Step 3 — Retrieval (Clause-Level Evidence Selection)

Objective

Transform a large set of legal document chunks into **focused, clause-specific evidence** that can be reliably consumed by an LLM for extraction.

Instead of sending entire contracts to the model, Step 3 retrieves **only the most relevant text fragments per clause family**, ensuring:

- higher precision,
- lower hallucination risk,
- full auditability (clear trace from extracted right → source text).

Inputs

- `chunks.jsonl` (output of Step 2)
  - One record per chunk
  - Includes:
    - `chunk_id`
    - `text`
    - `company_id`
    - `source_file`
    - `page_start`, `page_end`
    - `section_title`

Chunks may be nested per company and per document.

Outputs

- `retrieval.jsonl`

- Exactly **one record per clause family per company**
- Clause families (current):
  - `veto`
  - `liquidity`
  - `exit`
  - `anti_dilution`

Each record contains:

```
{
"company_id":"...",
"clause_family":"...",
"model":"bm25-okapi",
"top_k":12,
"queries":[...],
"hits":[
{
"chunk_id":"...",
"score":7.31,
"text":"...",
"source_file":"...",
"page_start":12,
"page_end":14,
"section_title":"..."
}
]
}
```

This file is the **only input** to Step 4 (LLM extraction).

Retrieval Strategy

Why retrieval is mandatory

Legal contracts are:

- long,

- heterogeneous in language,
- inconsistent across companies.

Passing full documents to an LLM leads to:

- diluted attention,
- higher hallucination risk,
- unclear provenance of extracted rights.

Retrieval enforces **evidence-first extraction**.

Algorithm Choice

BM25 (Lexical Retrieval)

The system uses **BM25 (Okapi)** instead of embeddings.

Why BM25 was chosen

- Deterministic and fully auditable
- Extremely strong on legal language (exact terms matter)
- No ML runtime dependencies (no Torch, no ONNX)
- Robust to platform constraints (Python 3.14, macOS)
- Easy to tune via vocabulary, not models

This makes Step 3 **stable, explainable, and reviewer-friendly**.

Clause-Family Query Packs

Retrieval is driven by **hand-crafted query packs**, one per clause family.

Each pack:

- is **French-first**, with English fallback
- intentionally redundant
- reflects real legal phrasing observed in documents

Example (conceptual):

- *Veto*: décisions collectives, approbation préalable, majorité qualifiée, conseil, comité

- *Liquidity*: cession de titres, agrément, préemption, ROFR, tag/drag, inaliénabilité
- *Exit*: liquidation, boni, changement de contrôle, IPO, cession de la société
- *Anti-dilution*: antidilution, relution, ajustement du prix, ratchet, BSA

Key design choice

Improving retrieval quality is done by **editing query packs**, not by changing algorithms.

Processing Logic

For each company:

1. Load all chunks
2. Build a BM25 index over chunk texts
3. For each clause family:
    - Run all associated queries
    - Aggregate scores
    - De-duplicate chunks
    - Rank by relevance
    - Select top-K (default: 12)
4. Write one consolidated record per family

If a clause family is weak or absent, it is **still emitted** with low-quality evidence — absence is a valid outcome.

Auditability Guarantees

Step 3 enforces:

- Explicit linkage: extracted rights → `chunk_id` → page numbers → source file
- Deterministic behavior (no randomness)
- Reproducible outputs given the same inputs
- Transparent failure modes (e.g. “clause not found”)

This makes the system defensible in:

- internal reviews,
- investment committees,
- legal audits.

Validation on Real Data

Validation is performed via:

- **Human inspection** of top retrieved chunks per family
- **Keyword coverage checks** (sanity heuristics)
- Page-level verification against original PDFs

Example outcome (Eurolysine):

- Liquidity clauses correctly detected
- Governance / veto language identified after query tuning
- Anti-dilution surfaced when present, absent otherwise