Suchen nach Bedeutung statt nach Wörtern (Semantische Suche & RAG)

Kapitel online

Warum die KI „Fahrzeug“ findet, wenn du „Auto“ tippst: Durchsuche ein Firmen-Handbuch einmal mit Stichwörtern und einmal mit Bedeutungs-Vektoren — und sieh, wie RAG daraus Antworten baut.

Aktuell · Kapitel 31 von 50 · ca. 10 Min. · 250 XP

Einfach gesagt: Suchst du „Auto“, findet die KI auch Sätze mit „Dienstwagen“ — sie vergleicht Bedeutung statt Buchstaben.

In „Latent Space“ hast du gesehen: ähnliche Bedeutung = nahe Punkte. Heute baust du damit das, was 2026 in fast jeder Firma läuft: **semantische Suche** — und darauf **RAG**, den Trick, mit dem Chatbots plötzlich DEINE Dokumente kennen.

Was du hier lernst

- Warum die **Stichwortsuche** den „Dienstwagen“ nicht findet, wenn du nach „Auto“ suchst
- Wie **Nähe im Latent Space** zum Suchtreffer wird — Bedeutung statt Buchstaben
- Was **RAG** in vier Schritten tut: Frage → Vektor → nächste Nachbarn → Prompt
- Warum die **Blockgröße** (Chunks) über die Trefferqualität entscheidet
- Wie die **Kosinus-Ähnlichkeit** von Hand gerechnet aussieht — und wo RAG an Grenzen stößt

Für den Unterricht

Einstieg (2 Min): Kapitel gemeinsam lesen, Alltagsbeispiel aufgreifen.

Erkunden am Gerät (6 Min): die interaktiven Module auf herrki.de/330_semantische_suche ausprobieren.

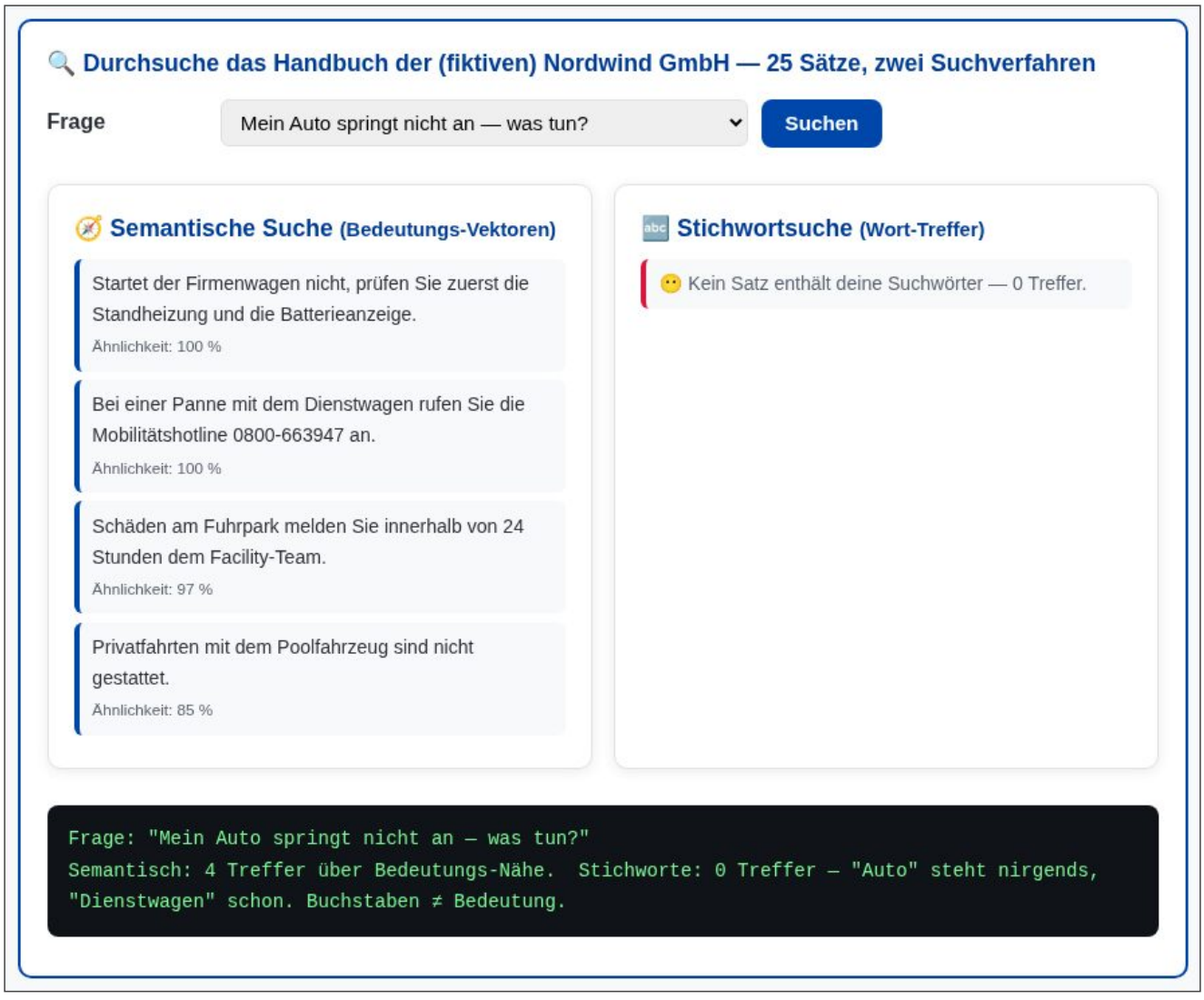
Sichern mit Quiz (2 Min): „Kurzer Check" besprechen, „Das nimmst du mit" festhalten.

Das Problem mit der Stichwortsuche

Klassische Suche vergleicht **Buchstaben**: Steht „Auto“ nicht im Text, findet sie nichts — auch wenn dort „Fahrzeug“, „Dienstwagen“ oder „PKW“ steht. Menschen meinen Bedeutungen, Computer vergleichen Zeichenketten. Die Lösung kennst du schon: **Mach aus Sätzen Punkte im Latent Space** — dann ist „ähnliche Bedeutung“ einfach „kleiner Abstand“.

Am Gerät ausprobieren: herrki.de/330_semantische_suche#abschnitt-2

Direkt zum Modul



Interaktives Modul – am Gerät ausprobieren

Und daraus wird RAG

Retrieval-**A**ugmented **G**eneration heißt nur: erst suchen, dann antworten lassen.

1**Frage → Vektor** — deine Frage wird zum Punkt im Latent Space

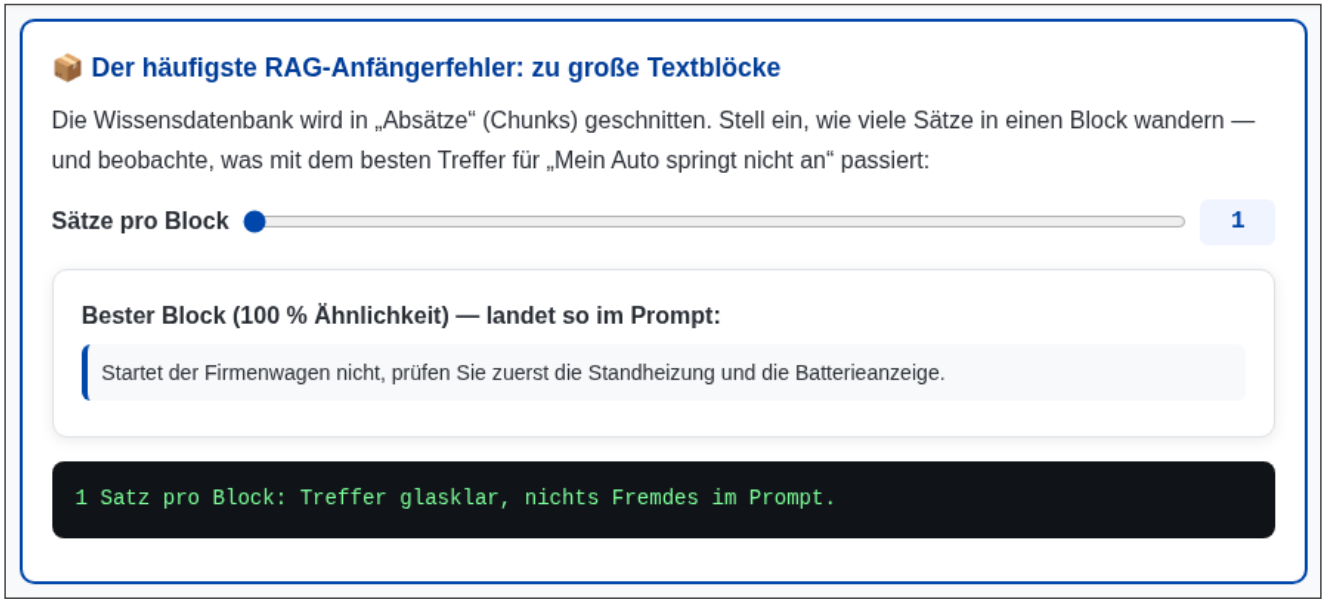
2**Nächste Nachbarn holen** — die 3–5 ähnlichsten Absätze aus der Wissensdatenbank

3**In den Prompt packen** — „Beantworte die Frage NUR mit diesen Absätzen: …“

4**Antwort mit Quelle** — das Modell formuliert, die Fakten kommen aus DEINEN Dokumenten

Am Gerät ausprobieren: herrki.de/330_semantische_suche#abschnitt-3

Direkt zum Modul



Interaktives Modul – am Gerät ausprobieren

Runter zum Tensor

Die ganze Magie ist ein Skalarprodukt (Kapitel 060) — und das rechnen wir jetzt mit deiner Auto-Frage von oben einmal von Hand nach. Die Sätze dieses Moduls haben acht Bedeutungs-Achsen; wir nehmen die drei, auf die es hier ankommt:

```
// Zum Mitrechnen auf 3 Achsen verkleinert — echte Embeddings haben 384 bis 1536.
                                Fahrzeug Notfall Urlaub
Frage „Auto springt nicht an" 1.0 0.6 0.0
Satz 6 „Panne mit dem Dienstwagen" 1.0 0.7 0.0
Satz 1 „Urlaubsanträge stellen Sie" 0.0 0.0 0.9
```

Erst Achse für Achse multiplizieren und aufaddieren, dann durch die Vektor-Längen teilen — damit lange und kurze Sätze fair verglichen werden. Das ist die **Kosinus-Ähnlichkeit**:

```
Frage · Dienstwagen-Panne
  = 1.0×1.0 + 0.6×0.7 + 0.0×0.0 = 1.42
  Längen: √(1.0²+0.6²) = 1.166 √(1.0²+0.7²) = 1.221
  1.42 / (1.166 × 1.221) = 0.997 → 100 % ✔ Volltreffer

Frage · Urlaubsantrag
  = 1.0×0.0 + 0.6×0.0 + 0.0×0.9 = 0.00
  0.00 / irgendwas = 0.00 → 0 % ✘ nichts gemeinsam
```

Kein einziges Wort ist gleich — „Auto“ steht in keinem der 25 Sätze. Trotzdem 100 %: Verglichen wurden Richtungen im Raum, nicht Buchstaben. Und so heißt dasselbe im Code, wo jeder Satz [384] lang ist und die ganze Wissensdatenbank eine Matrix [25, 384] bildet:

```
// Jeder Satz wurde EINMAL vorab in einen Vektor verwandelt (Embedding-Modell).
const frageVektor = embed("Mein Auto springt nicht an");
const aehnlichkeit = frageVektor.dot(satzVektor) // oben: 1.42
      .div( normFrage.mul(normSatz) ); // oben: 1.166 × 1.221 → 0.997
```

Kein Verstehen, kein Nachdenken — **Vektoren vergleichen**. Deshalb ist die Suche auch in Millionen Dokumenten in Millisekunden fertig.

Zurück zur Anwendung — und die ehrlichen Grenzen

⚠ **RAG dämpft Halluzinationen, es heilt sie nicht** (Kapitel „Halluzinationen“): Findet die Suche die falschen Absätze, formuliert das Modell aus falschem Material trotzdem eine schöne Antwort. Und wie du oben gesehen hast, hängt die Trefferqualität fast immer an der **Aufbereitung der Dokumente** (Blockgröße, saubere Texte) — nicht am Sprachmodell. Wer RAG verbessern will, arbeitet zu 80 % an den Daten.

Zum Weiterlesen

- **Das RAG-Paper:** Patrick Lewis u. a., *„Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks“*, 2020 — arxiv.org/abs/2005.11401 — hier bekam der Vier-Schritte-Trick von oben seinen Namen
- **Woher gute Satz-Vektoren kommen:** Nils Reimers & Iryna Gurevych, *„Sentence-BERT“*, 2019 — arxiv.org/abs/1908.10084 — erklärt, warum man ganze Sätze eigens dafür trainieren muss, dass Nähe wirklich Ähnlichkeit heißt
- **Vektorsuche schlägt Stichwortsuche — gemessen:** Vladimir Karpukhin u. a., *„Dense Passage Retrieval“*, 2020 — arxiv.org/abs/2004.04906 — der Zahlenvergleich zu deinem Zwei-Spalten-Experiment von oben

Kurzer Check – mit Lösung

Lösungen für Lehrkräfte – für Lernende die Online-Version nutzen.

Frage 1: Warum findet die semantische Suche „Dienstwagen-Panne“, wenn du „Auto springt nicht an“ fragst?

- Weil sie eine Liste aller Synonyme der deutschen Sprache enthält.
- ✔ **Weil beide Formulierungen als Vektoren nahe beieinander liegen — verglichen wird Bedeutung, nicht Buchstaben.**
- Weil sie das Internet durchsucht.

Die Sätze wurden vorab in Embedding-Vektoren verwandelt (Kapitel 310). Ähnliche Verwendung → ähnliche Vektoren → kleiner Abstand. Ein Wörterbuch braucht dafür niemand.

Frage 2: Was macht RAG mit den gefundenen Absätzen?

- Es trainiert das Modell damit neu.
- ✔ **Es packt sie als Kontext in den Prompt — das Modell antwortet dann auf dieser Grundlage.**
- Es schickt sie ungelesen an den Nutzer zurück.

RAG ändert KEINE Gewichte (das wäre Fine-Tuning, Kapitel 370). Es füttert das Wissen zur Laufzeit in den Eingabe-Tensor — deshalb ist es billig und sofort aktualisierbar.

Frage 3: Was passiert bei zu großen Text-Blöcken (Chunks)?

- ✔ **Der Block-Vektor wird zum Durchschnitt vieler Themen — Treffer verwässern, und Unpassendes landet mit im Prompt.**
- Die Suche wird genauer, weil mehr Text mehr Information ist.
- Nichts — die Blockgröße ist egal.

Ein Vektor pro Block: Je mehr Themen im Block, desto „durchschnittlicher“ zeigt der Vektor in keine Richtung mehr richtig. Faustregel: lieber kleine, thematisch saubere Absätze.

Das nimmst du mit

- **Bedeutung schlägt Buchstaben.** Die Stichwortsuche findet den „Dienstwagen“ nicht, wenn du „Auto“ tippst — die semantische Suche schon.
- **Der Treffer ist ein Abstand.** Frage und Dokument werden zu Vektoren, verglichen wird per Kosinus-Ähnlichkeit — kein Verstehen, kein Nachdenken.
- **RAG sind vier Schritte.** Frage → Vektor → nächste Nachbarn → ab damit in den Prompt. Kein Gewicht wird angefasst, deshalb ist es billig und sofort aktuell.
- **Die Blockgröße entscheidet.** Ein Vektor pro Block: Je mehr Themen darin stecken, desto durchschnittlicher — und desto schlechter der Treffer.
- **RAG dämpft Halluzinationen, es heilt sie nicht.** Wer RAG verbessern will, arbeitet zu 80 % an den Daten, nicht am Modell.

Weiter geht’s: Du hast gesehen, wie ein Modell sich Wissen von außen holt (RAG) — der Latent Space als Landkarte zum Nachschlagen. Im nächsten Kapitel „Nachschlagen statt rechnen: N-Gramm-Speicher im Modell“ siehst du den Gegenentwurf: Wissen, das im Modell selbst in einer Tabelle liegt — DeepSeeks Engram. Danach entsteht aus purem Rauschen Schritt für Schritt ein Bild — „Wie Bild-KIs wirklich malen (Diffusion)“.

So zitierst du dieses Kapitel

HerrKI: Suchen nach Bedeutung statt nach Wörtern (Semantische Suche & RAG). herrki.de/330_semantische_suche, ein privates Hobbyprojekt von Terragon.de, abgerufen am 09.09.2026.