Text Splitters

1. What is Text Splitting?

- **Definition:**
  Text Splitting is the process of breaking large chunks of text (articles, PDFs, HTML pages, books) into smaller, manageable pieces (chunks).
  → These chunks can then be processed effectively by **LLMs (Large Language Models)**.
- **Why it's needed:**
  - LLMs have **maximum input size constraints** (e.g., 50K tokens).
  - Splitting prevents exceeding these limits.
  - Smaller chunks improve accuracy, reduce hallucinations, and make downstream tasks more efficient.

---

2. Benefits of Text Splitting

- **Overcoming model limitations**:
  - Works around input length constraints.
  - Allows handling of large documents.
- **Downstream task improvements**:

Task	Why Splitting Helps
Embedding	Short chunks give more accurate vectors.
Semantic Search	Focused info, less noise.
Summarization	Prevents hallucination and topic drift.

- **Optimizing computational resources**:
  - Small chunks are **more memory-efficient**.
  - Easier to **parallelize** processing.

---

3. Types of Text Splitters

(a) Length-Based Splitters

- Split by **fixed size** (characters, words, or tokens).
- Example: Split every 100 characters.
- **Chunk Overlap**: Adding overlap ensures continuity and prevents loss of context.
    - Example: If chunk size = 100, overlap = 20 →
        - Chunk 1: characters 1–100
        - Chunk 2: characters 80–180
- Useful for embeddings & LLM training.

(b) Text-Structure-Based Splitters

- Splits based on **structural elements**:
    - Paragraphs (\n\n)
    - Lines (\n)
    - Sentences (e.g., . or !)
    - Words / characters
- **Example Input:**
- My name is Jawad
- I am 27 years old
- I live in Dhahran
- **Split by line →**
    - ["My name is Jawad", "I am 27 years old", "I live in Dhahran"]

(c) Document-Structure-Based Splitters

- Useful for structured docs like **Markdown, JSON, code**.
- Splits by **headings, sections, or code blocks**.
- **Markdown Example:**

- ## Features
- - Add new students
- - View details
- ## Tech Stack
- - Python 3.10

Splits into:

  - Features section
  - Tech Stack section

- **Code Example:**
  Split along **class** or **def** keywords:
- class Student:
-     def __init__(...):
-         ...
- def is_passing(...):
-     ...

Ensures logical chunks (functions/classes) remain intact.

(d) Semantic Meaning-Based Splitters

- Splits text by **meaningful semantic units** rather than just size/structure.
- Requires **NLP techniques** (e.g., sentence embeddings, similarity).
- **Example Input:**
- Farmers were working hard...
- The Indian Premier League is the biggest cricket league...
- Splits into **topic-based chunks**:
  - Agriculture/season context.
  - Cricket/entertainment context.

→ Prevents unrelated topics being grouped in one chunk.

---

4. Chunk Overlap (Image 5)

- Ensures **context preservation** across chunks.
- **Without overlap**: Info may be cut in half.
- **With overlap**: Smooth flow between chunks.
- Example:
  Text = "Space exploration has led to..."
  - Chunk size = 50 chars, Overlap = 10.
  - First chunk ends with "has led to"
  - Next chunk starts from "to incredible scientific..."

---

5. Document Splitter Patterns (Image 7 & 8)

- **Markdown-based splitting**:
  - Split on headings (##, ###).
  - Split on horizontal lines (---).
  - Split on code blocks (` ` `).
- **Code-based splitting**:
  - Look for class, def, or indentation.
  - Ensures **logical grouping** of code instead of arbitrary cuts.

📊 Comparison of Text Splitters

Splitter Type	How it Works	Pros	Cons	Best Use Cases
Length-Based	Splits by fixed length	- Simple to implement	- May cut sentences/paragraphs	- Embeddings - When size

Splitter Type	How it Works	Pros	Cons	Best Use Cases
	(characters, words, tokens)	- Predictable chunk size - Works with any text	awkwardly - Risk of losing context without overlap	control is crucial (e.g., LLM input windows)
Text-Structure-Based	Splits using natural structure (paragraphs, lines, sentences, words)	- Human-readable chunks - Preserves natural flow of text	- Chunks may vary widely in size - Not suitable for size-constrained tasks	- Chat logs - Documents with natural paragraph breaks
Document-Structure-Based	Splits based on document formatting (Markdown headings, code blocks, sections)	- Keeps logical sections intact - Works well for structured docs (reports, code, markdown)	- Depends on consistent formatting - Harder for unstructured text	- Technical docs - Codebases - Research reports
Semantic Meaning-Based	Uses NLP/embeddings to group text by meaning or topic	- Most context-aware - Prevents mixing unrelated topics - Produces coherent chunks	- Computationally expensive - Requires semantic models	- Summarization - Semantic search - Knowledge retrieval

✅ **Summary**

- Text splitting = breaking large text into chunks.
- Helps LLMs deal with size limits, improves embeddings, search, and summarization.
- **Types of splitters**:

1. Length-based → fixed size chunks (with overlap).
2. Text-structure-based → split by lines, paragraphs, words.
3. Document-structure-based → split by headings, code blocks.
4. Semantic-meaning-based → split by topic/context.