LAB FILE

for

AGENTIC AI LAB

CSCR3215

SEMESTER: VIth

Section: CS-F

Group: G2

SUBMITTED BY

Harshit Arora
2023277664

SHARDA SCHOOL OF ENGINEERING AND TECHNOLOGY

SHARDA UNIVERSITY, GREATER NOIDA – 201310

Text Chunking Methods for Language Models

Code Explanation (Chunking_Method.ipynb)

This document explains the workflow of the Chunking_Method.ipynb notebook, which demonstrates **multiple levels of text chunking techniques** used in modern **Large Language Model (LLM) pipelines**. The notebook progressively explores character-based, token-based, semantic, and agentic chunking strategies using Python and LangChain utilities.

---

1. Introduction to Text Chunking

Text chunking is the process of **splitting large text into smaller, manageable segments** to improve:

- Context handling in LLMs
- Retrieval-Augmented Generation (RAG)
- Embedding quality
- Model performance and cost efficiency

The notebook presents **five levels of chunking**, increasing in intelligence and adaptability.

---

2. Level 1: Character-Based Chunking

This is the most basic form of text splitting.

Description:

- The text is divided based on a **fixed number of characters**.
- No understanding of words, sentences, or meaning.

Implementation:

- A sample text string is defined.
- A loop slices the text using a fixed chunk_size.

- Each slice is appended to a list of chunks.

Limitations:

- May split words or sentences abruptly.
- Not suitable for real-world NLP applications.

---

3. Character Chunking Using LangChain

To simplify manual character splitting, the notebook uses **LangChain's CharacterTextSplitter**.

Key Parameters:

- chunk_size: Number of characters per chunk
- chunk_overlap: Overlapping characters between chunks
- separator: Character used for splitting
- strip_whitespace: Controls whitespace trimming

Advantages:

- Cleaner implementation
- Configurable chunking behavior
- Easier integration into LLM pipelines

---

4. Level 2: Recursive / Structured Chunking

At this level, chunking becomes **structure-aware**.

Concept:

- Attempts to split text by **logical separators** such as:
    - Paragraphs

    - Sentences
    - Lines
- Falls back to character splitting only if needed.

Benefit:

- Preserves textual structure
- Produces more meaningful chunks for embeddings

This approach is commonly used in **document ingestion pipelines**.

---

5. Level 3: Token-Based Chunking

Instead of characters, text is split using **tokens**, which align with how LLMs process input.

Why token-based?

- LLMs have **token limits**, not character limits.
- Ensures chunks fit within model constraints.

Usage:

- Tokenization-aware splitting
- Prevents truncation during inference

This method is more reliable for production LLM systems.

---

6. Level 4: Semantic Chunking

Semantic chunking introduces **meaning-awareness** into text splitting.

Implementation:

- Uses **OpenAI embeddings**
- Computes vector representations of sentences

- Measures similarity using **cosine similarity**

Process:

1. Sentences are embedded individually.
2. Similarity scores are calculated between sentences.
3. Sentences with high semantic similarity are grouped together.

Advantage:

- Chunks preserve **contextual meaning**
- Ideal for RAG and search-based systems

---

7. Level 5: Agentic Chunking

This is the most advanced chunking strategy demonstrated.

Concept:

- Uses an **LLM as an agent** to decide how text should be chunked.
- The model dynamically determines:
    - Chunk boundaries
    - Logical grouping
    - Context preservation

Tools Used:

- ChatOpenAI
- LangChain experimental modules

Benefits:

- Adaptive and intelligent
- Handles complex documents

- Mimics human understanding of text structure

---

8. Libraries and Tools Used

- **LangChain**
- **langchain_text_splitters**
- **langchain_experimental**
- **OpenAI Embeddings**
- **NumPy**
- **Scikit-learn (cosine similarity)**

These tools collectively enable efficient chunking and semantic analysis.

Conclusion

This notebook provides a comprehensive overview of **text chunking strategies**, ranging from simple character splitting to intelligent agent-based chunking. Understanding and applying the appropriate chunking method is critical for building scalable and high-performing LLM applications.