

AI Engineering

Retrieval, Chunking, Hybrid Search & Reranking

Vektorsuche liefert Kandidaten. Gutes Retrieval entsteht erst durch Chunking, hybride Signale und kontrollierte Fusion.

Band 1 · Kapitel 14

Öffentliche Leseprobe · Auszug (§1 „Von Kandidaten zum Kontext — warum Vektorsuche nur der Anfang ist“ bis §4 „Hybrid: Fusion statt Addition“)

Sebahattin Özata · Diplom-Informatiker

OMEX Press

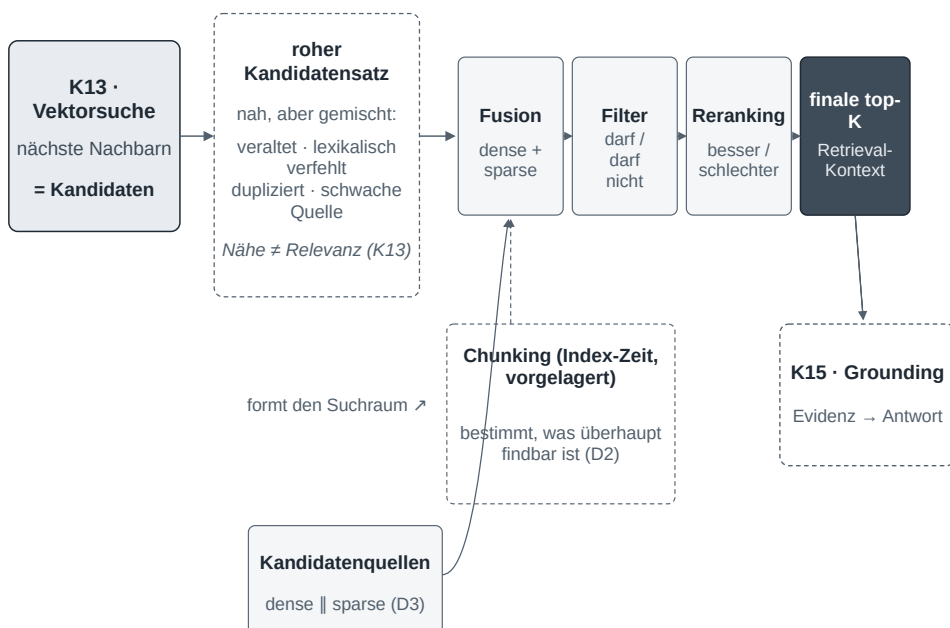
Aus Kapitel 14 in Band 1: der Weg vom Kandidaten zum Kontext.
Kapitelverweise beziehen sich auf das vollständige dreibändige Werk.

Kapitel 14 — Retrieval, Chunking, Hybrid Search & Reranking: vom Kandidaten zum Kontext

1. Von Kandidaten zum Kontext — warum Vektorsuche nur der Anfang ist

Retrieval ist eine Pipeline, kein Vektor-Such-Aufruf

Die nächsten Nachbarn aus K13 sind der Eingang — nicht das Ergebnis



Die drei Grenzen, die dieses Kapitel auseinanderhält

SECURITY — wer darf überhaupt in den Suchraum? (vorgelagert, nie ein Rang-Problem · D6)

RETRIEVAL — welche Kandidaten sind für diese Query am besten? (Chunking, Fusion, Filter, Reranking · K14)

GROUNDING — welche Evidenz darf eine konkrete Aussage tragen? (K15 — nicht dieses Kapitel)

K13 liefert die Geometrie · K14 baut daraus Retrieval · K15 macht daraus Grounding.

D1 · Retrieval ist eine Pipeline

Der häufigste Denkfehler in RAG-Systemen ist, „Retrieval“ mit „ein Vektor-Such-Aufruf“ gleichzusetzen. D1 zeigt, warum das zu kurz greift: Die nächsten Nachbarn aus K13 sind **ein Kasten in einer längeren Kette**, nicht ihr Ende. Der rohe Kandidatensatz ist genau das, was K13 als Grenze markiert hat — nah, aber ungeordnet nach Relevanz, Aktualität und Autorität. Jede folgende Stufe fügt eine **andere** Entscheidung hinzu.

Das Neue gegenüber K13 ist nicht mehr Geometrie, sondern **Auswahl unter Nebenbedingungen**. Und D1 nennt die drei Achsen, die das ganze Kapitel trennt: Wer darf überhaupt in den Suchraum (**Security**)? Welche Kandidaten sind für diese Query am besten (**Retrieval**)? Welche Evidenz darf eine konkrete Aussage tragen (**Grounding**, K15)? Diese Dreiteilung ist die zentrale Owner-Grenze des Buches; §5 macht sie scharf.

ARCHITEKTURREGEL. Retriever-Qualität **deckelt** alles Nachgelagerte. Kein noch so gutes Grounding rettet eine Antwort, deren Evidenz nie abgerufen wurde. Was hier verloren geht, ist unten nicht mehr zu holen.

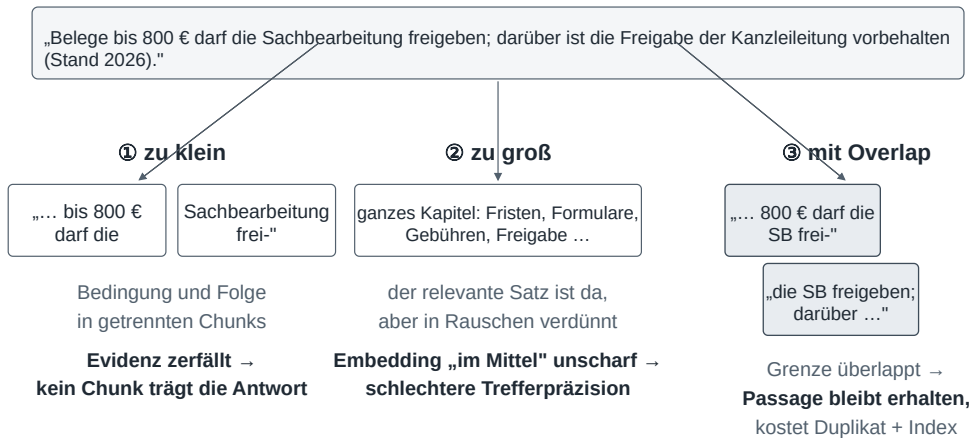
2. Chunking formt den Suchraum

Bevor irgendetwas gesucht wird, wird der Korpus in **indexierbare Einheiten** geschnitten. Diese Entscheidung fällt zur Index-Zeit, einmal, und sie bestimmt, was überhaupt als Treffer existieren kann.

Chunking formt den Suchraum

Die Chunk-Grenze entscheidet, was überhaupt als Einheit gefunden werden kann — vor jeder Suche

Dokument (eine zusammenhängende Regel)



Der Trade-off — und warum es keine universelle Chunk-Größe gibt

klein → präzise, aber Kontextverlust, fragmentierte Evidenz, viele Kandidaten. **groß** → mehr Kontext, aber verdünnte Embeddings, mehr Tokenkosten.

Overlap schützt Grenzen (Duplikations-/Indexkosten). **Struktur-bewusst** (Überschrift/Absatz) ist sinnvoll, aber dokumententypabhängig.

Semantisches Chunking (Schnitt an Bedeutungsgrenzen) ist plausibel, aber komplex — eine allgemeine Überlegenheit ist nicht etabliert (T1).

Die optimale Größe hängt von Korpus, Aufgabe und Embedding-Modell ab — sie ist ein Hyperparameter, kein Naturgesetz.

D2 · Chunking formt den Suchraum

D2 macht den Kern sichtbar; T1 ordnet die Strategien als Nachschlage-Raster. Im Text nur die Interpretation: Chunking ist kein Formatierungsschritt, sondern die **erste Recall-Grenze**. Ein zu kleiner Chunk zerreit zusammengehrige Evidenz — Bedingung und Folge, Frage und Antwort, Betrag und Ausnahme landen getrennt, und keiner der beiden Teile gewinnt die Suche. Ein zu groer Chunk verdnnt den relevanten Satz in Off-Topic-Tokens, sodass das Embedding „im Mittel“ unscharf wird und an Trefferprzision verliert.

T1 · Chunking-Strategien

Strategie	Stärke	Schwäche	Geeignet, wenn ...
Fixed-Size (Token-/Zeichenfenster)	einfach, deterministisch, vorhersagbare Indexgröße	schneidet mitten durch Sätze/Ideen	homogener Fließtext, schneller Start
+ Overlap (Stride < Fenster)	schützt Grenz- Passagen	Duplikation, größerer Index	Boundary-Fehler sind teuer
Satz-/Absatz	respektiert Sprachgrenzen	Einheiten stark ungleich lang	natürliche Sprache mit klaren Absätzen
Struktur-bewusst (Überschrift/HTML/Code)	folgt echter Dokumentstruktur	dokumenttyp- spezifisch, mehr Aufwand	strukturierte Dokumente (Verträge, Code)
Semantisch (Schnitt an Bedeutungsgrenzen)	Chunks sind in sich kohärent	komplex, modellabhängig, Nutzen umstritten	wenn empirisch für den Korpus belegt
Parent/Child (klein suchen, groß liefern)	scharfes Rangsignal + kohärenter Kontext	zwei Repräsentationen, mehr Komplexität	präzise Treffer + zusammenhängender Kontext nötig

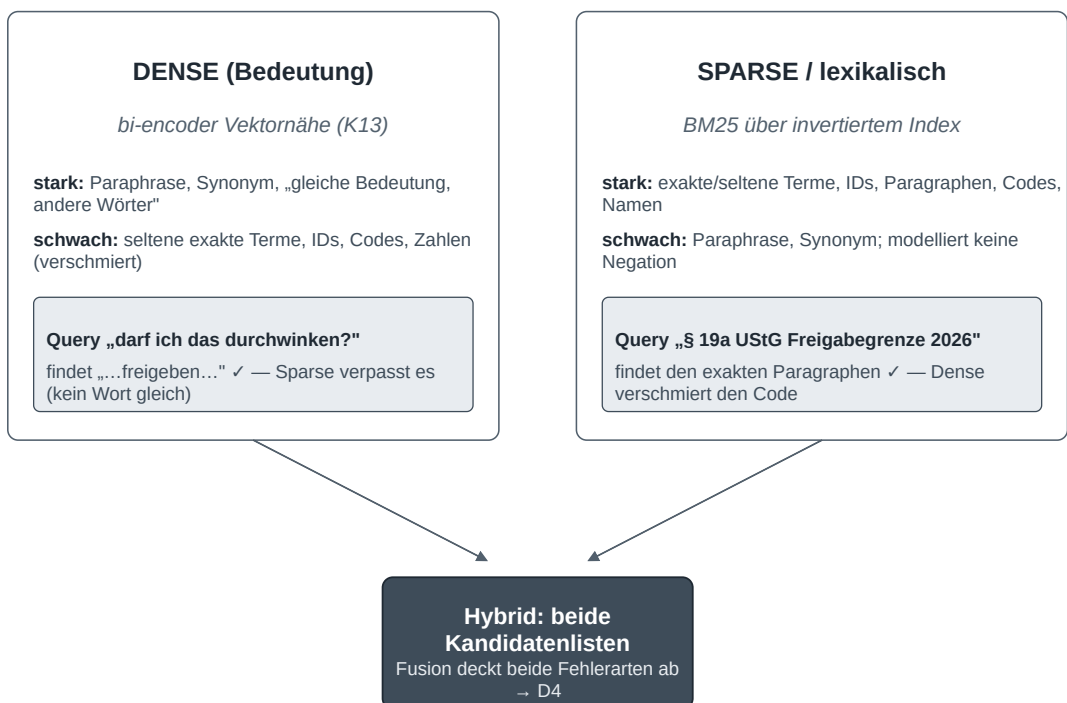
Die letzte Reihe von D2 ist die wichtigste und die am häufigsten überschätzte: **Es gibt keine universell optimale Chunk-Größe**. Viel verbreitete Chunking-Beratung ist Ingenieurs-Heuristik, keine belegte Wissenschaft. **Semantisches** Chunking (Schnitt an Bedeutungsgrenzen) ist eine plausible, verbreitete Strategie; eine **allgemeine** Überlegenheit über gut abgestimmtes Fixed-Size oder struktur-bewusstes Chunking ist jedoch **nicht etabliert** und bleibt korpus-, aufgaben- und setup-abhängig. Der wissenschaftliche Anker in dieser Gegend ist nicht eine „beste Größe“, sondern das **Passagen-Granularitäts-Prinzip**: Retrieval und offene QA arbeiten seit jeher auf Passagen statt ganzen Dokumenten, weil die Granularität die Findbarkeit bestimmt (Karpukhin et al., Dense Passage Retrieval/DPR, EMNLP 2020). Chunk-Größe und -Strategie sind ein **Hyperparameter**, den man je Korpus, Aufgabe und Embedding-Modell abstimmt.

3. Dense und Sparse — verschiedene Fehler

Ist der Korpus geschnitten und indexiert, gibt es zwei grundverschiedene Wege, Kandidaten zu holen — und ihr Verhältnis ist der Grund, warum es Hybrid gibt.

Dense und Sparse finden verschiedene Fehler

Deshalb ist keins von beiden allein genug — und deshalb gibt es Hybrid (D4)



Die Lehre — typische Stärken, keine Garantien

Dense und Sparse machen **unterschiedliche** Fehler. Wo das eine verschmiert, trifft das andere exakt — und umgekehrt.

Das sind *typische* Eigenschaften, keine Absolutregeln: BM25 „löst“ nicht Matching, Dense „versteht“ nicht jede Bedeutung.

Weil ihre Fehler sich ergänzen statt überlappen, lohnt sich Hybrid — meist, nicht immer (T2).

D3 · Dense und Sparse finden verschiedene Fehler

D3 zeigt ein Paar, T2 macht es zum Raster; der Text erklärt, warum beides zusammengehört. **Dense** (der bi-encoder aus K13) ist stark, wo die Bedeutung zählt: Paraphrase, Synonym, „gleiche Sache, andere Wörter“. Es ist schwach genau dort, wo ein Wort **wörtlich** stimmen muss — seltene Terme, Aktenzeichen, Paragraphen, Fehlercodes, Zahlen werden in nahe Richtungen des Vektorraums verschmiert (Karpukhin et al. 2020).

Sparse/lexikalisch — klassisch **BM25** über einem invertierten Index (Robertson & Zaragoza 2009) — ist die Umkehrung: stark bei exakten, seltenen Termen, blind gegenüber Paraphrase und Synonymie.

T2 · Dense / Sparse / Hybrid

Signal	Findet gut	Verpasst typischerweise
Dense (bi-encoder, K13)	Bedeutung, Paraphrase, Synonym, „anders formuliert“	seltene exakte Terme, IDs, Codes, Zahlen, unbekannte Wörter (out-of-vocabulary)
Sparse (BM25, lexikalisch)	exakte/seltene Terme, Paragraphen, Namen, Codes	Paraphrase, Synonymie; modelliert keine Negation
Hybrid (dense + sparse, D4)	beides — die Fehler ergänzen sich	garantiert nichts: Nutzen hängt vom Query-Mix ab

Zwei Präzisierungen, die T2 nicht ausschreiben kann. Erstens: BM25 **„löst“ kein Matching und modelliert keine Negation** — es zählt Term-Überlappung; „darf freigeben“ und „darf **nicht** freigeben“ sehen für BM25 fast gleich aus. Zweitens sind das **typische** Eigenschaften, keine Absolutregeln: lexikalisches und dichtes Retrieval haben je eigene Stärken, und weil ihre Fehler sich **ergänzen statt überlappen**, lohnt sich, beide zu befragen — der Übergang zu Hybrid. Das ist empirisch breit belegt (◆).

◆ Reales Experiment — kein Retriever gewinnt überall (BEIR, Thakur et al. 2021)

BEIR — ein heterogener Benchmark für **Zero-Shot**-Retrieval — prüft Verfahren über eine breite Sammlung sehr verschiedener Datensätze (Fakten-Checking, Fragen, Argumente, Fachdomänen), jeweils ohne domänenspezifisches Nachtraining. Befund: das klassische lexikalische **BM25** bleibt ein **schwer zu schlagender** Baseline, während dichte Modelle außerhalb ihrer Trainingsdomäne **ungleichmäßig** generalisieren — mal besser, mal schlechter ([Studienbefund] , 2021; konkrete Ränge driften mit Modellgenerationen).

Technische Lehre: genau die **Komplementarität** aus D3/T2 — dense und sparse machen *verschiedene* Fehler, kein einzelner Retriever dominiert über alle Query-Typen. Das ist die empirische Begründung für Hybrid (§4). **Beweist nicht:** dass BM25 „besser“ als dense ist oder dass Hybrid immer gewinnt — der Nutzen hängt vom Query-Mix ab (T2); ein datierter Benchmark, kein Naturgesetz.

4. Hybrid: Fusion statt Addition

Zwei Kandidatenlisten zu haben ist das Leichte. Sie **zusammenzuführen** ist die eigentliche Ingenieursfrage — und der Ort, an dem der häufigste Anfängerfehler sitzt.

Hybrid heißt nicht „Scores addieren“

Zwei Ranglisten in verschiedenen Zahlenräumen — naive Addition lässt eine Skala dominieren

DENSE-Rangliste

cosine $\in [-1, 1]$

1. Chunk A 0,82
2. Chunk B 0,79
3. Chunk C 0,75

kleine, gebundene Werte

SPARSE-Rangliste

BM25 unbeschränkt

1. Chunk C 14,2
2. Chunk D 9,1
3. Chunk A 6,3

große, korpusabhängige Werte

Reciprocal Rank Fusion (RRF)

fusioniert über RÄNGE, nicht Scores

$$\text{RRF}(d) = \sum 1 / (k + \text{Rang}_i(d)), k \approx 60$$

Chunk A: $1/61 + 1/63 \approx 0,0323 \rightarrow$ Rang 1

Chunk C: $1/63 + 1/61 \approx 0,0323 \rightarrow$ Rang 1–2

Chunk B: $1/62 + 0 \approx 0,0161 \rightarrow$ dahinter

Chunk D: $0 + 1/62 \approx 0,0161 \rightarrow$ dahinter

was in BEIDEN Listen oben steht, gewinnt —

ohne die Zahlenräume je zu vergleichen

Naiv addiert: $0,82 + 6,3 \approx 7,1$ vs. 14,2 — **BM25 überstimmt cosine allein durch seine Skala.**

Gegenmittel: Score-Normalisierung ODER rang-basierte Fusion.

Was RRF gut kann — und was es aufgibt

Vorteil: skalenfrei, tuning-arm, robust; braucht keine Score-Kalibrierung — ein starker Standard.

Preis: es verwirft die Score-Magnituden (die Konfidenz). Ein „0,95 vs. 0,72“ und ein „0,73 vs. 0,72“ sehen im Rang gleich aus.

Deshalb kann **gewichtete oder gelernte Fusion RRF schlagen**, wenn die Scores gut kalibriert sind — RRF ist Default, nicht Optimum.

Und: Hybrid gewinnt meist, aber nicht immer — der Nutzen hängt vom Query-Mix und der Fehler-Korrelation ab (T3).

RRF: Cormack, Clarke & Buettcher, SIGIR 2009 ($k = 60$). Alle Zahlen hier illustrativ, keine Messung.

D4 · Hybrid heißt nicht „Scores addieren“

Der Kern steht im Bild: **Dense- und BM25-Scores leben in verschiedenen Zahlenräumen.** cosine ist auf $[-1, 1]$ gebunden, BM25 ist unbeschränkt und korpusabhängig. Wer sie naiv addiert, lässt die größere Skala gewinnen — die Rangfolge sagt dann mehr über die Zahlenbereiche als über Relevanz. Es gibt zwei saubere Auswege: **Score-Normalisierung** (min-max, z-Score) oder **rang-basierte Fusion**, die die Scores gar nicht erst vergleicht.

Reciprocal Rank Fusion ist der verbreitetste rang-basierte Ansatz (Cormack, Clarke & Buettcher, SIGIR 2009): $RRF(d) = \sum 1/(k + Rang_i(d))$ über alle Systeme, mit $k \approx 60$. Ein Dokument, das in **beiden** Listen oben steht, sammelt aus beiden einen hohen Beitrag — ohne dass jemals cosine mit BM25 verglichen wird. T3 stellt die Fusions-Optionen nebeneinander:

T3 · Fusions-Strategien

Methode	Braucht Score-Kalibrierung?	Vorteil	Risiko
Naive Addition	— (und genau das ist der Fehler)	trivial	größere Skala dominiert
Score-Normalisierung (min-max / z)	ja	behält Score-Magnituden (Konfidenz)	empfindlich gegen Ausreißer, Verteilungsdrift
RRF (rang-basiert)	nein	skalenfrei, tuning-arm, robust	verwirft die Score-Magnituden
Gewichtete / gelernte Fusion	ja (Gewichte/Training)	kann RRF schlagen	braucht Kalibrierung/Labels; Overfit-Gefahr

Zwei Overclaims, die T3 ausdrücklich vermeidet. **RRF ist kein Universaloptimum:** Weil es nur Ränge nutzt, wirft es die Konfidenz weg — ein „0,95 vs. 0,72" und ein „0,73 vs. 0,72" sehen im Rang gleich aus; bei gut kalibrierten Scores kann gewichtete Fusion es schlagen. Und **Hybrid gewinnt meist, nicht immer:** Der Nutzen hängt vom Query-Mix (lexikalisch- vs. bedeutungslastig) und davon ab, wie korreliert die Fehler der beiden Ranker sind — ein schlecht gewichteter Hybrid kann hinter einem einzelnen starken Retriever zurückfallen.

Markierungen in dieser Leseprobe

Einzelne Sätze tragen eine Markierung, die sagt, woher die Aussage stammt und wie belastbar sie ist. Hier stehen nur die, die in dieser Leseprobe vorkommen.

Markierung	Bedeutung
[Studienbefund]	Ergebnis einer einzelnen Studie oder eines Berichts, datiert — kein Konsens.