Lab 03: Text Chunking

Splitting Documents for RAG Systems

CSI403 - Full Stack Program Development

Why Chunking?

LLMs have token limits!

- GPT-4: 8K-128K tokens
- Claude: 100K-200K tokens
- Local LLMs: 2K-8K tokens

Solution: Split into Chunks

Large Document $\rightarrow$ [Chunk 1] [Chunk 2] [Chunk 3] ...

Simple Chunking: String Slicing

```
text = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

# Slice from index 0 to 10
chunk1 = text[0:10]    # "ABCDEFGHIJ"
chunk2 = text[10:20]   # "KLMNOPQRST"
chunk3 = text[20:26]   # "UVWXYZ"
```

Syntax: text[start:end]

Returns characters from index start to end-1

Automatic Chunking with Loop

```
def simple_chunk(text, chunk_size):
    chunks = []
    for i in range(0, len(text), chunk_size):
        chunk = text[i:i+chunk_size]
        chunks.append(chunk)
    return chunks

result = simple_chunk("ABCDEFGHIJ", 3)
# ['ABC', 'DEF', 'GHI', 'J']
```

Chunking with Overlap

Overlap preserves context between chunks!

```
def chunk_with_overlap(text, size, overlap):
    chunks = []
    start = 0
    while start < len(text):
        end = start + size
        chunks.append(text[start:end])
        start = end - overlap
    return chunks

# size=10, overlap=3
# [ABCDEFGHIJ] [HIJKLMNOPQ] [OPQRSTUVWX]
```

Summary

- **Chunking** = splitting text into smaller pieces
- Use `text[start:end]` for slicing
- Use `range(0, len(text), chunk_size)` for looping
- **Overlap** helps maintain context

Next Lab

Lab 04: Text Search - finding relevant chunks!

Questions?

Let's start the Tutorial!