Handling Long Text in Large Language Models (LLMs)

1. Introduction

This document focuses **exclusively on techniques and strategies for handling long text in Large Language Models (LLMs)**. It does not cover general NLP concepts or unrelated LLM applications. The entire discussion is centered on overcoming context window limitations and enabling LLMs to work effectively with long documents such as reports, PDFs, books, logs, transcripts, and enterprise knowledge bases.

2. Understanding the Context Window Limitation

2.1 What is a Context Window?

The context window refers to the maximum number of tokens (words or subwords) that an LLM can read and reason over at once. Tokens are not the same as words; a single word can map to multiple tokens.

2.2 Why Context Limits Exist

Context limits exist due to: - Computational constraints (memory and attention complexity) - Model architecture (self-attention scales poorly with length) - Cost and latency considerations

2.3 Problems Caused by Context Limits

- Truncation of important information
- Loss of global document understanding
- Hallucinations due to missing context
- Increased costs when repeatedly sending large prompts

3. Chunking: The Foundation Technique

3.1 What is Chunking?

Chunking is the process of splitting a long document into smaller, manageable pieces (chunks) that fit within the LLM’s context window.

3.2 Chunk Size and Overlap

- Typical chunk sizes: 300–1000 tokens
- Overlap: 10–20% of chunk size
- Overlap helps preserve context between chunks

3.3 Types of Chunking

- Fixed-size chunking
- Sentence-based chunking
- Paragraph-based chunking
- Semantic chunking (topic-aware)

3.4 Advantages and Limitations

Advantages - Simple to implement - Works with any LLM

Limitations - Loses global context - Requires aggregation logic

4. Sliding Window Approach

4.1 Concept

The sliding window approach processes text sequentially with overlapping windows instead of independent chunks.

4.2 How It Works

- Window moves forward by a fixed stride
- Each window shares tokens with the previous one

4.3 Use Cases

- Long conversations
- Sequential narratives
- Time-series text data

4.4 Trade-offs

- Better continuity than chunking
- Higher computational cost

5. Hierarchical Summarization (Map-Reduce)

5.1 Overview

Hierarchical summarization summarizes long documents in multiple stages.

5.2 Process Flow

1. Split document into chunks
2. Summarize each chunk (Map step)
3. Combine summaries
4. Summarize combined output (Reduce step)

5.3 Benefits

- Scales to very large documents
- Preserves high-level meaning

5.4 Limitations

- Fine-grained details may be lost
- Requires careful prompt design

6.Memory-Based Techniques

6.1 Short-Term Memory

Stores recent conversation turns within the context window.

6.2 Long-Term Memory

Stores summarized or embedded past interactions externally.

6.3 Memory Management Strategies

- Periodic summarization
- Topic-based memory storage
- Time-based expiration

6.4 Applications

- Personal assistants
- Conversational AI
- CRM chat systems

7. Long-Context Models

7.1 Overview

Some modern LLMs support very large context windows (tens or hundreds of thousands of tokens).

7.2 Advantages

- Reduced need for chunking
- Simpler pipelines

7.3 Challenges

- High cost
- Latency issues
- Still benefit from retrieval techniques

8. Evaluation and Optimization

8.1 Quality Metrics

- Relevance
- Faithfulness
- Completeness
- Consistency

8.2 Performance Metrics

- Latency
- Cost per query
- Retrieval accuracy

8.3 Optimization Techniques

- Hybrid search (keyword + vector)
- Caching
- Prompt compression
- Adaptive chunk sizes

9. Best Practices and Real-World Architecture

9.1 Recommended Architecture

- Chunking + Embeddings
- Vector database
- RAG pipeline
- Memory layer
- Monitoring and evaluation

9.2 Common Mistakes

- Oversized chunks
- No overlap
- Poor embedding models
- Ignoring evaluation