



## **Agentic Ai Lab**

**Nirob Paul (2023000168)**

**Section – F**

**Group – 1**

# Working of 5 Levels of Text Splitting

## Introduction

Text splitting (also called chunking) is the process of breaking large text into smaller, manageable pieces so that language models can process, store, and retrieve information effectively. It is a **core concept in RAG (Retrieval-Augmented Generation), search systems, agents, and long-context applications.**

This document explains **how each level of text splitting works**, why it exists, and **when to use it**, without repeating any code.

---

## Level 1: Character Splitting

### How it Works

The text is divided purely based on a **fixed number of characters**. The splitter does not understand sentences, words, or meaning. It simply counts characters and cuts the text accordingly.

Optionally, a small overlap is kept between chunks to avoid losing information at boundaries.

### Key Idea

- Splitting is mechanical
- No understanding of language structure

### Strengths

- Very easy to implement
- Fast and predictable

### Weaknesses

- Breaks sentences and ideas
- Loses semantic meaning
- Poor retrieval quality

### When to Use

- Learning chunking concepts
- Very rough preprocessing

- Never recommended for production RAG systems
- 

## Level 2: Recursive Character Splitting

### How it Works

Instead of blindly cutting text, recursive splitting tries **multiple separators in order**:

1. Paragraph breaks
2. Line breaks
3. Spaces
4. Characters (last resort)

The splitter attempts to keep chunks within size limits **while respecting text structure as much as possible**.

### Key Idea

- Structure-aware splitting
- Falls back gracefully if ideal splits are not possible

### Strengths

- Preserves sentences and paragraphs
- Much better context retention
- Works well for most text types

### Weaknesses

- Still not meaning-aware
- Cannot understand topic shifts

### When to Use

- Default choice for most RAG applications
  - Blogs, articles, documentation, books
- 

## Level 3: Document-Specific Splitting

### How it Works

This level uses **custom splitters depending on document type**, such as:

- Python code split by functions and classes
- Markdown split by headers
- PDFs split by pages or sections

Each splitter understands the **structure rules of that document format**.

### **Key Idea**

- One document type  $\neq$  another
- Structure matters more than size

### **Strengths**

- Highly accurate chunk boundaries
- Preserves logical sections
- Excellent for code and technical docs

### **Weaknesses**

- Requires correct document detection
- More setup than generic splitters

### **When to Use**

- Codebases
- Markdown documentation
- PDFs, research papers, manuals

---

## **Level 4: Semantic Splitting**

### **How it Works**

Instead of using characters or separators, semantic splitting uses **embeddings**:

- Each sentence is converted into a vector
- Similar sentences are grouped together
- Topic changes trigger new chunks

This ensures each chunk contains **one coherent idea**.

## **Key Idea**

- Meaning over structure
- Similarity-based grouping

## **Strengths**

- Best contextual coherence
- Excellent retrieval accuracy
- Ideal for complex knowledge bases

## **Weaknesses**

- Computationally expensive
- Slower preprocessing
- Requires embedding models

## **When to Use**

- High-quality RAG systems
- Knowledge assistants
- Enterprise search

---

## **Level 5: Agentic Splitting**

### **How it Works**

An agent (rule-based or LLM-powered) decides **how and where to split text** based on intent, goals, or reasoning.

The agent may:

- Detect topic changes
- Identify important entities
- Decide chunk boundaries dynamically

### **Key Idea**

- Splitting is a decision-making process
- Context-aware and goal-driven

### **Strengths**

- Extremely flexible
- Adapts to task requirements
- Best for autonomous systems

**Weaknesses**

- Experimental
- Hard to debug
- Expensive and complex

**When to Use**

- AI agents
- Autonomous research systems
- Advanced multi-step RAG pipelines

---

**Comparison Summary**

	Level	Awareness	Accuracy	Cost	Typical Use
1	None	Very Low	Very Low	Very Low	Learning only
2	Structure	Good	Low		Default RAG
3	Format	High	Medium		Code, PDFs
4	Meaning	Very High	High		Production RAG
5	Reasoning	Highest	Very High		AI Agents

---

**Final Takeaway**

- **Start with Level 2** for most applications
- **Upgrade to Level 3** when document structure matters
- **Use Level 4** when accuracy is critical
- **Level 5** is for future-ready, agentic systems

Text splitting is not about breaking text — it is about **preserving meaning while respecting model limits.**

