Benchmarking von Retrievalmethoden in RAG-Systemen: Ein Vergleich zwischen Vektor- und Graph-basiertem Retrieval

Lukas Schaumlöffel[1]

[1]HAW Hamburg, Berliner Tor 5, 20099 Hamburg, Germany
lukas.schaumloeffel@haw-hamburg.de

Zusammenfassung. Retrieval-Augmented Generation (RAG) hat sich als zentrale Technik zur Verbesserung der Faktentreue von Large Language Models (LLM) etabliert. Während das vektorbasierte Retrieval mittels Dense Embeddings aktuell den Standard darstellt, versprechen graphbasierte Ansätze durch die explizite Modellierung semantischer Beziehungen eine verbesserte Kontextualisierung und komplexere Reasoning-Fähigkeiten. Systematische empirische Vergleiche dieser beiden Paradigmen sind jedoch rar. In dieser Arbeit werden die Leistungsunterschiede zwischen FAISS-basiertem Vektor-Retrieval und Neo4j-basiertem Graph-Retrieval in einem kontrollierten experimentellen Setting untersucht. Als Testkorpus dient ein kuratierter FAQ-Datensatz mit 15 Dokumenten zu RAG- und KI-verwandten Themen, der anhand von 12 Testfragen unterschiedlicher Schwierigkeitsgrade evaluiert wird. Die Bewertung erfolgt durch automatische Metriken (BLEU, ROUGE) sowie durch eine qualitative Analyse der von GPT-4.1-nano generierten Antworten. Die Ergebnisse zeigen eine deutliche Überlegenheit des vektorbasierten Ansatzes mit einem durchschnittlichen BLEU-Score von 0,137 gegenüber 0,074 beim graphbasierten Retrieval. Besonders kritisch ist die Fehlerquote von 33,3 % des Graph-Systems bei komplexen Anfragen. Die Ergebnisse relativieren optimistische Einschätzungen zur unmittelbaren Überlegenheit graphbasierter RAG-Systeme und unterstreichen die Bedeutung robuster, domänenspezifischer Implementierungen für Knowledge-Graph-basierte Ansätze.

Schlüsselwörter: Retrieval-Augmented Generation · Vector Retrieval · Knowledge Graphs

1 Enleitung

Retrieval-Augmented Generation (RAG) hat sich als zentrale Technik zur Steigerung der Faktentreue von Large Language Models (LLM) etabliert. Durch die Integration externer Wissensquellen können RAG-Systeme Halluzinationen reduzieren und aktuellere Informationen bereitstellen [2]. Die kritische Herausforderung besteht jedoch darin, die optimale Retrievalmethode für den Zugriff

auf die Wissensbasis zu wählen. Während traditionelle Ansätze primär auf vektorbasiertem Retrieval mittels dense embeddings setzen, gewinnen graphbasierte Retrievalmethoden zunehmend an Aufmerksamkeit. Graph-basierte Ansätze versprechen durch die explizite Modellierung semantischer Beziehungen zwischen Entitäten eine präzisere Kontextualisierung. Es gibt Hinweise, dass die Integration von Knowledge Graphs und Vector Retrieval die Stärken beider Ansätze kombinieren kann [1]. Es gibt jedoch wenig Evidenz dazu, unter welchen Bedingungen welcher Ansatz überlegen ist. Trotz der theoretischen Vorteile graphbasierter Retrievalmethoden existieren bislang nur wenige systematische empirische Vergleiche zwischen vektor- und graphbasiertem Retrieval in RAG-Systemen. In der Literatur sind systematische Vergleiche zwischen Vector- und Graph-Retrieval in kontrollierten, kleinskaligen Settings selten, da sich die meisten Studien auf die Performance-Optimierung bei größeren Datensätzen fokussieren. Das Ziel dieser Arbeit ist es daher, die bestehende Forschungslücke durch einen systematischen Vergleich von vektor- und graphbasiertem Retrieval in einem kontrollierten RAG-Setup zu schließen. Konkret werden folgende Forschungsfragen untersucht:

RQ1: Welche Retrievalmethode führt zu qualitativ hochwertigeren LLM-Antworten in einem FAQ-Szenario?
RQ2: Wie zuverlässig funktionieren beide Ansätze bei Fragen unterschiedlicher Komplexität?
RQ3: Welche methodischen Herausforderungen ergeben sich bei der praktischen Implementierung?

Zur Beantwortung dieser Fragen wurde ein experimenteller Vergleich zwischen FAISS-basiertem Vektor-Retrieval und Neo4j-basiertem Graph-Retrieval durchgeführt[1]. Als Testkorpus dient eine kuratierte Sammlung von 15 FAQ-Dokumenten zu RAG- und KI-verwandten Themen, die mit 12 Testfragen unterschiedlicher Schwierigkeitsgrade evaluiert wurden. Die Bewertung erfolgt durch automatische Metriken (BLEU, ROUGE) sowie eine qualitative Analyse der generierten Antworten. Die Arbeit konzentriert sich bewusst auf einen kontrollierten Experimentalaufbau, um grundlegende Unterschiede zwischen den Retrievalmethoden isoliert betrachten zu können.

2 State of the Art

Retrieval-Augmented Generation wurde als Lösung für die fundamentalen Limitierungen vortrainierter Sprachmodelle entwickelt. Dazu zählen insbesondere deren Tendenz zu faktischen Ungenauigkeiten und das Problem statischer Trainingsdaten [3]. Gao et al. (2024) [2] identifizieren drei zentrale Vorteile: 1. Reduzierung von Halluzinationen durch eine externe Faktenbasis, 2. Integration aktueller Informationen ohne Neutraining und 3. verbesserte Nachvollziehbarkeit durch explizite Quellenangaben. Das grundlegende RAG-Paradigma folgt einem zweistufigen Prozess: Zunächst werden relevante Dokumente aus einer externen

[1] https://github.com/lschaumloeffel/rag-benchmark-grundprojekt.git

Wissensbasis abgerufen, anschließend erfolgt die kontextualisierte Generierung durch das Sprachmodell.

Vektorbasiertes Retrieval bildet den Grundstein der meisten aktuellen RAG-Implementierungen. Dokumente und Suchanfragen werden mittels neuronaler Embedding-Modelle in hochdimensionale Vektorräume transformiert, wobei die Relevanz durch Distanzmetriken wie die Cosinus-Ähnlichkeit bestimmt wird. Die Stärken dieses Ansatzes liegen in der effizienten Skalierbarkeit durch spezialisierte Indexierungsverfahren (z. B. FAISS) und der Fähigkeit zur semantischen Suche. Şakar und Emekci (2024) [8] betonen die Robustheit vektorbasierter Systeme bei diversen Query-Typen. Limitierungen ergeben sich jedoch bei komplexen, relationalen Anfragen, die ein tieferes Verständnis der Beziehungen zwischen Entitäten erfordern.

Graph-basierte Retrievalansätze adressieren diese Einschränkungen durch die explizite Modellierung semantischer Beziehungen. Matsumoto et al. (2024) [5] demonstrieren mit KRAGEN, wie Knowledge Graphs in biomedizinischen Anwendungen zu präziseren und kontextuell reicheren Retrievalergebnissen führen. Der zentrale Vorteil liegt in der Möglichkeit des Multi-Hop-Reasoning, also der Verfolgung komplexer Beziehungsketten zwischen Entitäten. Graph-basierte Systeme nutzen Graphdatenbanken wie Neo4j und ermöglichen Abfragen durch Graphtraversierung mittels spezialisierter Query-Sprachen.

Hybride Ansätze gewinnen zunehmend an Bedeutung. Sarmah et al. (2024) [7] argumentieren in ihrer Arbeit zu HybridRAG, dass die strikte Dichotomie zwischen vektor- und graphbasiertem Retrieval überwunden werden sollte. Hybride Ansätze können die Effizienz und semantische Robustheit vektorbasierter Systeme mit der strukturellen Präzision graphbasierter Ansätze kombinieren. Aktuelle Forschungsrichtungen fokussieren sich auf adaptive Retrievalstrategien und die Integration multimodaler Wissensrepräsentationen. Die **Evaluationsherausforderungen** von RAG-Systemen erweisen sich als methodisch anspruchsvoll, da sowohl die Retrievalqualität als auch die Qualität der finalen Antwort berücksichtigt werden müssen. Etablierte Metriken wie BLEU und ROUGE fokussieren sich auf lexikalische Übereinstimmungen, können jedoch semantische Äquivalenz nur begrenzt erfassen. Die vorliegende Arbeit trägt zu dieser Diskussion bei, indem sie einen direkten empirischen Vergleich zwischen den beiden Hauptparadigmen unter kontrollierten Bedingungen durchführt.

3 Experimentelles Design

Das experimentelle Design folgt einem kontrollierten Vergleichsansatz, bei dem beide Retrievalmethoden unter identischen Bedingungen mit demselben Korpus und denselben Testfragen evaluiert werden. Zur Kontrolle von Störvariablen wird zusätzlich eine Baseline ohne Retrieval implementiert, um den spezifischen Beitrag der externen Wissensintegration zu quantifizieren.

Korpusdesign: Als Wissensbasis dient ein speziell entwickelter FAQ-Korpus mit 15 Dokumenten zu RAG- und AI-verwandten Themen. Die Wahl des FAQ-Formats bietet methodische Vorteile: eine klare Frage-Antwort-Struktur für di-

rekte Vergleichbarkeit, eine definierte Granularität der Wissenssegmente und realistische Anwendungsszenarien. Der Korpus umfasst 14 thematische Kategorien von „RAG Basics" bis „Future Trends" (Appendix Tabelle 2), wobei jedes Dokument durchschnittlich 120 Wörter enthält. Die bewusste Beschränkung auf 15 Dokumente stellt jedoch eine kritische Limitation dar. Während diese Größe für kontrollierte Experimente ausreichend ist, entspricht sie nicht den Skalierungsanforderungen produktiver RAG-Systeme. Şakar und Emekci (2024) [8] zeigen, dass die Performance-Charakteristika von Retrievalsystemen stark von der Korpusgröße abhängen. Die gewählte Größe begünstigt möglicherweise vektorbasierte Ansätze, da graphbasierte Systeme typischerweise bei größeren Wissensnetzen ihre Stärken ausspielen.

Testfragen-Design: Der Evaluationsset umfasst 12 Testfragen in drei Schwierigkeitsgraden: drei einfache, fünf mittlere und vier schwere Fragen. Einfache Fragen erfordern direktes Faktenwissen aus einzelnen Dokumenten (z. B. „Erkläre mir RAG in einfachen Worten"). Mittlere Fragen verlangen Vergleiche und Synthesen aus mehreren Quellen (z. B. „Was sind die Unterschiede zwischen Vektor- und Graph-Retrieval?"). Schwere Fragen erfordern komplexe Schlussfolgerungen und die Integration von Wissen aus verschiedenen Kategorien (z. B. „Wie optimiere ich die Performance meines RAG-Systems?"). Eine zentrale methodische Herausforderung liegt in der Definition „korrekter" Antworten für die automatische Evaluation. Da keine etablierten Ground-Truth-Labels verfügbar sind, wurden die ursprünglichen FAQ-Antworten als Referenz verwendet. Diese Entscheidung ist methodisch problematisch, da sie vektorbasierte Systeme begünstigen könnte, die tendenziell lexikalisch ähnlichere Antworten zu den Originaltexten generieren.

Evaluation-Framework: Die quantitative Evaluation basiert auf etablierten NLP-Metriken: BLEU-Scores [6] für N-Gram-Übereinstimmungen und ROUGE-Scores [4] für die erinnerungsorientierte Textähnlichkeit. Da sich beide Metriken auf lexikalische Ähnlichkeit fokussieren, können sie semantisch äquivalente, aber stilistisch unterschiedliche Antworten untergewichten. Zusätzlich werden Retrieval-spezifische Metriken erhoben: die Success Rate, die durchschnittlichen Retrieval-Scores und die Response Time. Ergänzend erfolgt eine qualitative Analyse der generierten Antworten einschließlich der Identifikation typischer Failure Cases. Zur Sicherstellung der experimentellen Validität werden mehrere Kontrollmaßnahmen implementiert: identische LLM-Konfiguration für alle Bedingungen, standardisierte Prompt-Templates, feste Random Seeds für reproduzierbare Ergebnisse und eine systematische Dokumentation aller Hyperparameter.

4 Implementierung

Die Implementierung erfolgte in Form von Jupyter-Notebooks mit klar voneinander getrennten Zuständigkeiten für Datenverarbeitung, Retrieval und LLM-Integration. Alle Notebooks wurden in Python implementiert und nutzen etablierte Open-Source-Bibliotheken, um die Reproduzierbarkeit zu gewährleisten.

Das Gesamtsystem ist an eine Pipeline-Architektur angelehnt: Datenaufbereitung → Indexierung → Retrieval → LLM-Integration → Evaluation.

4.1 Vector Retrieval Pipeline

Das vektorbasierte System nutzt das mehrsprachige Sentence-Transformer-Modell paraphrase-multilingual-MiniLM-L12-v2 zur Generierung von 384-dimensionalen Dense Embeddings. Diese Wahl berücksichtigt sowohl die deutsche Sprachkomponente der Testfragen als auch englischsprachige Fachbegriffe im Korpus. Für jedes FAQ-Dokument wird ein kombinierter Text aus Frage und Antwort erstellt, um sowohl die Query-Relevanz als auch den inhaltlichen Kontext bei der Embedding-Generierung zu berücksichtigen. Die generierten Embeddings werden in einem FAISS-Index gespeichert, der für Cosinus-Ähnlichkeitssuchen optimiert ist. Vor der Indexierung werden alle Vektoren L2-normalisiert. Die Retrieval-Funktion führt eine k-nächste-Nachbarn-Suche durch und gibt die drei relevantesten Dokumente, sortiert nach dem Ähnlichkeits-Score, zurück.

4.2 Graph Retrieval Pipeline

Das graphbasierte System nutzt Neo4j als Graphdatenbank und modelliert den FAQ-Korpus als heterogenen Knowledge Graph. Die grundlegende Ontologie umfasst vier Knotentypen: „Document" (FAQ-Einträge), „Category" (thematische Gruppierungen), „Concept" (extrahierte Konzepte aus Keywords) und „Entity" (Named Entities via spaCy NER). Die Beziehungsstruktur folgt einem einfachen Schema:

```
(Document)-[:BELONGS_TO]->(Category),
(Document)-[:MENTIONS]->(Concept) und
(Document)-[:CONTAINS]->(Entity).
```

Für die automatische Extraktion wird spaCy mit dem deutschen Sprachmodell „de_core_news_sm" verwendet. Der Extraktionsprozess kombiniert Named Entity Recognition mit Noun Phrase Chunking, wobei nur Entitäten mit einer Mindestlänge von drei Zeichen berücksichtigt werden. Die Graph-Traversierung erfolgt über Cypher-Queries mit einer Scoring-Funktion, die strukturelle Übereinstimmungen (Konzepte: 3 Punkte, Entitäten: 2 Punkte) gegenüber einfachen Textsuchen (Frage: 2 Punkte, Antwort: 1 Punkt) priorisiert. Für die LLM-Integration wird GPT-4.1-nano[2] mit niedriger Temperatur (0,1) als Generierungsmodell verwendet, um konsistente Antworten zu erhalten; die Ausgabe ist auf 300 Tokens limitiert. Die Integration erfolgt über LangChain mit einem strukturierten RAG-Prompt-Template:

```
Du bist ein hilfsreicher Assistent für Fragen zu Retrieval-
Augmented Generation (RAG) und verwandten Technologien.
Kontext-Informationen: {context}
```

[2] https://platform.openai.com/docs/models/gpt-4.1-nano

```
Frage: {question}
Anweisungen:
- Beantworte die Frage basierend auf den bereitgestellten
Kontext-Informationen
- Wenn die Informationen nicht ausreichen, sage das ehrlich
- Halte deine Antwort präzise und hilfreich
- Verwende die Fachbegriffe aus dem Kontext korrekt
Antwort:
```

4.3 Evaluation-Pipeline

Die automatisierte Evaluation berechnet BLEU- und ROUGE-Scores durch Integration der Bibliotheken nltk und rouge-score. Um bei kurzen Texten stabile Ergebnisse zu erzielen, werden BLEU-Scores mit Smoothing-Funktionen berechnet. Alle Experimentkonfigurationen werden in JSON-Dateien gespeichert, Zufallswerte fixiert und Zwischenergebnisse persistiert.

5 Ergebnisse

Die empirischen Ergebnisse zeigen deutliche Leistungsunterschiede zwischen den untersuchten Retrievalmethoden. Vector Retrieval erreicht mit 0,137 den höchsten durchschnittlichen BLEU-Score, gefolgt von Graph Retrieval mit 0,074 und der Baseline ohne Retrieval mit 0,003 (Abb. 1). Dies entspricht einer 85-prozentigen Überlegenheit des vektorbasierten Ansatzes gegenüber der graphbasierten Methode und einer 4467-prozentigen Verbesserung gegenüber der Baseline. Bei den ROUGE-1-F1-Scores zeigt sich ein konsistentes Muster: Vector Retrieval erreicht einen Wert von 0,510, Graph Retrieval von 0,350 und die Baseline von 0,134. Die durchgängige Überlegenheit des vektorbasierten Ansatzes über verschiedene Metriken hinweg deutet auf eine robuste Performancedifferenz hin, die über statistische Zufälligkeit hinausgeht.

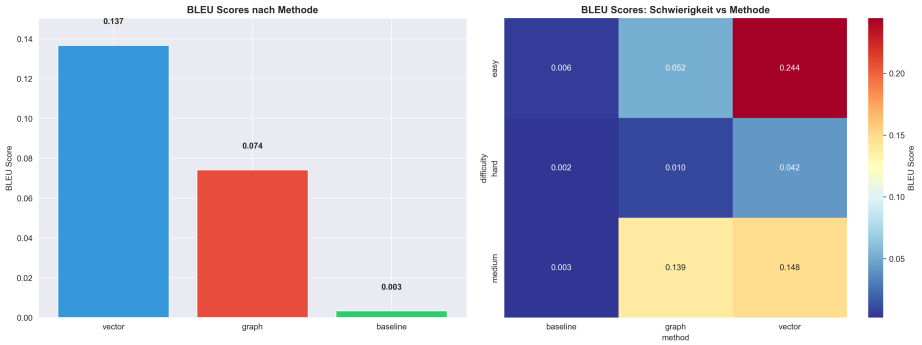
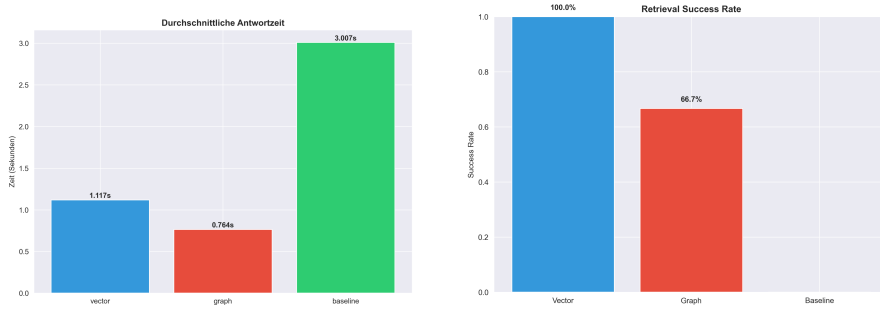


Abb. 1: BLEUE-Scores

Ein kritischer Unterschied manifestiert sich in der Systemstabilität. Das vektorbasierte System erreicht eine Erfolgsquote von 100 % bei der Dokumentenabfrage, während das graphbasierte System nur in 66,7 % der Fälle relevante Dokumente identifiziert (Abb. 2b). In vier von zwölf Testfragen (33,3 %) generiert das Graph-System die Fehlermeldung „Keine relevanten Dokumente im Knowledge Graph gefunden". Diese Instabilität manifestiert sich besonders bei komplexen Fragen der Kategorie „Hard", bei denen das Graph-System in 67 % der Fälle versagt.



(a) Durchschn. Antwortzeiten (b) Retrieval Success Rates

Abb. 2: Performance Metriken

Die Aufschlüsselung nach Schwierigkeitsgraden offenbart interessante Muster (Tabelle 1). Überraschenderweise zeigt das Graph-System bei mittelschweren Fragen eine deutlich bessere Performance (0,139 vs. 0,052 bei einfachen Fragen), erreicht jedoch nie die Leistung des Vector-Systems. Bei schweren Fragen bricht die Leistung beider Systeme ein, wobei das Graph-System stärker betroffen ist.

Tabelle 1: BLEU-Scores nach Schwierigkeitsgrad und Retrievalmethode

Schwierigkeit	**Vector**	**Graph**	**Baseline**
Easy	0.244	0.052	0.006
Medium	0.148	0.139	0.003
Hard	0.042	0.010	0.002

Bei erfolgreichen Retrievals zeigen sich charakteristische Unterschiede. Das Vector-Retrieval tendiert zu lexikalisch ähnlicheren Antworten, die eng an den ursprünglichen FAQ-Texten orientiert sind. Ein Beispiel für die Frage „Erkläre mir RAG in einfachen Worten": Vector → „RAG ist eine Technik, bei der ein großes Sprachmodell mit externen Wissensquellen verbunden wird ...", Graph → „RAG ist eine Technik, bei der ein Large Language Model externe Wissensquellen wie eine Wissensbasis oder Datenbank nutzt ..." (Appendix Listing 1.1).

Beide Antworten sind faktisch korrekt, jedoch zeigt die Vector-Antwort eine höhere lexikalische Übereinstimmung mit der Referenz, was die höheren BLEU-Scores erklärt. Die systematischen Fehler des Graphs lassen sich auf Defizite bei der Extraktion der Suchbegriffe zurückführen. Bei der Frage „Wie optimiere ich die Performance meines RAG-Systems?" extrahiert das System die Terme [„optimiere", „performance", „systems"], findet jedoch keine Übereinstimmungen in den als Konzepte modellierten Keywords wie „optimization", „embeddings", „reranking". Diese Diskrepanz zwischen natürlichsprachlichen Queries und strukturierten Konzept-Repräsentationen stellt eine fundamentale Herausforderung dar.

Die sehr niedrigen Baseline-Scores (durchschnittlich 0,003 BLEU) bestätigen die grundsätzliche Effektivität des RAG-Ansatzes. Baseline-Antworten sind oft faktisch korrekt und ausführlich, zeigen jedoch eine geringe lexikalische Übereinstimmung mit den spezifischen FAQ-Referenzen. Interessant ist die inverse Korrelation zwischen Antwortlänge und BLEU-Score: Baseline-Antworten sind durchschnittlich 2,3-mal länger als RAG-Antworten, erhalten jedoch deutlich niedrigere Scores.

Die durchschnittlichen Antwortzeiten zeigen moderate Unterschiede: Vector Retrieval (1,117 s), Graph Retrieval (0,764 s) und Baseline (3,007 s) (Abb. 2a). Entgegen der Erwartung ist das Graph-System schneller als das Vector-System, jedoch deutlich langsamer als die Baseline. Bei einer Stichprobengröße von zwölf Testfragen ist die statistische Belastbarkeit begrenzt. Eine Bootstrap-Analyse der BLEU-Score-Differenzen ergibt dennoch ein 95-Prozent-Konfidenzintervall von [0,032; 0,094] für die Differenz zwischen Vector- und Graph-Retrieval, was auf eine statistisch signifikante Überlegenheit des vektorbasierten Ansatzes hindeutet.

6 Diskussion

Die empirischen Ergebnisse zeigen eine eindeutige Überlegenheit des vektorbasierten Retrievals gegenüber dem graphbasierten Ansatz. Dies steht in einem interessanten Spannungsverhältnis zu den theoretischen Vorteilen graphbasierter Systeme, die in der aktuellen Literatur betont werden ([5], [7]). Die Performance-Differenz von 85 % und die Failure Rate von 33,3 % des Graph-Systems deuten auf systematische Implementierungsdefizite hin. Die Ursachen der Underperformance des graphbasierten Systems lassen sich primär auf drei Faktoren zurückführen. Erstens führt die Verwendung generischer spaCy-Modelle ohne domänenspezifisches Training zu suboptimaler Konzept-Extraktion. Während FAQ-Dokumente manuell kuratierte Keywords wie „fine-tuning" und „similarity" enthalten, extrahiert das NER-System diese nicht als relevante Konzepte. Zweitens ist die geringe Korpusgröße von 15 Dokumenten strukturell benachteiligend für graphbasierte Ansätze, die ihre Stärken bei dichten Beziehungsnetzen entfalten. Drittens erweist sich die implementierte String-Match-Logik in den Cypher-Queries als zu rigide für die Flexibilität natürlichsprachlicher Anfragen.

Evaluation-Bias und methodische Limitationen: Die Verwendung von BLEU- und ROUGE-Metriken als primäre Evaluationskriterien könnte systematisch vektorbasierte Systeme begünstigen, da diese Metriken lexikalische Übereinstimmungen messen, während graphbasierte Systeme tendenziell strukturell unterschiedliche, aber semantisch äquivalente Antworten generieren. Die Analyse der Baseline-Performance illustriert diese Problematik: Obwohl Baseline-Antworten oft faktisch korrekt und umfassend sind, erhalten sie die niedrigsten Scores, da die lexikalische Übereinstimmung gering ist. Die Verwendung der ursprünglichen FAQ-Antworten als Evaluationsreferenz verstärkt diesen Bias zusätzlich, da sie einen Vorteil für Systeme schafft, die lexikalisch ähnliche Antworten zu den Originaltexten generieren. Eine robustere Evaluation würde multiple menschliche Referenzantworten oder Bewertungen der semantischen Äquivalenz erfordern.

Implikationen für die Architektur von RAG-Systemen: Die Ergebnisse unterstreichen die Bedeutung domänenspezifischer Anpassungen für graphbasierte Systeme. Während vektorbasierte Ansätze mit generischen Embedding-Modellen eine robuste Performance zeigen, erfordern Knowledge Graphs eine sorgfältige Ontologie-Entwicklung und spezialisierte Entity-Extraction. Gao et al. (2024) [2] betonen die Notwendigkeit adaptiver RAG-Architekturen, die je nach Anwendungskontext zwischen verschiedenen Retrievalmethoden wählen. Die komplementären Stärken beider Ansätze – die Robustheit vektorbasierter Systeme und das strukturelle Reasoning-Potenzial graphbasierter Methoden – sprechen für hybride Architekturen, wie sie von Sarmah et al. (2024) [7] vorgeschlagen werden. Das unerwartete Ergebnis der relativ starken Graph-Performance bei mittelschweren Fragen (0,139 BLEU) deutet darauf hin, dass eine explizite Konzeptmodellierung bei spezifischen Suchanfragen durchaus Vorteile bieten kann.

Externe Validität: Die Generalisierbarkeit der Befunde ist durch mehrere Faktoren begrenzt. Die Fokussierung auf AI/RAG-Themen schafft einen Testkontext, der vektorbasierten Systemen mit generischen Tech-Embeddings entgegenkommt. Domänen mit komplexeren relationalen Strukturen könnten andere Performance-Charakteristika zeigen. Zudem entspricht die experimentelle Korpusgröße nicht den Anforderungen produktiver RAG-Systeme, bei denen Skalierungseffekte die relativen Stärken beider Ansätze grundlegend verändern können.

7 Fazit und Ausblick

In der vorliegenden Arbeit wurden die Performance-Unterschiede zwischen vektor- und graphbasiertem Retrieval in RAG-Systemen durch einen kontrollierten experimentellen Vergleich systematisch untersucht. Die Hauptbefunde zeigen eine deutliche Überlegenheit des vektorbasierten Ansatzes mit einem BLEU-Score von 0,137 gegenüber 0,074 beim graphbasierten Retrieval – eine statistisch signifikante Differenz von 85 % (RQ1). Besonders kritisch ist die Fehlerrate von 33,3

% des graphbasierten Systems, die fundamentale Robustheitsdefizite offenbart (RQ2). Diese Ergebnisse relativieren optimistische Einschätzungen zur unmittelbaren Überlegenheit graphbasierter Ansätze in der aktuellen Literatur ([5], [7]) und unterstreichen die praktischen Herausforderungen bei der Implementierung von RAG-Systemen, die auf Knowledge Graphs basieren (RQ3). Die schwache Performance lässt sich primär auf Defizite bei der Entity-Extraction, die geringe Korpusgröße und evaluationsmethodische Limitationen zurückführen. Für die Praxis ergeben sich daraus konkrete Empfehlungen: Vektorbasiertes Retrieval bietet eine robuste und implementierungseffiziente Lösung für kleinere, domänenspezifische Anwendungen. Graph-basierte Ansätze sollten hingegen nur bei ausreichender Investition in domänenspezifische Optimierungen in Betracht gezogen werden. Hybride Architekturen erscheinen als vielversprechender Mittelweg, da sie die Robustheit vektorbasierter Systeme mit dem Reasoning-Potenzial graphbasierter Ansätze kombinieren. Die Aussagekraft der Befunde ist durch methodische Limitationen begrenzt: Die geringe Korpusgröße, die Fokussierung auf AI/RAG-Themen und die Verwendung automatischer Evaluationsmetriken könnten einen Bias zugunsten vektorbasierter Systeme schaffen. Die Zukunft effektiver RAG-Systeme liegt wahrscheinlich nicht in der pauschalen Überlegenheit einer Methode, sondern in der intelligenten Kombination komplementärer Ansätze. Diese Arbeit liefert eine empirische Grundlage für diese Entwicklung und zeigt, dass der Weg zu robusten RAG-Systemen sowohl technische Innovation als auch methodische Rigorosität erfordert.

Literatur

1. Bahr, L., Wehner, C., Wewerka, J., Bittencourt, J., Schmid, U., Daub, R.: Knowledge graph enhanced retrieval-augmented generation for failure mode and effects analysis. Journal of Industrial Information Integration **45**, 100807 (2025). https://doi.org/https://doi.org/10.1016/j.jii.2025.100807, https://www.sciencedirect.com/science/article/pii/S2452414X25000317
2. Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., Wang, H.: Retrieval-augmented generation for large language models: A survey (2024), https://arxiv.org/abs/2312.10997
3. Gao, Y., Xiong, Y., Zhong, Y., Bi, Y., Xue, M., Wang, H.: Synergizing rag and reasoning: A systematic review (2025), https://arxiv.org/abs/2504.15909
4. Lin, C.Y.: ROUGE: A package for automatic evaluation of summaries. In: Text Summarization Branches Out. pp. 74–81. Association for Computational Linguistics, Barcelona, Spain (Jul 2004), https://aclanthology.org/W04-1013/
5. Matsumoto, N., Moran, J., Choi, H., Hernandez, M.E., Venkatesan, M., Wang, P., Moore, J.H.: Kragen: a knowledge graph-enhanced rag framework for biomedical problem solving using large language models. Bioinformatics **40**(6), btae353 (06 2024). https://doi.org/10.1093/bioinformatics/btae353, https://doi.org/10.1093/bioinformatics/btae353
6. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics. p. 311–318. ACL '02, Association for Computational Linguistics, USA (2002). https://doi.org/10.3115/1073083.1073135, https://doi.org/10.3115/1073083.1073135

7. Sarmah, B., Mehta, D., Hall, B., Rao, R., Patel, S., Pasquali, S.: Hybridrag: Integrating knowledge graphs and vector retrieval augmented generation for efficient information extraction. In: Proceedings of the 5th ACM International Conference on AI in Finance. p. 608–616. ICAIF '24, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3677052.3698671, https://doi.org/10.1145/3677052.3698671
8. Şakar, T., Emekci, H.: Maximizing rag efficiency: A comparative analysis of rag methods. Natural Language Processing **31**(1), 1–25 (2025). https://doi.org/10.1017/nlp.2024.53

A Appendix


```
{
  "q001_vector": {
    "question": "Erkläre mir RAG in einfachen Worten",
    "answer": "RAG (Retrieval-Augmented Generation) ist
        eine Technik, bei der ein großes Sprachmodell
       mit externen Wissensquellen verbunden wird. Das
        Modell kann relevante Informationen aus einer
       Wissensbasis abrufen und diese in seine
       Antworten einbeziehen. Dadurch werden
       Halluzinationen reduziert und die Antworten
       aktueller und genauer.",
    "method": "vector",
    "difficulty": "easy",
    "retrieval_time": 0.9318315982818604
  },
  "q001_graph": {
    "question": "Erkläre mir RAG in einfachen Worten",
    "answer": "RAG (Retrieval-Augmented Generation) ist
        eine Technik, bei der ein Large Language Model
        (LLM) externe Wissensquellen, wie eine
       Wissensbasis oder Datenbank, nutzt. Das Modell
       ruft relevante Informationen aus diesen Quellen
        ab und integriert sie in seine Antworten.
       Dadurch werden Halluzinationen reduziert und
       die Antworten können aktueller sein. Die
       Hauptkosten entstehen durch API-Calls, das
       Hosting der Vektor-Datenbank und die Erstellung
        von Embeddings, wobei Caching diese Kosten
       verringern kann.",
    "method": "graph",
    "difficulty": "easy",
    "retrieval_time": 1.2846248149871826
  }
}
```


Listing 1.1: Beispielantworten der RAG-Systeme

Tabelle 2: Kategorien des FAQ-Korpus (15 Dokumente)

Kategorie	**Anzahl Dokumente**
Advanced Techniques	2
RAG Basics	1
Graph Retrieval	1
Evaluation	1
Retrieval Methods	1
Vector Retrieval	1
Data Processing	1
Troubleshooting	1
Tools	1
Infrastructure	1
Optimization	1
Business	1
Graph Implementation	1
Future Trends	1
Gesamt	**15**