Text Splitting

Definition: Text Splitting is the process of breaking large bodies of text (such as articles, PDFs, HTML pages, or books) into smaller, manageable pieces called **chunks**. This process enables Large Language Models (LLMs) to process and analyze lengthy content efficiently and accurately.

Why Text Splitting is Important

1. Overcoming Model Limitations

Many LLMs and embedding models have a fixed **maximum input token size**. If the text exceeds this limit, it cannot be processed directly. **Solution:** By splitting text into smaller chunks, we ensure that large documents fit within model constraints while preserving contextual integrity.

2. Enhancing Downstream Tasks

Almost every LLM-powered task — including *question answering*, *summarization*, *semantic search*, and *document retrieval* — benefits from well-structured text chunks. Smaller chunks improve retrieval accuracy and reduce hallucinations.

3. Optimizing Computational Resources

Working with smaller pieces of text:

- Reduces memory usage.
- Enables better **parallelization** of tasks.
- Improves speed and efficiency for both training and inference.

Summary

Large Text ⇒ **Chunks** ⇒ Efficient LLM Processing

Types of Text Splitting Techniques

1. Length-Based Text Splitting

- Splits text purely based on **character count** or **token length**.
- Ensures each chunk fits within a model's token limit.
- Simple and fast, but may break context mid-sentence.

Example: Split every 500 tokens.

Pros: Easy to implement and deterministic.
Cons: May ignore sentence or paragraph boundaries.

2. Text-Structured Based Splitting

- Splits content using structural cues like **paragraphs**, **sentences**, or **headings**.
- Maintains logical flow while keeping chunks within a token threshold.

Example: Split at each paragraph or "." while maintaining 1000-character limit.

Pros: More natural boundaries and context preservation.
Cons: Uneven chunk sizes depending on document structure.

3. Document-Structured Based Splitting

- Used for well-defined formats like **PDFs**, **HTML**, or **Markdown**.
- Splits based on document elements (e.g., sections, tables, headers, bullet lists).

Example: Split by section headers or HTML tags like `<h1>`, `<p>`.

Pros: Preserves document hierarchy.
Cons: Requires parsing logic specific to file type.

4. Semantic Meaning-Based Splitting

- Uses embeddings or similarity metrics to split text based on **semantic coherence**.
- Ensures each chunk represents a meaningful unit of thought.

Example: Split when cosine similarity between adjacent sentences drops below a threshold.

Pros: Best preserves context and meaning.
Cons: Computationally expensive.

Conclusion

Text Splitting is a foundational step in building effective LLM-powered pipelines such as Retrieval-Augmented Generation (RAG), Document Question Answering, and Summarization. A well-chosen splitting strategy balances:

- **Chunk size vs. context retention**
- **Computational efficiency vs. semantic integrity**