Case A – Structuring Unstructured Documents

1. Design Objective

What "good" document segmentation means

To me, "good" segmentation means that each chunk of text satisfies four goals:

1. **Semantic Coherence** – each chunk expresses one complete idea (e.g., a clause, definition, paragraph, or table).

2. **Retrievability** – the chunk is neither too large (causing noise) nor too small (losing meaning).

3. **Continuity & Composability** – chunks can be stitched together for multi-step reasoning.

4. **Traceability** – every chunk preserves provenance: page number, clause number, table region, etc.

These principles match findings from several modern RAG chunking resources:

- The *Unstructured.io* best-practices guide stresses that chunking should respect **document element boundaries** (paragraphs, headings, tables), not random slices.

- The *Milvus legal-document vectorization notes* state that **clause-level structure must be preserved**, especially for contracts.

- RAG surveys highlight that chunking quality is often a bigger factor than the LLM model choice itself.

Together, these reinforce why semantic + structural integrity is central to good segmentation.

Different chunking approaches (3–4 relevant ones)

A. Structure-Aware Chunking

Uses headings, clause numbers, section boundaries.
Example:
Section → Subsection → Clause 2.1 → Clause 2.1.1

Why relevant:
Legal/technical documents are inherently structured, and research consistently recommends respecting these natural boundaries to maintain meaning.

B. Semantic Chunking

Splits text where the topic shifts, even if the structure is weak.

Example: A single page mixing “scope”, “payment terms”, and “penalties” becomes three chunks.

Supported by: Semantic chunking methods recommended in several RAG blog surveys, because semantic coherence improves retrieval accuracy.

C. Fixed-Size Token Chunking (with overlap)

Split every ~300–800 tokens with slight overlap.

Why relevant:
Useful when structure is absent.
Literature suggests this as a **baseline** approach for unstructured or messy PDFs.

D. Object-Based Chunking (Tables, Tables Rows, Forms)

Treats objects like tables as separate chunks.
May extract table rows as sub-chunks.

Supported by: Tools like PdfTable (2024, arXiv) emphasize structured extraction of tables from PDFs for better downstream reasoning.

2. Understanding the Trade-offs

Which to prefer: smaller or larger chunks?

I prefer **medium-sized, semantically complete chunks**, rather than extreme smallness or largeness.

This choice is shaped by findings across several chunking guides:

- Small chunks → high precision but lose context.
- Large chunks → rich context but noisy retrieval.
- Many practitioners now recommend **“semantically complete but size-bounded”** chunks (e.g., a full clause capped at ~512–1024 tokens).

Thus, the optimal strategy balances semantic coherence with token budget.

Impact of the decision

A. Retrieval Accuracy

- Smaller chunks → risk missing parts of a clause.
- Larger chunks → may retrieve irrelevant material bundled with target text.
- Medium chunks → best balance of precision vs recall.

B. Reasoning Cost

- Smaller → more chunks must be recombined → more LLM work.
- Larger → fewer chunks, but each more expensive to process.

C. Continuity of Understanding

- Too small → clause meaning breaks.
- Too large → over-broad context.
- Medium → preserves necessary continuity.

Several industry articles also warn that chunk size directly shapes embedding quality and LLM context behavior.

3. Handling Real-World Complexity: Multi-Page Clauses

Can clause-specific queries be answered via FAISS similarity search alone?

Example query:
"What does clause 2.1.2 say?"

Yes, but only if chunking is done correctly.
FAISS can identify the right clause only when:

- The entire clause is stored as **one coherent chunk**, and
- The clause number or metadata connects the question to the chunk.

Otherwise, FAISS may:

- return only part of the clause,
- confuse similar clauses (common in contracts),
- or retrieve adjacent but irrelevant text.

This aligns with legal-RAG analysis, which repeatedly stresses **structure-aligned segmentation** for clause lookup.

Is this a chunking problem or a retrieval problem?

Primarily a chunking problem.

Because:

- If clause boundaries are broken, even perfect retrieval will return fragmented or wrong chunks.
- If clause boundaries are preserved, even simple retrievers perform well.

Retrieval methods can improve accuracy (cross-encoders, hybrid search), but **correct chunk boundaries are the foundation**.

Modern RAG research confirms that retrieval errors often originate from **poor chunk formation**, not poor retrievers.

4. Structure and Format Variations

How to ingest and chunk scanned documents

A realistic pipeline:

1. **OCR text extraction**
    - Extract text + bounding boxes + confidence scores.
2. **Layout detection**
    - Identify paragraphs, headings, tables, signatures, etc.
      (Modern articles show DL-based layout detection significantly improves segmentation quality.)
3. **Post-processing**
    - Merge lines into paragraphs, fix hyphenation, strip headers/footers.
4. **Chunking**
    - Apply structure-aware or semantic chunking to the cleaned text.
5. **Tables and forms**
    - Treat them as special objects.

This mirrors modern OCR + layout pipelines described in tools like PdfTable, LayoutLM-based methods, and Unstructured.io.

How to extract tables: deterministic, ML, or hybrid?

I would use a **hybrid approach**, because:

Deterministic (rule-based)

- Best for digital PDFs where table lines and structures are explicit.
- Very fast and clean.

ML-Based

- Needed for scanned documents, rotated/blurred pages, or messy layouts.
- Deep learning reliably detects table regions even when no lines exist.

Hybrid = most robust

- Try deterministic first.
- Fall back to ML when confidence drops.

This matches how modern PDF-extraction tools operate, as shown in PdfTable (2024) and several recent surveys.

5. Metadata Thinking

Metadata makes chunks more useful for downstream agents.

Structural Metadata

- Document ID
- Page number(s)
- Clause number / section heading
- Chunk type (paragraph, clause, table, annex)
- Bounding box coordinates (for scanned pages)
- Token length

Semantic Metadata

- Short 1–2 line summary
- Keywords / NER entities (dates, parties, amounts)
- Topics or labels
- OCR confidence (if scanned)

These types of metadata reflect patterns found in multiple RAG guides that emphasize **metadata-assisted retrieval**, e.g., entity-based filtering.

Agents that benefit from metadata-rich chunks

- **Clause agent** → uses clause_number metadata to answer clause-specific queries.
- **Table reasoning agent** → uses table-type chunks + structured table JSON.
- **Summarisation agent** → uses section metadata to create chapter-level summaries.
- **Compliance agent** → uses entities (dates, amounts, responsibilities).
- **Reranking agent** → uses OCR confidence, chunk type, section boundaries.

Modern agentic-RAG papers underline that metadata provides *routing signals* that determine which agent processes which chunk.