IMEN891K / IDSC501 / AXGS-501 · DATA ENGINEERING · FALL 2026

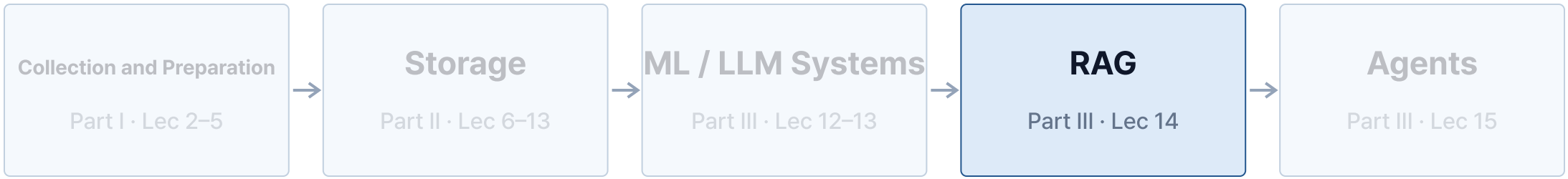
16.

RAG

Sangwoo Mo · POSTECH

Industrial & Management Engineering

Where We Are



Part III connects stored data to models, retrieval, and actions

Outline: RAG

01

How RAG Answers Questions

find supporting text before generating an answer

02

Preparing Documents for Search

turn source documents into searchable pieces

03

Finding Evidence and Checking Answers

find relevant text and verify the answer

01 · How RAG Answers Questions

01

How RAG Answers Questions

find supporting text before generating an answer

02

Preparing Documents for Search

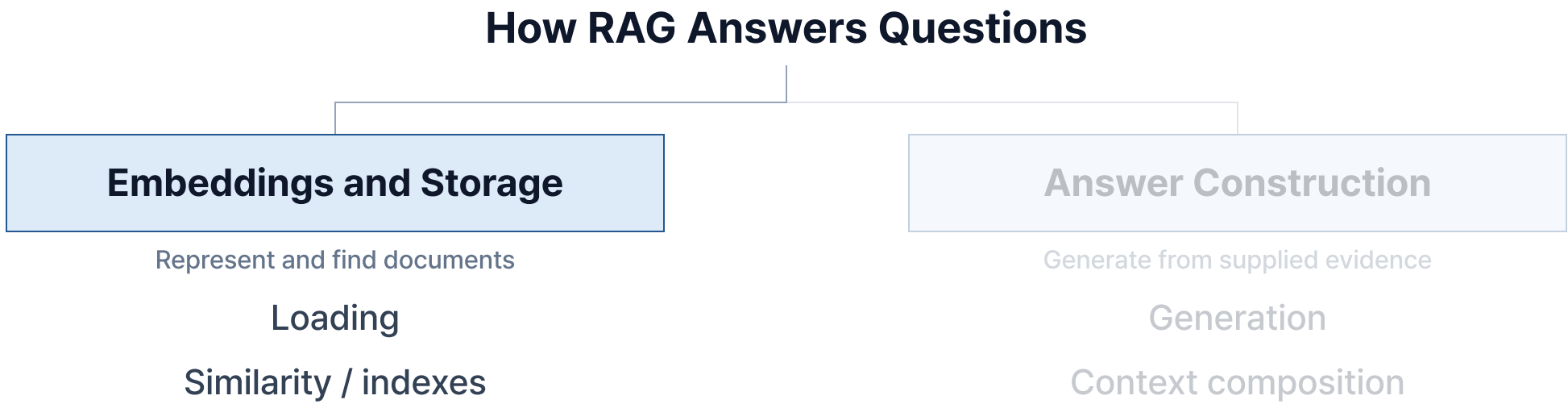
turn source documents into searchable pieces

03

Finding Evidence and Checking Answers

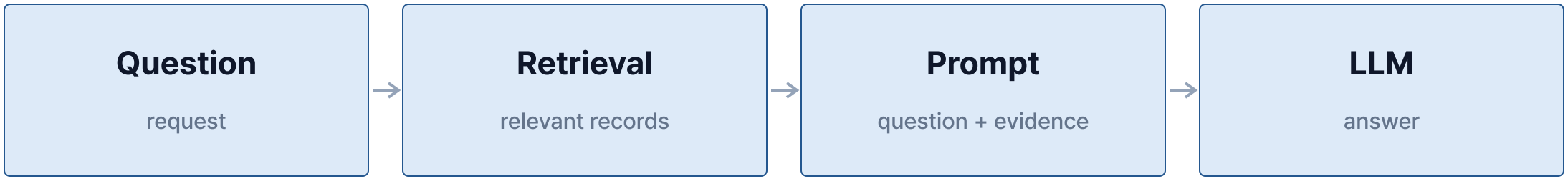
find relevant text and verify the answer

§ Embeddings and Storage



Recap: LLM Workflow

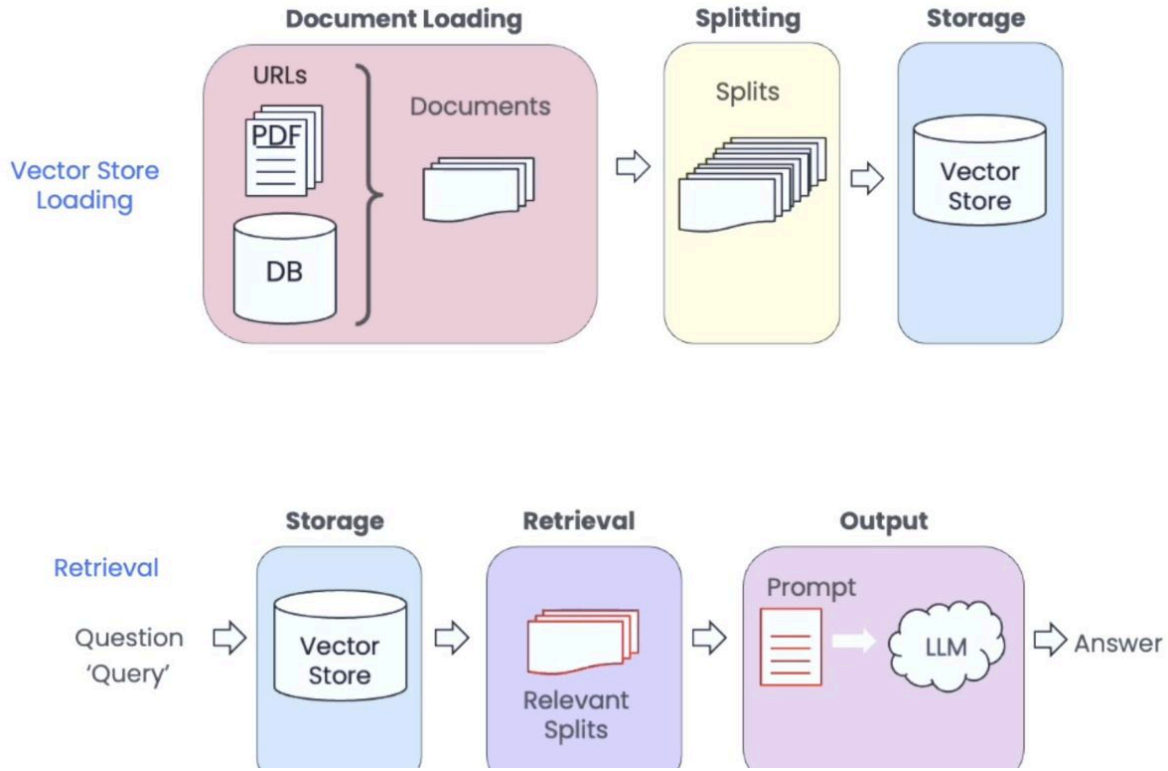
A retrieval step supplies external evidence to the model



Question → Retrieval → Prompt → LLM

Retrieval-Augmented Generation

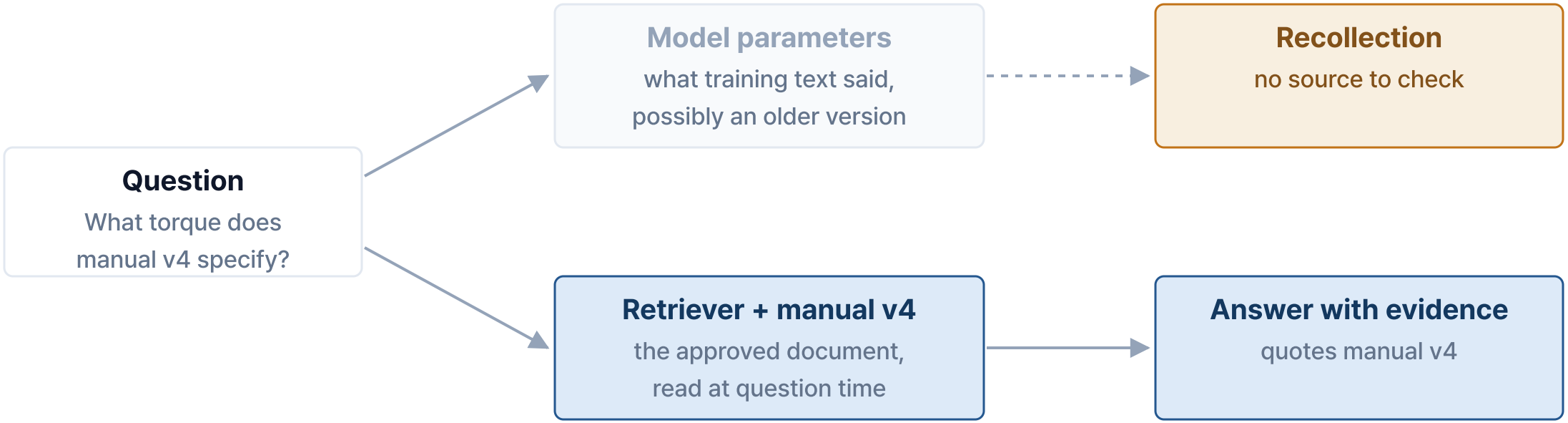
RAG conditions generation on evidence retrieved at query time



Indexing prepares documents; retrieval supplies answer context

Why Retrieval

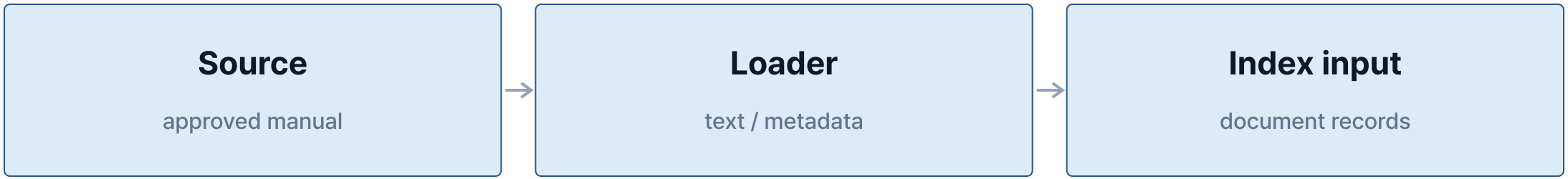
Private and changing facts need evidence outside model parameters



Retrieval reads the **approved manual** at question time

Document Loader

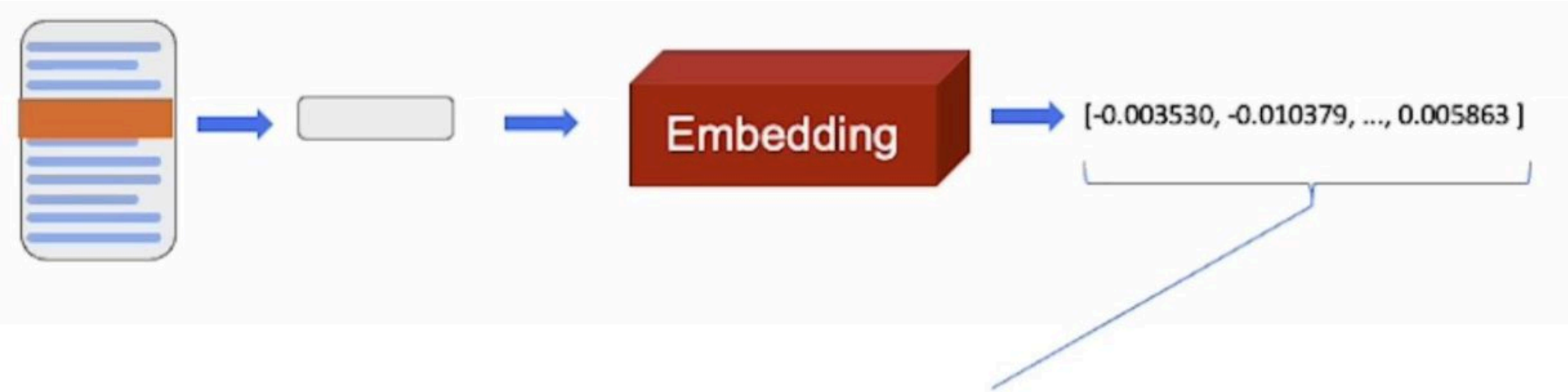
A loader supplies text and source metadata to the indexing pipeline



The source reference travels with the loaded content

Text Embeddings

An embedding maps text into a vector used for similarity search

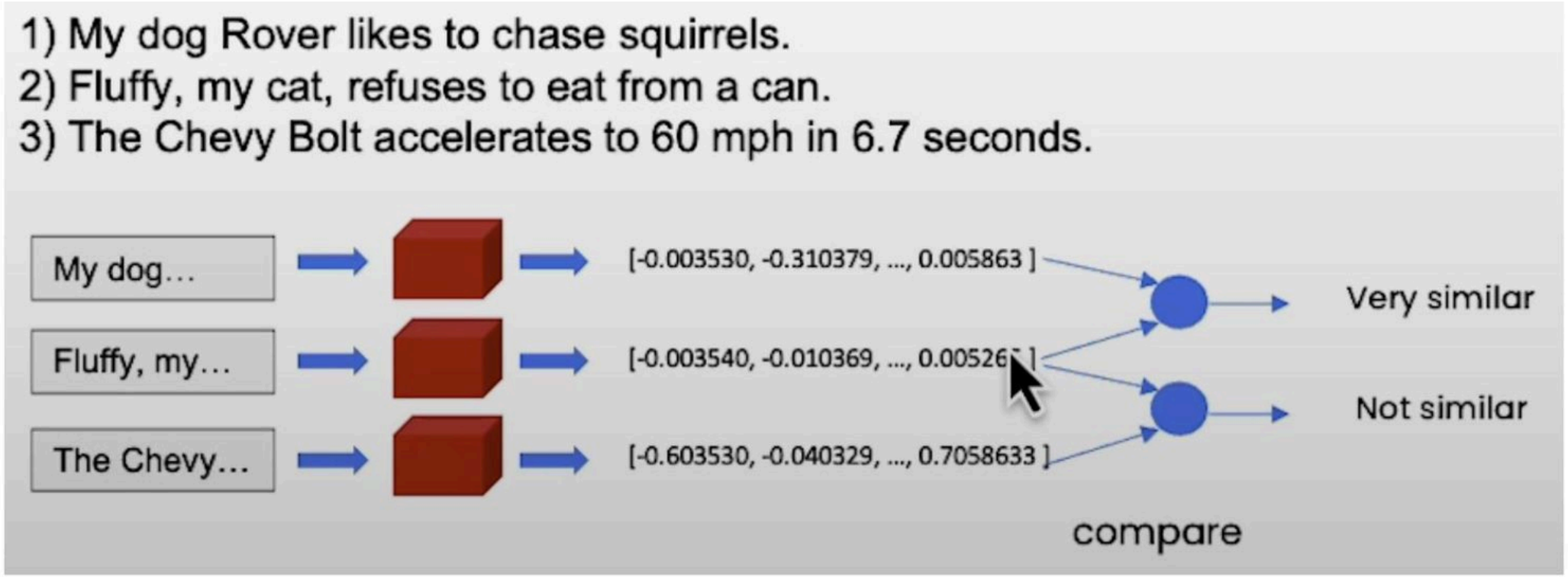


- Embedding vector captures content/meaning
- Text with similar content will have similar vectors

Similar wording or meaning can produce nearby vectors

Embedding Similarity

Semantic similarity can differ from exact word overlap

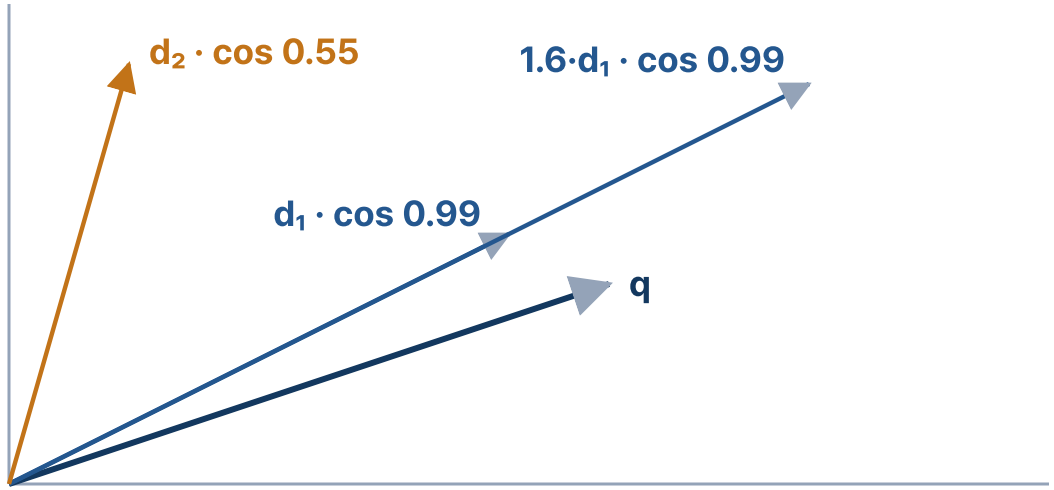


The two animal sentences are closer than the car sentence

Cosine Similarity

Cosine similarity compares vector direction

$$\cos(q, d) = (q \cdot d) / (\|q\| \|d\|)$$



Direction, not length

d_1 and a longer vector along d_1 get the same cosine

Two cautions

normalize if the index scores dot product

a score of 0.99 is not a probability

Same direction, same score, **whatever the length**

Embedding Interface

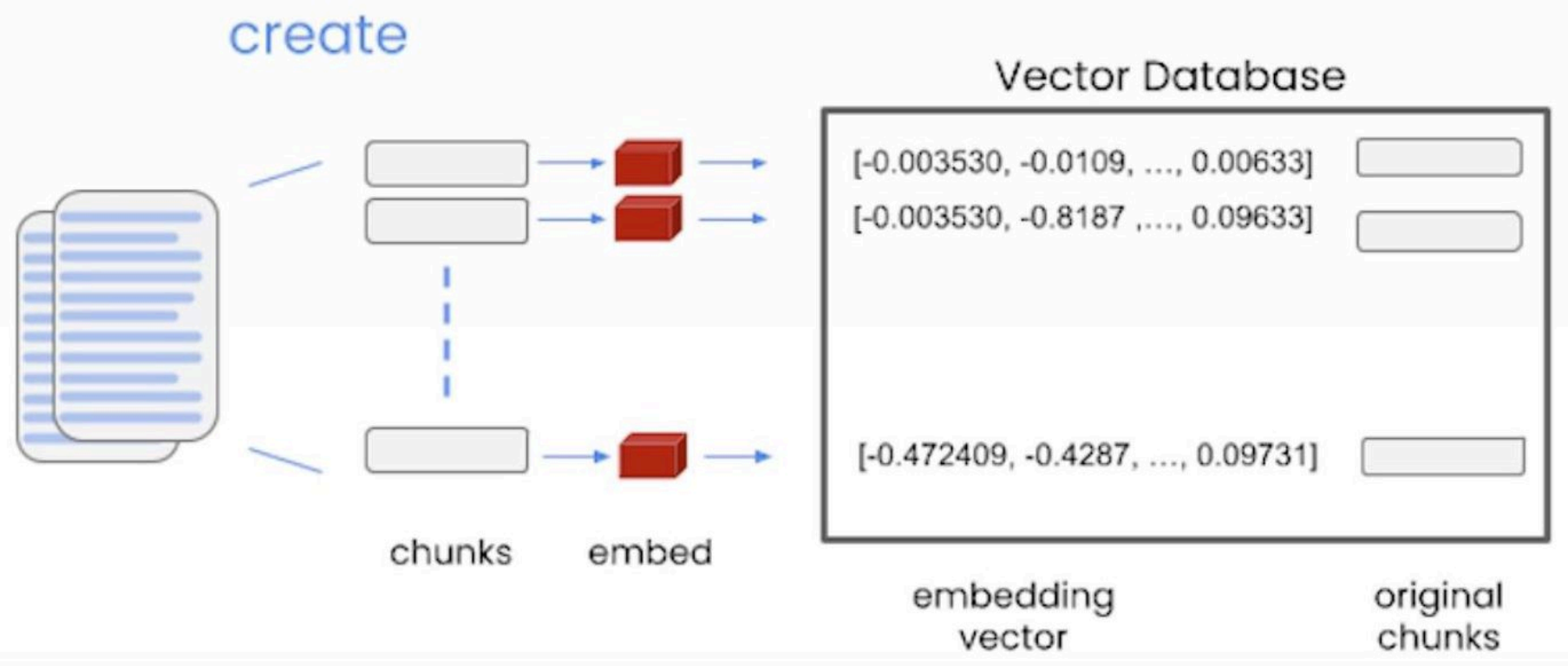
Use one compatible embedding space for documents and queries

```
import { OllamaEmbeddings } from '@langchain/ollama';

const embeddings = new OllamaEmbeddings({
  model: '<installed-embedding-model>',
  baseUrl: 'http://127.0.0.1:11434'
});
const documents = await embeddings.embedDocuments([
  'P-17 pressure alarm threshold is 5 bar.'
]);
const query = await embeddings.embedQuery('P-17 pressure threshold');
// Use one embedding model and compatible preprocessing for both.
```

Vector Storage

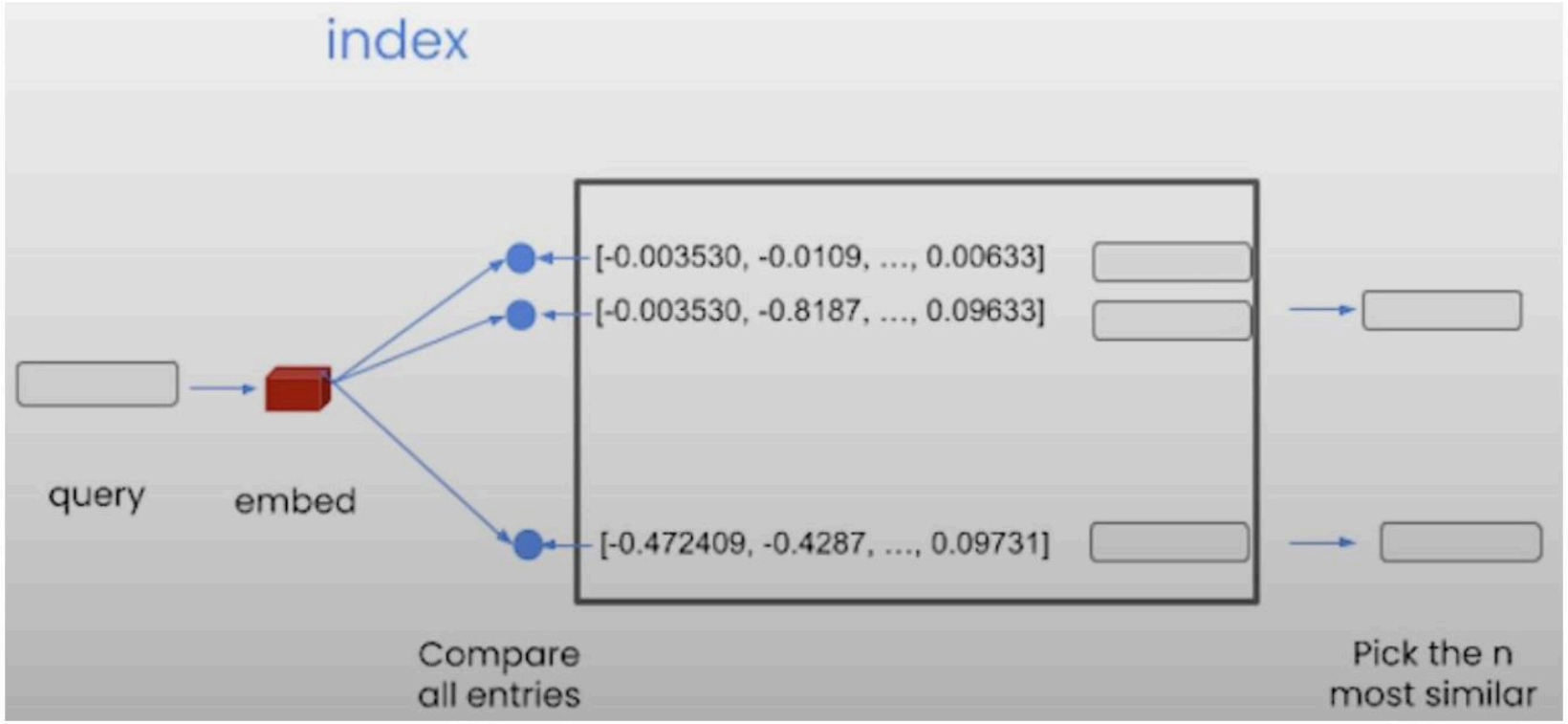
Store vectors together with the original chunks and metadata



A vector is a search key; the text remains answer evidence

Vector Search

A query vector retrieves nearby chunk vectors

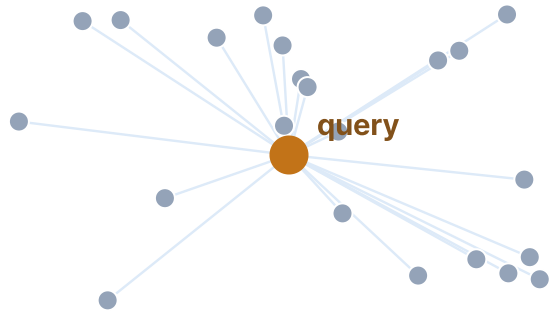


The query selects candidate documents through the vector index

Vector Index Choices

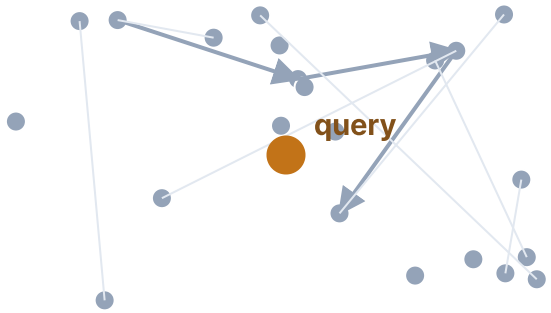
Select the index using collection size, latency, and recall

Exact scan



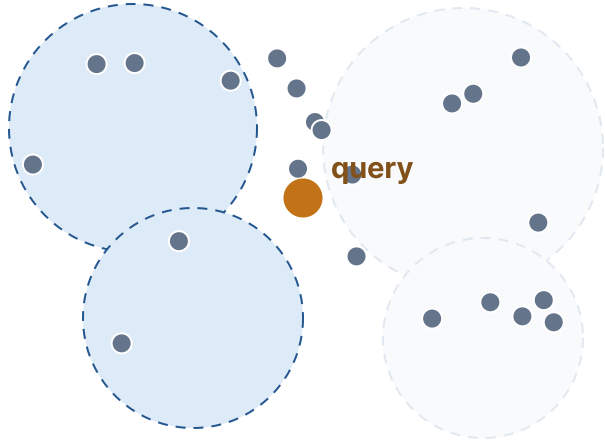
scores every vector: work grows with size

HNSW



hops a neighbour graph: memory vs effort

IVF

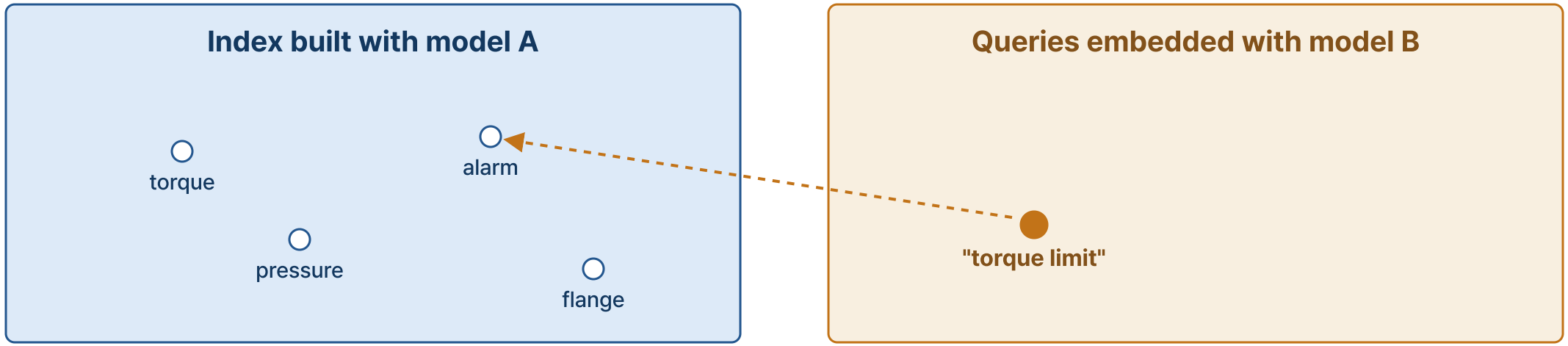


probes a few clusters: recall depends on probes

Index recall and embedding relevance are **separate questions**

Caveat: Embedding Version

A model change requires a compatible query and document index

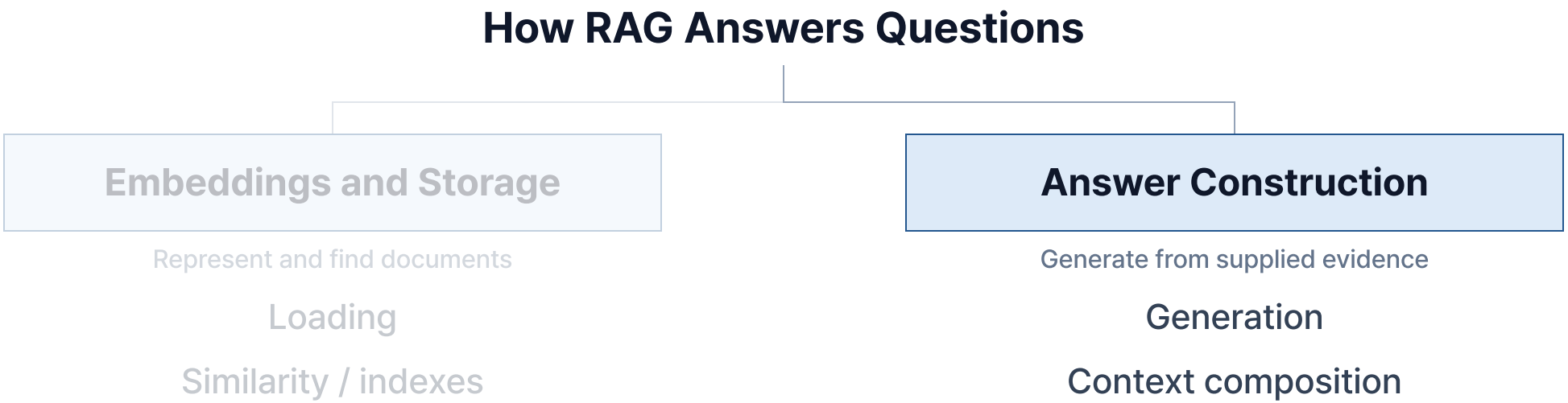


Same 768 dimensions, different coordinate systems: the nearest neighbour is meaningless

Re-embed every document, or keep both indexes until the switch

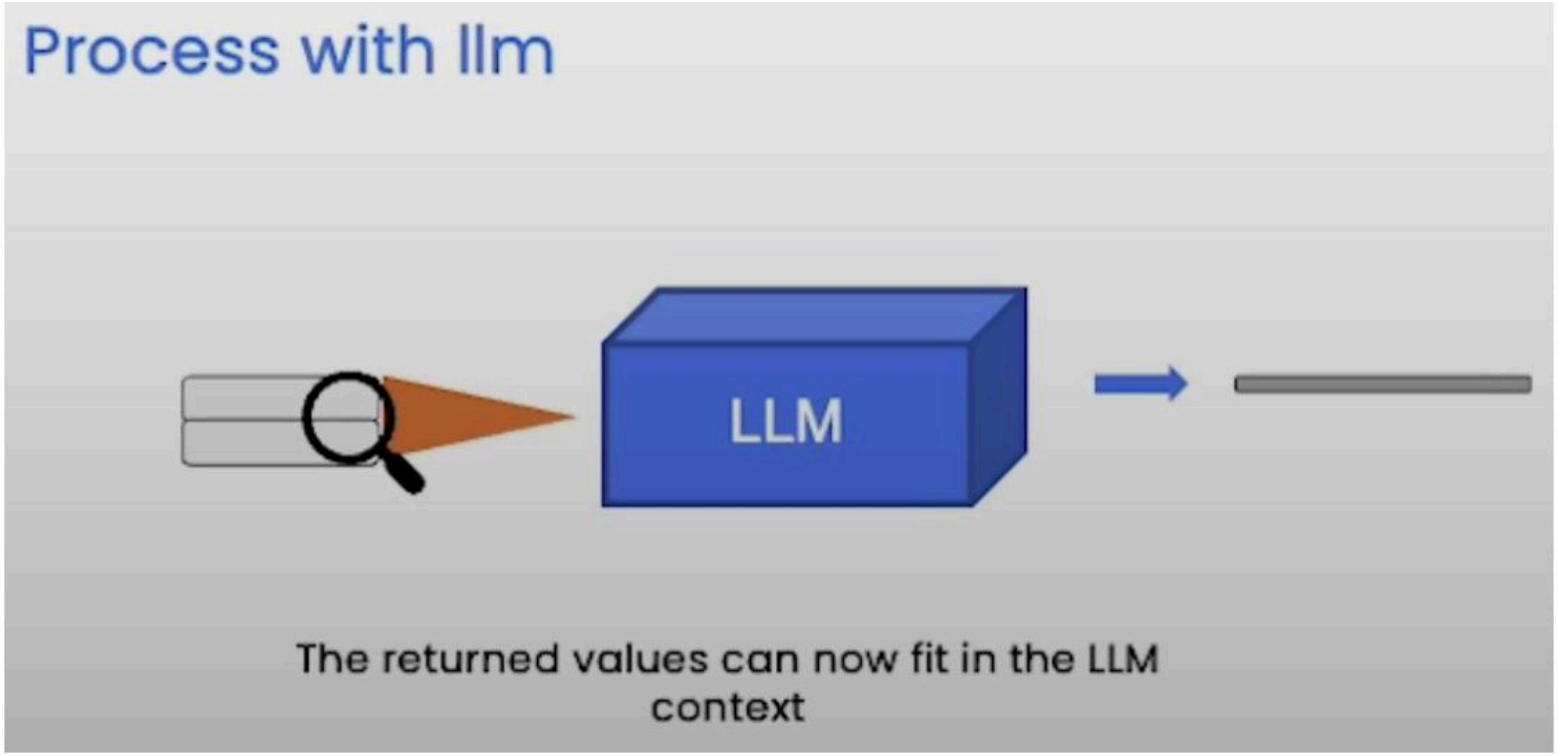
Mixing models compares points from **two different spaces**

§ Answer Construction



Answer Generation

The model receives the question and selected evidence



The answer prompt receives retrieved chunks, not the whole corpus

Context Record

Keep the selected text and its source reference together

```
context = [
  {'source_id': 'manual_v4:p12',
   'text': 'Steel flange: tighten to 12 N·m.'},
  {'source_id': 'manual_v4:p13',
   'text': 'Polymer flange: tighten to 8 N·m.'}
]
```

Grounded Answer Prompt

Ask for evidence references and an explicit unsupported outcome

```
# Prompt template
Use only the evidence below for factual claims.
If the evidence is insufficient, say what is missing.
Attach source IDs to the claims they support.

Question: {question}
Evidence: {context}
```

Context Composition Choices

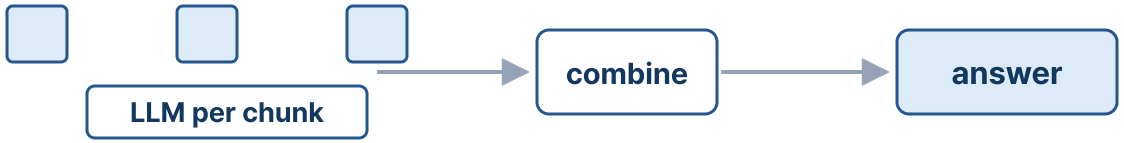
Choose how retrieved evidence enters generation

Stuff
evidence fits in context



one generation call

Map–reduce
many independent chunks



can lose cross-chunk links

Refine
ordered evidence



sequential; order-sensitive

Map–rerank
one candidate answer per chunk



compare answer scores

Four ways evidence enters **generation**

02 · Preparing Documents for Search

01

How RAG Answers Questions

find supporting text before generating an answer

02

Preparing Documents for Search

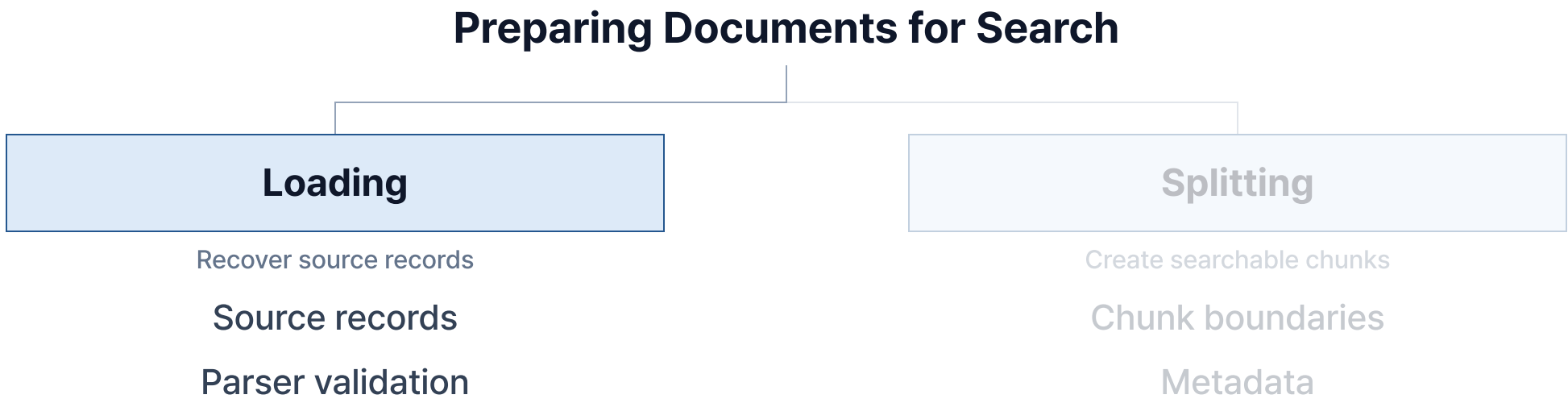
turn source documents into searchable pieces

03

Finding Evidence and Checking Answers

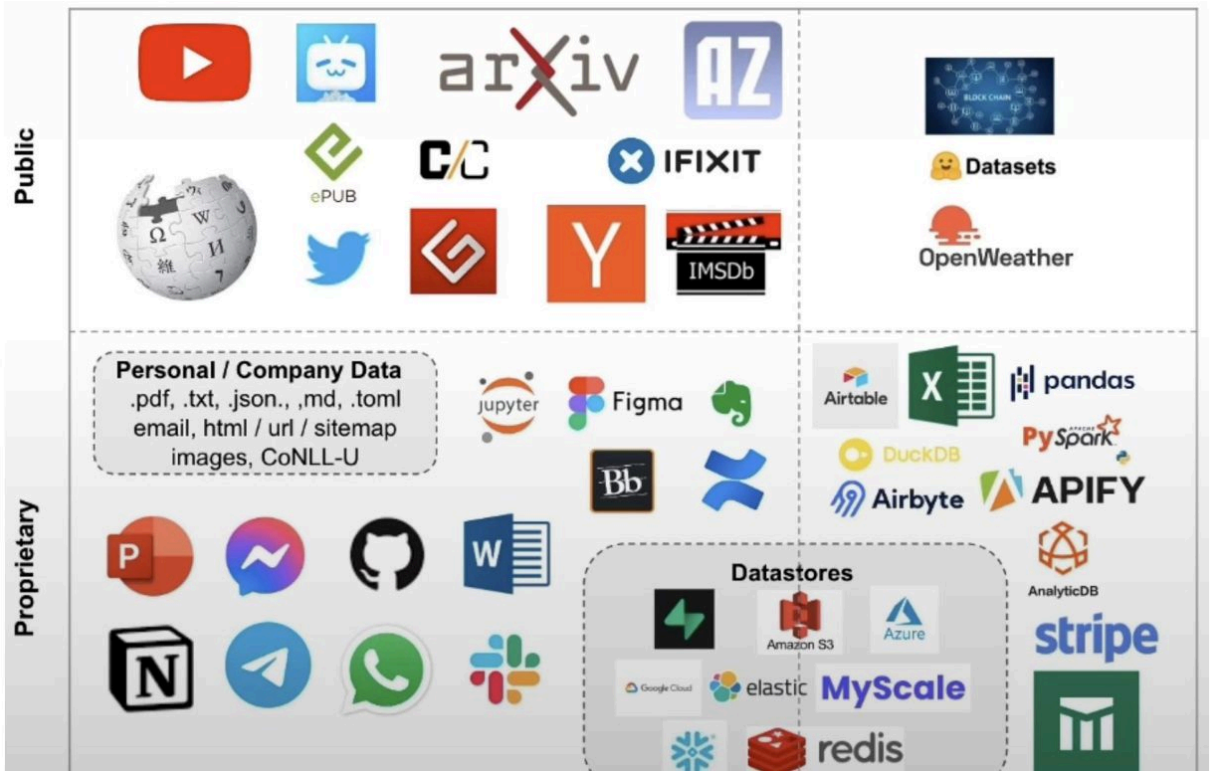
find relevant text and verify the answer

§ Loading


Preparing Documents for Search
Loading
Recover source records
Source records
Parser validation
Splitting
Create searchable chunks
Chunk boundaries
Metadata

Document Loading

A loader reads a source and returns content with metadata



Loaders normalize heterogeneous sources into documents

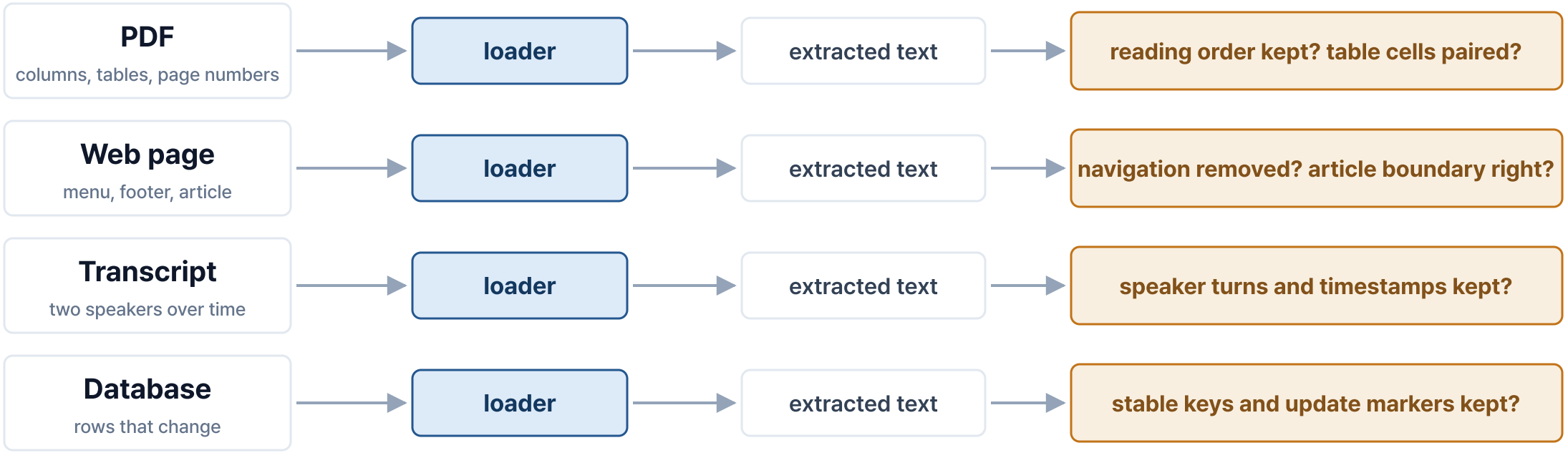
Document Record

Preserve identity and provenance alongside the text

```
document = {
  'doc_id': 'pump_manual_v4',
  'text': extracted_text,
  'metadata': {
    'source': 'manuals/pump_v4.pdf',
    'page': 12, 'version': 4,
    'access_group': 'maintenance'
  }
}
```

Loader Validation

Loading success does not imply faithful content extraction



Inspect extracted content **before** it enters the index

Caveat: PDF Reading Order

Flattening a table can detach a number from its meaning

Table in the PDF

Pump	Torque
Pump A	12 N·m
Pump B	18 N·m

flatten →

Text the index receives

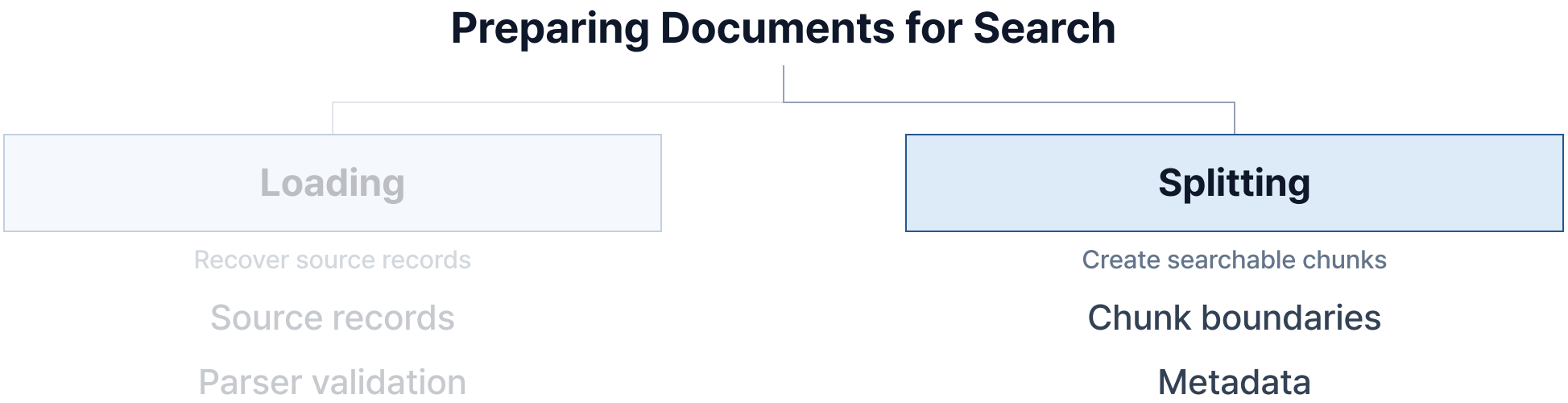
```
Pump A Pump B 12 18
```

Which torque belongs to Pump A?

Keep the layout, or store the table as rows: pump → torque

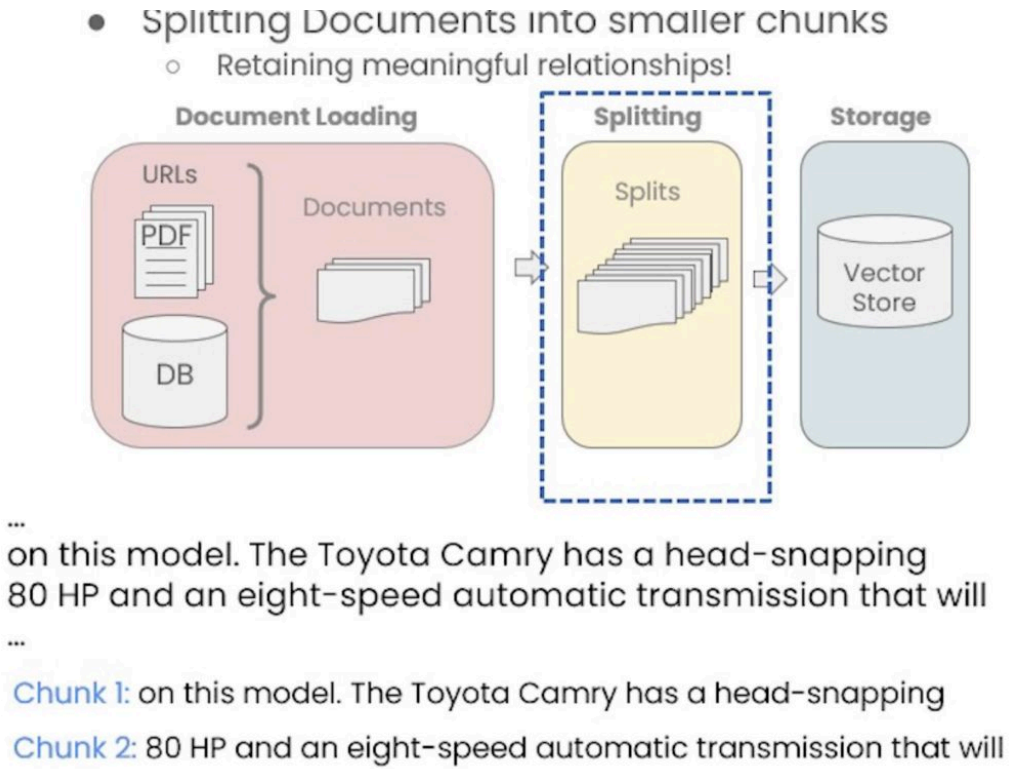
Flattening detaches **12** from **Pump A**

§ Splitting


Preparing Documents for Search
Loading
Recover source records
Source records
Parser validation
Splitting
Create searchable chunks
Chunk boundaries
Metadata

Document Splitting

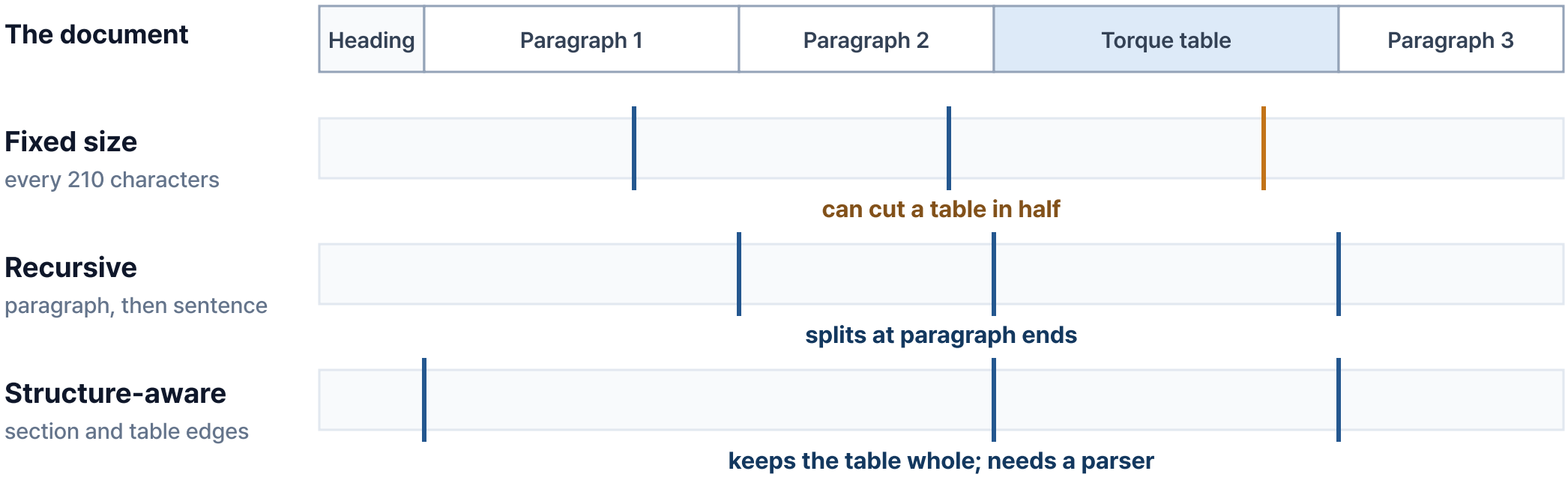
Splitting creates the retrieval units from loaded documents



Chunk boundaries determine which evidence travels together

Splitting Strategies

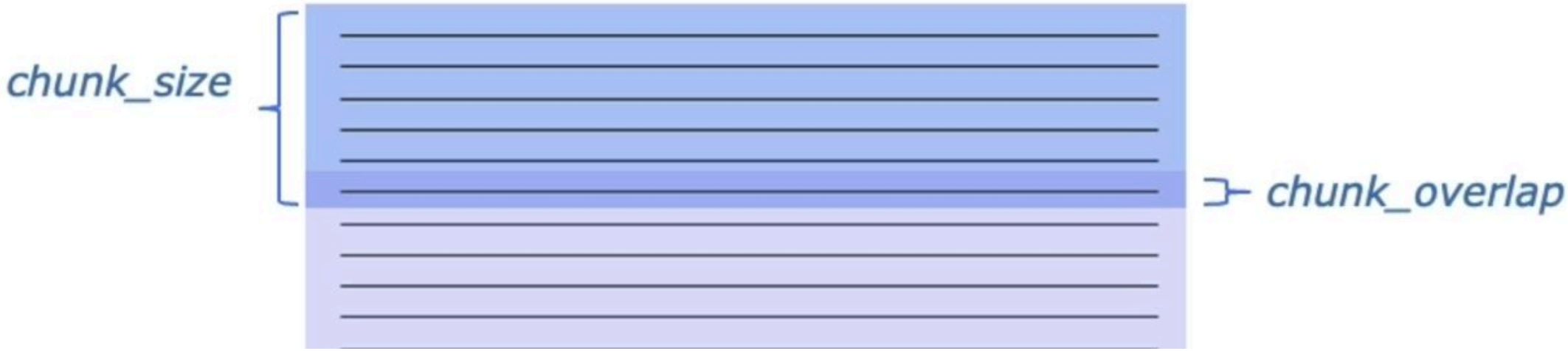
Choose boundaries that match the document structure



The chunk must keep the **evidence** an answer needs

Chunk Size and Overlap

Overlap keeps some boundary context at the cost of duplication



Neighboring chunks share a controlled overlap

Splitting the Manual

Run a real text splitter and inspect the retained metadata

```
const splitter = new RecursiveCharacterTextSplitter({
  chunkSize: 85, chunkOverlap: 15
});
const chunks = await splitter.splitDocuments(manuals);
return chunks.map(d => ({id:d.metadata.id,
  text:d.pageContent, length:d.pageContent.length}));
```

Demo: Manual Chunks

Change chunk size and overlap; inspect text and source IDs

1. Set chunkSize to 35 and inspect split conditions.
2. Compare chunkOverlap 0 and 15. Keep overlap smaller than chunk size.

```
Type code or click Starter
```

```
Run code to inspect its output.
```

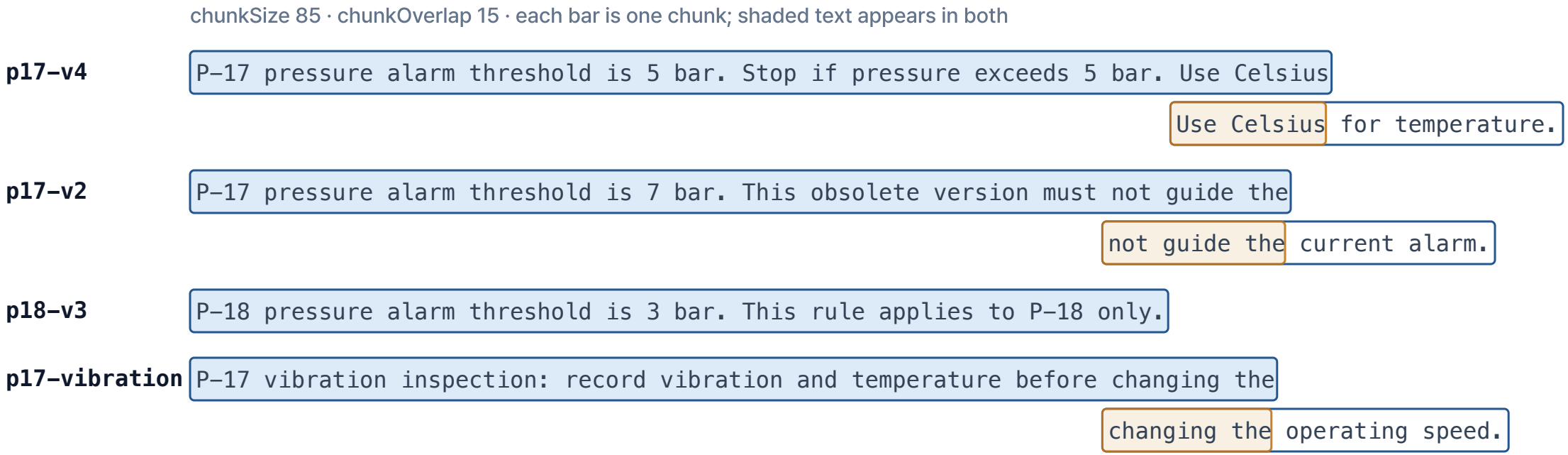
Run | Starter | Reset

Ready

LangChain.js · JavaScript · Ctrl/Cmd+Enter · Real library execution with local data

Answer: Manual Chunks

The baseline output comes from executing the shown code



Overlap repeats a few words; split **conditions** still separate

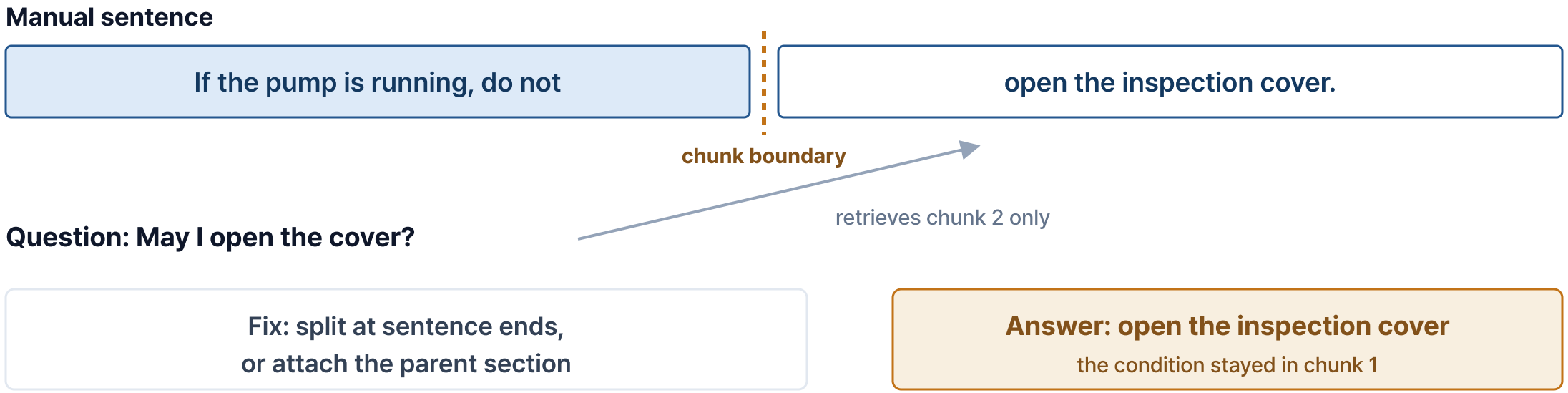
Chunk Identity

Assign each chunk a stable source reference

```
chunk = {
  'chunk_id': 'pump_manual_v4:p12:c2',
  'doc_id': 'pump_manual_v4',
  'text': 'Tighten the flange bolts to 12 N·m.',
  'metadata': {'page': 12, 'section': 'Flange assembly'}
}
```

Caveat: Broken Evidence

Splitting a condition from an instruction can reverse the answer



A chunk boundary can **drop the condition**

03 · Finding Evidence and Checking Answers

01

How RAG Answers Questions

find supporting text before generating an answer

02

Preparing Documents for Search

turn source documents into searchable pieces

03

Finding Evidence and Checking Answers

find relevant text and verify the answer

§ Retrieval

Finding Evidence and Checking Answers

Retrieval

Select evidence for a question

Search / filters

Ranking / context

Evaluation and Updates

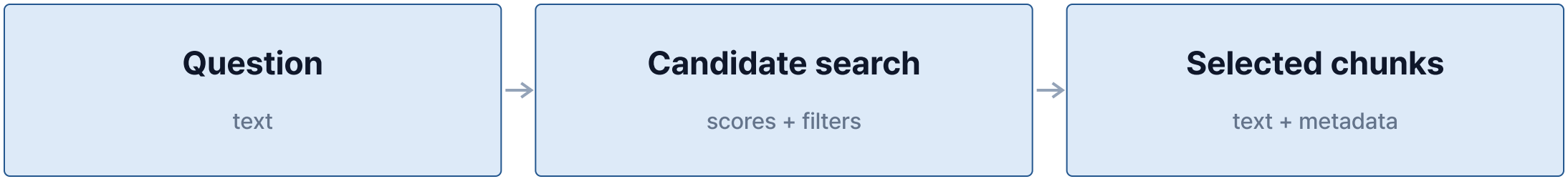
Check quality and freshness

Conversational QA

Evaluation / refresh

Retriever Interface

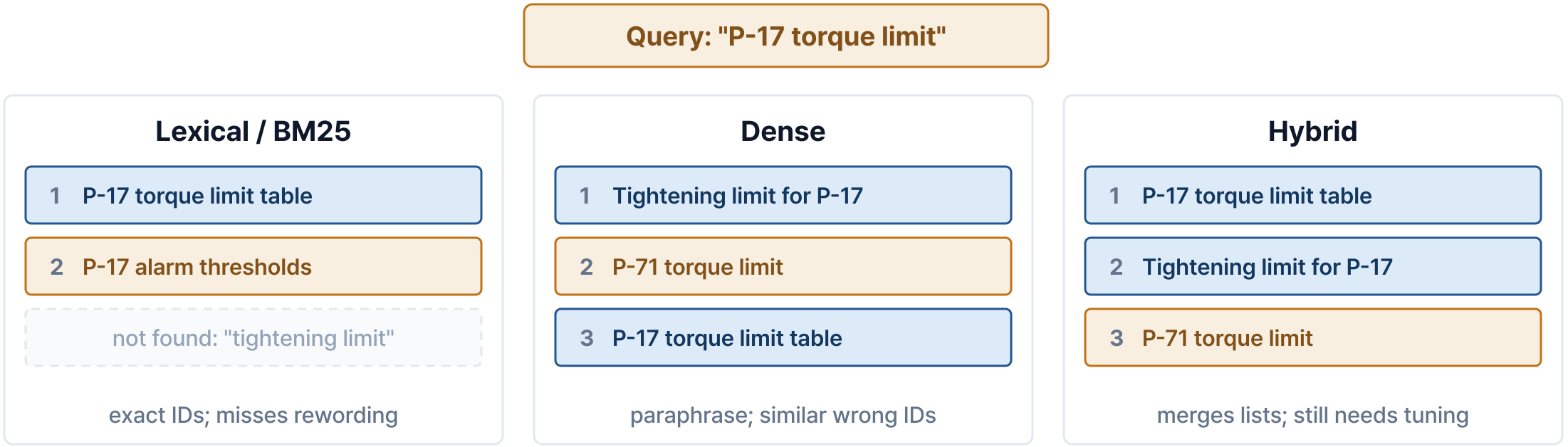
A retriever returns evidence records for a query



Question → Candidate search → Selected chunks

Search Methods

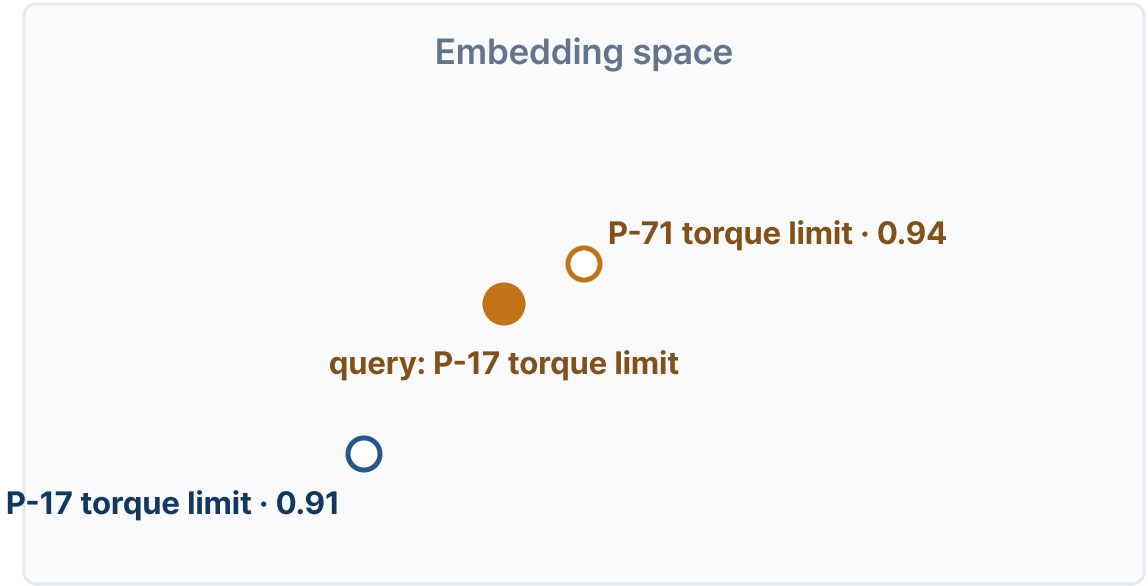
Lexical and dense search fail on different queries



Exact IDs need lexical matching; rewording needs **dense** search

Caveat: Similar but Wrong

A high similarity score can retrieve the wrong entity



Top hit is the wrong pump

two IDs differ by one transposed digit,
the sentences are almost identical

Exact constraint

filter equipment_id = P-17
before ranking by similarity

The closest vector belongs to the **wrong pump**

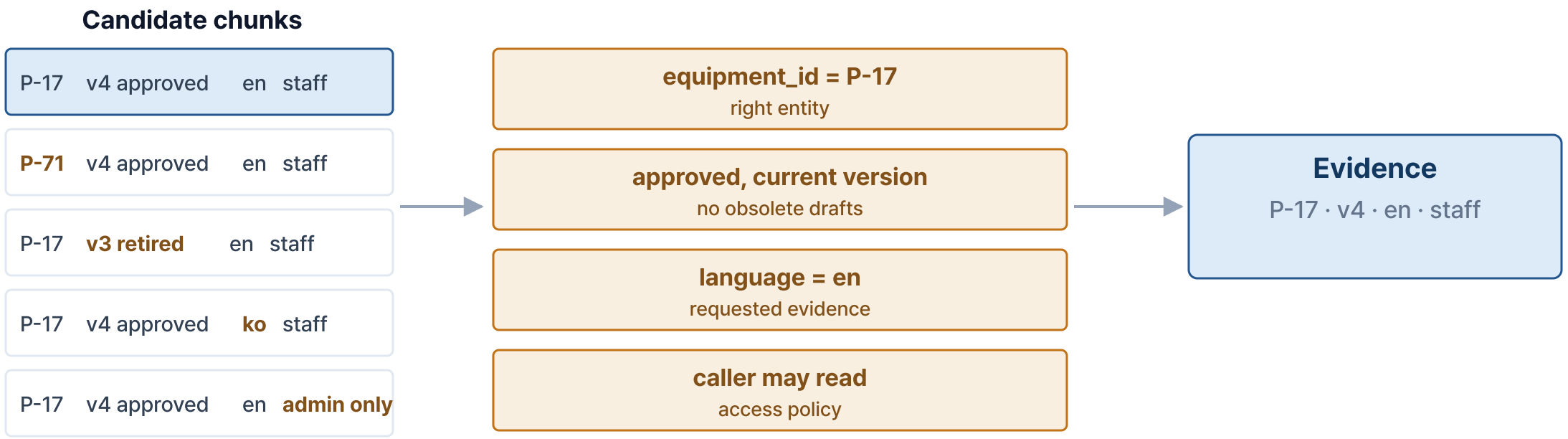
Retrieval Request

Carry constraints separately from the natural-language question

```
query = {
  'text': 'What is the flange torque?',
  'filters': {'equipment_id': 'P-17', 'approved': True},
  'top_k': 5
}
# Conceptual interface; actual filter syntax depends on the store.
```

Metadata Filters

Restrict candidate records using source properties



Each mismatched tag removes a **candidate**

Searching the Manual

Retrieve documents with a real vector store and an exact filter

```
const chunks = await new RecursiveCharacterTextSplitter({
  chunkSize: 100, chunkOverlap: 15
}).splitDocuments(manuals);
const store = await MemoryVectorStore.fromDocuments(chunks, makeEmbeddings());
const k = 2;
const filter = d => d.metadata.equipment === "P-17" && d.metadata.approved;
const hits = await store.similaritySearch("P-17 pressure alarm threshold", k, filter);
return {embedding:EMBEDDING_MODE, hits:hits.map(d => ({
  source:d.metadata.id, text:d.pageContent}))};
```

Demo: Filtered Retrieval

Change k and the filter; inspect which version becomes evidence

1. Remove approved from the filter. Does the obsolete 7 bar rule appear?
2. Search P-18 and update the equipment filter together.

Execution: Offline replay | Installed model na | Embedding model | http://127.0.0.1:11434

```
Type code or click Starter
```

```
Run code to inspect its output.
```

Run | Starter | RAG Chain | Reset

Ready

LangChain.js · JavaScript · Ctrl/Cmd+Enter · Offline replay uses fixed responses; local mode calls your model

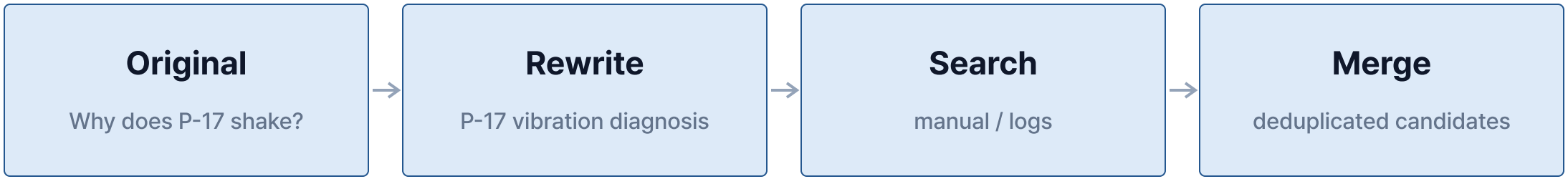
Answer: Filtered Retrieval

The baseline output comes from executing the shown code


```
{"embedding": "Local token-presence vectors (not learned)", "hits": [{"source": "p17-v4", "text": "P-17 pressure alarm threshold is 5 bar. Stop if pressure exceeds 5 bar. Use Celsius for temperature."}, {"source": "p17-vibration", "text": "P-17 vibration inspection: record vibration and temperature before changing the operating speed."}]}
```

Query Reformulation

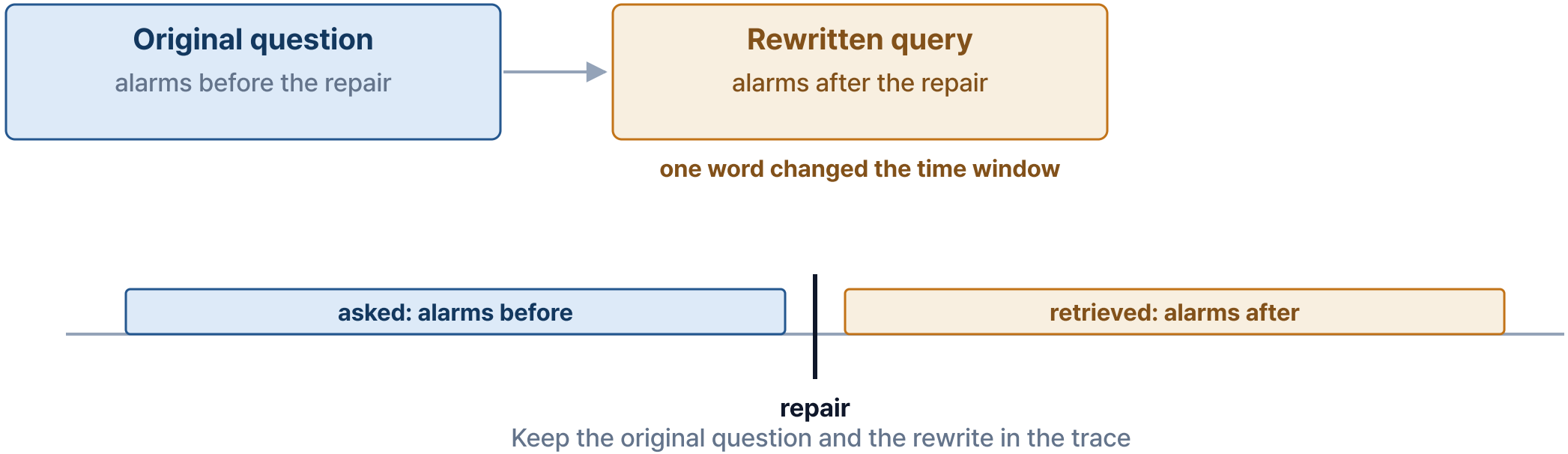
A rewritten query can recover evidence missed by the original wording



Original → Rewrite → Search → Merge

Caveat: Query Drift

A rewrite can silently change the question



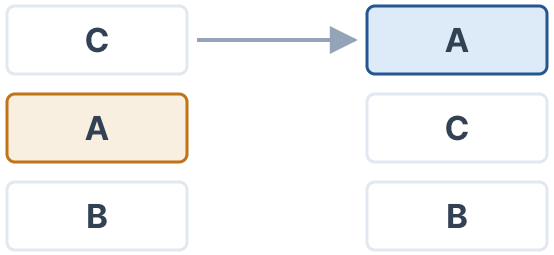
Retrieval succeeded on the **wrong question**

Evidence Selection Methods

Choose the operation by what the candidate set still lacks

Reranking

changes the order

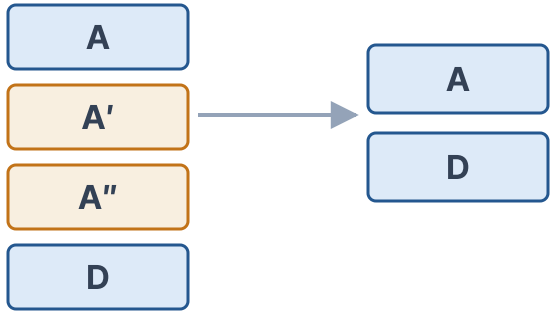


use when

relevant evidence ranked too low

MMR

changes the selected set

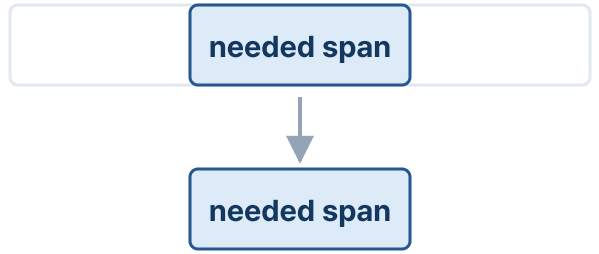


use when

top hits repeat the same content

Compression

changes the text spans



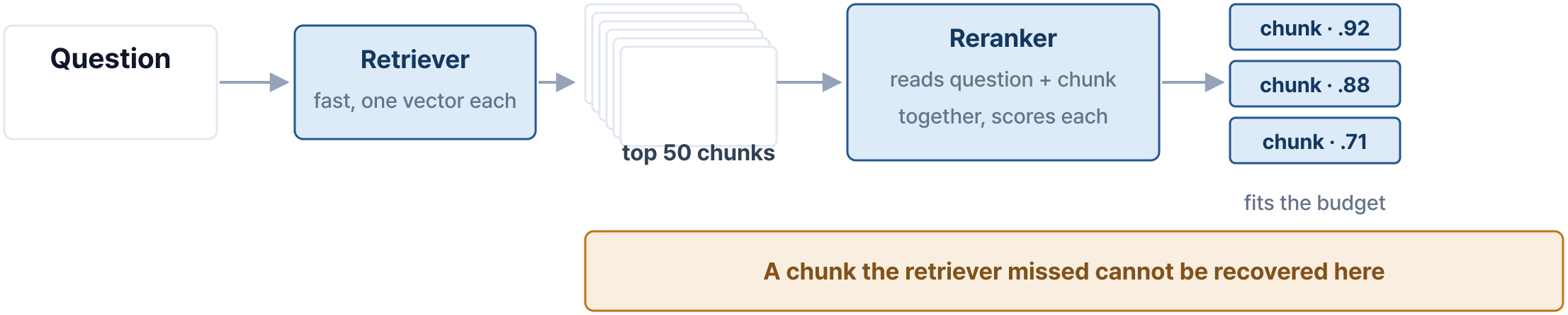
use when

chunks carry excess context

Choose by what the candidate set **still lacks**

Reranking

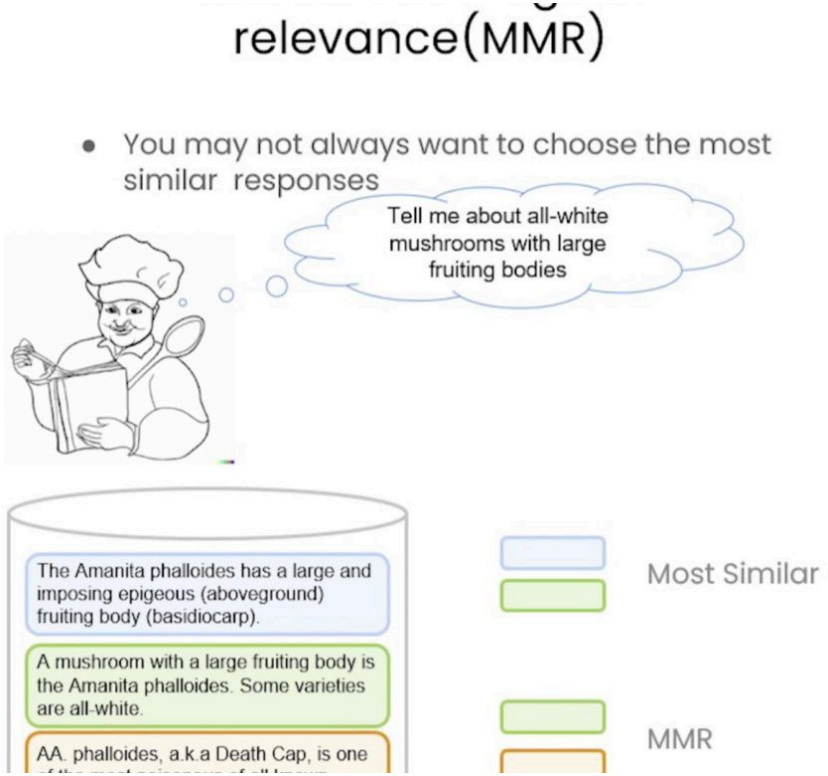
A reranker scores the query together with each candidate



Reranking reorders only what **retrieval found**

Maximum Marginal Relevance

MMR balances query relevance with novelty among selected chunks

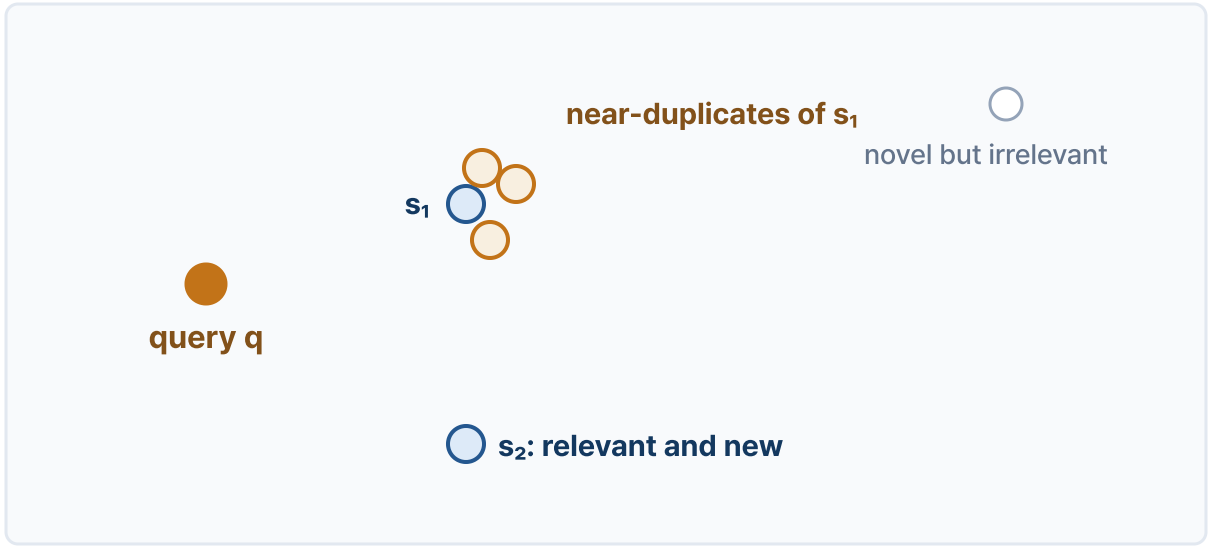


Diverse candidates reduce repeated evidence

MMR Objective

Penalize similarity to evidence that has already been selected

$$\text{argmax over } d \text{ in } C \setminus S \text{ of } \lambda \cdot \text{sim}(q,d) - (1-\lambda) \cdot \max \text{ over } s \text{ in } S \text{ of } \text{sim}(d,s)$$



λ near 1

relevance only: picks the duplicates

Lower λ

penalizes similarity to what is chosen

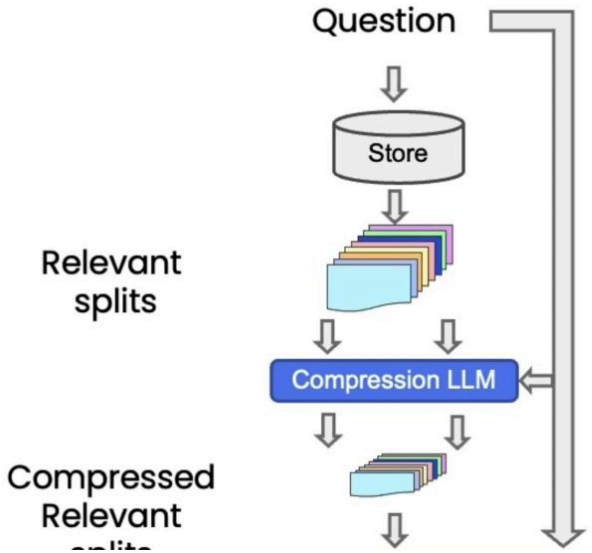
C candidates, S selected set

MMR trades **relevance** against **repetition**

Context Compression

Compression removes irrelevant text before generation

- Increase the number of results you can put in the context by shrinking the responses to only the relevant information.



Only selected evidence enters the answer prompt

Caveat: Compression Loss

Compressing away a condition can change a safety instruction

Source chunk

Tighten to 12 N·m

only for the steel flange

dropped by compression

Compressed

Tighten to 12 N·m

Now reads as a rule for every flange

check critical facts against the source span

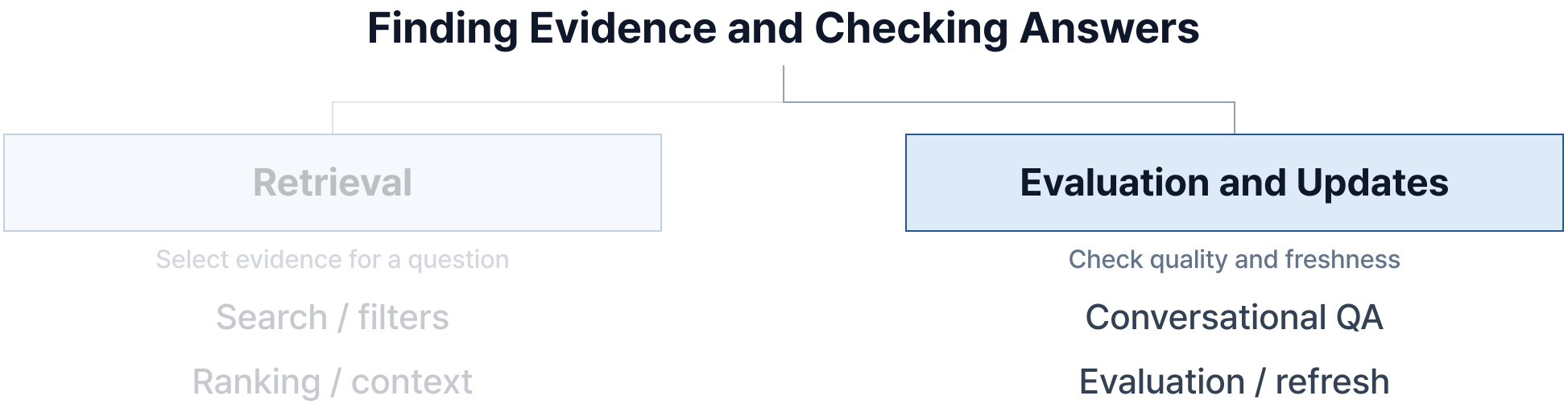
Compression removed the **condition**

Complete RAG Application

Search approved records, build context, and invoke the answer chain

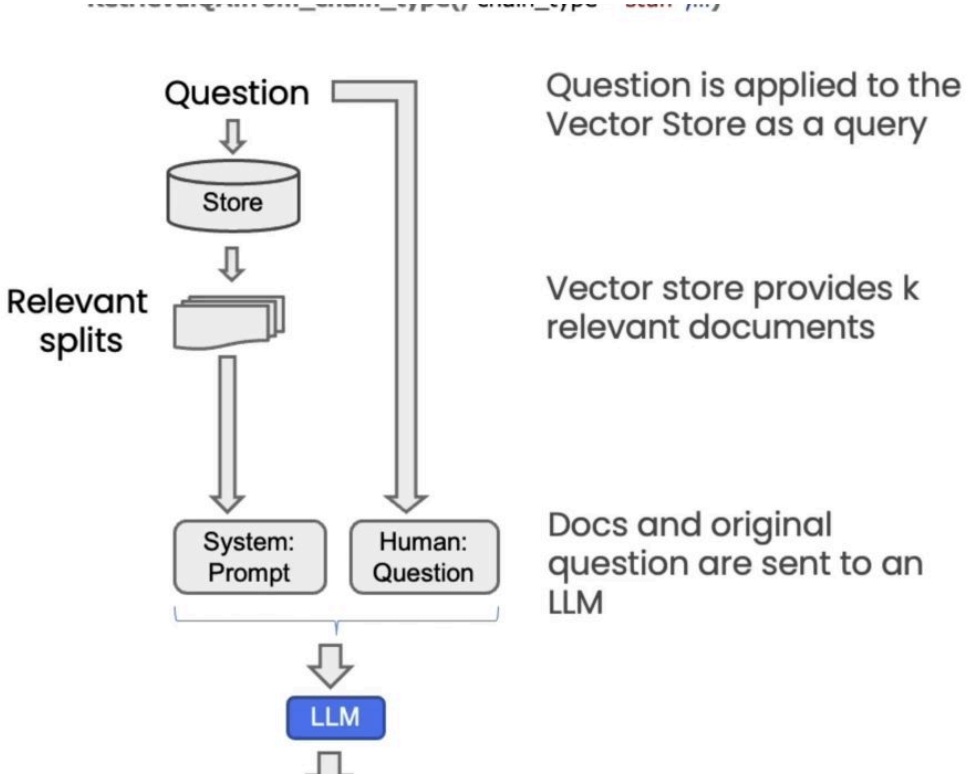
```
const chunks = await new
RecursiveCharacterTextSplitter({
  chunkSize: 100, chunkOverlap: 15
}).splitDocuments(manuals);
const store = await
MemoryVectorStore.fromDocuments(chunks,
makeEmbeddings());
const retriever = RunnableLambda.from(q =>
store.similaritySearch(q, 2,
  d => d.metadata.equipment === "P-17" &&
d.metadata.approved));
const prompt =
ChatPromptTemplate.fromMessages([
  ["system", "Use only evidence. Cite source
IDs. If missing, return unknown."],
  ["human", "Question: {question}\nEvidence:
{context}"]
]);
const model = makeModel(["P-17: 5 bar [p17-
v4]"]);
const chain = prompt.pipe(model).pipe(new
StringOutputParser());
const question = "What is the P-17 pressure
alarm threshold?";
const hits = await retriever.invoke(question);
const context = hits.map(d =>
`[${d.metadata.id}]
${d.pageContent}`).join("\n");
return {mode:MODEL_MODE,
sources:hits.map(d=>d.metadata.id),
  answer:await
chain.invoke({question,context})};
```

§ Evaluation and Updates



Question Answering Pipeline

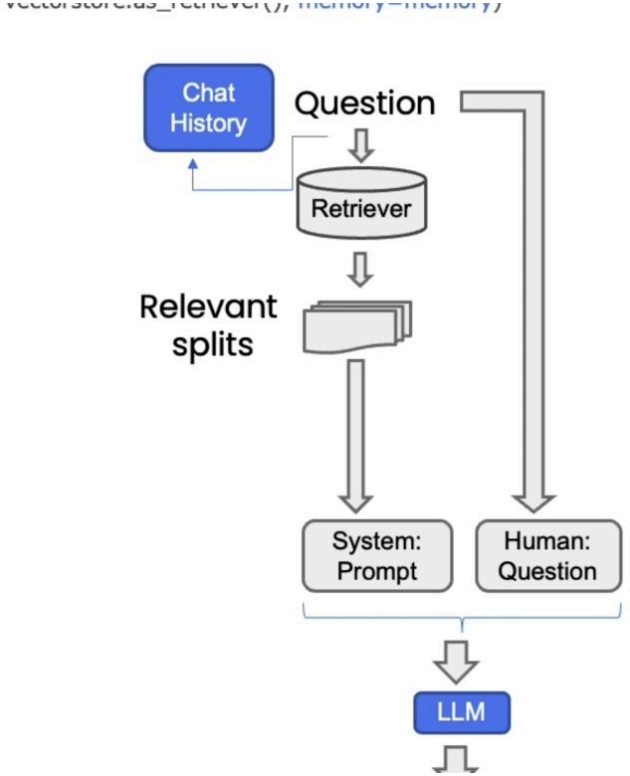
A QA pipeline combines a retriever with an answer generator



A question produces evidence before an answer

Conversational Retrieval

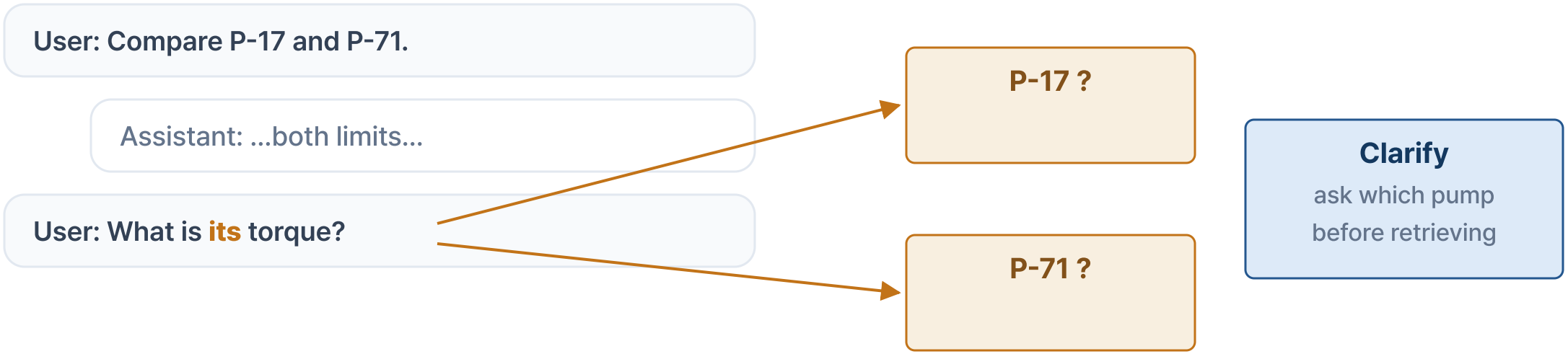
Resolve the current question before retrieving new evidence



Conversation history helps form the retrieval question

Caveat: Follow-Up Ambiguity

History can resolve a pronoun to the wrong equipment

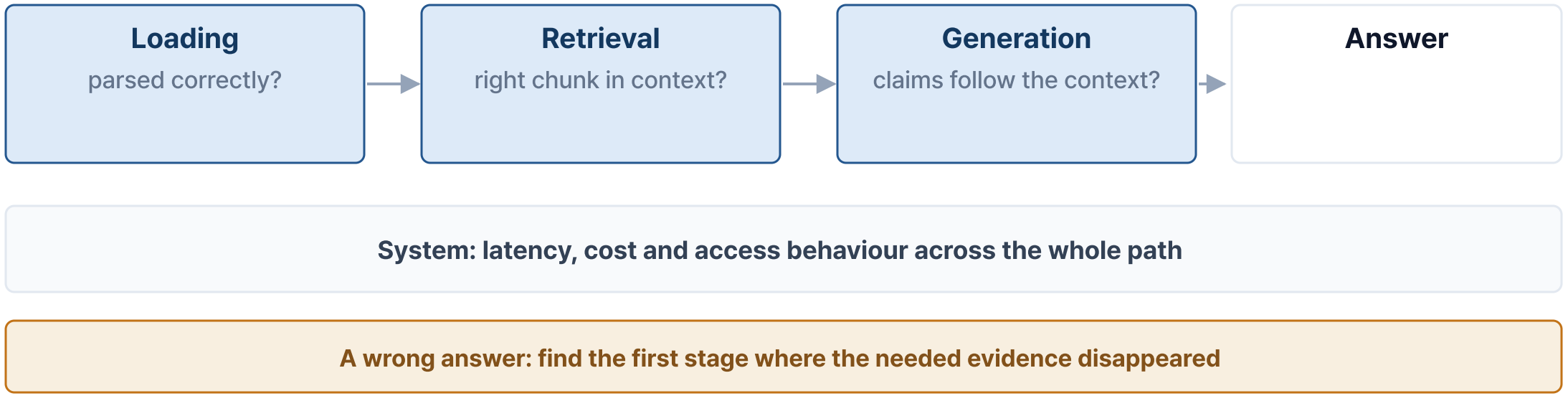


Guessing one pump turns an exact filter into confident error

"Its" points to **two pumps**

RAG Evaluation Layers

Diagnose the first stage that loses the required evidence



A wrong answer may start with **parsing or retrieval**

Retrieval Recall at k

Recall measures how many relevant chunks are retrieved

$$\text{Recall@k} = |\text{Relevant} \cap \text{Retrieved}_k| \,/\, |\text{Relevant}|$$

Retrieved, k = 5

1. **v4 pressure rule**
2. P-18 rule
3. **v4 stop condition**
4. vibration check
5. v2 obsolete rule

Relevant (labelled first)

- **v4 pressure rule**
- **v4 stop condition**
- **v4 unit note · missed**

Recall@5 = 2 / 3

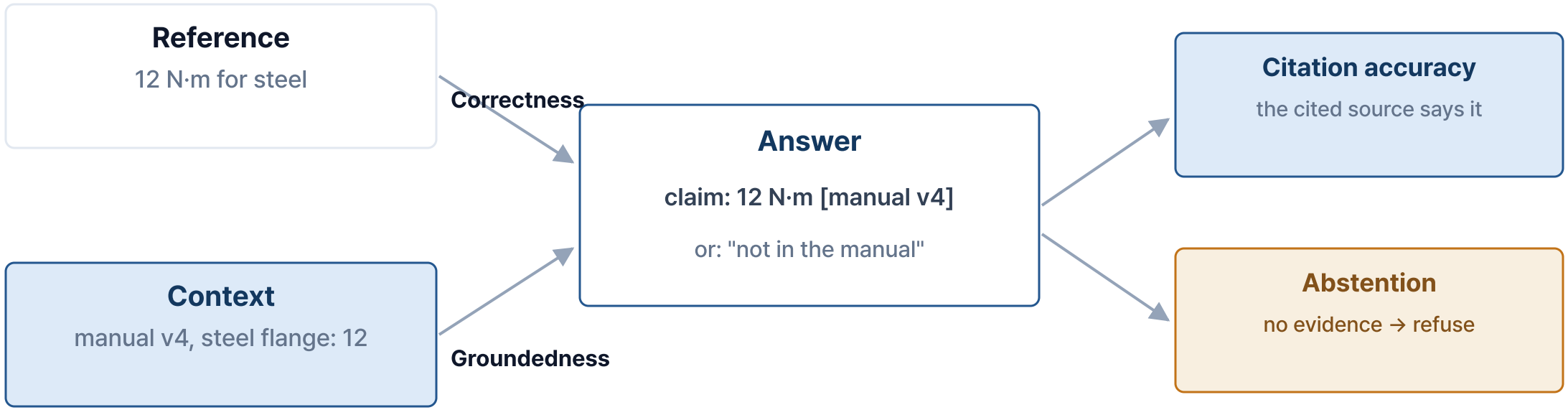
two of three relevant chunks reached the context

One required source: recall becomes a hit-or-miss check

Relevance labels come **before** the score

Answer Evaluation

Correctness and groundedness are different questions



A cited answer still needs **claim-level** checks

Case study: Versioned Manuals

A correct fact from an obsolete manual can be the wrong answer

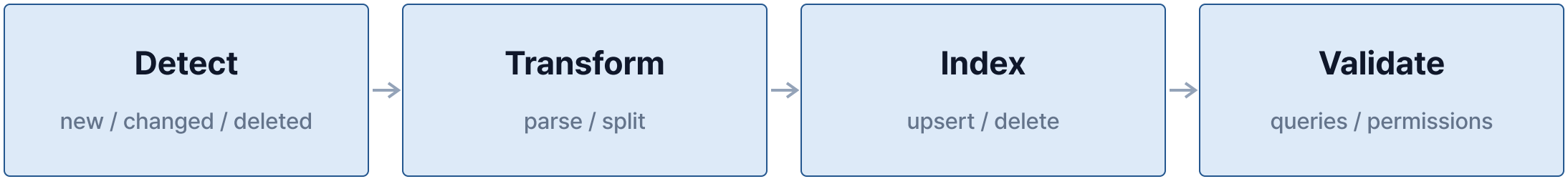
Query: approved manual, steel flange

v3 · retired	**v4 · approved**	**v4 · approved**
flange: Steel	flange: Steel	flange: Polymer
torque: 10 N·m	torque: 12 N·m	torque: 8 N·m
wrong version	the answer	wrong material

Entity, material and **version** decide the answer

Index Refresh

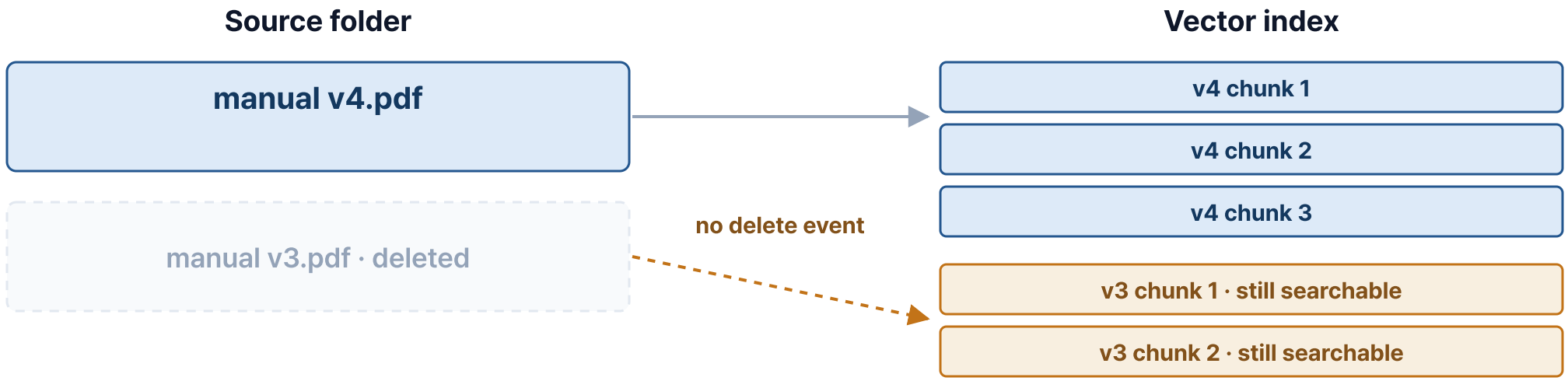
Source changes must reach chunks, vectors, and cached answers



Detect → Transform → Index → Validate

Caveat: Stale Chunks

Deleting a source file may leave its indexed text searchable

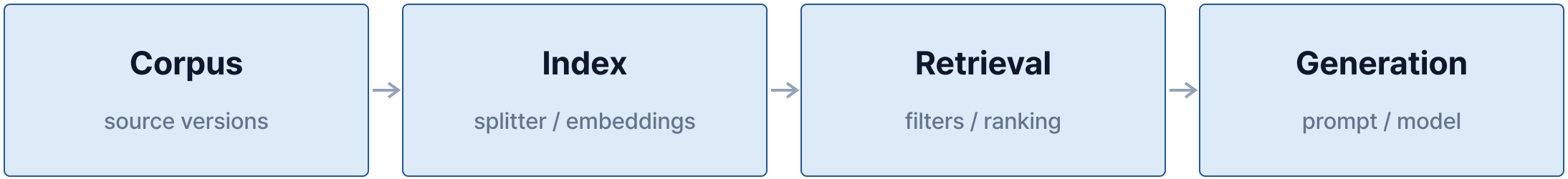


Track source identity so a deletion removes its chunks

Deleting the file left its **chunks searchable**

RAG Release Bundle

Version the corpus and index together with the answer system



Corpus → Index → Retrieval → Generation

Takeaway: Retrieve the evidence before trusting the answer

1. Index content together with identity, versions, and access metadata
2. Evaluate retrieval before changing the answer prompt
3. Return supported answers and keep the index current