Chunking Strategies for RAG applications

Compiled by D.Gueorguiev 8/13/25

# What is Chunking?

Chunking strategies in Retrieval-Augmented Generation (RAG) involve dividing documents into smaller, manageable pieces to improve retrieval accuracy and efficiency. Different methods exist, each with its own strengths and weaknesses, and the optimal approach depends on the specific data and application. Common strategies include fixed-size chunking, recursive chunking, semantic chunking, and document-based chunking.

The video [1] explains the basics of RAG and the importance of chunking.

A diagram of a user and a user

AI-generated content may be incorrect.

Here's a breakdown of common chunking strategies:

1. **Fixed-Size Chunking**:

Concept: splits text into chunks of a predetermined size (e.g., number of characters or tokens). Advantages: simple to implement and understand.

Disadvantages: can split sentences or paragraphs, potentially losing context, and may not align with semantic boundaries.

Considerations: can be improved with overlap to maintain context between chunks.

Example: using langchain\_text\_splitters.CharacterTextSplitter with a specified chunk\_size and chunk\_overlap.

1. **Recursive Chunking**:

Concept: attempts to split text at natural boundaries like paragraphs, then sentences, then words, if necessary, to preserve context.

Advantages: more likely to maintain semantic meaning within chunks than fixed-size chunking. Disadvantages: more complex to implement than fixed-size chunking.

Example: starting with paragraph breaks, then trying sentence breaks, and finally word breaks.

1. Semantic Chunking:

Concept: Splits text based on semantic similarity of embeddings, aiming for concise chunks that represent a single, coherent idea.

Advantages: Optimized for retrieval accuracy by ensuring each chunk has a clear meaning.

Disadvantages: Requires embedding calculations and can be more computationally expensive.

Considerations: May lead to smaller chunks than other methods.

Example: Grouping sentences based on the similarity of their embeddings.

4. Document-Based Chunking:

Concept:

Splits text based on document structure, such as sections, headings, or specific tags (e.g., in HTML or Markdown).

Advantages:

Preserves the original structure of the document, which can be helpful for understanding the context.

Disadvantages:

Requires knowledge of the document structure and might not be applicable to all document types.

Example:

Splitting a document into sections based on headings or splitting a table into chunks based on rows.

5. Agentic Chunking:

Concept:

Utilizes an LLM to determine the optimal chunking strategy, potentially considering semantic meaning, content structure, and document type.

Advantages:

Can adapt to different document types and potentially achieve the best results.

Disadvantages:

Relies on the LLM's ability to understand the data and may be more computationally expensive.

# References

[1] [Chunking Strategies in RAG: Optimizing Data for Advanced AI Responses, Mervin Praison, 08/2024 (youtube video)](https://youtu.be/pIGRwMjhMaQ?si=a5q02PhyfbXp1CPl)

[2] [Semantic Chunking in RAG, James Briggs, 08/2024 (youtube video)](https://www.youtube.com/watch?v=TcRRfcbsApw)

[3] [The 5 Levels Of Text Splitting For Retrieval, Greg Kamradt, 08/2024 (youtube video)](https://youtu.be/8OJC21T2SL4?si=e_rut78-goKkONb9)