

AGENTIC AI LAB

(CSCR3215)

BACHELOR OF TECHNOLOGY

In

Computer Science and Engineering

Submitted by: *Raj Kumar Shah (2023464858) CSE-I G1*

Submitted to: *Mr. Ayush Kumar Singh Assistant Professor*



Department of Computer Science and Engineering School of
Computing Science & Engineering Sharda University, Greater Noida

Jan-June 2026

REPORT ON TEXT SPLITTING TECHNIQUES

Based on the Jupyter Notebook: *5 Levels of Text Splitting*

Abstract

This report presents a detailed study of text splitting techniques used in Natural Language Processing (NLP) and Large Language Model (LLM) applications. Text splitting is a crucial preprocessing step for handling large documents, especially in Retrieval-Augmented Generation (RAG) systems. The report explains five levels of text splitting, ranging from simple character-based methods to advanced agentic chunking using large language models.

Introduction

Modern language models have token limitations, making it difficult to process large documents directly. To overcome this challenge, text is divided into smaller, manageable chunks. This process is known as text splitting or chunking. Efficient text splitting improves retrieval accuracy, memory handling, and response quality in AI systems.

Level 1: Character-Based Text Splitting

Character-based splitting divides text into fixed-length chunks based on the number of characters. This method is simple and fast but does not consider sentence or paragraph structure. In the notebook, this approach is demonstrated using manual Python logic and LangChain's `CharacterTextSplitter`.

Level 2: Recursive Character Text Splitting

Recursive character splitting improves upon basic character splitting by considering document structure. The splitter first attempts to divide text by paragraphs, then sentences, and finally characters if needed. This produces more meaningful chunks and is implemented using `RecursiveCharacterTextSplitter`.

Level 3: Document-Specific Splitting

Different document formats require customized splitting strategies. The notebook demonstrates:

- Markdown text splitting using headings
 - Python code splitting based on classes and functions
 - JavaScript code splitting using language-aware rules
- This ensures logical consistency in chunks.

Level 4: Semantic Chunking

Semantic chunking groups text based on meaning rather than size. Sentence embeddings are generated, and cosine similarity is used to measure semantic closeness. Sentences with similar meaning are grouped together, improving contextual understanding in downstream tasks.

Level 5: Agentic Chunking

Agentic chunking uses large language models to intelligently decide how text should be split. This method is context-aware and adaptive, making it suitable for complex, unstructured documents. It represents the most advanced chunking technique discussed in the notebook.

Applications

- Retrieval-Augmented Generation (RAG)
- Chatbots and virtual assistants
- Document search systems
- Knowledge bases
- AI-powered summarization tools

Conclusion

The notebook clearly demonstrates the evolution of text splitting techniques from simple character-based methods to intelligent, model-driven approaches. Selecting the appropriate splitting strategy depends on document type, size, and application requirements.

References

1. LangChain Official Documentation
2. OpenAI API Documentation
3. NLP and Machine Learning Research Papers