# RAG Chunking Strategy Guide

How to split text for optimal retrieval quality

## The Key Insight

**Chunking strategy = 70% of answer quality in RAG.** Not the vector database. Not the embeddings. Not the LLM. How you split text into chunks determines retrieval quality.

## Understanding RAG: The Library Analogy

Imagine a massive library with thousands of books (your documents):

**Chunks** = Individual chapters ripped out and filed separately. Search chapter-by-chapter, not whole books.

**Vectors** = A magical filing system where each chapter gets "meaning coordinates." Similar topics sit near each other—even if words differ. "How to bake bread" sits next to "Making sourdough at home."

**Semantic search** = Convert question to coordinates, walk to that spot, grab nearest chapters.

## The Core Principle

**Context preservation > arbitrary splits. Bad chunks split thoughts mid-sentence. Good chunks preserve complete ideas.**

## Use Case 1: Podcast/Video Transcripts

| | Bad Chunking | Good Chunking |
|---|---|---|
| **Strategy** | Fixed 500-word splits | Hybrid: Q&A pairs + logical sections + speaker turns |
| **Problem** | Splits Q&A mid-answer<br>Cuts speakers arbitrarily<br>Breaks topic flow | Preserves complete thoughts<br>Maintains attribution<br>Respects topic boundaries |
| **Impact** | Incomplete retrieval, lost context | Quality answers, complete thoughts retrieved |
| **Example** | Split at word 500 mid-answer | Keep "Question: How do you price? Answer: [full answer]" together |

## Use Case 2: Documentation & Knowledge Bases

| | Bad Chunking | Good Chunking |
|---|---|---|
| **Strategy** | Every 1000 words | Section-based with header preservation |
| **Problem** | Splits procedures mid-step<br>Breaks code examples<br>Separates headers from content | Headers with content<br>Complete procedures<br>Code context maintained |
| **Best For** | N/A | Company wikis, Notion, Confluence |
| **Example** | Step 3 in chunk A, Step 4 in chunk B | All steps 1-5 in one chunk with header |

## Use Case 3: Customer Support Tickets

| | Bad Chunking | Good Chunking |
|---|---|---|
| **Strategy** | Every 10 messages | Thread-based: issue + resolution together |
| **Problem** | Issue in chunk 1, resolution in chunk 2<br>Context scattered | Complete thread<br>Issue + resolution paired<br>Customer context preserved |
| **Best For** | N/A | Zendesk, Intercom, support emails |
| **Example** | Messages 1-10 chunk, 11-20 chunk | One thread = one chunk (issue to resolution) |

## Use Case 4: Research Papers & Academic Content

| | **Bad Chunking** | **Good Chunking** |
|---|---|---|
| **Strategy** | Paragraph-based splits | Section-based: methodology + results paired |
| **Problem** | Methodology separated from results<br>Abstract split from conclusions | Abstract complete<br>Method with results<br>Citations preserved |
| **Best For** | N/A | Scientific papers, whitepapers, technical reports |
| **Example** | Each paragraph = chunk | Introduction section = 1 chunk, Methods+Results = 1 chunk |

## Use Case 5: Legal Documents & Contracts

| | **Bad Chunking** | **Good Chunking** |
|---|---|---|
| **Strategy** | Page-based splits | Clause-based with cross-references |
| **Problem** | Definition on page 5, reference on page 12<br>Context scattered | One clause per chunk<br>Cross-references included<br>Definitions preserved |
| **Best For** | N/A | Contracts, terms of service, compliance docs |
| **Example** | Page 1 = chunk, Page 2 = chunk | Clause 3.2 (with sub-clauses) = 1 chunk |

## Use Case 6: Code Repositories

| | **Bad Chunking** | **Good Chunking** |
|---|---|---|
| **Strategy** | Entire files as chunks | Function/class-level with dependencies |
| **Problem** | 500-line file = one chunk<br>User finds 20 functions<br>Manual search needed | One function per chunk<br>Imports included<br>Context preserved |
| **Best For** | N/A | GitHub repos, internal codebases |
| **Example** | auth.py (500 lines) = 1 chunk | login_user() function = 1 chunk (with imports) |

## General Principles (Any Content Type)

| Principle | Implementation |
|---|---|
| Preserve complete thoughts | Don't split mid-sentence, mid-procedure, or mid-argument |
| Maintain attribution | Keep speaker/author/source with their content |
| Respect natural boundaries | Use headings, topic changes, speaker turns as split points |
| Add rich metadata | Include: source, date, author, category, keywords in each chunk |
| Test with real queries | Search expected questions, verify chunks are complete |

## Quick Decision Framework

| If Your Content Has... | Use This Strategy |
|---|---|
| Natural Q&A format | Preserve Q&A pairs together |
| Clear sections/headings | Chunk by section, keep headers with content |
| Multiple speakers/authors | Preserve speaker attribution, chunk by turns |
| Procedures/steps | Keep all steps of a process together |
| Definitions + usage | Keep definitions with their usage examples |
| High interconnection | Use overlapping chunks or add cross-references |

## Key Takeaways

1. **Context preservation > simplicity.** Arbitrary splits are easy but destroy context.
2. **Test with real queries.** Search expected questions, verify retrieved chunks are complete.
3. **Metadata is free context.** Add source, author, date, category to every chunk.
4. **Start simple, iterate.** Begin with basic strategy, add complexity only if needed.

Built by Rachitha Suresh • rachithasuresh.com • Based on learnings from building The Lenny Lens