

L2-M6: Collaborating with AI - Multi-turn Conversations

Managing Context, Maintaining Coherence, Guiding AI Through Extended Dialogues

Module ID: L2-M6

1. Introduction: The Challenge of Extended AI Dialogue

Large Language Models (LLMs) have revolutionized human-computer interaction, moving beyond single-query responses to enable complex, multi-turn conversations. However, this extended dialogue presents a fundamental technical challenge: **context management**. Unlike human memory, an LLM's "memory" of a conversation is constrained by its **context window**, a finite limit on the number of tokens it can process in a single input. As a conversation lengthens, the history quickly fills this window, leading to a phenomenon known as **context decay** or "**getting lost in conversation**" [1].

This module provides a technical deep dive into the mechanisms, strategies, and prompt engineering techniques required to manage context effectively, maintain conversational coherence, and guide AI through productive, extended dialogues.

1.1 The Technical Constraint: The Context Window

The context window is the primary technical limitation in multi-turn AI collaboration. It dictates the maximum length of the input prompt (including the system instruction, conversation history, and the current user query) that the model can consider when generating a response.

| Feature | Description | Implication for Multi-turn Conversation |
|---------------------|---|--|
| Fixed Size | Measured in tokens (e.g., 8k, 32k, 128k, 1M). | Longer conversations will inevitably exceed this limit. |
| Tokenization | Input text is broken down into tokens (words, sub-words, punctuation). | A single word can consume multiple tokens (e.g., "tokenization" is 5 tokens in some models). |
| Recency Bias | Models often exhibit better performance on information closer to the end of the context window. | Older turns of the conversation are more likely to be "forgotten" or discounted. |

The core objective of multi-turn context management is to ensure the **most relevant** information from the conversation history is always present within the available context window.

2. Context Management Techniques: Preserving Memory

Effective collaboration with an AI over an extended period requires a deliberate strategy to manage the conversation history before it is passed to the LLM. This is often implemented on the application layer, outside of the LLM itself.

2.1 Truncation (Sliding Window)

Truncation, or the **Sliding Window** approach, is the simplest and most common context management technique.

Step-by-Step Explanation: 1. **Store:** All user and AI messages are stored in a history buffer. 2. **Measure:** Before a new prompt is sent, the total token count of the system prompt, the new user query, and the conversation history is calculated. 3. **Truncate:** If the total count exceeds the context window limit, the oldest messages (from the beginning of the history) are removed until the total token count falls below the limit. The most recent messages are always prioritized.

| Advantage | Disadvantage |
|---|--|
| Simple to implement and computationally cheap. | Irrelevant information is retained while potentially critical, but older, context is lost. |
| Guarantees the most recent turns are always included. | Leads to abrupt loss of context, causing the AI to "forget" initial setup or goals. |

2.2 Summarization (Memory Buffering)

Summarization involves periodically condensing the conversation history into a shorter, more dense representation. This technique transforms the expansive history into a concise, token-efficient summary that can be prepended to the current context.

Step-by-Step Explanation: 1. **Trigger:** When the conversation history reaches a predefined token threshold (e.g., 75% of the context window), a summarization trigger is activated. 2. **Summarize:** The conversation history is sent to the LLM with a specific instruction, such as: "Summarize the preceding conversation between the user and AI, focusing on key decisions, open questions, and the user's primary goal, into a single paragraph." 3. **Replace:** The original history is replaced by the generated summary, which consumes significantly fewer tokens. New turns are added after the summary.

This method is more robust than simple truncation as it preserves the **semantic content** of the older conversation, even if the verbatim text is lost.

2.3 Retrieval-Augmented Generation (RAG) for Context

For highly technical or knowledge-intensive dialogues, **Retrieval-Augmented Generation (RAG)** can be adapted to manage conversational context. Instead of storing the entire conversation, key pieces of information (e.g., user requirements, project goals, technical specifications) are extracted and embedded into a vector database.

Step-by-Step Explanation: 1. **Chunk and Embed:** Each turn or key statement is chunked, converted into a vector embedding, and stored in a database. 2. **Query:** When the user asks a new question, the query is also embedded. 3. **Retrieve:** The system queries the vector database to retrieve the most semantically similar chunks from the *conversation history* (or external documents). 4. **Augment:** Only the

retrieved, relevant chunks are added to the prompt, along with the system instruction and the current user query.

This technique is highly effective for maintaining coherence on specific topics, as it selectively retrieves only the most pertinent context, regardless of when it occurred in the dialogue.

3. Maintaining Conversational Coherence

Coherence is the logical and semantic flow of a conversation. In multi-turn AI dialogue, coherence is maintained not just by managing context, but by actively guiding the AI's internal state and response generation.

3.1 System Prompt Refinement (Dynamic System Instructions)

The **System Prompt** is the foundational instruction that defines the AI's persona, goals, and constraints. In extended dialogues, this prompt should not be static; it should be **dynamically refined** by the application layer based on the conversation's progression.

Practical Application: Project Management Assistant

| Conversation Stage | Dynamic System Instruction Update | Coherence Goal |
|--------------------|---|--|
| Initial Setup | "You are a Senior Project Manager. Your goal is to define the scope and timeline for a new software feature." | Establishes persona and primary objective. |
| Scope Defined | "The project scope is now defined. Your new goal is to help the user break down the scope into technical tasks and estimate effort." | Shifts focus to the next logical phase of the project. |
| Error/Correction | "The user has corrected a previous assumption about the database schema. All future responses MUST adhere to the new schema: <code>Users (id, name, email)</code> . Do not reference the old schema." | Enforces adherence to corrected or updated facts. |

This dynamic approach ensures the AI's internal "mindset" is always aligned with the current phase of the collaboration.

3.2 Coreference Resolution and Entity Tracking

A common failure in multi-turn LLM conversations is the inability to correctly resolve **coreferences** (e.g., "it," "he," "the project") to the specific entities they represent.

Example of Coreference Failure: * **User Turn 1:** "I want to build a Python script to analyze stock data." * **User Turn 2:** "Can it also generate a chart?" (The AI might forget "it" refers to the "Python script" and generate a general chart-making suggestion).

To mitigate this, the application can use a smaller, specialized language model or a rule-based system to perform **entity tracking** and **coreference resolution** before passing the prompt to the main LLM. The system rewrites the user's ambiguous query into an explicit one:

Step-by-Step Resolution: 1. **Original Query:** "Can it also generate a chart?" 2. **Resolution Engine:** Identifies "it" as the "Python script to analyze stock data" from history. 3. **Rewritten Prompt (sent to LLM):** "The user is asking: Can the Python script to analyze stock data also generate a chart?"

This pre-processing step significantly improves the model's ability to maintain semantic coherence.

4. Guiding AI Through Extended Dialogues (Advanced Prompt Engineering)

Effective multi-turn collaboration is an art of **guiding** the AI. This involves structuring queries and responses to maximize information density and minimize context waste.

4.1 The "Constraint-Action-Output" (CAO) Pattern

For complex, multi-step tasks, the **CAO Pattern** provides a robust framework for structuring multi-turn prompts. It ensures the AI understands its limitations, its immediate task, and the required output format.

| Element | Description | Multi-turn Application |
|-------------------|--|--|
| Constraint | Defines the rules, limitations, and essential context. | <i>Example:</i> "You have a 500-word limit for this section. The tone must be academic." |
| Action | The specific, atomic task the AI must perform in this turn. | <i>Example:</i> "Draft the 'Context Management Techniques' section (2.1 and 2.2) based on the outline." |
| Output | Specifies the required format and structure of the response. | <i>Example:</i> "Output only the Markdown text for the section. Include a table with two columns: 'Advantage' and 'Disadvantage'." |

By applying the CAO pattern turn-by-turn, the user effectively breaks down a large, complex goal into a series of manageable, coherent micro-tasks, preventing the AI from straying or generating irrelevant content.

4.2 Step-by-Step: Iterative Refinement and State Management

For tasks requiring multiple drafts or complex data manipulation, an iterative, state-managed approach is essential. This technique treats the conversation as a formal state machine.

Practical Example: Drafting a Technical Report

| Turn | User Action (Prompt) | AI Response (State Change) | Context Management Technique |
|-------------------|---|--|--|
| 1 (Setup) | "Draft the introduction to the report on LLM context. The core thesis is 'Context is the new bottleneck.'" | Drafts Introduction. | System Prompt (Initial Goal) |
| 2 (Refinement) | "Review the introduction. The tone is too informal. Revise it to be more academic and cite two sources." | Revises Introduction, updating the text. | Truncation (Keeps only revised text) |
| 3 (New Task) | "Now, move to Section 2. Before you start, summarize the key points of the Introduction for your own reference." | Provides a 3-sentence summary of the Introduction. | Summarization (New summary replaces old intro text in history) |
| 4 (Guided Action) | "Draft Section 2 using the CAO pattern. Constraint: 400 words. Action: Explain RAG for context. Output: Markdown with a bulleted list." | Drafts Section 2 as requested. | CAO Pattern (Guides output structure) |

This process ensures that each turn builds logically on the previous one, and the user is actively managing the AI's internal state by confirming, correcting, or summarizing before proceeding.

4.3 The "Self-Correction" Meta-Prompt

A powerful technique for maintaining coherence is to instruct the AI to perform a **self-correction** or **self-reflection** step. This meta-prompt forces the LLM to review its previous output against the system instructions and the conversation history before generating a new response.

Self-Correction Prompt Template:

****System Instruction (Pre-pended to every turn):****
Before generating your response, perform the following two steps:

- 1. **Review:**** Read the last [N] turns of the conversation. Identify the user's current goal and any constraints established in the system prompt.
- 2. **Self-Correction:**** Check your previous response for any factual errors, inconsistencies, or deviations from the required format.
- 3. **Action:**** Generate the new response, ensuring it is coherent with the established context and corrects any identified errors.

While this consumes more tokens (due to the extra instruction), it drastically improves the quality and coherence of the output in extended, complex dialogues, especially when the conversation is nearing the context window limit [2].

5. Practical Applications and Best Practices

The principles of context management and coherence are critical across various professional applications.

5.1 Application: Code Generation and Refactoring

In software development, multi-turn conversations are used to iteratively generate, test, and refactor code.

| Best Practice | Description |
|----------------------|---|
| Code Block Isolation | Always enclose generated code in language-specific Markdown code blocks (e.g., <code>```python</code>). This prevents the code from polluting the natural language context and ensures the AI can easily identify and reference the code itself. |
| File-Based Context | Instead of pasting the entire codebase into the prompt, use RAG or a file-linking system. The user's prompt should reference files: "Refactor the <code>User.py</code> class (see file) to use asynchronous database calls." |
| Atomic Commits | Treat each successful turn (e.g., a function written, a bug fixed) as an "atomic commit." The user should confirm the change and explicitly state the next, isolated task. |

5.2 Application: Technical Documentation Drafting

Drafting long-form technical documents requires the AI to maintain consistency in terminology, tone, and structure across many sections.

Step-by-Step Guide: Drafting a Technical Whitepaper

1. **Establish the Style Guide (Turn 1):** Provide the AI with a comprehensive system prompt defining the persona ("Senior Technical Writer"), the target audience ("Advanced Engineers"), and the style guide (e.g., "Use active voice, define all acronyms on first use, cite sources in APA format").
 2. **Outline Generation (Turn 2):** Request a detailed, multi-level outline. Once approved, this outline is the **master context**.
 3. **Section-by-Section Drafting (Turns 3-N):** Use the CAO pattern for each section.
Constraint: "Adhere to the approved outline (Section 3.2: 'Context Decay')." **Action:** "Draft the section." **Output:** "Markdown text only."
 4. **Glossary and Acronym Management:** Maintain a running list of defined acronyms and key terms in a separate, high-priority part of the context window or in the dynamic system prompt. This prevents the AI from redefining terms or using inconsistent language.
-

6. Conclusion and Key Takeaways

Collaborating with AI through multi-turn conversations transforms the user from a simple query-sender into a **context manager** and **conversational guide**. The effectiveness of the collaboration is directly proportional to the user's ability to manage the technical constraints of the LLM.

The primary lesson is that the AI does not "remember" the conversation; it merely processes the tokens provided in the current prompt. Successful extended dialogue requires the user to strategically curate those tokens.

Key Takeaways

- **Context Window is the Bottleneck:** All context management techniques (Truncation, Summarization, RAG) are designed to mitigate the finite token limit of the LLM's context window.

- **Prioritize Semantic Content:** Summarization and RAG are superior to simple truncation because they preserve the *meaning* of the conversation, not just the most recent *words*.
 - **Be the Guide:** Use dynamic System Prompts and the Constraint-Action-Output (CAO) pattern to actively guide the AI's focus, ensuring coherence and preventing context drift.
 - **Resolve Ambiguity:** Pre-processing steps like coreference resolution (explicitly defining "it," "this," "that") are crucial for maintaining logical flow in complex technical discussions.
 - **Iterate and Confirm:** Treat extended dialogue as a state machine, confirming the AI's output and updating the context before moving to the next logical step.
-

References

- [1] OpenAI Community. (2025, May 20). *Interesting research: lost in conversation. All models get lost easily in multi-turn conversations*. <https://arxiv.org/html/2505.06120v1>
- [2] Chen, B., et al. (2025). Unleashing the potential of prompt engineering for large language models. *Cognitive Systems Research*, 91, 101234. <https://www.sciencedirect.com/science/article/pii/S2666389925001084> [3]
- PromptHub. (2025, June 26). *Why LLMs Fail in Multi-Turn Conversations (And How to Fix It)*. Retrieved from <https://www.prompthub.us/blog/why-llms-fail-in-multi-turn-conversations-and-how-to-fix-it>
- [4] Google Cloud. *Prompt Engineering for AI Guide*. Retrieved from <https://cloud.google.com/discover/what-is-prompt-engineering>
- [5] Wang, K., et al. (2024). Understanding users' effective use of generative conversational AI. *Computers in Human Behavior: Reports*, 14, 100412. <https://www.sciencedirect.com/science/article/pii/S266676492400047X>