RAG – Wissenssysteme mit LLMs entwickeln

Eigene Wissensbestände zuverlässig für Sprachmodelle nutzbar machen

01 INTRO

Sprachmodelle verfügen über umfangreiches allgemeines Wissen, kennen jedoch weder automatisch interne Unternehmensinformationen noch aktuelle oder projektspezifische Daten. Retrieval-Augmented Generation verbindet LLMs deshalb mit externen Wissensbeständen: Für eine Anfrage werden relevante Informationen gesucht, ausgewählt und dem Modell als Kontext für die Antwort bereitgestellt.

Der Kurs vermittelt die Entwicklung belastbarer RAG-Systeme von der Datenaufbereitung über Indexierung und Retrieval bis zur Generierung und Evaluation. Die Teilnehmer bauen schrittweise ein eigenes Wissenssystem auf und untersuchen, welche Architekturentscheidungen tatsächlich Einfluss auf die Qualität haben. Im Mittelpunkt stehen nicht Framework-Rezepte, sondern nachvollziehbares Retrieval, Quellenbezug, messbare Qualität und der Weg vom funktionierenden Prototyp zum betreibbaren System.

DAUER

3 Tage

FORMAT

Präsenz · Live Online · Inhouse

VORKENNTNISSE

Praktische Programmiererfahrung, vorzugsweise mit Python, und grundlegendes Verständnis von LLMs

02 TRAININGSPROGRAMM

Inhalte

RAG einordnen

- Grenzen des Modellwissens
- internes und aktuelles Wissen
- Wissensbestände zur Laufzeit einbinden
- Retrieval-Augmented Generation
- typische Anwendungsfälle
- dokumentenbasierte Assistenzsysteme
- Unternehmenswissen
- Grenzen von RAG

RAG, langer Kontext und Fine-Tuning

- Informationen direkt im Prompt
- große Kontextfenster
- Retrieval zur Laufzeit
- Modellwissen
- Fine-Tuning
- unterschiedliche Zielsetzungen
- Kombination der Verfahren
- Entscheidung nach Anwendungsfall

Architektur eines RAG-Systems

- Datenquellen
- Ingestion
- Aufbereitung
- Chunking
- Repräsentation
- Index
- Retrieval
- Kontextaufbau
- Generierung und Quellen

Vom Dokument zur Wissensbasis

- Dokumente erfassen
- Inhalte extrahieren
- normalisieren
- strukturieren
- segmentieren
- Metadaten ergänzen
- indexieren
- Verarbeitung reproduzierbar gestalten

Datenquellen

- Text und Markdown
- HTML
- PDF
- Office-Dokumente
- Wikis
- technische Dokumentationen
- Datenbanken
- strukturierte und unstrukturierte Informationen

Dokumentstruktur erhalten

- Überschriften
- Abschnitte
- Listen
- Tabellen
- Seiten und Kapitel
- Hierarchien
- Dokumentidentität
- Struktur als Retrieval-Signal

Datenqualität

- Dubletten
- veraltete Dokumente
- widersprüchliche Informationen
- unvollständige Inhalte
- OCR- und Extraktionsfehler
- irrelevante Bestandteile
- Versionen
- schlechte Daten werden durch RAG nicht automatisch gut

Chunking

- warum Inhalte segmentiert werden
- feste Chunk-Größen
- Überlappungen
- semantische Grenzen
- strukturorientiertes Chunking
- parent-child-Strukturen
- zu kleine und zu große Chunks
- Chunking als Qualitätsentscheidung

Chunking-Strategien vergleichen

- gleiche Dokumente unterschiedlich segmentieren
- Informationsverlust
- Kontextverlust
- redundante Treffer
- Retrieval-Qualität
- Antwortqualität
- Messung statt Bauchgefühl
- Strategie nach Dokumenttyp

Embeddings

- semantische Repräsentation
- Vektoren
- Embedding-Modelle
- Query- und Dokument-Embeddings
- Ähnlichkeit
- Modellwahl
- mehrsprachige Inhalte
- Grenzen von Embeddings

Ähnlichkeitssuche

- Similarity Search
- Distanz- und Ähnlichkeitsmaße
- Top-k
- Schwellenwerte
- relevante und nur ähnliche Inhalte
- Query und Dokument unterschiedlich formuliert
- Ergebnisse untersuchen
- Retrieval nicht mit Antwortqualität verwechseln

Vector Stores und Indizes

- Aufgabe eines Vector Stores
- lokale Lösungen
- FAISS
- Datenbanken mit Vektorunterstützung
- spezialisierte Vector Databases
- Persistenz
- Metadaten
- Auswahl nach Anforderungen

Metadaten

- Quelle
- Dokumenttyp
- Datum
- Version
- Fachgebiet
- Autor oder Herkunft
- Zugriffsbereich
- Metadaten als Bestandteil des Retrievals

Metadata Filtering

- Suche einschränken
- Zeiträume
- Dokumenttypen
- Versionen
- Mandanten
- Benutzerrechte
- semantische und strukturierte Kriterien kombinieren
- Filter bereits beim Retrieval anwenden

Klassische Informationssuche

- Keyword Search
- Volltextsuche
- BM25
- exakte Begriffe
- Produktnamen und Kennungen
- semantische Suche ist nicht immer überlegen
- Stärken klassischer Retrieval-Verfahren
- geeignete Kombinationen

Hybrid Search

- lexikalisches und semantisches Retrieval
- unterschiedliche Trefferlisten
- Scores
- Fusion
- Ranking
- Fachbegriffe und semantische Ähnlichkeit verbinden
- typische Einsatzfälle
- Qualität empirisch vergleichen

Query Processing

- Benutzerfrage und Retrieval Query
- Query Normalization
- Schlüsselbegriffe
- Synonyme
- Mehrdeutigkeiten
- Query Expansion
- mehrere Suchanfragen
- ursprüngliche Nutzerintention erhalten

Query Rewriting

- Suchanfrage aus der Frage ableiten
- unklare Fragen präzisieren
- Gesprächskontext berücksichtigen
- mehrere Teilfragen
- unterschiedliche Formulierungen
- LLM für Query Transformation
- Fehler durch falsches Rewriting
- Originalfrage nicht verlieren

Multi-Query Retrieval

- mehrere Suchperspektiven
- alternative Formulierungen
- Teilfragen
- Treffer zusammenführen
- Duplikate
- zusätzlicher Recall
- zusätzliche Kosten
- nur einsetzen, wenn messbarer Nutzen entsteht

Reranking

- Retrieval erzeugt Kandidaten
- zweite Ranking-Stufe
- semantisches Reranking
- Cross Encoder
- LLM-basiertes Reranking
- Relevanz gegenüber Ähnlichkeit
- Kosten und Latenz
- Nutzen evaluieren

Kontext für das LLM aufbauen

- relevante Chunks auswählen
- Reihenfolge
- Quelleninformationen
- redundante Inhalte
- widersprüchliche Inhalte
- Kontextbudget
- Nutzerfrage und Retrieval-Ergebnisse
- klare Trennung von Instruktion und Wissensmaterial

Kontextkompression

- irrelevante Bestandteile entfernen
- relevante Passagen auswählen
- Zusammenfassungen
- Informationsverlust
- zusätzliche Modellaufrufe
- Kosten
- Nutzen
- Originalquellen erhalten

Antwortgenerierung

- Antwort aus bereitgestelltem Kontext
- System Instructions
- Quellenbezug
- keine Information vorhanden
- Unsicherheit
- Antwortformat
- Zitate und Referenzen
- Modell nicht zum Erfinden fehlender Fakten verleiten

Quellen und Nachvollziehbarkeit

- Dokumentquelle erhalten
- Fundstelle
- Chunk und Ursprungsdokument
- Quellen in Antworten
- überprüfbare Aussagen
- Link zum Original
- Provenance
- Vertrauen durch Nachprüfbarkeit statt durch Formulierung

Wenn keine Antwort vorhanden ist

- Retrieval findet nichts
- irrelevante Treffer
- Wissensbasis enthält die Information nicht
- Schwellenwerte
- Abstention
- Rückfragen
- alternative Suche
- eine ausbleibende Antwort als valides Ergebnis

Halluzinationen in RAG-Systemen

- Retrieval verhindert Halluzinationen nicht automatisch
- falscher Kontext
- unzureichender Kontext
- Modell ignoriert Kontext
- Modell ergänzt eigenes Wissen
- widersprüchliche Quellen
- erfundene Quellenbezüge
- Ursachen getrennt analysieren

Fehlerquellen systematisch trennen

- Ingestion-Fehler
- Chunking-Fehler
- Embedding-Fehler
- Retrieval-Fehler
- Ranking-Fehler
- Kontextfehler
- Generierungsfehler
- Pipeline statt nur Endantwort untersuchen

Retrieval evaluieren

- Testfragen
- relevante Dokumente
- Ground Truth
- Trefferquote
- Precision
- Recall
- Hit Rate
- Ranking-Metriken
- Retrieval unabhängig vom LLM messen

Antwortqualität evaluieren

- fachliche Richtigkeit
- Relevanz
- Vollständigkeit
- Faithfulness
- Quellenbezug
- unbelegte Aussagen
- unbeantwortbare Fragen
- menschliche und automatische Bewertung

Evaluationsdatensatz entwickeln

- repräsentative Fragen
- einfache und schwierige Fälle
- unterschiedliche Dokumentbereiche
- bekannte Antworten
- relevante Quellen
- unbeantwortbare Fragen
- Edge Cases
- Datensatz versionieren

Automatisierte Evaluation

- regelbasierte Prüfungen
- Retrieval-Metriken
- strukturierte Antwortprüfungen
- semantische Bewertung
- LLM-as-a-Judge
- Grenzen automatischer Bewertung
- Stichproben durch Menschen
- mehrere Signale kombinieren

Regressionstests

- Änderungen an Chunking
- neues Embedding-Modell
- Retrieval-Parameter
- Reranker
- Promptänderungen
- Modellwechsel
- Evaluation vorher und nachher
- Verbesserung nicht nur anhand einzelner Beispiele beurteilen

Berechtigungen und Zugriffsschutz

- Benutzer darf nicht jedes Dokument sehen
- Dokumentberechtigungen
- Metadatenfilter
- Mandantentrennung
- Zugriff bereits beim Retrieval begrenzen
- Modell ist keine Zugriffskontrolle
- Quellen nicht versehentlich offenlegen
- Least Privilege

Prompt Injection über Wissensquellen

- Dokumente als untrusted Input
- manipulierte Inhalte
- indirekte Prompt Injection
- Dokumenttext und System Instructions trennen
- Retrieval-Ergebnisse nicht als Befehle behandeln
- Tool-Zugriffe
- technische Sicherheitsgrenzen
- Defense in Depth

Aktualisierung der Wissensbasis

- neue Dokumente
- geänderte Dokumente
- gelöschte Dokumente
- Versionen
- inkrementelle Indexierung
- Re-Embedding
- veraltete Chunks entfernen
- konsistenter Indexzustand

Dokument-Lifecycle

- Quelle
- Import
- Version
- Indexierung
- Aktualisierung
- Archivierung
- Löschung
- Nachvollziehbarkeit über den gesamten Lebenszyklus

Performance

- Anzahl der Dokumente
- Indexgröße
- Retrieval-Latenz
- Embedding-Kosten
- Reranking
- Modelllatenz
- Caching
- Qualität und Geschwindigkeit ausbalancieren

Kosten

- Embeddings
- Indexierung
- Storage
- Retrieval
- zusätzliche Modellaufrufe
- Reranking
- große Kontexte
- Qualität gegen Kosten messen

Observability

- Benutzerfrage
- erzeugte Retrieval Query
- gefundene Dokumente
- Scores
- ausgewählter Kontext
- Antwort
- Laufzeit
- Fehler

RAG-Debugging

- schlechte Antwort reproduzieren
- Retrieval-Ergebnisse sichtbar machen
- erwartete Dokumente suchen
- Chunking überprüfen
- Query untersuchen
- Ranking analysieren
- Prompt und Kontext kontrollieren
- Fehler an der richtigen Pipeline-Stufe beheben

Framework oder eigene Pipeline?

- Frameworks beschleunigen Prototypen
- Abstraktionen
- versteckte Defaults
- Debugging
- Abhängigkeiten
- eigene Komponenten
- Framework gezielt einsetzen
- Architektur nicht vom Framework bestimmen lassen

RAG als Service

- API für Wissensabfragen
- Trennung von UI und RAG-Pipeline
- mehrere Anwendungen
- Authentifizierung
- Benutzerkontext
- Skalierung
- Versionierung
- wiederverwendbarer Knowledge Service

Integration in Anwendungen

- Chat
- Suche
- Support-Systeme
- Fachanwendungen
- interne Wissensassistenten
- APIs
- Benutzerfeedback
- Quellen direkt zugänglich machen

RAG und Agenten

- Retrieval als Tool
- Agent entscheidet über Suche
- mehrstufige Recherche
- unterschiedliche Wissensquellen
- Retrieval-Ergebnisse als Agentenkontext
- Agentenarchitektur bleibt getrennt
- zusätzliche Autonomie nur bei Bedarf
- RAG funktioniert auch ohne Agenten

Vom Prototyp zum produktiven Wissenssystem

- Datenqualität sichern
- Evaluation etablieren
- Berechtigungen
- Aktualisierungsprozess
- Monitoring
- Regressionstests
- Kosten und Performance
- klare Betriebsverantwortung

Praktische Übungen

- Dokumentbestand analysieren und aufbereiten
- unterschiedliche Chunking-Strategien vergleichen
- Embeddings erzeugen
- Vektorindex aufbauen
- semantische Suche implementieren
- Metadatenfilter einsetzen
- klassische und semantische Suche vergleichen
- Hybrid Search implementieren
- Reranking ergänzen
- Quellen in Antworten zurückführen
- unbeantwortbare Fragen behandeln
- Evaluationsdatensatz erstellen
- Retrieval und Antwortqualität messen
- Fehler einer RAG-Pipeline systematisch analysieren
- vollständiges RAG-System entwickeln und bewerten

03 DURCHFÜHRUNG

Rahmen und Organisation

Dauer	3 Tage
Durchführung	Präsenz oder Live Online, vorzugsweise als Inhouse-Schulung
Schwerpunkt	Inhouse-Schulungen für Unternehmen und Teams
Sprache	Deutsch, englische Fachbegriffe und Dokumentation
Unterlagen	Kursunterlagen, Beispiele und Übungsaufgaben
Technik	Eigener Rechner mit Python, Entwicklungsumgebung und Zugang zu einem aktuellen Sprachmodell

> **Inhouse-Schulungen**
> Geeignete eigene Dokumentbestände, Wissensquellen und fachliche Fragestellungen können als Grundlage für die Übungen verwendet werden.

04 ZIELGRUPPE

Für wen ist der Kurs gedacht?

Der Kurs richtet sich an Softwareentwickler, AI Engineers, Data Engineers, Softwarearchitekten und technische Consultants, die Sprachmodelle mit eigenen Dokumenten und Wissensbeständen verbinden möchten.

Er eignet sich insbesondere für Teams, die über einfache Prototypen zum Befragen von Dokumenten hinaus belastbare, nachvollziehbare und evaluierbare Wissenssysteme entwickeln wollen.

> **Voraussetzungen**
> Praktische Programmiererfahrung wird vorausgesetzt; Python-Kenntnisse sind für die Übungen sinnvoll. Grundlegendes Verständnis generativer KI und Sprachmodelle wird erwartet. Kenntnisse zu Vektordatenbanken, Information Retrieval oder Machine Learning sind nicht erforderlich.

05 LERNZIELE

Was Sie nach dem Kurs können

Nach dem Kurs können die Teilnehmer:

- RAG gegenüber Modellwissen, langen Kontexten und Fine-Tuning abgrenzen
- eine vollständige RAG-Architektur entwerfen
- unterschiedliche Datenquellen aufbereiten
- geeignete Chunking-Strategien entwickeln und vergleichen
- Embeddings und Vektorsuche einsetzen
- Vector Stores nach Anforderungen auswählen
- Metadaten und Zugriffsfilter verwenden
- klassische, semantische und hybride Suche kombinieren
- Queries für Retrieval aufbereiten
- Reranking gezielt einsetzen
- geeigneten Kontext für ein Sprachmodell zusammenstellen
- Quellen und Provenance erhalten
- unbeantwortbare Fragen kontrolliert behandeln
- Fehler in einzelnen Pipeline-Stufen lokalisieren
- Retrieval und Antwortqualität getrennt evaluieren
- Evaluationsdatensätze und Regressionstests entwickeln
- Zugriffsrechte und Prompt-Injection-Risiken berücksichtigen
- Wissensbestände konsistent aktualisieren
- Performance und Kosten analysieren
- ein RAG-System vom Prototyp zu einer betreibbaren Lösung weiterentwickeln

06 FAQ

Häufige Fragen

Was unterscheidet diesen Kurs vom LLM-Grundlagenkurs?

Der Grundlagenkurs gibt einen Überblick über generative KI und zeigt RAG als eine Möglichkeit, eigenes Wissen einzubinden. Dieser Kurs behandelt die technische Entwicklung eines RAG-Systems vollständig – von Datenaufbereitung und Retrieval bis Evaluation und Betrieb.

Was unterscheidet ihn vom Agentenkurs?

Der Agentenkurs behandelt Agenten, die unter anderem Retrieval als Werkzeug einsetzen können. Hier steht das Wissenssystem selbst im Mittelpunkt. Ein RAG-System benötigt keinen Agenten und kann als eigenständiger Dienst oder Bestandteil einer Anwendung betrieben werden.

Ist der Kurs eine Schulung für eine bestimmte Vektordatenbank?

Nein. Unterschiedliche technische Lösungen werden praktisch betrachtet, der Schwerpunkt liegt jedoch auf den zugrunde liegenden Retrieval- und Architekturprinzipien. Die Wahl eines Produkts folgt den Anforderungen des jeweiligen Systems.

Ist RAG einfach eine Vektorsuche vor einem LLM-Aufruf?

Das ist die einfachste Variante, reicht für belastbare Wissenssysteme aber häufig nicht aus. Datenqualität, Chunking, Metadaten, klassische Suche, Hybrid Retrieval, Reranking, Quellenbezug und Evaluation können erheblichen Einfluss auf das Ergebnis haben.

Wird Hybrid Search behandelt?

Ja. Klassische lexikalische und semantische Suche werden zunächst getrennt betrachtet und anschließend kombiniert.

Wird Reranking behandelt?

Ja. Reranking wird als zusätzliche Retrieval-Stufe behandelt und praktisch gegen einfaches Top-k-Retrieval verglichen.

Werden RAG-Systeme auch getestet?

Ja. Evaluation ist ein zentraler Bestandteil des Kurses. Retrieval und generierte Antwort werden bewusst getrennt bewertet, damit Fehler nicht nur anhand der fertigen Antwort beurteilt werden.

Wird LLM-as-a-Judge eingesetzt?

Es kann als ein Evaluationssignal eingesetzt werden. Der Kurs behandelt aber ausdrücklich dessen Grenzen und kombiniert automatische Modellbewertungen mit Retrieval-Metriken, deterministischen Prüfungen und menschlichen Stichproben.

Wird Prompt Injection behandelt?

Ja. Dokumente und andere Wissensquellen sind externe Eingaben und können manipulierte Instruktionen enthalten. Deshalb werden Trust Boundaries und technische Schutzmechanismen für RAG-Systeme behandelt.

Brauche ich eine Vektordatenbank?

Nicht zwangsläufig. Die geeignete Speicher- und Suchtechnik hängt von Umfang, Datenstruktur und Anforderungen ab. Der Kurs betrachtet deshalb auch klassische Informationssuche und hybride Ansätze.

Warum dauert der Kurs drei Tage?

Ein belastbares RAG-System besteht nicht nur aus Embeddings und einer Vektordatenbank. Datenaufbereitung, Chunking, Retrieval, Hybrid Search, Reranking, Kontextaufbau, Quellen, Evaluation, Security und Betrieb müssen gemeinsam betrachtet und praktisch erprobt werden. Zwei Tage würden diese Themen auf einen Prototypen-Workshop reduzieren.

Kann der Kurs mit unseren eigenen Dokumenten durchgeführt werden?

Ja. Bei Inhouse-Schulungen können geeignete eigene Dokumentbestände, Wissensquellen und fachliche Fragestellungen als Grundlage für die Übungen verwendet werden.

UC IT Service · Ulrich Cuber · kontakt@uc-it.de
https://www.uc-it.de/coaching/rag-wissenssysteme/