Modulare Benchmarking-Architektur für Retrievalmethoden in Retrieval-Augmented Generation

Lukas Schaumlöffel[1]

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

Zusammenfassung. Retrieval-Augmented Generation (RAG) verbindet Large Language Models (LLMs) mit externen Wissensquellen und wird als eine zentrale Technik angesehen, um die Faktentreue generativer Systeme zu verbessern. Vektorbasierte Ansätze stellen gegenwärtig den Standard dar, während graphgestützte Verfahren durch die Modellierung semantischer Relationen eine höhere Kontexttiefe versprechen. Das Ziel dieses Hauptprojekts bestand im Aufbau einer modularen Vergleichsumgebung, mit der unterschiedliche Retrievalmethoden unter identischen Bedingungen systematisch evaluiert werden können. Die entwickelte Architektur umfasst Module für Datenvorbereitung, Vector-, Graph- und Hybrid Retrieval, eine Baseline ohne Retrieval sowie eine automatisierte Evaluationspipeline. Die vorliegenden Experimente auf einem kleinen FAQ-Korpus zeigen eine leichte Überlegenheit des Vector Retrievals, während das Hybrid Retrieval vergleichbare Ergebnisse erzielt. Im Durchschnitt zeigte sich das Graph Retrieval-Verfahren als weniger leistungsstark, obgleich bei einer mittleren Frageschwierigkeit in einigen Fällen eine Konkurrenzfähigkeit festgestellt werden konnte. Die insgesamt niedrigen BLEU- und ROUGE-Werte verdeutlichen die Limitation klassischer Evaluationsmetriken und unterstreichen die Notwendigkeit semantisch robuster Bewertungsverfahren. Das Projekt liefert damit nicht nur empirische Befunde, sondern vor allem eine reproduzierbare und erweiterbare Infrastruktur für zukünftige RAG-Experimente.

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

1 Enleitung

1.1 Motivation und Problemstellung

In den zurückliegenden Jahren hat sich die Retrieval-Augmented Generation (RAG) als ein zentrales Paradigma etabliert, dessen Ziel die Verbesserung der Faktentreue und Aktualität von Large Language Models (LLMs) ist. Die Kombination generativer Sprachmodelle mit externen Wissensquellen ermöglicht die Reduktion von Halluzinationen und die Nutzung von Informationen auch über

den Trainingskorpus hinaus. Ein zentraler Erfolgsfaktor für die Leistungsfähigkeit von RAG-Systemen liegt dabei in der Wahl der Retrievalmethode.

Vektorbasierte Ansätze fokussieren sich auf dichte Embeddings und effiziente Approximate Nearest Neighbor-Suchen. Graphbasierte Methoden hingegen zielen darauf ab, explizite semantische Beziehungen zwischen Entitäten nutzbar zu machen. Der Versuch, die Stärken beider Paradigmen zu vereinen, wird als "hybrides Verfahren" bezeichnet. In der wissenschaftlichen Diskussion besteht jedoch weiterhin eine gewisse Unsicherheit bezüglich der Bedingungen, unter denen eine bestimmte Methode als vorteilhafter zu erachten ist. In der wissenschaftlichen Literatur werden die theoretischen Vorteile graphgestützter Systeme vielfach hervorgehoben. Jedoch mangelt es an systematischen empirischen Vergleichen (vgl. [3]; [8]).

Die vorliegende Problemstellung des Projekts ergibt sich aus dem zuvor dargelegten Hintergrund. Es besteht bislang ein Mangel an einer leicht nutzbaren, modularen Umgebung, die verschiedene Retrievalmethoden in RAG-Systemen systematisch vergleichbar macht.

1.2 Zielsetzung des Projekts

Das Hauptprojekt[1] zielt auf den Aufbau einer experimentellen Vergleichsumgebung ab, die verschiedene Retrievalmethoden – Vektor-, Graph- und Hybrid Retrieval – sowie eine Baseline ohne Retrieval unter identischen Bedingungen evaluierbar macht. Im Folgenden werden die Anforderungen definiert, die zu erfüllen sind:

- Modularität: Die einzelnen Komponenten (Daten, Retrieval, Evaluation) werden als wiederverwendbare Module implementiert
- Konfigurierbarkeit: Experimente können über YAML/JSON-Dateien flexibel angepasst werden
- Reproduzierbarkeit: Ergebnisse werden automatisiert in standardisierten Formaten (CSV JSON) ausgegeben
- Erweiterbarkeit: Neue Retrievalmethoden oder Evaluationsmetriken sollen ohne tiefgreifende Änderungen integrierbar sein

Das Hauptaugenmerk liegt weniger auf der Optimierung einzelner Methoden, sondern auf der Schaffung einer robusten Infrastruktur für vergleichende Experimente.

1.3 Abgrenzung zum Grundprojekt

Das Grundprojekt widmete sich der Durchführung eines Vergleichs zwischen einem auf FAISS basierenden Vektor-Retrieval und einem auf Neo4j basierenden Graph-Retrieval. Dieser Vergleich wurde anhand eines kleinen FAQ-Korpus

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

durchgeführt. Die Ergebnisse der Untersuchung offenbarten zwar erste Hinweise auf signifikante Leistungsunterschiede, jedoch waren diese aufgrund methodischer Einschränkungen nur eingeschränkt übertragbar. Zu den identifizierten Einschränkungen zählen eine begrenzte Datenbasis, das Fehlen von Modularität und Konfiguration. Das Hauptprojekt knüpft an diesen Befund an, geht aber in zwei Punkten deutlich weiter:

1. **Systemische Weiterentwicklung:** Statt einmaliger Jupyter-Notebooks wurde eine vollwertige Python-Pipeline mit Konfigurationsmanagement, Logging und automatisierter Evaluation implementiert
2. **Erweiterter Methodenfokus:** Neben Vektor- und Graphretrieval wurde ein Hybridansatz integriert, der eine flexible Fusion beider Methoden erlaubt.

Damit bildet das Hauptprojekt die methodische Grundlage für die anschließende Masterarbeit, in der die wissenschaftliche Untersuchung der Einbindung strukturierter Wissensquellen in RAG-Systemen im Mittelpunkt stehen soll.

2 Stand der Forschung

2.1 Retrieval-Augmented Generation (RAG) im Überblick

Die zugrunde liegende Idee besteht in der Kopplung eines generativen Sprachmodells mit einer externen Wissensbasis. In einem zweistufigen Prozess werden zunächst relevante Dokumente abgerufen und anschließend zur kontextualisierten Antwortgenerierung herangezogen. Dieser Ansatz adressiert zwei zentrale Schwächen reiner LLMs: Halluzinationen sowie die Abhängigkeit von statischem Trainingswissen. RAG hat sich in den letzten Jahren als Standardparadigma etabliert und bildet die Grundlage für eine Vielzahl industrieller und akademischer Anwendungen – von Chatbots bis hin zu spezialisierten Expertensystemen. Ein zentrales Forschungsfeld betrifft die Frage, welche Retrievalmethode in RAG-Systemen am besten geeignet ist.

2.2 Vektorbasierte Retrievalmethoden

Vector Retrieval ist derzeit der dominierende Ansatz in produktiven RAG-Systemen. Er basiert auf der Transformation von Dokumenten und Suchanfragen in hochdimensionale Embeddings durch neuronale Modelle wie SentenceTransformers oder OpenAI Embeddings. Anschließend wird die Ähnlichkeit zwischen Query und Dokument über Metriken wie die Cosinus-Ähnlichkeit oder die euklidische Distanz bestimmt. Für die effiziente Suche in großen Korpora kommen spezialisierte Bibliotheken wie FAISS oder Annoy zum Einsatz. Vector Retrieval zeichnet sich durch hohe Robustheit und Skalierbarkeit aus [11]. Ein Nachteil besteht jedoch darin, dass semantische Relationen zwischen Entitäten implizit in den Embeddings „versteckt" bleiben und nicht explizit nutzbar sind.

2.3 Graphbasierte Retrievalmethoden

Graph Retrieval ist ein Ansatz, der darauf abzielt, explizite semantische Beziehungen zwischen Entitäten darzustellen und diese für komplexe Abfragen nutzbar zu machen. Knowledge Graphs (KGs) modellieren Entitäten als Knoten und Relationen als Kanten; das Retrieval erfolgt typischerweise durch Traversierungen oder Pattern Matching mit Graphsprachen wie Cypher.

Matsumoto et al. [5] zeigen in ihrer Studie, dass graphgestützte RAG-Ansätze insbesondere in spezialisierten Domänen, wie beispielsweise der Biomedizin, eine höhere Antwortqualität liefern können. Auch neuere Arbeiten betonen das Potenzial, komplexe Reasoning-Aufgaben über Multi-Hop-Abfragen besser zu unterstützen (vgl. [2]). Allerdings sind graphbasierte Ansätze anfällig für Implementierungsprobleme. Dies betrifft das Ontologie-Design, die Entity Extraction und die Modellierung konsistenter Relationen, für die hohe Anforderungen gestellt werden. Darüber hinaus ist die Performance in kleinen oder spärlich besetzten Graphen oft limitiert. Dieser Befund wurde bereits im Grundprojekt bestätigt.

2.4 Hybride Ansätze in RAG-Systemen

Die vorliegende Arbeit nutzt u.a. Hybrid Retrieval, eine Methode, die die Stärken von Vector- und Graphmethoden vereint. Dabei werden diese beiden Ansätze kombiniert, um die jeweiligen Vorteile zu maximieren. Während das Vector Retrieval eine robuste semantische Suche ermöglicht, ermöglicht Graph Retrieval strukturierte Reasoning-Prozesse [1]. In ihrer Publikation aus dem Jahr 2024 präsentieren Sarmah et al. [8] ein Framework, das die beiden Paradigmen mittels diverser Fusionsstrategien integriert. Dieses Framework wird als HybridRAG bezeichnet. Erste Ergebnisse weisen darauf hin, dass Hybridmethoden insbesondere bei heterogenen Query-Typen Vorteile bieten. Darüber hinaus werden adaptive Ansätze, die eine dynamische Gewichtung in Abhängigkeit zur Komplexität der Abfrage ermöglichen, derzeit in der Forschung untersucht. Diese Entwicklung lässt die Vermutung zu, dass die Zukunft nicht in einem "Entweder-Oder" zwischen Vektor und Graph liegt, sondern in intelligenten Kombinationen beider Verfahren.

2.5 Evaluationsmetriken für RAG-Systeme

Die Bewertung von RAG-Systemen erfolgt traditionell über Textähnlichkeitsmetriken wie BLEU [6] und ROUGE [4]. Diese Metriken messen N-Gram-Überlappungen zwischen generierten Antworten und Referenztexten. Sie sind leicht automatisierbar, weisen aber eine zentrale Schwäche auf: sie begünstigen Systeme, die lexikalisch ähnliche Antworten erzeugen, und können semantisch äquivalente, aber stilistisch abweichende Antworten stark abwerten. Im Grundprojekt zeigte sich, dass BLEU/ROUGE die Baseline-Antworten systematisch unterschätzten,

obwohl diese inhaltlich korrekt waren. Neuere Forschungsansätze schlagen deshalb LLM-basierte Evaluationen („LLM-as-a-Judge“ [10]) oder semantische Metriken (z. B. BERTScore [9]) vor. Diese Methoden sind jedoch rechenintensiver und weniger standardisiert.

3 Systemdesign und Implementierung

3.1 Architektur und Aufbau der Vergleichsumgebung

Das Hauptprojekt wurde nicht als singulärer Notebook-Prototyp realisiert, sondern als modulare Python-Architektur, die eine flexible Konfigurierbarkeit und Erweiterbarkeit aufweist. Die vorliegende Gesamtstruktur lässt sich in folgende Kernkomponenten untergliedern:

- **Data-Module** (loader.py, preprocessor.py): Laden, Validieren und Vorverarbeiten des Korpus sowie der Testfragen
- **Retrieval-Module** (vector_retrieval.py, graph_retrieval.py, hybrid_retrieval.py): Implementierung der drei untersuchten Retrievalmethoden
- **Evaluation-Module** (metrics.py, evaluator.py): Berechnung der Metriken, Durchführung und Aggregation von Experimenten
- **Validation-Module** (config_validator.py): Überprüfung und Normalisierung der YAML-/JSON-Konfigurationen
- **Experiment-Runner** (run_experiment.py): Orchestrierung des gesamten Ablaufs, inkl. Logging und Ergebnisexport

Die vorliegende Modularisierung ist als klar strukturiert zu bezeichnen, was eine Trennung von Zuständigkeiten, eine einfache Integration neuer Methoden sowie die Reproduzierbarkeit von Experimenten ermöglicht. Ein zentrales Designelement stellt das YAML-basierte Konfigurationssystem dar, welches sowohl die Retrieval-Parameter (beispielsweise Modellname, Top-k-Werte, Gewichtungen im Hybridansatz) als auch Evaluationsmetriken und Pfade zu Datensätzen steuert. Diese Vorgehensweise verhindert die Anwendung von Hard-Coding und ermöglicht die Durchführung von Experimenten durch eine unkomplizierte Anpassung der Konfiguration.

3.2 Datenbasis und Preprocessing

Als Datenbasis dient ein Korpus an FAQ zu RAG und verwandten Themen, der aus 15 Dokumenten besteht und thematisch in Kategorien von Basics bis Future Trends gegliedert ist (identisch zu Grundprojekt). Jede Instanz ist durch eine Frage, eine Antwort, eine Kategorie und Schlagwörter charakterisiert. Für die Evaluation wurde eine Fragenliste im CSV-Format erstellt, die 12 Testfragen unterschiedlicher Schwierigkeitsgrade (drei einfach, fünf mittel, vier schwer) enthält. Das Preprocessing umfasst folgende Schritte:

- **Kombination von Frage und Antwort** zu Embedding-Texten (für Vector Retrieval)
- **Entity- und Concept-Extraction** mit spaCy, um Dokumente für den Graph aufzubereiten
- **Chunking** zur Vorbereitung von größeren Texten (in dieser Arbeit kaum relevant wegen des kleinen Korpus, aber für Erweiterungen vorgesehen)

Die Selektion eines Korpus von FAQs wird durch das Ziel motiviert, eindeutige Ground-Truth-Referenzen zur Verfügung zu stellen. Gleichzeitig stellt die geringe Größe eine Einschränkung dar (siehe Diskussion).

3.3 Retrieval-Methoden

Vector Retrieval Das vektorbasierte Retrieval nutzt SentenceTransformers zur Generierung von Embeddings. Diese werden in einem FAISS-Index gespeichert, der für Cosinus-Suche optimiert ist. Die Retrievalzeit beträgt weniger als eine Sekunde pro Abfrage. Zu den spezifischen Vorzügen zählen Robustheit, Skalierbarkeit und einfache Implementierung. Zu den Schwachstellen zählen: Es ist festzustellen, dass es an expliziter Darstellung von Entitäten und Relationen mangelt.

Graph Retrieval Für das Graph Retrieval wurde Neo4j eingesetzt. Dokumente, Kategorien, Konzepte und Entitäten werden als Knoten modelliert, während Beziehungen die semantischen Verbindungen repräsentieren. Die Ausführung von Queries erfolgt mittels Cypher, während Scores aus Konzept- und Entitätsmatches berechnet werden (vgl.[7]). Zu den spezifischen Kompetenzen, über die das Modell verfügt, zählt die explizite Modellierung semantischer Relationen. Als Schwachstelle ist eine Abhängigkeit von einer effektiven Entity Extraction zu benennen.

Hybrid Retrieval Das Hybridmodul kombiniert Vector- und Graph-Scores. Unterstützt werden drei Fusionsstrategien:

- **Weighted Sum:** Gewichtete Kombination der normalisierten Scores (=default)
- **Reciprocal Rank Fusion (RRF):** Kombination basierend auf Rangpositionen
- **Adaptive Fusion:** Anpassung der Gewichtung an die Query-Länge

Die komplementäre Nutzung beider Verfahren stellt eine Stärke dar. Zu den Schwachstellen zählen: Es wurde eine zusätzliche Komplexität festgestellt, die durch eine starke Kontextsensitivität der Fusionsparameter bedingt ist.

Baseline Als Referenz diente ein LLM ohne Retrieval, das nur auf den Prompt reagierte. Dies erlaubt die Quantifizierung des spezifischen Beitrags des Retrievals.

3.4 Evaluationspipeline und Metriken

Die Evaluation erfolgt automatisiert durch das Metrics-Modul. Verwendet werden:

- **BLEU-Score** (lexikalische Ähnlichkeit mit Smoothing)
- **ROUGE-1, ROUGE-2, ROUGE-L** (Fokus auf Recall-orientierte Ähnlichkeiten)
- **Custom-Metriken**, u.a.:
  - *Semantic Overlap* (Token-Overlap als Approximation semantischer Ähnlichkeit)
  - *Answer Completeness* (Abdeckung der Referenzinhalte)
  - *Factual Consistency* (Übereinstimmung mit abgerufenen Dokumenten)

Die Ergebnisse werden sowohl pro Frage als auch aggregiert in CSV- und JSON-Dateien ausgegeben, um eine systematische Analyse zu ermöglichen.

4 Experimente und Ergebnisse

4.1 Experimentkonfiguration und Testumgebung

Die Experimente wurden unter Verwendung des entwickelten Experiment Runners (run_experiment.py) durchgeführt, welcher sämtliche Schritte von der Datenladung über die Retrievalpipelines bis zur Evaluation orchestriert.

- **Datengrundlage:** FAQ-Korpus mit 15 Dokumenten und 12 Testfragen (3 einfach, 5 mittel, 4 schwer)
- **Verglichene Methoden:** Baseline (ohne Retrieval), Vector Retrieval, Graph Retrieval, Hybrid Retrieval
- **Metriken:** BLEU, ROUGE (1/2/L), Custom-Metriken (semantic overlap, completeness, factual consistency)
- **Testumgebung:** Python 3.13, 4-Kern-CPU, 16 GB RAM, Laufzeit ∼160 s pro Gesamtexperiment

Die Ergebnisse wurden automatisiert in CSV- und JSON-Dateien gespeichert und zum Teil mit Jupyter-Notebooks visualisiert.

4.2 Ergebnisse der Retrieval-Methoden

BLEU- und ROUGE-Scores Die Auswertung der BLEU-Scores zeigt, dass Vector Retrieval im Mittel leicht vor Hybrid und Graph liegt, während die Baseline erwartungsgemäß schwach abschnitt (Abb. 1).

- Vector Retrieval: BLEU Ø 0.0145
- Hybrid Retrieval: BLEU Ø 0.0140
- Graph Retrieval: BLEU Ø 0.0120
- Baseline: BLEU Ø 0.0041

Auch bei ROUGE zeigen Vector und Hybrid die höchsten Werte, Graph liegt knapp dahinter, Baseline deutlich darunter (Abb. 2).

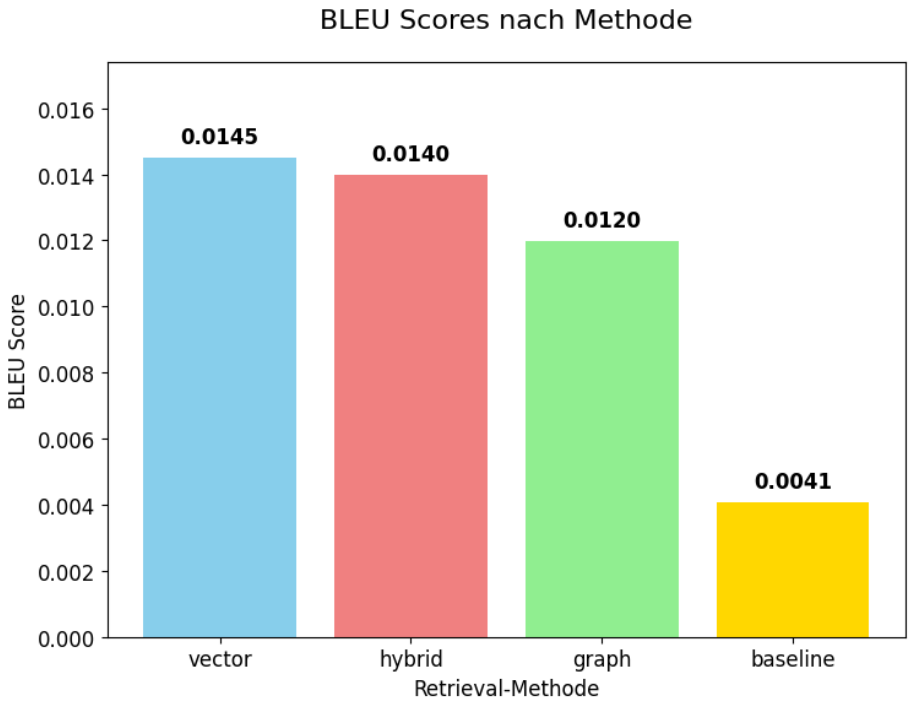

Abb. 1: Vergleich der BLEU-Scores nach Methode

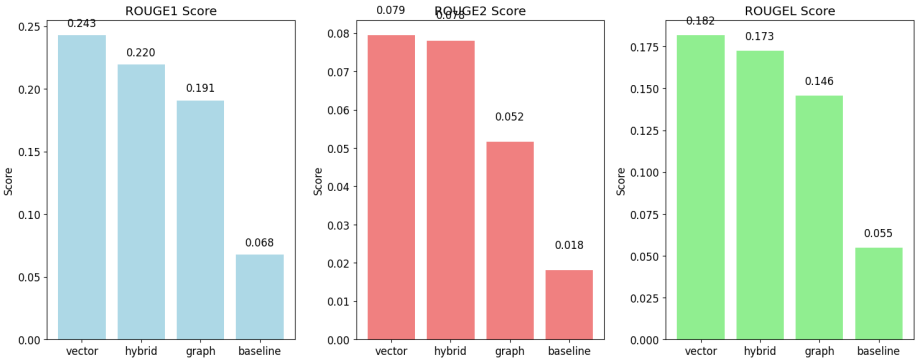

Abb. 2: Vergleich der ROUGE-Scores nach Methode

Laufzeiten und Success Rate Neben den Qualitätsmetriken wurde auch die Systemperformance untersucht. Die Retrievalzeiten lagen bei Vector und Hybrid im Mittel knapp unter zwei Sekunden, Graph Retrieval war etwas langsamer, während die Baseline im Durchschnitt fünf Sekunden benötigte (Abb. 3). Der Success Rate (Anteil an Queries mit mindestens einem relevanten Dokument) lag bei allen Methoden bei 100 %.

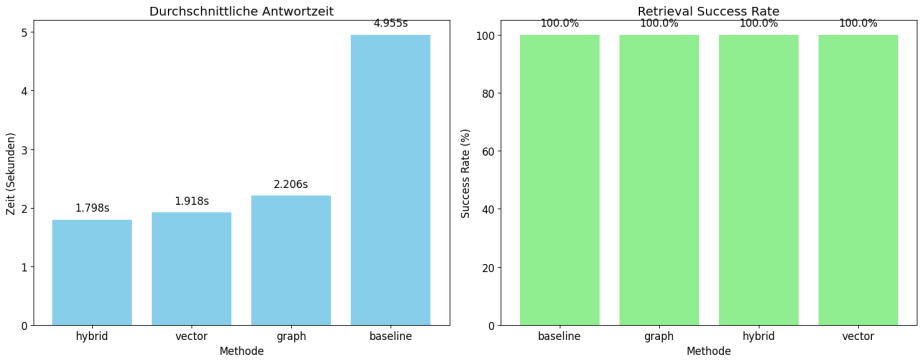

Abb. 3: Antwortzeiten (ms) und Success Rates (%) der Methoden

Ergebnisse nach Schwierigkeitsgraden Die Auswertung nach Schwierigkeitsgraden zeigt ein differenziertes Bild (Abb. 4):

- **Einfach Fragen:** Vector und Hybrid erreichen die besten Werte (BLEU Ø ∼0.023), Graph ist ähnlich stark
- **Mittlere Fragen:** Graph liegt näher an Vector, Hybrid liefert vergleichbare Ergebnisse
- **Schwere Fragen:** Alle Methoden brechen ein, Vector und Hybrid halten sich leicht besser, Graph fällt stärker ab

Diese Analyse verdeutlicht, dass Graph Retrieval bei mittlerer Komplexität durchaus konkurrenzfähig ist, während bei komplexen Fragen die Schwächen der Entity- und Concept-Extraction dominieren.

4.3 Interpretation der Ergebnisse

Die Ergebnisse lassen sich wie folgt zusammenfassen:

1. **Vector Retrieval** erzielte insgesamt die besten Ergebnisse und bestätigt seine Rolle als Standardansatz in RAG-Systemen
2. **Hybrid Retrieval** lag knapp hinter Vector und profitierte insbesondere bei Queries mit gemischter Struktur

3. **Graph Retrieval** schnitt im Durchschnitt schwächer ab, zeigte aber bei mittleren Fragen eine relative Stärke
4. **Baseline** zeigte erwartungsgemäß die schlechtesten Werte, was die zentrale Bedeutung des Retrievals für RAG bestätigt

Ein auffälliges Muster ist die insgesamt niedrige absolute Höhe der Scores (BLEU im Bereich 0,01–0,02, ROUGE-1 im Bereich 0,2). Diese Beobachtung ist weniger als Defizit des Systems zu interpretieren, sondern vielmehr als Hinweis auf die Limitationen der Metriken. Lexikalische Overlap-Messungen erfassen die semantische Ähnlichkeit nur unzureichend und benachteiligen stilistisch abweichende, aber korrekte Antworten. Die Experimente liefern somit nicht nur einen Methodenvergleich, sondern auch kritische Evidenz für die Notwendigkeit semantisch robuster Evaluationsansätze in RAG-Systemen.

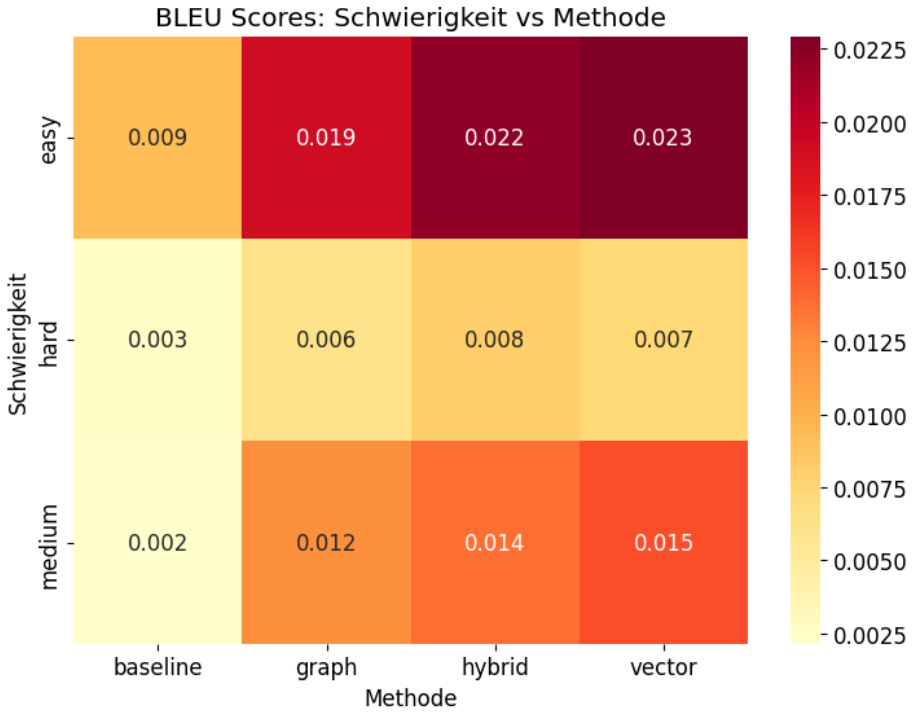

Abb. 4: Heatmap der BLEU-Scores nach Schwierigkeitsgrad (easy, medium, hard)

5 Diskussion

5.1 Ursachenanalyse und Limitationen

Die Resultate des Experiments demonstrieren ein eindeutiges Muster: Vektor- und Hybrid Retrieval erzielen im Durchschnitt höhere BLEU- und ROUGE-Scores als Graphretrieval, während die Baseline erwartungsgemäß eine geringe Leistung aufweist. Die Gründe für die festgestellten Diskrepanzen sind auf eine Vielzahl von Einflussfaktoren zurückzuführen:

1. **Korpusgröße und Domäne:** Der verwendete FAQ-Korpus ist bewusst klein gehalten (15 Dokumente). Vektorbasiertes Retrieval ist in der Lage, auch bei geringen Datenmengen zuverlässig zu arbeiten, während Graphansätze ihre Stärken typischerweise erst in dichteren Wissensnetzen entfalten (vgl. [5])
2. **Entity- und Concept-Extraction:** Die im Projekt eingesetzte generische spaCy-Extraktion wies keine domänen-spezifische Optimierung auf. In der Konsequenz manifestierte sich eine Diskrepanz zwischen den natürlichen Suchanfragen und den im Graph gespeicherten Konzepten. Dieses Problem wurde bereits im Grundprojekt evident
3. **Evaluationsmetriken:** Die niedrigen absoluten Scores sind weniger als Defizit der Methoden zu werten, sondern vielmehr ein Resultat der Limitierungen von BLEU und ROUGE. Die vorliegenden Metriken messen lediglich lexikalische Überlappung und unterschätzen demnach inhaltlich richtige, aber stilistisch abweichende Antworten. Dies resultiert in einem Informationsvorsprung zugunsten vektorbasierter Systeme, die eine größere lexikalische Nähe zu den Referenzantworten aufweisen

Zudem ist die experimentelle Validität aufgrund der geringen Anzahl an Testfragen eingeschränkt. Eine erhöhte Variabilität der Query-Typen hätte mutmaßlich zu differenzierteren Resultaten geführt.

5.2 Modularität als Projektergebnis

Von Relevanz ist weniger der Aspekt der reinen Scores, sondern vielmehr die Architektur des Systems. Im Rahmen des Hauptprojekts wurde eine modulare Vergleichsumgebung entwickelt, welche:

- verschiedene Retrievalmethoden in einer einheitlichen Pipeline integriert,
- Konfigurationen flexibel über YAML steuerbar macht,
- Ergebnisse automatisiert dokumentiert und exportiert,
- und damit eine erweiterbare Testumgebung für zukünftige Experimente bereitstellt.

Dieses Resultat ist von signifikanter praktischer Relevanz, da es Forschenden und Entwickler:innen ermöglicht, neue Retrievalmethoden oder Metriken ohne substanzielle Eingriffe in die bestehende Codebasis zu implementieren. Gemäß dieser Prämisse konstituiert sich der Mehrwert des Projekts nicht in der Optimierung einzelner Methoden, sondern in der Schaffung einer reproduzierbaren Infrastruktur für systematische Vergleiche.

6 Fazit und Ausblick

6.1 Zusammenfassung und Ergebnisse

Im Rahmen des Hauptprojekts wurde eine experimentelle Vergleichsumgebung für Retrievalmethoden in RAG-Systemen erfolgreich entwickelt. Im Gegensatz zum Grundprojekt, das lediglich einen einmaligen Vergleich von Vector- und Graphretrieval vornahm, wurde in diesem Projekt eine modulare und konfigurierbare Infrastruktur entwickelt. Diese ermöglicht den systematischen Vergleich verschiedener Methoden (Vector, Graph, Hybrid, Baseline) unter identischen Bedingungen. Die durchgeführten Experimente zeigten:

- **Vector Retrieval** erzielte die besten BLEU- und ROUGE-Werte und bestätigte damit seine Rolle als Standardverfahren.
- **Hybrid Retrieval** erreichte nahezu gleichwertige Ergebnisse und unterstreicht die Relevanz kombinierter Ansätze.
- **Graph Retrieval** blieb im Durchschnitt schwächer, konnte aber bei mittleren Fragen punktuell konkurrenzfähig sein.
- Die **Baseline** ohne Retrieval schnitt erwartungsgemäß am schlechtesten ab, was die Bedeutung des Retrievalschritts für RAG verdeutlicht.

Auffällig ist die insgesamt niedrige absolute Höhe der Scores. Dieser Aspekt weist weniger auf methodische Defizite als vielmehr auf die Limitationen der eingesetzten Evaluationsmetriken hin.

6.2 Erfüllte Projektziele

Die zentralen Ziele des Projekts wurden erreicht: Aufbau einer modularen Vergleichsumgebung, Integration mehrerer Retrievalmethoden inkl. Hybridverfahren, Nutzung einer einheitlichen Evaluationspipeline mit automatisierter Ergebnisdokumentation, Bereitstellung einer erweiterbaren Architektur für zukünftige Arbeiten. Damit liefert das Projekt eine methodische Grundlage für weiterführende Forschung und stellt einen direkten Fortschritt gegenüber dem Grundprojekt dar.

6.3 Ausblick auf die Masterarbeit

Das Hauptprojekt hat gezeigt, dass die Wahl der Retrievalmethode signifikante Unterschiede in der Antwortqualität bewirken kann – allerdings nur eingeschränkt messbar mit klassischen Metriken. Für die geplante Masterarbeit ergeben sich daraus mehrere Ansatzpunkte:

1. **Erweiterung der Datengrundlage:** Einsatz größerer und domänenspezifischer Korpora, um die Stärken von Graphretrieval in komplexen Beziehungsnetzen besser untersuchen zu können
2. **Qualitativere Evaluation:** Ergänzung der automatischen Scores durch LLM-basierte Bewertungen („LLM-as-a-Judge“) oder semantische Metriken (z. B. BERTScore)

3. **Integration weiterer Methoden:** Untersuchung von Re-Ranking, adaptiven Retrievalstrategien und domänenspezifischem Fine-Tuning von Embeddings
4. **Fokus auf Hybridarchitekturen:** systematische Analyse der Fusionsstrategien, um adaptive Kombinationen von Vector- und Graphansätzen zu optimieren

Langfristig soll die Arbeit zu einem tieferen Verständnis beitragen, wie strukturierte Wissensquellen die Leistungsfähigkeit von LLMs in RAG-Systemen beeinflussen – und welche Designprinzipien sich daraus für die Praxis ableiten lassen.

Literatur

1. Doan, N.N., Härmä, A., Celebi, R., Gottardo, V.: A hybrid retrieval approach for advancing retrieval-augmented generation systems. In: Abbas, M., Freihat, A.A. (eds.) Proceedings of the 7th International Conference on Natural Language and Speech Processing (ICNLSP 2024). pp. 397–409. Association for Computational Linguistics, Trento (Oct 2024), https://aclanthology.org/2024.icnlsp-1.41/
2. Fan, W., Ding, Y., Ning, L., Wang, S., Li, H., Yin, D., Chua, T.S., Li, Q.: A survey on rag meeting llms: Towards retrieval-augmented large language models. In: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. p. 6491–6501. KDD '24, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3637528.3671470, https://doi.org/10.1145/3637528.3671470
3. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., Riedel, S., Kiela, D.: Retrieval-augmented generation for knowledge-intensive nlp tasks. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS '20, Curran Associates Inc., Red Hook, NY, USA (2020)
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. Patel, P.: Graph-enhanced retrieval-augmented question answering for e-commerce customer support (2025), https://arxiv.org/abs/2509.14267
8. 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

9. Zhang, T., Kishore, V., Wu, F., Weinberger, K.Q., Artzi, Y.: Bertscore: Evaluating text generation with bert (2020), https://arxiv.org/abs/1904.09675
10. Zheng, L., Chiang, W.L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E.P., Zhang, H., Gonzalez, J.E., Stoica, I.: Judging llm-as-a-judge with mt-bench and chatbot arena. In: Proceedings of the 37th International Conference on Neural Information Processing Systems. NIPS '23, Curran Associates Inc., Red Hook, NY, USA (2023)
11. 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