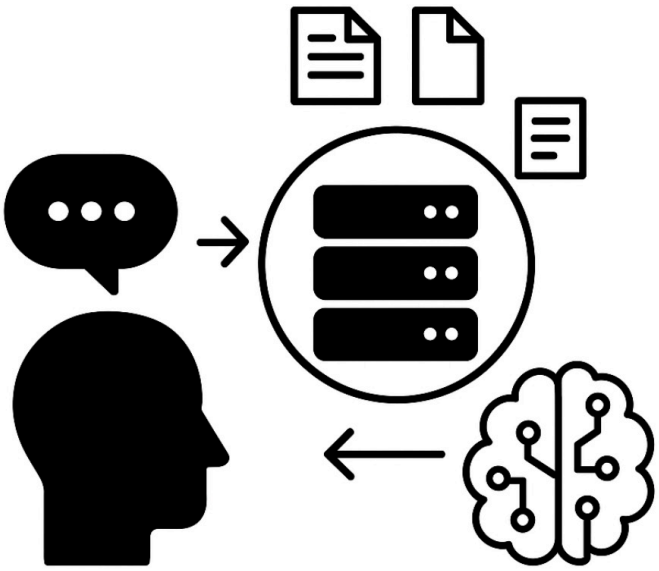
makandra



LOKALES LLM MIT RAG

Interne Daten sicher und smart nutzen

KI AUS EIGENER HAND – WARUM LOKALE MODELLE SINNVOLL SIND

Einleitung

Generative Sprachmodelle (LLMs) sind zu einem wichtigen Werkzeug im Unternehmen geworden. Sie unterstützen bei Aufgaben wie Textgenerierung, Kundenservice und Wissensmanagement. Gleichzeitig wächst der Anspruch: Wie stellt man sicher, dass KI-Modelle auf aktuelle und vertrauliche Informationen zugreifen – ohne Daten preiszugeben oder Abhängigkeiten zu schaffen?

Lokale LLMs, betrieben auf eigener Infrastruktur, bieten hierfür eine Lösung: volle Kontrolle über Daten, hohe Flexibilität und Unabhängigkeit von Cloud-Anbietern. Allerdings haben auch leistungsfähige Modelle wie Mistral oder LLaMA eine natürliche Grenze: Sie basieren auf dem Wissen, das ihnen zum Zeitpunkt ihres Trainings zur Verfügung stand.

Hier setzt Retrieval-Augmented Generation (RAG) an: Durch RAG wird das Sprachmodell erweitert, indem es aktuelle Informationen aus externen Quellen dynamisch abrufen kann – zum Beispiel aus Unternehmensdokumenten oder Datenbanken.

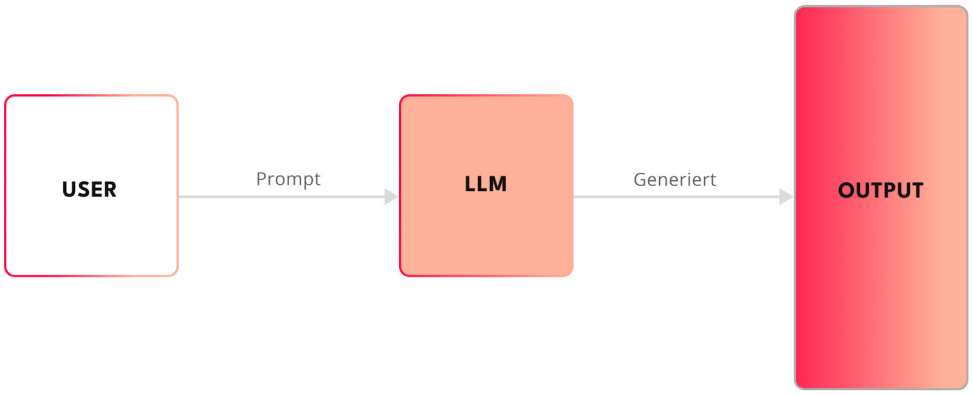
Gerade im Zusammenspiel mit Open-Source-LLMs entsteht so eine leistungsfähige, datenschutzfreundliche Architektur: Sprachliche Intelligenz kombiniert mit unternehmensspezifischem Wissen.

INHALTSVERZEICHNIS

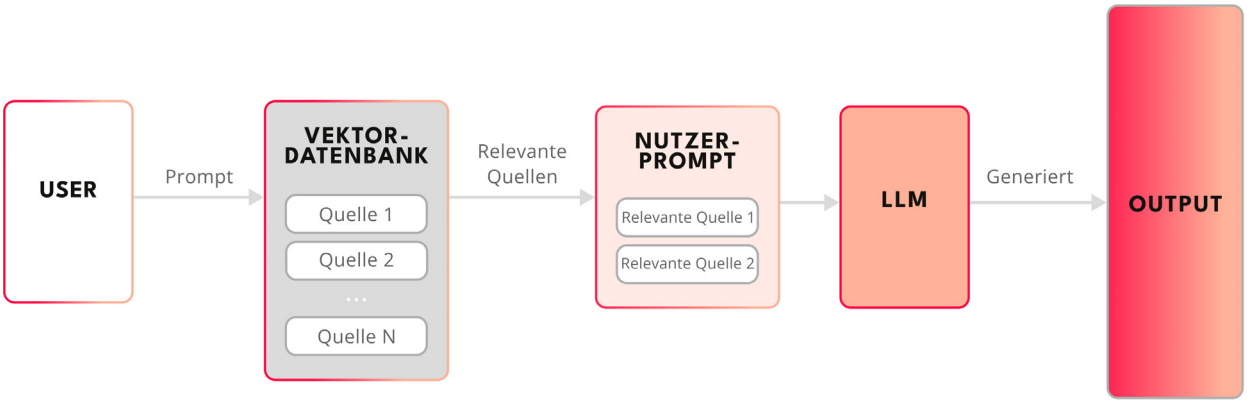
1. Was ist Retrieval-Augmented Generation (RAG)?	01
2. RAG-Grundlagen: Vektor-Datenbank und Embedding-Modell	02
3. Datenvorbereitung: Grundlage jeder RAG-Anwendung - Schritte	03
4. Herausforderungen bei einer RAG-basierten Anwendung	05
5. Evaluation von RAG-Systemen: Wie misst man Qualität?	06
6. Use Case: Wissenstransfer mit lokalem LLM + RAG	07
Fazit	09

1. WAS IST RETRIEVAL-AUGMENTED GENERATION (RAG)?

Ein **Large Language Model (LLM)**, das ausschließlich mit öffentlichen Daten trainiert wurde, verfügt über keinerlei Kenntnisse interner Unternehmensinformationen. Ohne ergänzenden Kontext kann es daher entweder keine brauchbare Antwort geben oder beginnt, Inhalte zu halluzinieren.



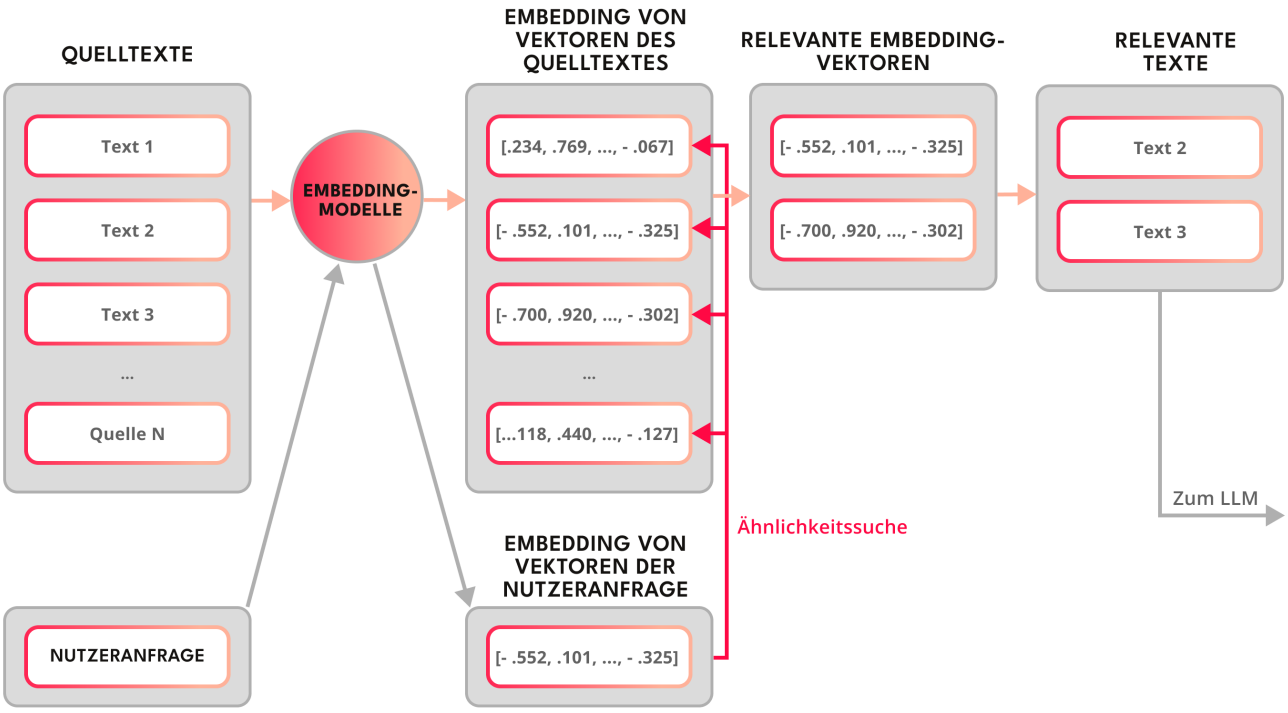
Retrieval-Augmented Generation (RAG) löst dieses Problem, indem es relevante Informationen aus externen Quellen dynamisch in den Prompt integriert. Das bedeutet: Vor der eigentlichen Anfrage wird der ursprüngliche Prompt durch passende Inhalte – etwa aus Textdokumenten, Podcasts, Videos oder strukturierten Datenbanken – angereichert. Diese Informationen werden mithilfe einer Vektordatenbank ermittelt und zusammen mit der eigentlichen Frage dem LLM, beispielsweise LLaMA, als Kontext übergeben. Auf diese Weise entstehen Antworten, die präziser, aktueller und spezifischer auf die jeweilige Informationsbasis zugeschnitten sind.



2. RAG-GRUNDLAGEN: VEKTOR-DATENBANK UND EMBEDDING-MODELL

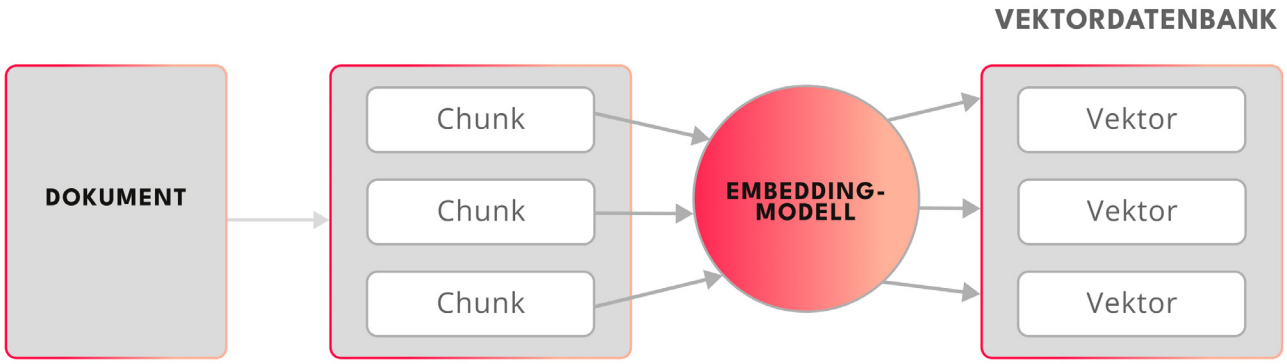
Eine RAG-Anwendung muss in der Lage sein, aus großen Datenmengen genau die Informationen herauszufiltern, die zum Prompt des Nutzers passen – und diese dem LLM gezielt bereitstellen. Gerade bei umfangreichen Dokumentbeständen ist das eine Herausforderung: Aus Tausenden oder Millionen von Texten müssen in Millisekunden die relevantesten Passagen gefunden werden.

Die Lösung dafür ist eine sogenannte Vektorsuche in einer Vektordatenbank. Für den Aufbau einer Vektordatenbank übersetzt ein spezieller Typ von Sprachmodell – ein sogenanntes Embedding-Modell – jeden zu durchsuchenden Text in einen numerischen Vektor: eine Zahlenreihe, die die Bedeutung des Textes repräsentiert. Dasselbe Modell wandelt auch die Nutzeranfrage in einen vergleichbaren Vektor um. Dadurch wird es möglich, die Anfrage mathematisch mit allen Texten zu vergleichen und diejenigen zu identifizieren, die inhaltlich am ähnlichsten und somit am relevantesten sind. Auf dieser Basis lassen sich semantisch passende Inhalte effizient und präzise identifizieren – auch wenn keine exakten Begriffe übereinstimmen.



3. DATENVORBEREITUNG: DIE BASIS JEDER RAG-ANWENDUNG

Damit ein RAG-System zuverlässig funktioniert, muss die zugrunde liegende Vektordatenbank gepflegt und aktuell sein. **Die Datenvorbereitung ist daher kein einmaliger Schritt, sondern ein fortlaufender Prozess.** Gerade darin liegt ein zentraler Vorteil von RAG: Neue oder aktualisierte Inhalte lassen sich jederzeit in die Datenbank einpflegen, ohne dass das zugrunde liegende Sprachmodell neu trainiert werden muss.



Die folgenden Schritte haben sich bei der Vorbereitung unstrukturierter Textdaten für RAG in der Praxis bewährt:

Quellen identifizieren 1

Im ersten Schritt erfolgt die Auswahl relevanter Dokumente, z. B. PDFs, Webseiten, technische Anleitungen, Support-Wikis oder interne Berichte.

Vorverarbeitung (Preprocessing) 2

Rohdokumente liegen oft nicht in einem Format vor, das sich direkt für RAG mit Vektorsuche eignet. Bilder müssen ggf. in Text umgewandelt, Tabellen oder Grafiken weiterverarbeitet und überflüssige Inhalte wie Kopfzeilen oder Seitenzahlen bereinigt werden. Meist ist es nötig, die Dokumente vorab zu analysieren und in ein geeignetes Textformat zu bringen, das sich in die RAG-Pipeline integrieren lässt.

Textaufteilung (Chunking) 3

Anschließend sollten die Texte in kleine, sinnvolle Abschnitte („Chunks“) unterteilt werden. Die Chunk-Größe beeinflusst die Antwortqualität: Zu kleine Chunks enthalten womöglich nicht genug Kontext, zu große verwässern die Relevanz. Es gibt keine ideale Standardgröße – sie hängt vom Dokumenttyp und Anwendungsziel ab. Am besten: verschiedene Größen testen.

Einbettung (Embedding) 4

Mithilfe eines Embedding-Modells werden die Chunks nun in numerische Vektoren übersetzt. Ein Embedding-Modell ist ein spezielles Sprachmodell, das die Bedeutung eines Textes als Vektor – also eine Zahlenreihe – abbildet. Der eigentliche Vorteil von Embeddings liegt in ihrer mathematischen Vergleichbarkeit: Texte mit ähnlicher Bedeutung ergeben ähnliche Vektoren. So kann später im RAG-Prozess der Vektor einer Nutzeranfrage mit den gespeicherten Text-Vektoren verglichen und gezielt die relevantesten Inhalte ausgewählt werden.

Speichern in der Vektordatenbank 5

Embeddings werden in einer speziellen Datenbank gespeichert, der Vektordatenbank. Sie ist darauf ausgelegt, große Mengen an Vektordaten effizient zu verwalten und durchsuchen zu können. Werden zu jedem Textabschnitt auch Metadaten gespeichert, lassen sich Suchergebnisse gezielter filtern und transparenter darstellen. Eine RAG-Anwendung kann damit z. B. die Quelle, das Veröffentlichungsdatum oder eine Seitenzahl mit ausgeben – oder gezielt nur Inhalte aus bestimmten Dokumenten oder Zeiträumen berücksichtigen.

4. HERAUSFORDERUNGEN BEI EINER RAG-BASIERTEN ANWENDUNG

1

Kontextfenster: Wie viel passt in den Prompt?

LLMs können nur eine begrenzte Menge Text gleichzeitig verarbeiten. Ein RAG-System muss sicherstellen, dass Prompt und Zusatzinformationen gemeinsam in dieses Kontextfenster passen. Ohne gezieltes Kürzen kann der eingefügte Kontext das Kontextfenster sprengen – und dazu führen, dass das Modell den System Prompt abschneidet und nicht mehr berücksichtigt. Außerdem ist mehr Kontext nicht immer besser: Modelle neigen dazu, Inhalte in der Mitte zu übersehen („lost in the middle"). Deshalb ist es wichtig, Inhalte gezielt auszuwählen und sinnvoll zu platzieren – auch bei großen Kontextfenstern.

2

Gutes Chunking ist anspruchsvoller als es klingt

Das Aufteilen von Dokumenten in sinnvolle Abschnitte ist oft komplexer als erwartet. Der Chunking-Algorithmus muss mit sehr unterschiedlichen Strukturen umgehen können. Wenn Abschnitte ungünstig geschnitten werden, leidet die Antwortqualität. Gutes Chunking braucht daher immer ein Verständnis für den Aufbau und die Logik der Quelltexte.

3

Prompt Engineering beim System Prompt

Ein oft übersehener Faktor in RAG-Systemen ist die Qualität des System Prompts – also jener unsichtbare Teil der Eingabe, der das Modellverhalten steuert. Ob Rollen, Regeln oder Formatvorgaben: Schon kleine Anpassungen im System Prompt können die Klarheit, Präzision und Konsistenz der Antworten deutlich verbessern.

4

Halluzinationen bei fehlenden Inhalten in der Wissensdatenbank

Fehlen relevante Informationen in der Vektordatenbank, kann das LLM trotz RAG falsche Antworten liefern – einfach, weil die richtige Antwort nicht auffindbar ist. In solchen Fällen „halluziniert" das Modell häufig, um die Lücke zu füllen – auch wenn die Frage nur teilweise zum vorhandenen Kontext passt. Gut gestaltete Prompts können das LLM dazu bringen, Wissenslücken zu erkennen und stattdessen eine passende Nicht-Antwort zu geben.

5

Datenqualität entscheidet über die Antwortqualität

Die Wirksamkeit eines RAG-Systems steht und fällt mit der Qualität der eingebundenen Inhalte. Veraltete, unvollständige oder fehlerhafte Daten führen logischerweise zu ungenauen oder irreführenden Antworten – selbst bei einem leistungsfähigen Modell.

6

Datenschutz und Zugriffskontrolle im Blick behalten

Gerade im europäischen Raum unterliegt der Umgang mit Daten strengen Vorgaben. RAG-Systeme müssen sicherstellen, dass sensible Informationen nur für berechtigte Nutzer zugänglich sind – etwa durch die Integration von Rechten- und Rollenkonzepten in den Retrieval-Prozess. Ohne solche Mechanismen besteht das Risiko, dass vertrauliche Inhalte ungewollt offengelegt werden.

5. EVALUATION VON RAG-SYSTEMEN: WIE MISST MAN QUALITÄT?

Die Bewertung von RAG-Anwendungen ist komplexer als bei reinen LLMs – denn hier wirken mehrere Komponenten zusammen: Suche, Embeddings, Vektordatenbank, Prompting und das LLM. Eine wirksame Evaluation muss daher mehrschichtig erfolgen. Ziel ist es, sowohl die Qualität der gefundenen Inhalte als auch die Qualität der Antworten im Kontext dieser Inhalte zu bewerten.

Qualität der Suchergebnisse

Um die Fähigkeiten des RAG-Modells zu messen, kann man sich klassischer Kennzahlen aus dem Machine Learning bedienen:

Precision@k: Die Kennzahl "Precision@k" misst den Anteil relevanter Treffer unter den Top-K gefundenen Ergebnissen. Beispiel: Von den 5 gefundenen Quellen sind 3 relevant: Das Modell hat eine Precision@5 von 60%.

Recall@k: Die Kennzahl "Recall@k" misst, wie gut das System in der Lage ist, alle relevanten Treffer unter den Top-K Ergebnissen zu erfassen – und bewertet damit die Vollständigkeit der Suche. Aufbauend auf dem vorherigen Beispiel: Es hätte in der Datenbank 8 relevante Quellen gegeben, von denen das Modell 3 gefunden hat. Recall@5 = 37%.

Qualität der Antworten des LLM

Hier beurteilt man, ob das Modell eine inhaltlich richtige und verständliche Antwort auf Basis des Kontexts liefert. Beurteilungskriterien können sein:

- **Kontexttreue:** Hält sich die Antwort eng an die bereitgestellten Informationen?
- **Sachliche Korrektheit:** Stimmt die Antwort inhaltlich mit den Fakten überein?
- **Nützlichkeit:** Ist die Antwort praktisch, hilfreich und verständlich?
- **Prägnanz:** Vermeidet das Modell unnötige Ausschweifungen oder erfundene Inhalte?

Aufgrund der dynamischen Natur sowohl der Vektordatenbank, als auch des zu testenden LLMs muss im besten Fall eine fortlaufende Evaluation erfolgen.

6. USE CASE: WISSENSTRANSFER MIT LOKALEM LLM + RAG

In vielen Unternehmen existieren zahlreiche verteilte Wissensquellen, in denen wichtige Informationen zu Prozessen, Produkten oder Best Practices dokumentiert sind. Die Herausforderung besteht darin, dieses Wissen zur richtigen Zeit der richtigen Person bereitzustellen – schnell, gezielt und ohne aufwendige Recherche. In der Praxis verbringen Mitarbeitende jedoch oft viel Zeit damit, relevante Inhalte manuell zu suchen, was ineffizient und fehleranfällig ist.

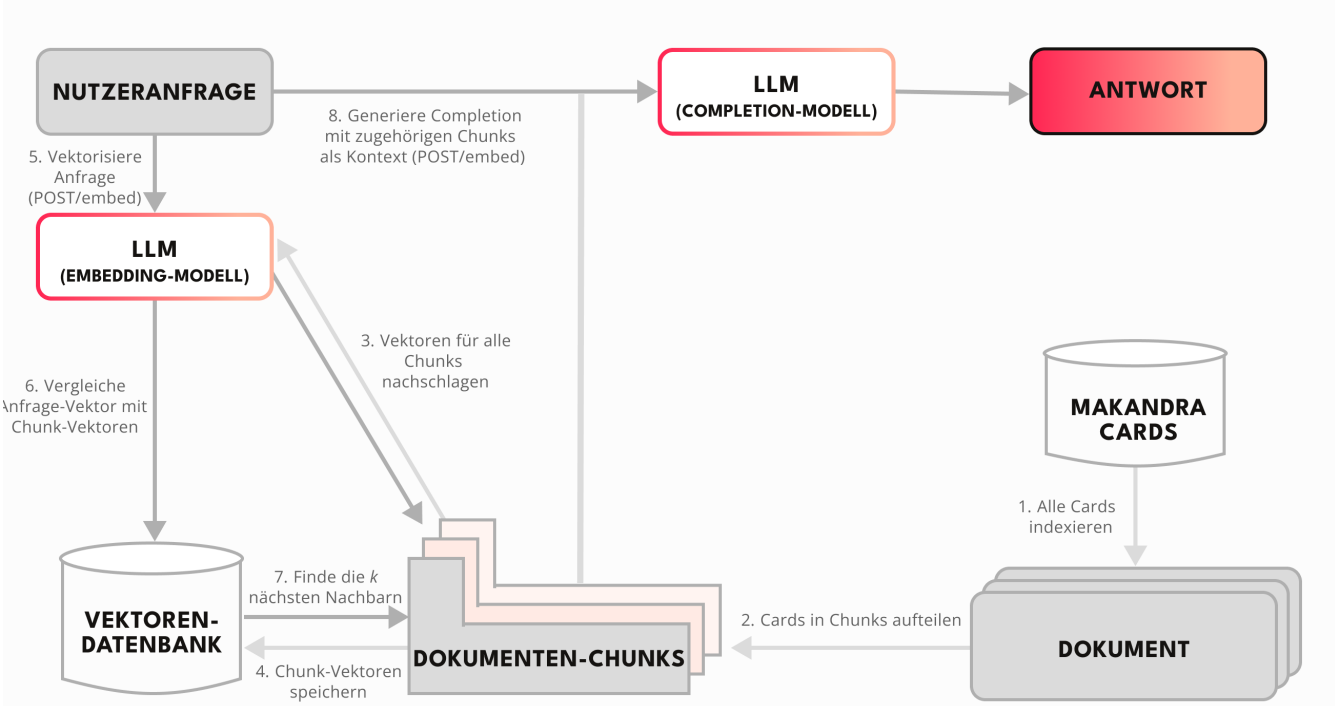
In unserem Fall nutzen wir eine interne Wissensdatenbank (makandra cards), in der tausende Beiträge zu Arbeitsweisen, Standards und Prozessen dokumentiert sind. Da diese Informationen sensible Unternehmensdaten enthalten, war für uns klar: Dieses Wissen sollte unter keinen Umständen in eine Cloud-Umgebung ausgelagert werden.

Unsere Lösung mit lokaler KI

Um ein System zu entwickeln, das automatisiert alle Fragen auf Basis unseres unternehmensinternen Wissens beantworten kann, setzen wir auf auf eine lokal betriebene KI (Open Source LLM), die durch eine sogenannte Retrieval-Augmented Generation (RAG)-Architektur erweitert wurde – vollständig lokal ausgeführt und datenschutzkonform.

Technisch funktioniert das so: Das Sprachmodell (zb. LLaMA oder Mistral) wird lokal über Ollama auf einem leistungsfähigen Server betrieben. Alle Daten verbleiben im Unternehmen, die Kontrolle über Modellversionen und Infrastruktur liegt vollständig intern.

Da ein LLM nur öffentliches Wissen bis zu seinem Trainingszeitpunkt kennt, wird für firmenspezifische Fragen die interne Wissensdatenbank angebunden. Ihre Inhalte werden in sinnvolle Textabschnitte („Chunks“) zerlegt, über sogenannte Embedding-Modelle in Vektoren umgewandelt und in einer Datenbank gespeichert – bei jeder Nutzerfrage wird der semantisch passende Kontext automatisch erkannt und sowohl in die Frage als auch Antwort eingebunden.



Weitere Anwendungsfälle

Support & Kundenservice: 1st-Level-Support-Assistent für interne oder externe Nutzer*innen oder Analyse und Kategorisierung eingehender Anfragen und automatisierte Antwortvorschläge auf Support-Tickets oder Emails.

Fachabteilungen (Vertrieb, HR, Recht, etc.): Generierung und Prüfung von Angebots- oder Vertragstexten (z. B. auf Basis bestehender Vorlagen), Analyse von Bewerbungsunterlagen oder HR-Anfragen, Beantwortung von rechtlichen Standardfragen basierend auf internen Policies, Onboarding und Einarbeitung neuer Mitarbeitender.

Produktion & Fertigung: Schneller Zugriff auf Arbeitsanweisungen, SOPs oder Wartungshandbücher per Chatbot – auch für Maschinenführer:innen, Fehleranalyse durch semantische Suche in Störmeldungen, Logs oder Qualitätsberichten, Qualitätsmanagement: KI-gestützte Auswertung von Prüfdaten und Ableitung von Maßnahmen.

Von den Unternehmen, die bisher keine KI-Technologien nutzen, haben 18% deren Einsatz bereits in Betracht gezogen. Nach den Gründen für den Nichtgebrauch gefragt, nannten diese Unternehmen: Fehlendes Wissen (71%), Unklarheit über die rechtlichen Folgen (58%), Bedenken hinsichtlich der Wahrung des Datenschutzes und der Privatsphäre (53%).

STATISTISCHES BUNDESAMT

FAZIT

Retrieval-Augmented Generation (RAG) ist ein entscheidender Baustein für den produktiven Einsatz lokaler LLMs in Unternehmen. Durch die Kombination von generativer Sprachkompetenz mit gezieltem Zugriff auf externe Daten entsteht eine Architektur, die leistungsfähig, anpassbar und datenschutzfreundlich zugleich ist.

Gerade in Kombination mit Open-Source-Modellen lässt sich RAG so gestalten, dass Unternehmen die volle Kontrolle über ihre Daten, Systeme und Weiterentwicklungen behalten – ohne auf die Möglichkeiten moderner KI verzichten zu müssen.

RAG ist dabei keine Plug-and-Play-Lösung: Die Qualität eines RAG-Systems hängt maßgeblich von der Datenbasis, der Gestaltung des Prompts, der Modellwahl und einer sauberen Evaluation ab. Wer bereit ist, diese Komponenten gezielt aufzubauen, erhält ein starkes Werkzeug für wissensbasierte Assistenzsysteme, Support-Automatisierung oder interne Recherche-Anwendungen.

Wie geht es jetzt weiter?

Wir hoffen, dass dieses Whitepaper Unklarheiten aus dem Weg räumen und Ihnen Ihre Entscheidung erleichtern konnte. Wenn Sie noch Fragen haben, rufen Sie uns gerne an oder vereinbaren Sie einen Termin. **Wir melden uns umgehend und stehen Ihnen in einem ersten Gespräch gerne kostenfrei beratend zur Seite.**

Wir bei makandra haben Erfahrung aus über 200 Projekten und eine überragende Kundenzufriedenheit. Wir beraten ehrlich und liefern nur Resultate ab, auf die wir auch stolz sein können.

KONTAKTIEREN SIE UNS

+49 821 58866 167

fabian.rimpl@makandra.de

https://makandra.de

Melli-Beese-Straße 5
86159 Augsburg