3-large	X	X	X
ollama/nomic-embed-text	X	X	X

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

Für die Experimente werden folgende Dokumentmengen und -typen verwendet:

- Für die Experimente mit **10 Dokumenten** werden existierende Dokumente ausgewählt.
- Für die Experimente mit **100** und **1000 Dokumenten** müssen zusätzliche Dokumente generiert werden, vorzugsweise mit einem LLM.

In allen Versuchen werden Dateien der folgenden Typen verwendet:

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

4.2.2 Testset-Generierung

Die Question Synthesizer bleiben in allen Experimenten gleich. Weitere Informationen: https://docs.ragas.io/en/stable/concepts/test_data_generation/rag/

- **SingleHopSpecificQuerySynthesizer** (Gewichtung: 0,5)
- **MultiHopAbstractQuerySynthesizer** (Gewichtung: 0,25)
- **MultiHopSpecificQuerySynthesizer** (Gewichtung: 0,25)

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

Dokumentanzahl	Anzahl Fragen pro Testset	Anzahl Testsets pro Modell
10	15, 30	2
100	50, 100	2
1000	150, 300	2
Summe Testsets pro Modell		6

Tabelle 4.2: Kombinationen aus Dokumentanzahl 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 Experimente (2 Embeddings \times 2 Bewerter \times 6 Testset-Varianten).

Verwendete Abkürzungen in der Tabelle:

- **OAI-E** = openai/text-embedding-3-large
- **OLL-E** = ollama/nomic-embed-text
- **GPT-4** = openai/gpt-4
- **DSK-R** = ollama/deepseek-r1:7b

Experiment	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	4/4
5	OAI-E	1000	150	GPT-4	GPT-4	1
6	OAI-E	1000	300	GPT-4	GPT-4	1
7	OAI-E	10	15	DSK-R	DSK-R	1
8	OAI-E	10	30	DSK-R	DSK-R	1
9	OAI-E	100	50	DSK-R	DSK-R	1
10	OAI-E	100	100	DSK-R	DSK-R	1
11	OAI-E	1000	150	DSK-R	DSK-R	1
12	OAI-E	1000	300	DSK-R	DSK-R	1
13	OLL-E	10	15	GPT-4	GPT-4	1
14	OLL-E	10	30	GPT-4	GPT-4	1
15	OLL-E	100	50	GPT-4	GPT-4	1
16	OLL-E	100	100	GPT-4	GPT-4	1
17	OLL-E	1000	150	GPT-4	GPT-4	1
18	OLL-E	1000	300	GPT-4	GPT-4	1
19	OLL-E	10	15	DSK-R	DSK-R	1
20	OLL-E	10	30	DSK-R	DSK-R	4
21	OLL-E	100	50	DSK-R	DSK-R	1
22	OLL-E	100	100	DSK-R	DSK-R	1
23	OLL-E	1000	150	DSK-R	DSK-R	1
24	OLL-E	1000	300	DSK-R	DSK-R	1

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

5 Ergebnisse und Diskussionen

5.1 Antworten aus den Experimenten

5.2 Identifikation von Interessenten

6 Zusammenfassungen

6.1 Zukunftsausblick

6.2 Diskussion der Ergebnisse mit potenziellen Anwendern

6.3 Reflektieren der Arbeit

7 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.

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.

7.1 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.

7.1.1 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. Die Präzision@k ist das Verhältnis der Anzahl relevanter Textabschnitte auf Rang k zur Gesamtanzahl der Textabschnitte auf Rang k. (eigene Übersetzung nach [5])

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.

7.1.2 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 [6])

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.

7.1.3 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 [4])

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

7.1.4 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 [8])

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

7.1.5 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 [10])

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.

7.1.6 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:

$$\text{Faithfulness Score} = \frac{\text{Anzahl der durch den Kontext gestützten Aussagen in der Antwort}}{\text{Gesamtanzahl der Aussagen in der Antwort}} \quad (7.1)$$

(eigene Übersetzung nach [7])

7.1.7 Multimodal Faithfulness/Multimodal Relevance

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

7.2 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.

7.2.1 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 [9])

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.

7.2.2 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.

7.2.3 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 Experimenten interessant zu vergleichen wie diese Metriken zusammenhängen.

7.3 Natural Language Comparison

7.3.1 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.[12] Aus dieser Matrix lassen sich dann precision, recall und f1 score berechnen.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \quad (7.2)$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \quad (7.3)$$

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (7.4)$$

[12]

7.3.2 Semantic Similarity

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

7.4 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.

7.4.1 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.

7.4.2 ROUGE Score

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

7.4.3 String Presence

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

7.4.4 Exact Match

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

7.5 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

7.6 Andere Metriken

7.6.1 Summarization

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

7.7 Irrelevante Metriken

7.7.1 SQL

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

7.7.2 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.

Literatur

- [1] 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.
- [2] 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/>.
- [3] Luka Panic. *RAG in der Praxis – Generierung synthetischer Testdatensätze*. Abgerufen am 30. Mai 2025. 2024. URL: <https://pixon.co/blog/rag-in-practice-test-set-generation>.
- [4] Ragas. *Context Entities Recall*. Accessed: 2024. 2024. URL: https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/context_entities_recall/.
- [5] Ragas. *Context Precision*. Accessed: 2024. 2024. URL: https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/context_precision/.
- [6] Ragas. *Context Recall*. Accessed: 2024. 2024. URL: https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/context_recall/.
- [7] Ragas. *Faithfulness*. Accessed: 2024. 2024. URL: https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/faithfulness/.
- [8] Ragas. *Noise Sensitivity*. Accessed: 2024. 2024. URL: https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/noise_sensitivity/.
- [9] Ragas. *Nvidia Metrics*. Accessed: 2024. 2024. URL: https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/nvidia_metrics/.
- [10] Ragas. *Response Relevancy*. Accessed: 2024. 2024. URL: https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/answer_relevance/.
- [11] 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>.
- [12] Wikipedia. *Confusion matrix*. Accessed: 2024. 2024. URL: https://en.wikipedia.org/wiki/Confusion_matrix.

- [13] 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