Agentic AI Lab

B. TECH 3rd YEAR

SEMESTER: 6TH

SESSION:2026-27

SUBMITTED BY: **AYUSHI TIWARI (2023334003)**

SECTION H AND GROUP: G2

SUBMITTED TO **Mr. AYUSH KUMAR SINGH**

FACULTY DESIGNATION

ASSISITANT PROFESSOR @ SHARDA UNIVERSITY

NOIDA, UTTAR PRADESH, INDIA

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING SHARDA

SCHOOL OF COMPUTER SCIENCE & ENGINEERING SHARDA

UNIVERSITY, GREATER NOIDA

5 Levels of Text Splitting

This document explains everything that is happening step by step in the notebook 5_Levels_Of-Text_Splitting.ipynb. The notebook demonstrates different strategies (levels) of text splitting, which are essential in NLP, Information Retrieval, and LLM-based applications such as RAG (Retrieval-Augmented Generation).

1. What Is Text Splitting and Why It Matters

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

Why text splitting is important:

- LLMs have **context length limits**
- Improves **retrieval accuracy** in vector databases
- Helps preserve **semantic meaning**
- Reduces noise and irrelevant context

This notebook shows **5 increasing levels of sophistication** in text splitting.

2. Level 1: Character-Based Text Splitting

What happens:

- Text is split purely based on **fixed character count**
- No understanding of words, sentences, or meaning

Example logic:

- Split every *N* characters (e.g., 1000 chars)

Advantages:

- Very simple
- Fast to compute

Limitations:

- Can cut sentences or words in the middle
- Loses semantic structure

Use case:

- Quick preprocessing when structure does not matter

3. Level 2: Recursive Character Text Splitting

What happens:

- Text is split using **hierarchical separators**
- Typical order:
    1. Paragraphs (\n\n)
    2. Lines (\n)
    3. Sentences (.)
    4. Characters

Key idea:

- Try **larger semantic units first**, fall back only if chunk is too large

Advantages:

- Better structure retention than Level 1
- Avoids unnecessary sentence breaks

Limitations:

- Still rule-based
- No real semantic understanding

Use case:

- Most commonly used splitter in RAG pipelines

4. Level 3: Token-Based Text Splitting

What happens:

- Text is split based on **token count**, not characters
- Tokens correspond to how models (GPT, BERT, etc.) read text

Why this matters:

- LLM limits are measured in **tokens**, not characters
- Prevents exceeding model context windows

Advantages:

- Model-aligned splitting
- Predictable context size

Limitations:

- Tokenizers vary by model
- Less intuitive for humans

Use case:

- Production LLM systems
- OpenAI / Hugging Face based pipelines

5. Level 4: Semantic Text Splitting

What happens:

- Text is split based on **meaning and topic shifts**
- Often uses:
    - Embeddings
    - Similarity scores

Key idea:

- Keep semantically related sentences together
- Split when topic changes

Advantages:

- High-quality chunks
- Better retrieval relevance

Limitations:

- Computationally expensive
- Requires embedding models

Use case:

- High-accuracy RAG systems
- Research and enterprise search

6. Level 5: Agentic / LLM-Based Text Splitting

What happens:

- An **LLM decides** how to split the text
- Uses instructions like:
    - "Split this document into coherent sections"

Advantages:

- Best semantic coherence
- Understands discourse, headings, intent

Limitations:

- Costly
- Slower
- Less deterministic

Use case:

- Complex documents (legal, medical, research papers)

7. Chunk Size and Overlap

Chunk Size:

- Number of characters/tokens per chunk

Overlap:

- Repeats some text between chunks

Why overlap is used:

- Prevents loss of context at chunk boundaries
- Improves answer continuity

8. Comparison Summary

Level	**Method**	**Semantic Quality**	**Cost**	**Common Use**
1	Character	Very Low	Very Low	Quick split
2	Recursive	Medium	Low	RAG default
3	Token	Medium	Medium	LLM-safe chunks
4	Semantic	High	High	Accurate retrieval
5	LLM-based	Very High	Very High	Complex docs

9. How This Fits Into RAG Pipelines

Typical RAG flow:

1. Load documents
2. Split text (this notebook)

3. Generate embeddings
4. Store in vector database
5. Retrieve relevant chunks
6. Generate answers with LLM

Text splitting directly impacts **answer quality**.

10. Final Takeaway

This notebook teaches that:

- Text splitting is **not one-size-fits-all**
- Better splitting → better retrieval → better LLM answers
- Choice depends on **accuracy, cost, and scale**

Understanding these 5 levels is essential for **modern NLP and GenAI systems**.

Outcome After This Notebook

After completing this notebook, you will:

- Understand multiple text splitting strategies
- Know when to use which splitter
- Be able to design efficient RAG pipelines