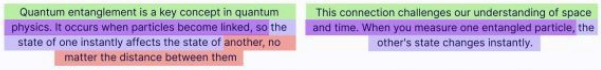
Chunking Techniques

for retrieval-intensive applications

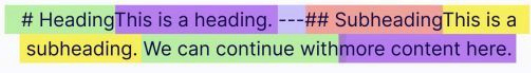
Fixed-Size Chunking

This technique **splits** the text into chunks of a **fixed size**, without considering natural breaks or the structure of the content. It's **simple** and **cost-effective**, but lacks **contextual awareness**. To improve this, **overlapping chunks** can be used, allowing adjacent chunks to share some content.



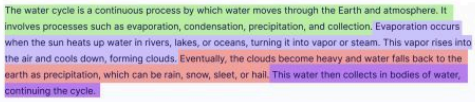
Recursive Chunking

Text is initially split using a **primary separator,** like paragraphs. If the resulting chunks are too large, **secondary separators**, like sentences, are applied recursively until the desired chunk size is achieved. This technique respects the **document's structure** and is **flexible** for various use cases.



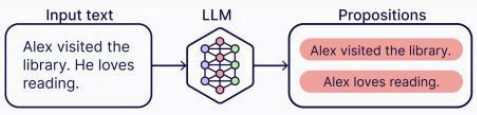
Document-Based Chunking

This technique creates chunks based on the **natural divisions** within the document, such as headings or sections. It's very effective for structured data like **HTML**, **Markdown**, or **code files** but it's less useful when the data lacks clear structural elements.



Semantic Chunking

In this technique, the text is divided into **meaningful units**, such as sentences or paragraphs, which are then **vectorized**. These units are then combined into chunks based on the **cosine distance** between their embeddings, with a new chunk formed whenever a significant **context shift** is detected.

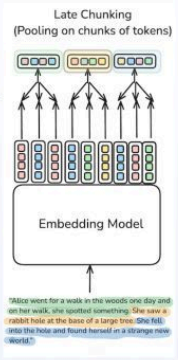


LLM-Based Chunking

This advanced technique uses a **Language Model (LLM)** to generate chunks. The LLM processes the text and generates **semantically isolated** sentences or **propositions** that can stand alone. While this method is **highly accurate**, it is also the most **computationally demanding**.

Late Chunking

Late chunking inverts traditional document processing by embedding first, then chunking. While traditional methods chunk text before embedding each piece separately, late chunking embeds the entire document first using a long context model. The resulting token-level embeddings are then divided and pooled into multiple chunk representations. This approach better preserves contextual information across large documents.



weaviate.io/ebooks/advanced-rag-techniques