15 RAG Chunking Techniques Every AI Engineer Should Know

Retrieval-Augmented Generation (RAG) depends *heavily* on how we chunk our data.

If you want the LLM to retrieve context that actually makes sense, you must **chunk your data thoughtfully.**

Below are 15 key chunking strategies, explained in detail, each with a *realistic example* and practical splitting.

1. Line-by-Line Chunking

What: Split at every new line.

When to use:

- Chat logs, transcripts, or data where each line is a full thought.
- Great for: Support chats, interview Q&A, messaging apps.

Example Input:

```
Alice: Hey Bob, are you free for a call at 3 PM today?

Bob: Sure, Alice. Do you want to discuss the project updates?

Alice: Yes, and we need to talk about the client meeting.

Bob: Sounds good! See you at 3.
```

Chunked Output:

```
CHUNK1: Alice: Hey Bob, are you free for a call at 3 PM today?
```

```
CHUNK2: Bob: Sure, Alice. Do you want to discuss the project updates?
```

```
CHUNK3: Alice: Yes, and we need to talk about the client meeting.
```

```
CHUNK4: Bob: Sounds good! See you at 3.
```

Why:

- Each message is a distinct context.
- Good for granular retrieval — LLM can fetch the exact Q&A pair.
- **Warning:** If lines are too short, LLM may hallucinate due to lack of context.

2. Fixed-Size Chunking

What: Break text into fixed numbers of words/characters, regardless of meaning.

When to use:

- Messy, unstructured text with no natural boundaries.
- Good for: OCR dumps, web-crawled raw text, old scanned docs.

Example Input:

```
Python is a high-level, interpreted programming language. Its simple syntax and dynamic typing make it popular for rapid application development and scripting. Python supports multiple programming paradigms, including structured, object-oriented, and functional programming. It is widely used for web development, data analysis, AI, scientific computing, and more.
```

Suppose fixed size = 20 words.

Chunked Output:

```
CHUNK1: Python is a high-level, interpreted programming language. Its simple syntax and dynamic typing make it popular for rapid application development
```

```
CHUNK2: and scripting. Python supports multiple programming paradigms, including structured, object-oriented, and functional programming. It is widely used
```

```
CHUNK3: for web development, data analysis, AI, scientific computing, and more.
```

Why:

- Forces uniformity.
- May split sentences/ideas, which can harm understanding.
- **Tip:** Use only if there's no structure, and tweak size for your LLM's token limit.

3. Sliding Window Chunking

What: Overlap chunks by a set number of words/tokens to preserve context.

When to use:

- When ideas/sentences run across chunk boundaries.
- Good for: Narrative texts, legal docs, technical writing.

Example Input:

```
Machine learning models require large datasets for training. The quality and quantity of data significantly affect model performance. Data preprocessing involves cleaning and transforming raw data into usable input.Machine learning models require large datasets for training. The quality and quantity of data significantly affect model performance. Data preprocessing involves cleaning and transforming raw data into usable input.
```

Suppose window size = 15 words, overlap = 5 words.

Chunked Output:

```
CHUNK1: Machine learning models require large datasets for training.
The quality and quantity of data
```

```
CHUNK2: quantity of data significantly affect model performance.
Data preprocessing involves cleaning and transforming
```

```
CHUNK3: transforming raw data into usable input.
```

Why:

- Maintains continuity, so retrieval doesn't lose context at boundaries.
- Chunks overlap, which means some redundancy (increases storage, but worth it for context).

4. Sentence-Based Chunking

What: Each sentence is a chunk.

When to use:

- Clean, well-edited prose.
- Good for: Articles, documentation, textbooks.

Example Input:

```
Deep learning has transformed many fields of technology. Neural networks can now outperform humans in image recognition. Training these models requires substantial computational resources.
```

Chunked Output:

```
CHUNK1: Deep learning has transformed many fields of technology.
```

```
CHUNK2: Neural networks can now outperform humans in image recognition.
```

`CHUNK3: Training these models requires substantial computational resources.`

Why:

- Each chunk is a focused idea.
- Easy for LLM to reassemble context.
- **Risk:** Some sentences might be too short or lack context, so sometimes combine 2–3 sentences per chunk.

5. Paragraph Chunking

What: Each paragraph is a chunk.

When to use:

- Well-formatted docs, blogs, essays.
- Each paragraph covers a single idea or topic.

Example Input:

```
Data science combines domain expertise, programming skills, and
knowledge of mathematics and statistics to extract meaningful
insights from data.

It's an interdisciplinary field that uses techniques from computer
science, statistics, machine learning, and data visualization to
solve complex problems.

Data scientists work with large datasets to identify trends, make
predictions, and drive strategic decisions.
```

Chunked Output:

```
CHUNK1: Data science combines domain expertise, programming skills,
and knowledge of mathematics and statistics to extract meaningful
insights from data.

CHUNK2: It’s an interdisciplinary field that uses techniques from
computer science, statistics, machine learning, and data
visualization to solve complex problems.
```

```
CHUNK3: Data scientists work with large datasets to identify trends,
make predictions, and drive strategic decisions.
```

Why:

- Preserves logical flow and context.
- Great for retrieving larger “thought blocks”.

6. Page-Based Chunking

What: Each page (in a paginated doc) becomes a chunk.

When to use:

- PDFs, books, scanned docs, legal contracts.
- Needed if referencing by page number.

Example Input:

Page 1:

Section 1: Introduction to RAG

Retrieval-Augmented Generation (RAG) systems combine LLMs with information retrieval. RAG improves factual accuracy and extends the model’s knowledge beyond its training data.

Page 2:

Section 2: Architecture

The main components are the retriever, which fetches relevant documents, and the generator, which synthesizes an answer using the retrieved context.

Chunked Output:

CHUNK1 (Page 1): Section 1: Introduction to RAG

Retrieval-Augmented Generation (RAG) systems combine LLMs with information retrieval. RAG improves factual accuracy and extends the model’s knowledge beyond its training data.

CHUNK2 (Page 2): Section 2: Architecture

`The main components are the retriever, which fetches relevant documents, and the generator, which synthesizes an answer using the retrieved context.`

Why:

- Essential when page structure matters (e.g., legal evidence, contracts, textbooks).

7. Section or Heading-Based Chunking

What: Split at headings/sections (H1/H2/etc., or "## Section Title").

When to use:

- Documents with clear, logical sections.
- Technical documentation, books, whitepapers.

Example Input:

```
# Introduction

Retrieval-Augmented Generation (RAG) allows language models to use external information to improve answers.

# How RAG Works

RAG first retrieves relevant documents, then generates responses based on both the user query and the context.

# Benefits

RAG improves factual accuracy and lets you use private or updated data.
```

Chunked Output:

```
CHUNK1: # Introduction

Retrieval-Augmented Generation (RAG) allows language models to use external information to improve answers.
```

```
CHUNK2: # How RAG Works
```

```
RAG first retrieves relevant documents, then generates responses
based on both the user query and the context.
```

```
CHUNK3: # Benefits
```

```
RAG improves factual accuracy and lets you use private or updated
data.
```

Why:

- Chunks match natural topic boundaries — improves retrieval accuracy.
- Users get entire topics/sections on retrieval.

8. Keyword-Based Chunking

What: Split whenever a keyword appears (e.g., “Step”, “Diagnosis”, “Note”).

When to use:

- Forms, logs, technical instructions with recurring keywords.
- Example: Medical records, step-by-step guides.

Example Input:

```
Diagnosis: Acute bronchitis.

Symptoms: Persistent cough, mild fever, chest discomfort.

Prescription: Amoxicillin 500mg three times daily for 7 days.

Note: Advise patient to rest and hydrate.
```

Keyword: “Note:”

Chunked Output:

```
CHUNK1: Diagnosis: Acute bronchitis.

Symptoms: Persistent cough, mild fever, chest discomfort.

Prescription: Amoxicillin 500mg three times daily for 7 days.

CHUNK2: Note: Advise patient to rest and hydrate.
```

Why:

- Keeps related info together (everything before “Note:” is one chunk).
- Perfect for structured records.

9. Entity-Based Chunking

What: Use Named Entity Recognition (NER) to group sentences/paragraphs by entity (person, organization, product, etc.).

When to use:

- News, legal docs, product reviews — where references to entities matter.

Example Input:

```
Apple announced a new iPhone model at their annual event. Tim Cook presented several new features focused on camera improvements and battery life. Meanwhile, Samsung is rumored to launch a competing device next month.
```

NER picks: “Apple”, “Tim Cook”, “Samsung”

Chunked Output:

```
CHUNK1: Apple announced a new iPhone model at their annual event. Tim Cook presented several new features focused on camera improvements and battery life.
```

```
CHUNK2: Meanwhile, Samsung is rumored to launch a competing device next month.
```

Why:

- Enables entity-focused retrieval — LLM can answer, “What did Apple announce?” by fetching all “Apple” chunks.

10. Token-Based Chunking

What: Split by number of tokens (model’s processing units), not just words.

When to use:

- When LLM context size is limited (e.g., 1024, 2048 tokens).

Example Input:

```
The rapid growth of generative AI has created a surge in
applications for chatbots, document summarization, and data
extraction. As models get larger, they require more memory and
computation, but also open up new possibilities for automation
across industries. Organizations are exploring hybrid systems that
combine classic algorithms with large language models for improved
performance and cost efficiency.
```

Suppose each chunk = 25 tokens

(Simulate: Each ~10 words = 10 tokens. We'll split by sentence to avoid breaking sentences.)

Chunked Output:

```
CHUNK1: The rapid growth of generative AI has created a surge in
applications for chatbots, document summarization, and data
extraction.
```

```
CHUNK2: As models get larger, they require more memory and
computation, but also open up new possibilities for automation
across industries.
```

```
CHUNK3: Organizations are exploring hybrid systems that combine
classic algorithms with large language models for improved
performance and cost efficiency.
```

Why:

- Controls model input size, avoids cutoff errors.
- Works well for API-driven apps.

11. Table Chunking

What: Extract each table as a separate chunk (optionally row-by-row or as a whole).

When to use:

- Invoices, financial reports, scientific papers — any doc with tables.

Example Input:

```
Table 1: Quarterly Revenue

| Quarter | Revenue (USD) |

|---------|---------------|

| Q1 2024 | $1,000,000    |

| Q2 2024 | $1,200,000    |

The company experienced steady growth, with a noticeable increase in Q2.
```

Chunked Output:

```
CHUNK1: Table 1: Quarterly Revenue

| Quarter | Revenue (USD) |

|---------|---------------|
```

```
| Q1 2024 | $1,000,000    |
```

```
| Q2 2024 | $1,200,000    |
```

```
CHUNK2: The company experienced steady growth, with a noticeable
increase in Q2.
```

Why:

- Tables can be parsed as structured data.
- Retrieval can answer “What was the revenue in Q2 2024?” by fetching the table chunk.

12. Recursive Chunking

What: Start big (paragraphs or sections), then split oversized chunks further (by sentence, then words), until every chunk fits the size you want.

When to use:

- Long, rambling transcripts, interviews, or documents with unevenly sized paragraphs.

Example Input:

```
Interview transcript:

John: So, in the beginning, we focused mostly on user experience. We ran several surveys, collected feedback, and iterated quickly.

Later, as the product matured, we started addressing scalability and infrastructure. This phase was more challenging because we needed to maintain uptime while scaling.
```

Suppose chunk size = max 20 words.

Step 1: Split by paragraph

- Paragraph 1: “John: So, in the beginning, we focused mostly on user experience. We ran several surveys, collected feedback, and iterated quickly.”
- Paragraph 2: “Later, as the product matured, we started addressing scalability and infrastructure. This phase was more challenging because we needed to maintain uptime while scaling.”

Step 2: Paragraphs still too big → split by sentences

Chunked Output:

```
CHUNK1: John: So, in the beginning, we focused mostly on user
experience.
```

```
CHUNK2: We ran several surveys, collected feedback, and iterated
quickly.
```

```
CHUNK3: Later, as the product matured, we started addressing
scalability and infrastructure.
```

```
CHUNK4: This phase was more challenging because we needed to
maintain uptime while scaling.
```

Why:

- Guarantees no chunk is too big for your system.

13. Semantic Chunking

What: Use embeddings/AI to group together sentences/paragraphs that discuss the same topic.

When to use:

- Mixed-topic data (e.g., support tickets, Q&A docs, FAQ).

Example Input:

```
Q: How do I reset my password?

A: Go to the login page and click 'Forgot Password'.

Q: How can I change my email address?

A: Visit your profile settings and enter your new email.

Q: What is the refund policy?

A: Refunds are available within 30 days of purchase.
```

Suppose semantic model detects "Account management" and "Payments" topics.

Chunked Output:

```
CHUNK1: Q: How do I reset my password?
```

```
A: Go to the login page and click 'Forgot Password'.

Q: How can I change my email address?

A: Visit your profile settings and enter your new email.
```

```
CHUNK2: Q: What is the refund policy?

A: Refunds are available within 30 days of purchase.
```

Why:

- Great for retrieving all related answers to a user's intent.
- Reduces context-missing/hallucination in retrieval.

14. Hierarchical Chunking

What: Multi-level chunking — split by chapters, then sections, then paragraphs, etc.

When to use:

- Large, well-structured texts (books, technical docs, legal codes).

Example Input:

```
Chapter 1: Introduction
```

```
Section 1.1: What is RAG?
```

```
Retrieval-Augmented Generation (RAG) combines LLMs with external data sources to provide up-to-date answers.
```

```
Section 1.2: Why use RAG?
```

```
RAG extends model capabilities, enhances factual accuracy, and supports private or dynamic information.
```

Chunked Output:

```
CHUNK1: Chapter 1: Introduction
```

```
CHUNK2: Section 1.1: What is RAG?
```

```
Retrieval-Augmented Generation (RAG) combines LLMs with external
data sources to provide up-to-date answers.
```

```
CHUNK3: Section 1.2: Why use RAG?
```

```
RAG extends model capabilities, enhances factual accuracy, and
supports private or dynamic information.
```

Why:

- Lets your RAG system retrieve broad (chapter) or detailed (section) information.

15. Content-Type Aware Chunking

What: Use different chunking for tables, lists, images, and plain text.

When to use:

- Docs with mixed content: PDFs, research papers, reports.

Example Input:

```
Abstract:

This research investigates chunking strategies for RAG pipelines.
Results show that chunking method impacts answer quality.

Table 1: Test Results
```

| Method | Accuracy |

|---|---|

| Sentence-based | 85% |

| Sliding window | 90% |

Figure 1: Pipeline diagram (image not shown here)

Chunked Output:

CHUNK1: Abstract:

This research investigates chunking strategies for RAG pipelines. Results show that chunking method impacts answer quality.

CHUNK2: Table 1: Test Results

```
| Method          | Accuracy |

|-----------------|----------|

| Sentence-based | 85%      |

| Sliding window | 90%      |
```

```
CHUNK3: Figure 1: Pipeline diagram (image not shown here)
```

Why:

- Ensures retrieval doesn’t mix up tables, text, and images.
- Enables targeted retrieval: “Show me the results table,” or “Fetch the abstract.”

Conclusion

Key Takeaways:

- **No single chunking strategy fits all data.**
- Match your chunking method to your document's format, use case, and the questions users will ask.
- Test on your real data — *and always check the LLM's output for context drift and hallucination.*