SHARDA UNIVERSITY

Beyond Boundaries

Agentic_AI_Lab1

Student Name-Harshit Singh

System id-2023491064

Roll no-2301010372

CSE-F(2023-27)

OVERALL PURPOSE OF THE CODE

This notebook demonstrates **5 levels of text splitting techniques** used in **NLP / LLM pipelines**, especially for:

- Large document processing
- Chunking text for embeddings
- Preparing data for **RAG (Retrieval-Augmented Generation)**
- Improving search, summarization, and QA systems

The idea is:

Different text splitting strategies are needed depending on data size, structure, and task.

🔷 **STEP 1: ENVIRONMENT SETUP & DRIVE MOUNTING**

```
from google.colab import drive

drive.mount('/content/drive')
```

🧠 **What's happening?**

- Mounts Google Drive into Colab
- Allows access to:
  - PDFs
  - Text files
  - Datasets stored in Drive

📌 Needed when working with **large documents**

🔷 **STEP 2: IMPORTING REQUIRED LIBRARIES**

You import libraries related to:

- Text processing
- LangChain text splitters
- File loaders

Typical imports include:

```
from langchain.text_splitter import (
    CharacterTextSplitter,
    RecursiveCharacterTextSplitter,
    TokenTextSplitter
```

)

꼰꽝 Why these?

LangChain provides **advanced text splitting utilities** designed for LLM workflows.

🔷 LEVEL 1: BASIC CHARACTER TEXT SPLITTING

🔷 Code Logic

```
CharacterTextSplitter(
  separator="\n",
  chunk_size=1000,
  chunk_overlap=200
)
```

꼰꽝 Working

- Splits text **purely based on character count**
- Uses a newline (\n) as separator
- Ensures:
    - Each chunk ≤ chunk_size
    - chunk_overlap preserves context between chunks

Use Case

- Simple text files
- When structure doesn't matter

Limitation

- Can break:
    - Sentences
    - Paragraph meaning

🔷 LEVEL 2: RECURSIVE CHARACTER TEXT SPLITTING (MOST IMPORTANT)

```
RecursiveCharacterTextSplitter(
  chunk_size=1000,
  chunk_overlap=200
)
```

꼽 Working (Very Important Concept)

This splitter tries separators **in order**:

1. \n\n (paragraphs)
2. \n (lines)
3. (spaces)
4. Characters (last resort)

It **recursively** splits until chunk size condition is met.

Advantages

- Preserves:
  - Paragraphs
  - Sentences
  - Semantic meaning

This is the industry-standard splitter for RAG

LEVEL 3: TOKEN-BASED TEXT SPLITTING

```
TokenTextSplitter(
  chunk_size=500,
  chunk_overlap=50
)
```

꼽 Working

- Splits text based on **tokens**, not characters
- Tokens ≠ words (depends on tokenizer like GPT, BERT)

Why important?

LLMs have **token limits**, not character limits.

Example

- "ChatGPT is amazing" → maybe 5 tokens, not 3 words

LEVEL 4: DOCUMENT-BASED SPLITTING

This level involves:

- Loading documents (PDF, TXT)

- Splitting while **preserving metadata**

Example logic:

docs = loader.load()

split_docs = splitter.split_documents(docs)

꼼꼼 Working

- Each chunk retains:
  - Page number
  - Source file
  - Document metadata

Useful for:

- Search results with citations
- Academic & legal documents

🔷 LEVEL 5: SEMANTIC / SMART SPLITTING (ADVANCED)

This level focuses on:

- Meaning-aware splitting
- Chunking based on **content relevance**

May involve:

- Embeddings
- Similarity-based grouping

꼼꼼 Working Conceptually

1. Convert text → embeddings
2. Detect semantic boundaries
3. Split where meaning changes

🎯 Used in:

- Advanced RAG systems
- Research-level NLP
- Production AI assistants