Context Window Management in Large Language Models (LLMs)

This document provides a comprehensive guide to **strategies and techniques for managing the context window** when using large language models (LLMs). It consolidates insights from Mohammed Al Salboukh's Medium article and several advanced research and engineering references.

---

1. Token Truncation

What it is

Token truncation means simply **removing the oldest or least relevant parts** of the conversation or text once the total token count exceeds the model's maximum context limit.

How it works

You track the total number of tokens currently in context. If adding new content would exceed the threshold, you drop older parts until the sum fits comfortably within the available context window.

Advantages

- Simple and fast to implement.
- Guarantees you never exceed the context window.
- No extra computation cost.

Drawbacks

- Loses potentially useful older information.
- Can make the model "forget" earlier reasoning or preferences.
- Does not discriminate between relevant and irrelevant parts — everything old gets dropped.

When to use

- Short conversations or low-context tasks.
- Situations where recent content is most important.
- Prototyping or early-stage systems that need reliability more than memory.

---

2. Summarisation (Context Compression)

What it is

Instead of discarding information, **summarisation compresses earlier content** into a concise textual form that still retains meaning.

How it works

When the context becomes large, older segments are summarised using a smaller number of tokens. These summaries replace the detailed content, preserving the gist of what has been discussed earlier.

Advantages

- Maintains semantic continuity.
- Balances token budget and memory retention.
- Prevents catastrophic forgetting of early context.

Drawbacks

- Summaries might omit or distort key information.
- Summarisation consumes additional tokens and inference time.
- Difficult to verify the quality or faithfulness of the summary.

When to use

- Long-running chat sessions where previous context still matters.

- Scenarios requiring compact retention of facts or decisions.
- When accuracy can tolerate some compression loss.

---

3. Chunking / Sliding Window

What it is

Chunking divides long texts into **manageable sections (chunks)** that fit within the model's context size.
A sliding window version processes overlapping chunks to preserve continuity.

How it works

A document is split into smaller overlapping parts, each small enough for the model to process. The model works on each chunk individually and results are later merged or summarised.

Advantages

- Enables handling of very large inputs such as books, documents, or long logs.
- Preserves full coverage without losing parts.
- Works well for summarisation, extraction, or information retrieval tasks.

Drawbacks

- Loses global coherence between chunks unless a second summarisation pass is done.
- Increases computational cost due to multiple model invocations.
- May duplicate or break context around chunk boundaries.

When to use

- When input length far exceeds model limits.
- Document-level processing, code analysis, or summarisation pipelines.

- When you can afford more processing time.

---

4. Contextual Prioritisation / Selective Retrieval

What it is

Rather than keeping all past information, **only the most relevant parts** are selected based on similarity or relevance to the current query.

How it works

Each message or document is embedded into a vector representation. When a new query arrives, its embedding is compared against stored ones. The top-K most similar items are chosen to build the current context.

Advantages

- Efficient token usage — only relevant pieces are included.
- Improves quality by reducing noise.
- Works seamlessly with retrieval-augmented generation.

Drawbacks

- Needs an embedding model and a vector database.
- Relevance scoring may fail, leading to missing context.
- More complex than simple truncation or summarisation.

When to use

- When dealing with large histories or knowledge bases.
- For chatbots, assistants, and multi-document search systems.
- As part of RAG pipelines.

---

5. Dynamic Context Adjustment

What it is

This approach **adjusts the size of context dynamically** based on the complexity of the user query or task.

How it works

The system estimates whether a query is simple or complex. Simple queries use smaller context windows; complex ones use extended context with additional summarisation or retrieval. This conserves tokens and speeds up responses.

Advantages

- Adapts to workload and task complexity.
- Balances latency, cost, and accuracy.
- Reduces unnecessary token usage.

Drawbacks

- Requires good heuristics or classification of query complexity.
- Misclassification can cause missing or redundant context.
- Adds system logic overhead.

When to use

- When user queries vary greatly in complexity.
- In systems prioritising token efficiency and cost control.
- For production environments where speed and performance matter.

---

6. Retrieval-Augmented Generation (RAG)

What it is

RAG combines retrieval and generation: **fetching relevant data from an external knowledge base** and feeding it to the model, keeping the context window small but informative.

How it works

Documents are embedded and stored in a vector store. A query retrieves the top relevant chunks. These are concatenated with the user question and passed to the model for reasoning and answer generation.

Advantages

- Effectively bypasses the limits of the model's fixed context window.
- Keeps knowledge external, allowing updates without retraining.
- Provides factual grounding for model responses.

Drawbacks

- Requires embedding pipelines and vector database management.
- Retrieval quality determines final accuracy.
- Still must manage token budget for retrieved chunks.

When to use

- Knowledge-heavy applications.
- Chatbots requiring reference to long-term data.
- Systems needing scalable, factual responses.

---

7. Hybrid / Semantic Compression

What it is

Hybrid methods combine multiple techniques — for example, **chunking + summarisation + retrieval** — or use more advanced "semantic compression" to distil entire documents.

How it works

Large documents are chunked, each chunk summarised or compressed, and those summaries are further merged or summarised again. The model receives a dense, highly compressed semantic overview of the source material.

Advantages

- Can handle extremely large inputs.
- Retains both breadth and essence of the information.
- Well suited for hierarchical or multi-layer summarisation.

Drawbacks

- Multi-step process increases complexity and cost.
- Cascading summarisation can cause information loss.
- Hard to tune automatically for optimal compression depth.

When to use

- Large-scale document analysis (e.g., legal, research, audit data).
- Multi-hour conversations or memory-based agent systems.
- Enterprise systems with structured pipelines.

---

8. Context Engineering / Ordering

What it is

This is about **structuring, ordering, and formatting** the content within the prompt so the model interprets it effectively — i.e., “what to keep and in what order”.

How it works

System instructions, critical background, summaries, and the most recent exchanges are arranged logically. Important information appears near the beginning or end of the prompt (where LLMs typically pay more attention).

Advantages

- Maximises “value per token”.
- Helps mitigate recency or primacy bias.
- Clarifies context, reducing hallucinations and confusion.

Drawbacks

- Requires manual design and ongoing tuning.
- Adds maintenance overhead for multi-stage prompts.
- Still constrained by total token capacity.

When to use

- Production-grade systems where reliability and interpretability matter.
- Multi-agent or multi-tool architectures with layered prompts.
- Any scenario requiring deterministic prompt control.