// KI-Hub VON STRUKTURAFLOW

KI PRAKTISCH

Was ist RAG und wann lohnt es sich wirklich? Ein Leitfaden für KMU

RAG ist gerade in aller Munde — aber was steckt wirklich dahinter, wann lohnt sich der Aufwand und wann ist es schlicht zu viel für das, was gebraucht wird?

AUTOR
Natascha Reiner

VERÖFFENTLICHT
30. Mai 2026

ONLINE LESEN
https://wissen.strukturaflow.it.com/was-ist-rag-und-wann-lohnt-es-sich-wirklich-ein-leitfaden-fuer-kmu/

RAG ist gerade in aller Munde. Wer sich mit KI beschäftigt, begegnet dem Begriff spätestens dann, wenn die Frage auftaucht: Wie kann ein Sprachmodell das eigene Unternehmenswissen kennen — ohne nachtrainiert zu werden?

Die Erklärungen dazu bewegen sich meistens in zwei Extremen: entweder so vereinfacht, dass man danach immer noch nicht weiß, was es wirklich bedeutet, oder so technisch, dass man drei Studienabschlüsse braucht, um durchzublicken.

Dieser Artikel ist weder das eine noch das andere. Er erklärt, was RAG ist, warum es funktioniert, wann der Aufwand gerechtfertigt ist — und wann nicht. Mit konkreten Beispielen, ehrlicher Einschätzung und ohne das übliche Buzz-Vokabular.

Das Problem, das RAG löst

Jedes Sprachmodell hat eine harte Grenze: Es weiß nur, was beim Training in seinen Daten enthalten war. Veröffentlichte Artikel, öffentliche Dokumentationen, wissenschaftliche Paper, Bücher, Webseiten — das ist die Grundlage. Was darin nicht vorkommt, kennt das Modell nicht.

Das klingt abstrakt. Konkret bedeutet es:

Das Sprachmodell weiß nichts über die internen Abläufe Ihres Betriebs. Es kennt Ihre Produktpalette nicht. Es hat keine Ahnung von den Sondervereinbarungen mit Ihrem Lieferanten, den Erfahrungen aus den letzten zwanzig Kundenprojekten oder dem Handbuch, das Ihr Team für neue Mitarbeitende geschrieben hat.

Dieser Mangel ist kein Fehler des Modells — er ist eine strukturelle Eigenschaft. Ein Modell weiß, was im Training drin war. Punkt.

Was also tun, wenn Sie einer KI das eigene Firmenwissen beibringen wollen?

Option A: Das Modell neu trainieren.
Teuer. Langsam. Komplex. Für die allermeisten Betriebe schlicht nicht machbar.

Option B: Das gesamte Wissen bei jeder Anfrage mitschicken.
Funktioniert bei kleinen Wissensmengen. Bei hundert Dokumenten wird es zu teuer, zu langsam und zu ungenau.

Option C: RAG.
Das ist der mittlere Weg. Und meistens der richtige.

Was RAG bedeutet – ohne Abkürzungs-Bingo

RAG steht für Retrieval-Augmented Generation. Auf Deutsch: Abruf-gestützte Generierung. Das klingt nicht besser, also lieber im Klartext:

Bevor das Sprachmodell eine Frage beantwortet, sucht das System in einer eigenen Wissensdatenbank nach den relevantesten Informationen zu genau dieser Frage. Diese Informationen werden zusammen mit der Frage an das Modell übergeben. Das Modell antwortet dann auf Basis dieser frischen, konkreten Informationen — nicht aus dem Trainingswissen.

Das klingt einfach, weil es im Prinzip einfach ist. Die Komplexität steckt in der Umsetzung — aber das Grundprinzip ist so zugänglich wie das Nachschlagen in einem Nachschlagewerk, bevor man antwortet.

Ein Vergleich: Stellen Sie sich vor, Sie stellen einem neuen Mitarbeitenden eine Frage zu einem internen Projekt. Er könnte aus dem Gedächtnis antworten. Oder Sie geben ihm vorher den Projektordner mit den relevanten Unterlagen, er liest kurz nach — und antwortet dann auf Basis der tatsächlichen Dokumentation. RAG ist das Nachschlagen. Das Sprachmodell ist der Mitarbeitende, der dann antwortet.

Die drei Schritte, die dahinterstecken

Schritt 1: Indexierung – einmalig

Ihre Dokumente werden in kleine Abschnitte zerlegt — sogenannte Chunks. Jeder Chunk wird durch ein Embedding-Modell in einen mathematischen Vektor umgewandelt: eine Zahl-Repräsentation der Bedeutung dieses Textabschnitts. Diese Vektoren werden in einer Vektordatenbank gespeichert.

Was ist ein Embedding-Modell? Ein spezialisiertes KI-Modell, das Text nicht liest wie ein Mensch — sondern seine Bedeutung in Zahlenmuster übersetzt. Zwei Sätze mit ähnlichem Sinn landen im Vektorraum nah beieinander, auch wenn sie völlig verschiedene Wörter verwenden. Das ist das Herzstück der semantischen Suche.

Was ist eine Vektordatenbank? Eine spezialisierte Datenbank, die für Ähnlichkeitssuche auf Vektoren optimiert ist. Sie findet nicht den Eintrag mit denselben Wörtern — sie findet den Eintrag mit der ähnlichsten Bedeutung.

Schritt 2: Retrieval – bei jeder Anfrage

Wenn eine Frage gestellt wird, wird auch die Frage in einen Vektor umgewandelt. Das System vergleicht diesen Vektor mit allen gespeicherten Chunks und findet die inhaltlich ähnlichsten Abschnitte. Das sind die drei bis fünf Textblöcke aus Ihrer Wissensdatenbank, die am relevantesten für die gestellte Frage sind.

Das Entscheidende: Das ist keine Keyword-Suche. Die Frage „Wie gehe ich vor, wenn ein Kunde eine Rechnung beanstandet?„ findet den Abschnitt „Ablauf bei Rechnungsreklamationen“ — auch wenn das Wort Reklamation in der Frage nicht vorkommt.

Schritt 3: Generation – die Antwort

Die gefundenen Textabschnitte werden zusammen mit der ursprünglichen Frage an das Sprachmodell übergeben. Das Modell antwortet dann auf Basis dieser konkreten Informationen. Nicht aus dem Training. Nicht erfunden. Aus Ihren Dokumenten.

Warum das kein Nachtraining ist – und warum das wichtig ist

Fine-Tuning — das tatsächliche Nachtrainieren eines Modells mit eigenen Daten — wird oft als Alternative zu RAG genannt. Der Unterschied ist fundamental:

Fine-Tuning verändert das Modell dauerhaft. Es lernt aus Ihren Daten — aber es vergisst dabei auch anderes, es veraltet, sobald Ihre Daten sich ändern, und es ist teuer, langsam und erfordert technisches Know-how, das die meisten Betriebe intern nicht haben. Dazu kommt: Fine-Tuning ist gut darin, Stil und Ton zu lernen. Es ist schlecht darin, sich Fakten zu merken.

RAG verändert das Modell nicht. Es liefert dem Modell bei jeder Anfrage frische, aktuelle Informationen aus Ihrer Wissensdatenbank. Ändert sich ein Dokument? Sie aktualisieren die Datenbank, fertig. Kein Nachtraining nötig.

Das ist der Grund, warum RAG für faktisches, betriebliches Wissen die überlegene Methode ist — besonders für Wissensmanagement-Systeme im Unternehmen. Und warum es sich trotz des höheren initialen Aufwands für die meisten Anwendungsfälle lohnt.

Die Bausteine – was ein RAG-System braucht

Ein vollständiges RAG-System besteht aus fünf Komponenten. Keine davon muss zwingend ein Cloud-Dienst sein.

Die Dokumente

Bevor irgendeine Technologie ins Spiel kommt, steht die Frage: Welches Wissen soll die KI kennen? Und in welchem Zustand ist dieses Wissen?

Typische Quellen in einem Betrieb: interne Handbücher und Prozessbeschreibungen, Produktdatenblätter, FAQ-Listen aus dem Kundensupport, Angebotsvorlagen, Meeting-Protokolle, Dokumentationen aus vergangenen Projekten.

Das klingt selbstverständlich — ist es aber nicht. Ein schlecht strukturiertes, veraltetes Handbuch in der Wissensdatenbank erzeugt eine KI, die mit großer Überzeugung falsche Informationen liefert. Verlässlicher als ein Mensch, der den Fehler ignorieren würde — aber genauso falsch. Garbage in, garbage out gilt hier wie überall.

Der erste echte Arbeitsschritt ist deshalb immer ein Wissens-Audit: Was haben wir? Wo liegt es? Ist es aktuell? Widerspricht sich irgendetwas?

Das Embedding-Modell

Wandelt Text in Vektoren um. Für deutschsprachige Inhalte sind Modelle empfehlenswert, die explizit mehrsprachig trainiert wurden. **bge-m3** ist aktuell das stärkste frei verfügbare Modell für diesen Zweck — läuft lokal via Ollama, kein Cloud-API-Call, keine laufenden Kosten.

Wer Cloud akzeptiert: OpenAIs **text-embedding-3-small** ist günstig (ca. 0,02 USD pro Million Token) und auf Deutsch gut. Coheres **Embed v3** ist stark auf mehrsprachige Inhalte spezialisiert und bietet EU-Hosting an.

Die Vektordatenbank

Speichert die Vektoren, macht die Ähnlichkeitssuche. **Qdrant** ist die Empfehlung für selbst gehostete Setups: Open-source, Rust-basiert, sehr performant, läuft als Docker-Container, unterstützt Metadaten-Filter. Wer bereits PostgreSQL betreibt: **pgvector** als Extension ist die schlanke Alternative ohne neuen Dienst.

Chroma funktioniert gut für Prototypen und kleinere Setups. Für den Produktiveinsatz mit echten Anforderungen empfehle ich Qdrant.

Das Sprachmodell

Das Modell, das antwortet — austauschbar. Cloud-API (Anthropic Claude, OpenAI GPT) oder lokal via Ollama (Llama 3, Mistral). Lokale Modelle erfordern Hardware, sind dafür DSGVO-technisch unkompliziert. Cloud-Modelle sind einfacher, erfordern entsprechende Datenschutzverträge.

Eine pragmatische Zwischenlösung: Vektordatenbank und Embedding lokal, Sprachmodell über eine DSGVO-konforme Cloud-API. Dann gehen nur die Frage plus die drei relevantesten Chunks nach außen — nie die gesamte Wissensdatenbank.

Die Orchestrierung

Irgendetwas muss die Schritte zusammenhalten. **n8n** ist für Betriebe, die bereits auf No-Code-Automatisierung setzen, der natürliche Kandidat: native Nodes für Vektorspeicher und Embedding-Modelle, visuell konfigurierbar, kein Code nötig für einfache Szenarien. Für komplexere Anforderungen kommen Frameworks wie LangChain oder LlamaIndex ins Spiel.

Drei Anwendungsfälle aus der Praxis

Der interne Wissens-Assistent

Ein Planungsbüro mit vierzig Mitarbeitenden hat über Jahre eine beachtliche Wissensbasis aufgebaut: Normen, Richtlinien, interne Projekterfahrungen, Vertragsvorlagen, Checklisten. Diese liegen verteilt auf einem Netzlaufwerk — wer etwas braucht, sucht selbst, fragt Kollegen oder findet nichts.

Lösung: Ein interner KI-Assistent, der auf diese Wissensbasis zugreift. Mitarbeitende stellen Fragen in natürlicher Sprache und erhalten strukturierte Antworten auf Basis der tatsächlichen Firmendokumente — mit Quellenangabe, auffindbar, ohne Google, ohne die Kollegin zu fragen, die gerade im Urlaub ist.

Die gesamte Wissensbasis bleibt auf dem eigenen Server. Kein Dokument verlässt das Netzwerk. Das System indexiert neue Dokumente automatisch, sobald sie auf dem Laufwerk landen.

Der Chatbot mit echtem Produktwissen

Ein Großhändler betreibt einen Webshop mit dreitausend Produkten. Kundenanfragen kommen täglich: Welches Modell für Einsteiger? Unterschied zwischen diesen zwei Produkten? Passendes Zubehör?

Ein Standard-Chatbot antwortet aus allgemeinem Marketingtext. Ein RAG-gestützter Chatbot zieht bei jeder Frage die tatsächlichen Produktdatenblätter: Materialien, Gewichte, Einsatzbereiche, Vergleiche. Bei Aktualisierungen läuft ein Workflow, der die Datenbank automatisch aktualisiert.

Das ist kein Zukunftsprojekt. Das ist heute mit n8n und Qdrant in einem überschaubaren Aufwand umsetzbar.

Das Techniker-Wissen im Außendienst

Ein Unternehmen im Anlagenbau schickt Techniker in den Außendienst. Dieselben Fragen tauchen immer wieder auf: Wie war das Vorgehen bei Fehlercode XY bei Anlage Z? Was haben wir beim letzten ähnlichen Einsatz gemacht?

Das institutionelle Wissen steckt in Einsatzberichten und Fehlerprotokollen — irgendwo in einem System, das niemand durchsucht, weil es zu aufwändig ist. Ein RAG-System macht dieses Wissen abrufbar: Der Techniker stellt eine Frage, das System durchsucht alle Einsatzberichte der vergangenen Jahre nach ähnlichen Fehlerbildern und gibt eine strukturierte Zusammenfassung zurück. Das Wissen der erfahrensten Kollegin — abrufbar, auch wenn sie im Urlaub ist.

Wann RAG sinnvoll ist – und wann nicht

Ich halte wenig davon, Technologie als universelle Lösung zu verkaufen. RAG löst ein spezifisches Problem. Wenn dieses Problem nicht vorhanden ist, ist es der falsche Ansatz.

RAG ist sinnvoll, wenn:

- Die Wissensbasis mehr als 50–100 Dokumente umfasst, die regelmäßig abgefragt werden
- Informationen sich ändern und immer aktuell sein müssen — ohne Nachtraining
- Quellenangaben und Nachvollziehbarkeit wichtig sind (Compliance, interne Qualitätssicherung)
- Verschiedene Nutzer sehr unterschiedliche Fragen stellen und ein statischer FAQ-Bot nicht reicht
- Dokumente durchsuchbar sein sollen, ohne manuellen Aufwand

RAG ist nicht nötig, wenn:

- Der gesamte Kontext in einen übersichtlichen System-Prompt passt (unter ca. 20–30 Seiten)
- Die Wissensbasis stabil ist und sich kaum ändert
- Immer dieselben fünf Fragen gestellt werden — dann reicht ein gut gemachter FAQ-Chatbot
- Das Budget und der Aufwand für den erwarteten Nutzen nicht verhältnismäßig wären
- Allgemeines Weltwissen für den Anwendungsfall ausreicht

Ein Betrieb mit zwanzig standardisierten Leistungen und einer dreiseitigen FAQ braucht kein RAG. Ein Ingenieurbüro mit zwanzig Jahren Projektdokumentation, das sein internes Wissen zugänglich machen will, schon.

Was das Token-Budget kostet – und warum es relevant ist

Das ist ein Aspekt, über den wenig gesprochen wird: RAG erhöht den Token-Verbrauch pro Anfrage. Jeder Chunk, der retrieved und mitgeschickt wird, kostet Input-Tokens — und die sind nicht gratis, wenn man über eine Cloud-API arbeitet.

Eine typische Rechnung:

POSTEN	TOKENS
System-Prompt	~300
Nutzerfrage	~30
3 retrieved Chunks à 400 Token	~1.200
Generierte Antwort	~200
Gesamt pro Anfrage	**~1.730**

Ohne RAG wäre dieselbe Anfrage bei ~530 Token. RAG verdreifacht den Token-Verbrauch — bei Claude Sonnet (ca. 3 USD pro Million Input-Token) sind das Kosten im Cent-Bereich pro Anfrage. Bei hundert Anfragen täglich über dreißig Tage: ca. 15 USD Mehrkosten pro Monat. Für die meisten Betriebe unkritisch — aber es ist gut, das zu wissen und im Blick zu behalten.

Die Stellschrauben: Chunk-Größe, Anzahl retrieved Chunks (Top-K), Reranking (ein zweiter Schritt, der die besten Chunks nochmal filtert, bevor sie mitgeschickt werden).

Die DSGVO-Perspektive: Was technisch zählt

Datenschutz ist kein Checkbox-Thema — und kein Artikel ersetzt einen Datenschutzbeauftragten oder Rechtsbeistand. Aber ich kann die technischen Parameter benennen, die rechtlich relevant sind.

Wo liegen die Dokumente?
Auf dem eigenen Server: volle Kontrolle, keine Drittland-Problematik. Cloud: Auftragsverarbeitungsvertrag nötig.

Was verlässt das System beim API-Aufruf?
Nicht die gesamte Wissensdatenbank — nur die Frage plus die retrieved Chunks. Das ist ein wesentlicher Unterschied zu Systemen, die alle Dokumente in die Cloud hochladen.

Werden personenbezogene Daten indexiert?
Kundendaten, Mitarbeiterdaten in der Wissensdatenbank sind besonders kritisch und erfordern ein klares Konzept.

Wo ist der API-Anbieter?
EU-Anbieter: unkompliziert. US-Anbieter: Standardvertragsklauseln (SCCs) nötig. Cohere bietet EU-Hosting an. Bei OpenAI und Anthropic sind SCCs Pflicht.

Self-hosted Komponenten (Qdrant, Ollama, n8n):
Keine Drittland-Problematik, volle Datenkontrolle.

Die einfachste DSGVO-Lösung ist ein vollständig selbst gehosteter Stack: Vektordatenbank, Embedding-Modell und Sprachmodell lokal. Kein API-Anbieter, kein Drittland-Transfer, keine Auftragsverarbeitungsverträge. Die Einschränkung: Lokal betriebene Sprachmodelle sind bei allgemeinen Aufgaben noch schwächer als die großen Cloud-Modelle. Bei spezifischem Domänenwissen mit gutem Retrieval rückt der Abstand aber näher, als viele denken.

Die häufigsten Fehler – und wie man sie vermeidet

Die Wissensbasis wird nicht gepflegt

Das ist der häufigste Grund, warum RAG-Systeme nach ein paar Monaten schlechte Antworten geben. Das Handbuch aus 2022 ist noch drin. Die Produktliste von vor dem letzten Sortimentswechsel ist noch drin. Das Modell antwortet auf Basis veralteter Dokumente — mit großer Überzeugung.

Lösung: Klare Ownership. Wer ist verantwortlich für welche Dokumente? Automatische Trigger für Neuindexierung bei Dateiänderungen über n8n.

Zu viele Dokumente auf einmal

Der Impuls, alles sofort zu indexieren, ist verständlich — führt aber zu einem System, das schwer zu testen ist. Wenn die Qualität nicht stimmt, weiß man nicht, wo das Problem liegt: am Embedding-Modell? An der Chunk-Größe? An den Dokumenten selbst?

Lösung: Klein anfangen. Eine klar definierte Dokumentengruppe. Qualität validieren. Dann erweitern.

Chunk-Größen werden ignoriert

Zu kleine Chunks verlieren Kontext. Zu große bringen irrelevante Informationen mit. Technische Handbücher funktionieren mit anderen Chunk-Größen als Meeting-Protokolle oder Produktbeschreibungen.

Lösung: Chunk-Parameter pro Dokumenttyp testen. Overlap zwischen Chunks einplanen, damit kein Kontext an Grenzen verloren geht.

Das Retrieval wird nicht evaluiert

Viele prüfen die Ausgabe des Modells — aber nicht, ob das Retrieval die richtigen Chunks findet. Ein Modell kann aus einer schlechten Grundlage trotzdem einen kohärenten Text machen. Das macht Fehler schwerer zu erkennen.

Lösung: Testfragen mit bekannter Antwort und bekanntem Quelldokument verwenden. Prüfen, ob das Retrieval die richtigen Chunks zurückgibt — unabhängig von der generierten Antwort.

Die ehrliche Einschätzung

RAG ist kein Selbstläufer und kein Plug-and-Play-Produkt. Es entfaltet seinen Nutzen, wenn zwei Grundbedingungen erfüllt sind: eine gepflegte, strukturierte Wissensbasis — und ein klarer Anwendungsfall.

Wer RAG als ersten KI-Schritt plant, ohne vorher zu wissen, welches Wissen zugänglich gemacht werden soll und für wen, setzt an der falschen Stelle an.

Wer aber konkret sagen kann: „Unsere Techniker fragen täglich dieselben Dinge, die Antworten stecken in zweihundert Einsatzberichten, und aktuell muss jeder selbst suchen„ — der hat einen Anwendungsfall, der sich lohnt.

Die Technologie ist verfügbar, dokumentiert und für Betriebe ohne eigene Softwareentwicklung handhabbar — insbesondere wenn jemand den Aufbau begleitet. Der initiale Aufwand ist real, aber überschaubar. Die laufenden Kosten sind bei selbst gehosteten Setups minimal.

Was es wirklich braucht: die Bereitschaft, das eigene Wissen zu strukturieren. Das ist mehr Organisations- als Technologiefrage. Und genau deshalb scheitern mehr RAG-Projekte an der Datenlage als an der Technik.

FAQ

Muss ich für RAG programmieren können?
Für einfache Setups mit n8n: nein. n8n hat native Nodes für Vektordatenbanken und Embedding-Modelle, die ohne Code konfigurierbar sind. Für komplexe Szenarien — mehrstufiges Retrieval, benutzerdefinierte Chunk-Strategien, Evaluierungspipelines — ist technische Unterstützung sinnvoll.

Was kostet ein RAG-System?
Bei self-hosted Komponenten (Qdrant, Ollama, n8n): die Serverkosten. Ein kleiner dedizierter Server in einem EU-Rechenzentrum für 20–40 EUR/Monat reicht für die meisten Setups. Wenn Cloud-APIs für das Sprachmodell genutzt werden, kommen laufende Token-Kosten dazu — bei moderatem Nutzungsvolumen meist im zwei- bis dreistelligen Euro-Bereich pro Monat.

Was ist der Unterschied zwischen RAG und einem normalen Chatbot?
Ein normaler Chatbot antwortet aus dem Trainings-Wissen des Sprachmodells — allgemeines Wissen, nichts Betriebsspezifisches. Ein RAG-gestützter Chatbot sucht bei jeder Anfrage in Ihrer eigenen Wissensdatenbank und antwortet auf Basis Ihrer Dokumente.

Kann ich RAG auch mit lokal betriebenen Modellen nutzen?
Ja. Embedding-Modell (bge-m3 via Ollama) und Sprachmodell (Llama 3, Mistral via Ollama) können beide lokal betrieben werden. Dann verlässt kein einziges Datum das eigene Netzwerk. Die

Einschränkung: Lokale Modelle sind bei allgemeinen Aufgaben noch schwächer als die großen Cloud-Modelle — mehr dazu im Praxistest Claude vs ChatGPT. Für spezifisches Domänenwissen mit gutem Retrieval ist der Unterschied kleiner als erwartet.

Wie lange dauert die Erstindexierung?
Bei einigen hundert Dokumenten mit einem lokalen Embedding-Modell: Minuten bis wenige Stunden, abhängig von Dokumentgröße und Hardware. Danach werden nur neue oder geänderte Dokumente inkrementell indexiert.

Was, wenn sich meine Dokumente oft ändern?
RAG ist genau dafür gemacht. Kein Nachtraining nötig — die Wissensdatenbank wird aktualisiert, fertig. n8n kann so konfiguriert werden, dass Dateiänderungen automatisch eine Neuindexierung auslösen.

Ist RAG DSGVO-konform?
Das hängt vom Stack ab. Self-hosted Komponenten: unkompliziert. Cloud-APIs: Auftragsverarbeitungsvertrag nötig, EU-Anbieter bevorzugen. Eine verbindliche Einschätzung gibt ein Datenschutzbeauftragter — nicht dieser Artikel.

Nächste Schritte

Wenn Sie konkret prüfen wollen, ob und wie RAG für Ihren Betrieb sinnvoll ist — welche Dokumente als erste Wissensbasis geeignet wären, welcher Stack zu Ihrer Infrastruktur passt und was realistisch in welchem Aufwand umsetzbar ist — klären wir das in einem 30-Minuten-Gespräch.

Beratungsgespräch vereinbaren →

NÄCHSTER SCHRITT

Mehr praktische KI-Anleitungen für KMU

Dieser Artikel ist Teil des KI-Hubs von Strukturaflow — einer deutschsprachigen Plattform für den praktischen KI-Einsatz in kleinen und mittleren Unternehmen.

https://wissen.strukturaflow.it.com