

---

# 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<sub>a</sub>lexander.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 Kompetenzwerkstatt<sup>1</sup>.

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

---

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

## Inhaltsverzeichnis

## Tabellenverzeichnis

## Abbildungsverzeichnis

# 1 Ablauf des Experiments

## 1.1 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)

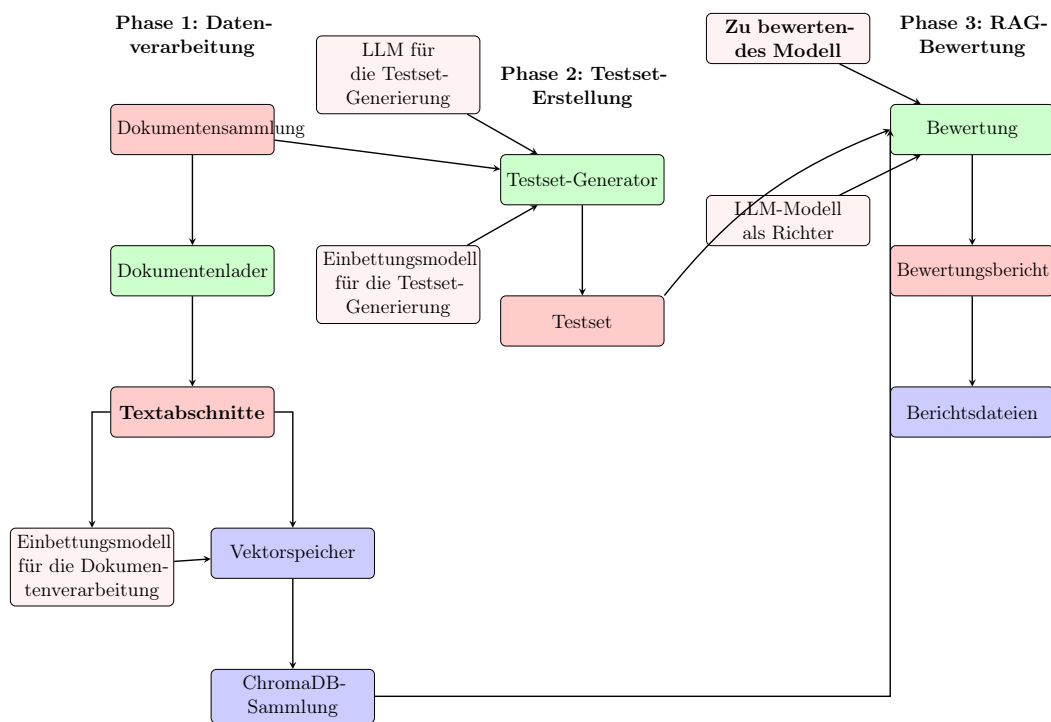


Abbildung 1.1: Flussdiagramm des RAG-Bewertungsprozesses, das die Interaktion zwischen verschiedenen Komponenten und Modellen zeigt. Spezifische Modellnamen (z.B. gpt-4-turbo, text-embedding-3-large) sind im Haupttext beschrieben.

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.

## 1.2 Experimentplan

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

### 1.2.1 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?

### 1.2.2 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.

### 1.2.3 Kosten- und Zeitanalyse

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

### 1.2.4 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.

### 1.2.5 Bewertungskriterien für die Geschäftsfähigkeit

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.

## 2 Metrics

In diesem Kapitel geht es um die verschiedenen Metriken, die für die Bewertung von RAG Evaluations Tools 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 Scores berechnen lassen. Für die Extraktion der Fakten wird häufig ein LLM verwendet, welcher als Richter fungiert.

### 2.1 Retrieval Augmented Generation

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

#### 2.1.1 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 `[ragas__context__precision]`)

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

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

### 2.1.2 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 [ragas\_context\_recall])

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.

### 2.1.3 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 [ragas\_context\_entities\_recall])

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

### 2.1.4 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 [ragas\_noise\_sensitivity])

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

### 2.1.5 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 [ragas\_response\_relevancy])

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.

### 2.1.6 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:

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

(eigene Übersetzung nach [ragas\_faithfulness])

### 2.1.7 Multimodal Faithfulness/Multimodal Relevance

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

## 2.2 Nvidia Metrics

Diese Metriken sind subjektiver Art und benutzen wieder eine LLM um die Bewertung zu treffen. Hier werden einzelne Scores generiert welche keinen tieferen Einblick in die Bewertung gewähren.

### 2.2.1 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 [ragas\_nvidia\_metrics])

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.

### 2.2.2 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.

### 2.2.3 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.

## 2.3 Natural Language Comparison

### 2.3.1 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.[\[wikipedia\\_confusion\\_matrix\]](#) Aus dieser Matrix lassen sich dann precision, recall und f1 score berechnen.

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

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

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

[\[wikipedia\\_confusion\\_matrix\]](#)

### 2.3.2 Semantic Similarity

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

## 2.4 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.



### 2.4.1 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.

### 2.4.2 ROUGE Score

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

### 2.4.3 String Presence

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

### 2.4.4 Exact Match

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

## 2.5 General purpose

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

- Aspect critic
- Simple Criteria Scoring
- Rubrics based Scoring
- Instance Specific Rubrics Scoring

## 2.6 Andere Metriken

### 2.6.1 Summarization

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

## 2.7 Irrelevante Metriken

### 2.7.1 SQL

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

### 2.7.2 Agents or Tool use cases

Metriken zum Bewerten des Einsatzes von Agents 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 faktuale Fragen, eine gute Bewertung liefern kann.

## 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