

Ein RAG-System wie meins, Schritt für Schritt nachgebaut

Begleitdokument zum Video «Ein echtes RAG-System von innen». Derselbe Text wie im Artikel:
www.latzerus.ch/lernen/echtes-rag-system-von-innen/

Inhalt · 1. Was ein RAG ist und wann du es brauchst · 2. Weg rein: Wie eine Seite ins System kommt · 3. Weg raus: Wie eine Frage ihre Antwort findet · 4. Drei Fehler aus meinem System und drei Sicherheitsregeln · 5. Bauanleitung: das RAG-System mit Claude oder ChatGPT nachbauen (Prompt) · 6. Deine Prüfliste in 30 Minuten

DAS WICHTIGSTE

In 5 Punkten

1. RAG heisst: erst in deinen Unterlagen suchen, dann antworten. Die KI bekommt vor jeder Antwort die passenden Stellen mitgegeben.
2. Es gibt zwei Wege. Rein: Eine Seite wird in Stücke geschnitten, in Zahlen übersetzt und im Archiv abgelegt. Raus: Die Frage wird übersetzt, gesucht, geprüft und beantwortet.
3. Zwei Dinge machen den Unterschied: eine zweite Suche nach exakten Wörtern und ein Prüfer, der die besten Treffer noch einmal liest.
4. Ein RAG rechnet nicht. «Wie viel habe ich letztes Jahr ausgegeben?» gehört in die Buchhaltung, nicht in ein RAG.
5. Miss mit einer Prüfliste aus 10 Fragen, ob dein System gut ist. Die komplette Bauanleitung und ein Prompt zum Nachbauen stehen in diesem Artikel.

1 · GRUNDLAGE

Was ein RAG ist und wann du es brauchst

RAG steht für Retrieval-Augmented Generation. Auf Deutsch: erst suchen, dann antworten. Eine KI weiss nichts über deine Firma, deine Preise, deine Notizen. Also gibt man ihr vor jeder Antwort die passenden Stellen aus deinen Unterlagen mit. Wie ein neuer Mitarbeiter, der vor jeder Antwort kurz im Archiv nachschaut.

Aber nicht jede Frage gehört in ein RAG. «Wie viel habe ich letztes Jahr für Software ausgegeben?» ist eine **Rechnung, keine Suche**. Ein RAG holt die fünf passendsten Stellen, und die KI zählt diese fünf zusammen. Die Zahl klingt sauber und ist falsch, denn dafür braucht es alle Belege. Rechnungen gehören in die Buchhaltung oder in eine Tabelle.

Prüf der Reihe nach:

1. **Allgemeinwissen?** Dann frag die KI direkt.
2. **Du weisst, wo es steht?** Dann such selbst.
3. **Nur ein paar Seiten?** Dann kopier sie einfach in den Chat.

4. **Zu viel für den Chat, und die Antwort steht an wenigen Stellen?** Dann lohnt sich ein RAG. Mein Wiki hat gut 200 Seiten, da lohnt es sich.

2 · WEG REIN

Weg rein: Wie eine Seite ins System kommt

Das Original ist mein Wiki. Das Programm heisst **BookStack**. Nur dort wird geschrieben und korrigiert. Alles andere ist eine Kopie davon, die man jederzeit neu bauen kann. Bei mir dauert ein kompletter Neuaufbau etwa drei Stunden.

Die Kopie macht **n8n**, eine Art Förderband. Es holt jede Seite, schneidet sie in Stücke, übersetzt jedes Stück in Zahlen und legt es im Archiv ab.

In Stücke schneiden (Chunks): Bei mir etwa 1200 Zeichen pro Stück, das ist ein langer Absatz. Das Ende eines Stücks wird am Anfang des nächsten wiederholt, damit kein Satz zerrissen wird. Vor jedes Stück kommt eine **Brotkrume**: Buch, Seite, Kapitel. Ohne sie weiss ein Stück oft nicht, wovon es handelt. Meine Beispielseite ergibt so 15 Stücke.

In Zahlen übersetzen (Vektor): Ein spezielles KI-Modell gibt jedem Stück Koordinaten, wie auf einer Landkarte. «Der Ferienanspruch beträgt 25 Tage» und «Wie viele Tage habe ich frei?» landen nah beieinander, obwohl sie kaum ein Wort teilen. Meine Landkarte hat 4096 Richtungen. Das Modell läuft über **Ollama** auf meinem eigenen Rechner, darum verlässt kein Text das Haus.

Im Archiv ablegen: Das Archiv ist **Qdrant**, eine sogenannte Vektordatenbank. Jedes Stück ist dort ein Punkt mit seinem Text und seinen Koordinaten.

3 · WEG RAUS

Weg raus: Wie eine Frage ihre Antwort findet

Eine Frage läuft in vier Phasen durch das System:

1. **Übersetzen:** Die Frage wird mit demselben Modell in Zahlen übersetzt wie vorher die Seiten. Zusätzlich wird sie in einzelne Wörter zerlegt.
2. **Suchen:** Das Archiv sucht auf zwei Arten und liefert die besten 30 Kandidaten.
3. **Prüfen:** Ein Prüfer liest jeden Kandidaten zusammen mit der Frage und gibt eine Note von 0 bis 1.
4. **Antworten:** Was fast 0 ist, fliegt raus. Die KI liest die Treffer und antwortet mit Quelle.

Zwei Arten suchen (Hybrid Search): Stell dir eine Bibliothek vor. Die Bibliothekarin versteht, was du meinst: Du fragst nach freien Tagen, sie bringt das Kapitel über Ferien. Bei exakten Namen ist sie aber ungenau. Das Stichwortregister hinten im Buch versteht nichts, findet aber exakte Wörter wie «Qwen2.5-VL». Mein System fragt beide und legt die Listen zusammen. Wer auf beiden weit oben steht, gewinnt.

Der Prüfer (Reranker) ist langsamer, aber genauer als die Suche. Achte auf die Form der Noten:

- «Was antworte ich, wenn der Kunde sagt, das ist zu teuer?» → $0,99 \cdot 0,98 \cdot 0,91 \cdot 0,84 \cdot 0,19$. Vier passende Artikel, dann ein Absturz. **Das System weiss es.**
- «Wie merke ich, dass ein Kunde innerlich schon gekündigt hat?» → $0,13 \cdot 0,09 \cdot 0,03 \cdot 0,02 \cdot 0,02$. Alles tief. **Das System rät**, denn dazu gibt es keinen Artikel.

Wenn es rät, hilft oft eine andere Formulierung. «KI erfindet Zahlen» fand bei mir den richtigen Artikel mit 0,97, «Halluzinationen vermeiden» fand ihn fast nicht.

4 · FEHLER UND SICHERHEIT

Drei Fehler aus meinem System und drei Sicherheitsregeln

Fehler 1, 86 Prozent unsichtbar: Ich habe nachgezählt. Erwartet waren rund 2000 Stücke, im Archiv lagen rund 200, von jeder Seite nur das erste. Schuld war ein einziger Baustein im Förderband. Die Suche hat trotzdem geantwortet, nur immer vom Anfang der Seiten. *Lehre: Zähl nach, wie viele Stücke drin sein müssten und wie viele drin sind.*

Fehler 2, der stille Ausfall: Der Prüfer schläft nach 10 Minuten ohne Arbeit ein und braucht rund 33 Sekunden zum Aufwachen. Die Suche wartete nur 30 Sekunden und lieferte dann ungeprüfte Treffer, ohne Warnung. Heute wartet sie 2 Minuten, und fehlt der Prüfer, erscheint eine Warnung. *Lehre: Ein Fehler, den man sieht, ist besser als einer, der still passiert.*

Fehler 3, eine Technik ohne Wirkung: Contextual Retrieval steht in vielen Anleitungen: Eine KI schreibt vor jedes Stück einen Satz, worum es im Ganzen geht. Ich habe gemessen. Die Brotkrume verbesserte die Note des Prüfers deutlich, der Zusatzsatz obendrauf nicht, eher im Gegenteil. *Lehre: Was in einer Anleitung steht, misst du bei dir nach.*

Drei Sicherheitsregeln. Wer an das Archiv kommt, sieht alles darin. Ein Filter ist keine Berechtigung.

1. **Archiv abschliessen:** Ein Schlüssel zum Schreiben für das Förderband, einer nur zum Lesen für die Suche. Qdrant ist nach der Installation offen, setz den Schlüssel, bevor du etwas hineinlegst.
2. **Keine Schlüssel und internen Adressen in Screenshots und Videos.** Auch Adresszeile und Fehlermeldungen prüfen.
3. **Keine KI installieren lassen, was du nicht gelesen hast.** Jeden Befehl lesen, bevor er läuft, und Schritt für Schritt installieren.

5 · BAUANLEITUNG

Bauanleitung: das RAG-System mit Claude oder ChatGPT nachbauen (Prompt)

Diese Anleitung baut eine kleine Demo meines Systems auf deinem Rechner. Du brauchst keine Programmierkenntnisse, aber etwas Geduld. Kopiere den Prompt unten in einen neuen Chat, in Claude, ChatGPT oder einen anderen KI-Assistenten. Der Assistent fragt dich zuerst nach deinem Rechner und führt dich dann Schritt für Schritt. Er führt nichts selbst aus. Du entscheidest bei jedem Befehl. Alle Unterlagen dazu gibt es kostenlos als Paket, der Download steht unten auf dieser Seite.

Was du baust: BookStack als Wiki, n8n als Förderband, Ollama für die Übersetzung in Zahlen, Qdrant als Archiv. Alles läuft lokal. Den Prüfer (Reranker) baust du erst dazu, wenn die Grundversion läuft und gemessen ist.

Förderband 1, Wiki einlesen (n8n): Start → Einstellungen → Archiv anlegen → Bücher holen → Seiten holen → Seiten auswählen → Seite lesen → in Stücke schneiden (1200 Zeichen, Überlappung, Brotkrume) → in Zahlen übersetzen (Ollama) → Punkte bauen (Bedeutung und Stichwörter) → ins Archiv schreiben.

Förderband 2, Prüfliste testen (n8n): Start → Prüfliste (Frage und erwartete Seite) → Fragen vorbereiten → Frage übersetzen → auf zwei Arten suchen → auswerten: Wie oft steht die richtige Seite auf Platz 1, wie oft unter den ersten 3?

Der Prompt zum Kopieren:

Du bist mein technischer Begleiter. Ich will ein kleines, lokales RAG-System als Demo nachbauen, nach dem Bauplan unten. Er stammt aus einem System, das im Alltag läuft. Arbeite Schritt für Schritt mit mir.

SPIELREGELN

1. Frag mich zuerst: Betriebssystem, Arbeitsspeicher, ob Docker installiert ist, ob ich schon ein Wiki habe, und welches Ziel ich habe: (A) nur verstehen oder (B) die Demo auf meinem Rechner bauen.
2. Du führst nichts selbst aus und installierst nichts. Du gibst mir jeden Befehl einzeln, erklärst in einem Satz, was er tut, und wartest auf mein Ergebnis.
3. Keine echten oder privaten Daten in der Demo. Wir arbeiten mit 5 bis 10 Beispielseiten, die du mir vorschlägst.
4. Keine Schlüssel und Passwörter im Chat. Wenn einer nötig ist, sagst du mir, wo ich ihn selbst eintrage.
5. Bist du dir bei einem Befehl, einer Version oder einer Adresse nicht sicher, sag es offen.
6. Antworte auf Deutsch, kurz, ohne Marketing-Sprache.

BAUPLAN (nicht vereinfachen, ohne mich zu fragen)

- Quelle der Wahrheit: ein Wiki (BookStack). Geschrieben wird nur dort. Der Index ist abgeleitet und jederzeit neu baubar.
- Förderband: n8n. Liest Bücher und Seiten über die API, zerlegt sie, bettet ein und schreibt nach Qdrant.
- Zerlegen: 1200 Zeichen pro Abschnitt, 150 Zeichen Überlappung, Schnitt an Absatzgrenzen, Überschriften bleiben erhalten. Vor jeden Abschnitt kommt eine Brotkrume «Buch > Seite > Kapitel».
- Feste Kennung pro Abschnitt: $\text{page_id} \times 1000 + \text{Abschnittsnummer}$. Erneutes Einlesen überschreibt, statt zu verdoppeln.
- Wichtig in n8n: Jeder Code-Baustein läuft über ALLE Eingaben und gibt eine Liste zurück. Sonst landet pro Seite nur der erste Abschnitt im Index.
- Einbetten: lokal mit Ollama. Wähle das Modell nach meinem Arbeitsspeicher (gross: qwen3-embedding:8b mit 4096 Dimensionen; kleiner, wenn der Rechner wenig Speicher hat). Beim Einlesen und beim Fragen immer dasselbe Modell.
- Speicher: Qdrant, eine Collection mit zwei Vektoren: «dense» (Cosine, Länge passend zum Modell) und «sparse» (Modifiziert idf).
- Wortsuche (sparse): ein Tokenizer, der Unterstrich, Punkt und Bindestrich innerhalb eines Wortes behält, damit Modellnamen, Slugs und Ports ganze Wörter bleiben. Wörter per Hash in feste Eimer.
- Suche: dense und sparse parallel, verschmolzen mit Reciprocal Rank Fusion (RRF). Ein Reranker (Cross-Encoder) kommt erst dazu, wenn die Grundversion läuft und gemessen ist.
- Kein Kontextsatz (Contextual Retrieval) in der ersten Version. Zuerst messen, ob er bei mir etwas bringt.
- Sicherheit: Qdrant nur mit API-Key, ein Schreib-Key für n8n, ein Lese-Key für die Suche, nur im eigenen Netz erreichbar. Schlüssel nur in n8n-Credentials oder Umgebungsvariablen.
- Messen: eine Prüfliste (Eval-Set) mit 10 echten Fragen und der erwarteten Seite. Platz-1-Quote und Top-3-Quote vor und nach jeder Änderung.

ABLAUF

1. Voraussetzungen prüfen (deine Fragen aus Regel 1).
2. Docker Compose für BookStack, Qdrant und n8n. Ollama läuft direkt auf dem Rechner. Erkläre mir jede Zeile der Compose-Datei.
3. 5 bis 10 Beispielseiten in BookStack anlegen. Du lieferst die Texte: eine Seite mit einem exakten Modellnamen, eine lange Seite mit mindestens 8000 Zeichen, der Rest kurz.
4. Die zwei n8n-Workflows importieren (05_n8n_Foerderband_1_Wiki_einlesen.json und 06_n8n_Foerderband_2_Pruefliste_testen.json aus dem Latzerus-Paket) und die Konfiguration anpassen. Habe ich sie nicht, baust du sie mit mir Baustein für Baustein nach dem Bauplan nach.
5. Einlesen und prüfen: Stimmt die Zahl der erwarteten Abschnitte mit der Zahl der Punkte in Qdrant überein?
6. Die Prüfliste laufen lassen und die zwei Quoten notieren.
7. Genau eine Änderung machen (zum Beispiel die Abschnittsgrösse) und neu messen.

Beginne mit Schritt 1 und stell mir deine Fragen.

Hinweis: Das ist eine Lern-Demo, kein fertiges Produkt. Lies jeden Befehl, bevor du ihn ausführst. Ein Assistent kann sich bei Versionen und Adressen irren.

Deine Prüfliste in 30 Minuten

1. **Zehn echte Fragen aufschreiben.** Nimm Fragen, die du oder dein Team wirklich stellen, aus Mails, Chats oder Telefonnotizen. Mindestens zwei davon mit einem exakten Namen oder einer Nummer.
2. **Die Seite mit der Antwort daneben schreiben.** Zu jeder Frage der Titel der Seite, auf der die Antwort steht. Findest du selbst keine, fehlt die Antwort in deinen Unterlagen. Das ist schon ein Ergebnis.
3. **Zwei Fragen von ganz hinten dazunehmen.** Zwei Fragen, deren Antwort weit hinten auf einer langen Seite steht. So merkst du, ob wirklich alle Stücke im Archiv angekommen sind.
4. **Fragen stellen und den Platz notieren.** Steht die richtige Seite auf Platz 1, unter den ersten 3 oder gar nicht? Daraus werden zwei Zahlen, zum Beispiel 6 von 10 auf Platz 1 und 8 von 10 unter den ersten 3.
5. **Vor und nach jeder Änderung wiederholen.** Bevor du etwas änderst, misst du. Danach misst du nochmals mit denselben Fragen. Wird eine Zahl schlechter, war die Änderung keine Verbesserung, auch wenn sie sich so anfühlt.

Im Paket dabei: Arbeitsheft für deine Prüfliste, Handout mit 5 Mythen und 3 Prüffragen, die zwei n8n-Förderbänder zum Importieren, der Prompt als Textdatei, die interaktive Seite «So läuft eine Suchanfrage» und ein vereinfachter Nachbau des Suchprogramms und des Prüfers zum Lesen.