Step 4 — Extraction (LLM over Retrieved Evidence)

Objective

Transform *retrieved legal text fragments* into **structured, auditable investor rights**, without hallucination, using a constrained LLM call.

Step 4 converts *textual evidence → normalized legal facts*.

Inputs

- `retrieval.jsonl` (output of Step 3)
  - One line per `clause_family` (e.g. veto, liquidity, exit, anti-dilution)
  - Each line contains the top-k relevant text chunks with provenance:
    - `text`
    - `source_file`
    - `page`, `section`, `article` (when available)

Outputs

- `extraction.jsonl`
  - One line per `clause_family`
  - Each line contains a list of extracted rights (`items`)
  - Fully structured, schema-validated JSON

Each extracted item includes:

- `present` (boolean, no inference)
- `clause_type`
- `beneficiary`
- `trigger`

- `effect`
- `thresholds` (optional)
- **Evidence**:
  - verbatim quote (≤ 25 words)
  - source file
  - page number
- `confidence` (0–1)
- `flags` (`missing_evidence`, `ambiguous`, `conflicting`)

Core Design Principles

1. Evidence-only extraction (no guessing)

- The model may **only** use provided context
- If a right is not explicitly stated → `present = false`
- No synthesis across chunks unless text explicitly supports it

2. Provenance is mandatory

Every extracted right must reference:

- exact quote
- file name
- page number

  This enables legal auditability and downstream validation.

3. Strong schema enforcement

- Output is validated with **Pydantic**
- Invalid JSON or schema mismatch fails fast
- Prevents silent corruption of downstream steps

Technical Implementation

Model

- **Local LLM via Ollama**
- Default: `qwen2.5:3b-instruct` (Q4, CPU-only)
- Project-scoped model storage (fully deletable)

Context control (critical on CPU)

- Retrieved chunks concatenated with headers
- Hard cap: **`max_chars ≈ 3,000`**
- Prevents exponential latency and memory issues

Prompting strategy

- Short, instruction-only prompt
- JSON-only output (`format="json"`)
- Output length capped (`num_predict`)
- Temperature = 0 (deterministic extraction)

CLI Usage

```
lre extract \
  --company-id eurolysine \
  --retrieval path/to/retrieval.jsonl \
  --out path/to/extraction.jsonl \
  --max-chars 3000 \
  --num-ctx 2048 \
  --timeout-s 900
```

Verbose mode (recommended during development):

```
LRE_VERBOSE=1 lre extract ...
```

Observability & Robustness

To avoid “silent freezes” on CPU:

- Timestamped progress logs per `clause_family`
- Context size + hit count logged
- Per-clause try/except: one failure does not kill the pipeline
- Immediate flush after each write

Typical runtime on CPU:

- ~20–40s per clause family
- ~2–5 minutes per document