WHITEPAPER // ENTERPRISE RAG-ARCHITEKTUREN (V2)

Dynamische Datenqualität: RAG-Optimierung durch Query Transformation und Re-Ranking

Wie moderne KI-Infrastrukturen das fundamentale Problem unpräziser Suchabfragen lösen und Datenströme zur Laufzeit veredeln.

Autor: **Michael Kettel**

Herausgeber: **rms (relationship marketing solutions GmbH)**

Datum: **Juli 2026**

Klassifizierung: **Technischer Leitfaden / Public**

Die klassische Vorstellung von Datenqualität im IT-Umfeld war jahrelang statisch geprägt: Datenbanken wurden bereinigt, Dubletten gelöscht, Schemata validiert und Pflichtfelder erzwungen. Liegen die Rohdaten sauber im Datentopf, stimmte per Definition die Qualität. Doch beim Einsatz von Retrieval-Augmented Generation (RAG) greift diese Denkweise zu kurz.

Selbst eine perfekt kuratierte und fehlerfreie Unternehmenswissensbasis liefert in der Praxis oft miserable oder unvollständige Ergebnisse. Der Grund hierfür liegt selten an den Daten selbst, sondern an der

Schnittstelle zum Menschen: Unpräzise Suchanfragen der Nutzer, mangelndes Verständnis des Retrieval-Systems für linguistische Nuancen oder das "Rauschen" innerhalb großer Ergebnismengen.

Datenqualität darf bei RAG-Systemen nicht mehr als statischer Zustand begriffen werden – sie ist ein dynamischer Prozess, der direkt während der Abfrage zur Laufzeit stattfindet. Die entscheidenden Architektur-Stellschrauben moderner Systeme sitzen daher unmittelbar vor und nach der Vektordatenbank: die Vorwärtsoptimierung (Query Transformation & Expansion) und die Rückwärtsoptimierung (Re-Ranking).

1. Vorwärtsoptimierung: Brücken bauen mit Query Transformation

Nutzer im geschäftlichen Alltag sind keine Information-Retrieval-Experten. Sie stellen vage Fragen, nutzen informellen Slang, Abkürzungen oder tippen schlicht fehlerhafte Suchphrasen ein. Würde man diese initialen Anfragen direkt als mathematischen Vektor an die Vektordatenbank senden, blieben die Ergebnisse weit hinter den Erwartungen zurück, da das Embedding einer unpräzisen Anfrage relevante semantische Aspekte häufig nicht vollständig abbildet.

Die Vorwärtsoptimierung löst dieses Problem, indem sie spezialisierte Query-Rewriter, kleinere LLMs, Distilled Models oder regelbasierte Verfahren gezielt *vor* die eigentliche Suche schaltet, um den Prompt semantisch anzureichern und zu transformieren. Ergänzend steuert ein intelligentes **Query Routing** die Anfrage je nach Komplexität an maßgeschneiderte Pipelines, während **Adaptive Retrieval** flexibel entscheidet, wann überhaupt eine externe Suche notwendig ist.

Query Expansion & Paraphrasierung

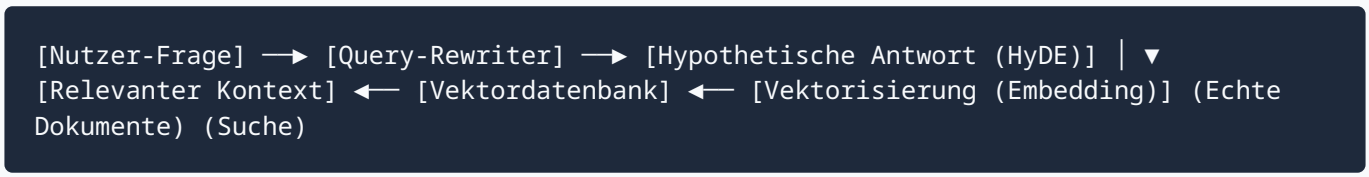
Gibt ein Mitarbeiter beispielsweise die unpräzise Frage *„Wie läuft die Kündigung bei uns?“* ein, transformiert das vorgeschaltete System diese Anfrage parallel in mehrere hochpräzise Suchabfragen:

- „Welche gesetzlichen und betrieblichen Fristen gelten für eine Arbeitsvertragsbeendigung?“
- „Prozessablauf und notwendige Schritte bei einer Eigenkündigung des Mitarbeiters.“
- „Aufhebungsvertrag, Resturlaub und Kündigungsfristen laut HR-Richtlinie.“

Durch das parallele Abfeuern dieser Varianten kann das System den Recall deutlich erhöhen. Es spielt keine Rolle mehr, ob im Zieldokument exakt das Wort „Kündigung“ oder stattdessen „Vertragsbeendigung“ steht. Zudem stellt ein vorgeschaltetes **Metadata Filtering** sicher, dass die Suche von vornherein auf relevante Zeiträume oder Abteilungen eingegrenzt wird.

HyDE (Hypothetical Document Embeddings)

Eine besonders innovative Technik ist das sogenannte **HyDE-Verfahren**. Anstatt direkt nach der Frage des Nutzers im Datenbestand zu suchen, generiert die Pipeline eine rein hypothetische, künstliche Antwort.



Der mathematische Hintergrund: Das hypothetische Dokument besitzt häufig eine ähnliche semantische Struktur wie relevante Zieldokumente und erzeugt dadurch ein Embedding, das näher an den relevanten Dokumenten liegt als das Embedding der ursprünglichen Frage. Das HyDE-Dokument fungiert hierbei als hocheffizienter „Suchmagnet“, der präzise die echten Dokumente mit ähnlicher linguistischer Textstruktur anzieht. Eventuelle inhaltliche Fehler in der hypothetischen Antwort sind irrelevant, da das Dokument nach der Suche verworfen wird.

2. Rückwärtsoptimierung: Rauschen filtern mit Cross-Encoder Re-Ranking

Moderne Enterprise-Systeme setzen im ersten Schritt meist auf ein **Hybrid Retrieval**, das die klassische Keyword-Suche (BM25) mit dichten Vektoren (Dense Retrieval) kombiniert, oder nutzen hochentwickelte Multi-Vector Retrieval-Ansätze wie **ColBERT**. Dennoch berechnen die initialen Retriever die Relevanz extrem schnell, was zu Lasten der tiefen, feingranularen semantischen Beziehung zwischen Frage und Textabschnitt geht.

Die logische Konsequenz: Die Datenbank wirft zwar eine Menge relevanter Daten aus (z.B. die Top 25 Chunks), unter den ersten Plätzen befinden sich jedoch häufig partielle semantische Überlappungen oder oberflächliche Ähnlichkeiten, die die Frage eigentlich gar nicht beantworten. An diesem Punkt setzt die Rückwärtsoptimierung an.

Der Cross-Encoder als ultimativer Qualitätsfilter

Ein nachgelagertes **Cross-Encoder-Modell** (wie beispielsweise ein *BGE-Reranker* oder *Cohere Re-rank*) nimmt die originale Nutzerfrage und analysiert diese zusammen mit jedem einzelnen der abgerufenen Textfragmente in einem einzigen, tiefen Rechenschritt. Es bewertet die direkte Interaktion und logische Passfähigkeit von Frage und Antwort Satz für Satz.

> ***Die goldene Regel moderner RAG-Architekturen lautet:*** *Weit fischen (z.B. Top 25-50 via Query Expansion für maximalen Recall), aber extrem spitz und präzise füttern (Top 3-5 via Re-Ranking für maximale Precision).*

Der Re-Ranker sortiert die Dokumente radikal um: Ein Textfragment, das bei der reinen Vektorsuche aufgrund abweichender Begrifflichkeiten weit hinten landete, wird vom Cross-Encoder als logisch exakte Antwort erkannt und nach vorne gehoben. Durch eine anschließende **Context Compression** werden zudem redundante Informationen aus den Chunks herausgefiltert, bevor die Daten an das finale Generierungs-Modell übergeben werden.

Vermeidung des „Lost in the Middle“-Phänomens & Evaluierung

Das bekannte **„Lost in the Middle“-Phänomen** belegt, dass LLMs Informationen am Anfang und am Ende eines langen Prompts extrem stark gewichten. Durch die Rückwärtsoptimierung mittels Re-Ranking wird sichergestellt, dass dem Modell ausschließlich die absolut essenziellen Chunks übergeben werden. Dies senkt nicht nur die Token-Kosten und minimiert die Latenzzeiten der Anwendung drastisch, sondern verringert das Risiko halluzinierter Antworten deutlich.

Um diesen dynamischen Kreislauf dauerhaft abzusichern, ist eine kontinuierliche **RAG-Evaluierung** unabdingbar. Während klassische Information-Retrieval-Metriken wie **Recall@k**, **nDCG** (Normalized Discounted Cumulative Gain) und **MRR** (Mean Reciprocal Rank) die Qualität der Vor- und Rückwärtsoptimierung mathematisch präzise messen, bewerten automatisierte LLM-as-a-Judge-Frameworks fortlaufend die **Faithfulness** (Faktentreue) und Antwortrelevanz des Gesamtsystems.

Fazit und Ausblick

Unternehmen, die Datenqualität im Kontext von Generativer KI und RAG-Infrastrukturen isoliert als statische Bereinigung von Quellsystemen betrachten, verschenken massives Potenzial und riskieren instabile Systeme im Produktivbetrieb.

Erst die Implementierung dynamischer Veredelungsstufen im Datenfluss – durch intelligente Vorwärtsoptimierung bei der Anfrage und präzise Rückwärtsoptimierung nach dem Retrieval – transformiert ungenaue Nutzerinteraktionen in verlässliche, hochqualitative Systemantworten. Für zukunftssichere Enterprise-Anwendungen bildet dieser Ansatz das qualitative Fundament.