BACHELOR OF TECHNOLOGY

In

Computer Science and Engineering

Submitted by:

Kusuma M

(2023443738) CSE-F G2

Department of Computer Science and Engineering

School of Computing Science & Engineering

Sharda University, Greater Noida

Jan-June 2026

Overview

This document explains the **working of the notebook 5_Levels_Of-Text_Splitting.ipynb** for documentation and conceptual understanding. The notebook demonstrates **different levels and strategies of text splitting**, which is a crucial preprocessing step in NLP applications such as search engines, RAG (Retrieval-Augmented Generation), chatbots, and document analysis.

The explanation focuses on *what each level does, why it exists, and where it is used*.

What is Text Splitting?

Text splitting is the process of **breaking large text documents into smaller, manageable chunks**.

Why text splitting is needed:

- Large documents exceed model context limits
- Smaller chunks improve retrieval accuracy
- Helps preserve semantic meaning
- Reduces noise in embeddings and vector databases

Core idea:

"Models understand better when text is fed in meaningful pieces rather than as one huge block."

Level 1: Character-Based Text Splitting

What happens here:

- Text is split based on a fixed number of characters.
- Each chunk contains a specific character length.

Why this level exists:

- Simplest and fastest splitting method
- Guarantees chunk size control

Limitations:

- May cut sentences or words in the middle
- Does not preserve meaning

Use cases:

- When structure doesn't matter
- Early experiments or raw processing

Level 2: Recursive Character Text Splitting

What happens here:

- Text is split hierarchically using separators like:
  - Paragraphs
  - Sentences
  - Words
  - Characters (as last fallback)

Why this level exists:

- Attempts to preserve semantic boundaries
- Prevents abrupt text cuts

How it works conceptually:

1. Try splitting by paragraphs
2. If chunk is too large → split by sentences
3. If still too large → split by words
4. Finally → split by characters

Use cases:

- Most commonly used splitter in LangChain
- Ideal for RAG pipelines

Level 3: Token-Based Text Splitting

What happens here:

- Text is split based on **tokens**, not characters
- Tokens correspond to model-specific tokenization

Why this level exists:

- Models have token limits, not character limits
- Prevents runtime errors due to token overflow

Important insight:

- 1 token ≠ 1 word
- Token size varies by language and tokenizer

Use cases:

- LLM applications
- Chatbots and summarization systems

Level 4: Semantic Text Splitting

What happens here:

- Text is split based on **semantic similarity**
- Embeddings are used to group related sentences together

Why this level exists:

- Maintains meaning across chunks
- Avoids splitting logically connected ideas

How it works conceptually:

- Generate embeddings
- Measure similarity
- Split where topic shifts occur

Use cases:

- High-quality document search
- Knowledge-base construction
- Advanced RAG systems

Level 5: Document-Aware / Structure-Based Splitting

What happens here:

- Splitting uses document structure such as:
    - Headings
    - Sections
    - Bullet points
    - Tables

Why this level exists:

- Documents already contain semantic hints
- Structure-aware splitting is closest to human understanding

Use cases:

- PDFs, research papers, legal documents
- Technical documentation
- Academic content ingestion

Overall Workflow Summary

1. Load raw text

2. Choose splitting strategy based on use case
3. Generate text chunks
4. Use chunks for embeddings, retrieval, or model input

Key Takeaway

This notebook shows that **text splitting is not just technical preprocessing** — it directly impacts:

- Model accuracy
- Retrieval quality
- Context understanding

Choosing the right level depends on:

- Document type
- Application goal
- Model constraints

Proper text splitting is the foundation of effective NLP and GenAI systems.