Don't Scroll Back: Missing-Evidence Memory for Streaming Dialogue Summarization

Hyangsuk Min[1], **Hwanjun Song**[1,2,*]
[1]KAIST, [2]Cluvion
{hyangsuk.min, songhwanjun}@kaist.ac.kr

Abstract

Users of modern platforms repeatedly need summaries of recent dialogue, but the window rarely contains enough context to be interpreted on its own. We formalize this setting as streaming dialogue summarization, where a system must summarize a current window using selective memory from an unbounded history under a fixed budget. We show that the central challenge is not how much history is accessed, but whether memory recovers the evidence that the current window presupposes. We construct a benchmark and evaluation protocol that separately assess whether memory contains gap-resolving evidence and whether the generated summary reflects it. We propose ReMEMBER, a missing-evidence memory framework that conditions retrieval on unresolved window dependencies and refines retrieved chunks into evidence-dense memory under a fixed budget. Experiments on dialogues with histories up to 160K tokens show that ReMEMBER improves memory recall and gap-resolution completeness over memory construction baselines under the same budget. Our code and benchmark dataset are available at https://github.com/DISL-Lab/ReMEMBER.

1 Introduction

Modern communication platforms, such as messaging apps (e.g., WhatsApp, WeChat) and collaborative workspaces (e.g., Slack, MS Teams), generate continuous streams of multi-turn dialogues (Yi et al., 2025; Kirstein et al., 2025; Deshpande et al., 2025). Users rarely seek a global summary, and instead repeatedly need summaries of recent segments (Wang et al., 2024; Ghebriout et al., 2025; Wu et al., 2025) to catch up on recent discussions or decisions. We refer to this as *streaming dialogue summarization*. The core challenge is that the current window is seldom self-contained. Speakers rely on shared history, leaving pronouns without antecedents, entities without attributes and decisions without rationales. Summarizing the window alone yields ungrounded fragments, while resolving these dependencies requires retrieving evidence from earlier turns under a strict budget.

Existing dialogue summarization, however, is largely designed under a *closed*-dialogue assumption (Zhu et al., 2025b; Jin et al., 2026), where the full conversation is available and a single global summary is produced. To scale to long dialogues, prior methods adopt either *incremental* summarization (Wu et al., 2025; Wang et al., 2025), which updates a running summary as new turns arrive, or *hierarchical* summarization (Kim and Kim, 2025; Li et al., 2025b; Ou and Lapata, 2025), which summarizes small segments independently and then merges them. While effective for constructing a global dialogue summary, they are misaligned with streaming scenarios. The former attenuates earlier context through repeated compression, while the latter favors global abstraction and discards the fine-grained evidence needed to interpret a specific window. Neither preserves the prior context that resolves dependencies in the current window.

One might simply feed the entire dialogue history to recover the missing context. In practice, however, histories routinely span tens to hundreds of thousands of tokens (Lee et al., 2025a), where this approach is computationally prohibitive and unreliable due to well-known *long-context* pathologies, including lost-in-the-middle effects (Liu et al., 2024a; Song et al., 2026), needle-in-a-haystack failures (Lee et al., 2025b) and inference overhead (Sun et al., 2025). As in Figure 1, neither full-context nor local-only summarization suffices. The central challenge of streaming dialogue summarization is thus one of *memory*, that is, selectively recovering the past context that the current window depends on, under a strict memory budget.

To diagnose this challenge, we examine three representative memory paradigms, *recency*-based,

*Corresponding Author.

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 22728–22760
October 24-29, 2026 ©2026 Association for Computational Linguistics

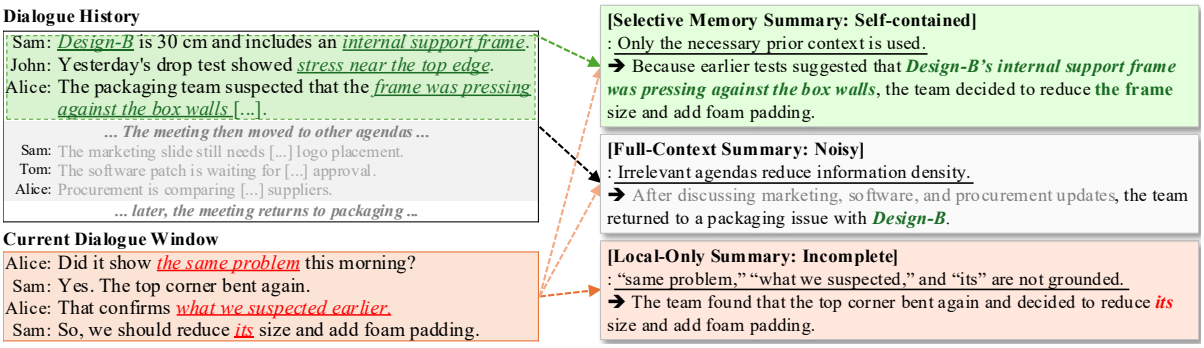


Figure 1: Context utilization in streaming dialogue summarization. *Local-Only* leaves contextual dependencies unresolved (red) and *Full-Context* introduces irrelevant history (gray). *Selective Memory* addresses both by retaining only necessary prior evidence for a self-contained summary (green).

summarization-based and *retrieval*-based (Liu et al., 2024b; Wu et al., 2025; Li et al., 2025b). Our analysis identifies retrieval as the strongest existing paradigm, since it preserves explicit historical evidence and selects past turns relevant to the current window. Yet, even retrieval reveals two fundamental bottlenecks under streaming settings. On the *construction* side, retrieval is driven by what the window already expresses, while the memory it actually needs is defined by what the window leaves unresolved. On the *utilization* side, retrieved chunks mix resolving evidence with neighboring turns, repetitions and topic drift, lowering evidence density under a fixed memory budget.

In response, we propose ReMEMBER (Retrieving Missing Evidence Memory By Elicited Recall), a context-gap-centered memory framework for streaming dialogue summarization. ReMEMBER operates in two stages: *(i) Gap-conditioned evidence retrieval*, which elicits unresolved dependencies in the current window and issues targeted queries to retrieve historical chunks that resolve what standard retrieval overlooks; and *(ii) Gap-conditioned chunk refinement*, which reduces memory noise by identifying the most relevant turns across retrieved chunks and retaining only those turns within a fixed memory budget. This shifts memory construction from similarity-based chunk retrieval to evidence-dense memory construction. Memory stores not history that resembles the window, but evidence that completes it—without any external query. Experiments on long-context chit-chat and workplace dialogues spanning up to 160K tokens show that ReMEMBER improves memory recall and gap-resolution completeness over recency-, summarization- and retrieval-based memory baselines under the same budget.

Our main contributions are summarized as:

(1) We introduce streaming dialogue summarization as a new task with a benchmark that separately evaluates memory evidence and summary reflection; (2) We analyze representative memory paradigms and identify two bottlenecks of retrieval-based memory in streaming settings: similarity-driven construction and noisy memory utilization; (3) We propose ReMEMBER, a missing-evidence memory construction framework using gap-conditioned retrieval and turn-level chunk refinement to extract resolving evidence under a fixed budget; (4) We show that ReMEMBER improves memory recall and gap-resolution completeness over memory construction baselines, with consistent gains up to 160K-token histories.

2 Related Work

Long Dialogue Tasks. Long dialogue research studies how models use information when conversations exceed a single context window. Existing work includes multi-session response generation (Xu et al., 2022; Chen et al., 2025; Liu et al., 2025), which predicts the next utterance from accumulated history, long-dialogue understanding such as QA or entity tracking over a complete conversation (Kim et al., 2024a; Maharana et al., 2024; Bai et al., 2025; Lee et al., 2025a), and conversational search, which rewrites prior turns into explicit retrieval queries (Yoon et al., 2025; Zhu et al., 2025a). Streaming dialogue summarization departs from these settings in two respects. No retrieval target is specified, and the model must identify what the current window leaves unresolved. The output is not a response to a question but a coherent completion of the window, rendering its content interpretable rather than answering it.

Dialogue Summarization. Dialogue summarization initially focused on compressing dialogue-specific structures, including multi-turn dependencies, speaker roles, topic shifts and discourse relations (Tang et al., 2022; Jia et al., 2023; Gao et al., 2023b; Lin et al., 2023; Xiao et al., 2024; Yi et al., 2025). For longer dialogues, prior work scales summarization through incremental updates of a running summary (Hwang et al., 2024; Ravaut et al., 2024) or hierarchical compression of segmented chunks (Zhu et al., 2020; Li et al., 2021; Zhang et al., 2021). Query-focused and instructive dialogue summarization assumes an explicit information need is given with the input (Zhong et al., 2021; Wang et al., 2023a). However, these methods are largely formulated in closed settings, where the complete dialogue is available, and the output is a global summary. This assumption does not specify which past information is necessary for interpreting a particular target window, making it insufficient for streaming dialogue summarization.

Long-Context Memory Construction. Long-context memory modeling selects or compresses past information. Summarization-based memory maintains compact histories through recursive or bounded summaries (Wang et al., 2025, 2026), but compression may remove evidence needed for local contextual gaps. Retrieval-based memory fetches relevant past utterances (Zhong et al., 2024; Tan et al., 2025; Pan et al., 2025; Li et al., 2025a), but high-recall retrieval can introduce irrelevant or redundant context. Intent- and goal-driven memory improves selectivity through missing slots or user-centric needs (Du et al., 2026; Yan et al., 2026), but targets agent response generation rather than contextual evidence construction for summarization. Structured memory organizes history through timelines or self-questioning (Ong et al., 2025; Yang et al., 2026), but is optimized for general context management. These methods motivate long-history memory construction, but they do not construct evidence for resolving target-window gaps.

Query Decomposition for Retrieval. Retrieval-oriented query reformulation improves evidence access by rewriting underspecified or complex inputs into explicit retrieval queries (Elgohary et al., 2019; Yoon et al., 2025; Zhu et al., 2025a). Related methods also use generated queries or hypothetical documents to guide retrieval (Mao et al., 2021; Gao et al., 2023a; Wang et al., 2023b). ReMEMBER shares the goal of targeted retrieval, but differs in its trigger and target. It is triggered by discourse incompleteness in the current window, and its queries specify non-hypothetical evidence needs for missing prior dialogue context. This distinction matters because streaming summaries must recover referents, prior states and rationales from earlier turns rather than infer plausible answers.

3 Streaming Dialogue Summarization: Formulation and Evaluation

In the absence of a formal task definition and dedicated benchmark for streaming dialogue summarization, we present the problem formulation (Section 3.1), benchmark construction (Section 3.2) and evaluation axes (Section 3.3).

3.1 Problem Formulation

We formulate *streaming dialogue summarization* as a sequential task over an utterance stream. At each step t, the model observes a current window $\mathcal{W}_t$ of recent utterances, while all preceding utterances form the accumulated history $\mathcal{H}_t$. The goal is to generate a self-contained summary of $\mathcal{W}_t$ by leveraging history $\mathcal{H}_t$. The summary reflects the *salient utterances* within $\mathcal{W}_t$. A subset of these salient utterances is independently comprehensible. The interpretation of the remaining salient utterances strictly depends on prior context in $\mathcal{H}_t$. This dependence introduces *contextual gaps*. Resolving such gaps necessitates retrieving missing evidence from $\mathcal{H}_t$. The history $\mathcal{H}_t$ grows unboundedly over time, rendering exhaustive access computationally impractical. Therefore, the model must construct and maintain a memory module $\mathcal{M}_t$. Instead of storing all past interactions, $\mathcal{M}_t$ selectively extracts and preserves the critical evidence necessary to resolve contextual gaps. The success of the task is determined by how effectively $\mathcal{M}_t$ provides the missing context needed to construct the summary S_t.

3.2 Benchmark Construction

Streaming dialogue summarization requires utterance-level grounding over long-term dialogue histories, which existing dialogue summarization benchmarks generally lack. Without links between history-dependent utterances in $\mathcal{W}_t$ and resolving evidence in $\mathcal{H}_t$, we cannot assess whether memory is adequate or whether the summary is self-contained. We therefore construct a benchmark with explicit utterance–evidence links over histories up to 160K tokens.

Streaming Dialogue Data. We source 35 long dialogues from chit-chat and workplace domains across three datasets (20 chit-chat from LoCoMo (Maharana et al., 2024) and REALTALK (Lee et al., 2025a); 15 workplace collaborative dialogues from EverMemBench (Hu et al., 2026)). The two domains are selected because workplace dialogues sustain a single topic across sequential agendas, concentrating evidence contiguously in $\mathcal{H}_t$, whereas chit-chat dialogues shift topics frequently, dispersing evidence discontinuously. Each dialogue is segmented into 1,024-token windows, each approximating five minutes of conversation, where $\mathcal{H}_t$ comprises all preceding utterances. Windows with $|\mathcal{H}_t| < 7$K tokens or whose salient utterances are entirely self-contained are excluded. The resulting 1,079 candidates span $\mathcal{H}_t$ lengths of 7K to 160K tokens, ensuring diversity in history length. From these, 900 instances are sampled with domain stratification, reflecting that workplace dialogues are substantially longer. Detailed statistics are provided in Appendix A.1.

Gap Annotation. Evaluating whether a generated summary is contextually self-contained requires two references. *Salient utterances* define the current-window content to preserve, and *contextual gaps* specify the missing historical information needed to make history-dependent utterances self-contained. Together, they serve as ground truth for memory adequacy and summary quality.

A contextual gap is annotated only when incompleteness stems from missing prior context. Following completeness-related error types in prior dialogue summarization studies (Liu et al., 2021; Tang et al., 2022; Zhu et al., 2023; Kirstein et al., 2024), we define three history-resolvable gap types. (i) Referential gaps denote missing antecedents of pronouns, mentions, or discourse references. (ii) Attribute gaps refer to missing states, preferences, or entity properties. (iii) Relational gaps encompass missing causal, logical, or discourse-level relations. Each history-dependent salient utterance receives one representative gap, resolved by one or more evidence utterances from $\mathcal{H}_t$. Appendix A.2 provides definitions and examples.

Salient utterances and contextual gaps are constructed through a five-stage LLM-assisted annotation pipeline, detailed in Appendix A.3. Each stage employs three independent LLMs from distinct model families. Outputs are retained only by majority agreement (Kim et al., 2024b; Lee et al., 2024; Thakur et al., 2025). This design reduces single-family dependence and yields a reproducible reference standard for comparing memory construction methods. Each retained gap records the target salient utterance, gap type, resolving evidence utterances from history and resolution statement. The final benchmark instance pairs the current window and accumulated history with the reference salient utterances and contextual gaps.

Validation Target	Accuracy	IAA
Gap-label validity	0.93	0.41
Gap-evidence linking	0.95	0.65

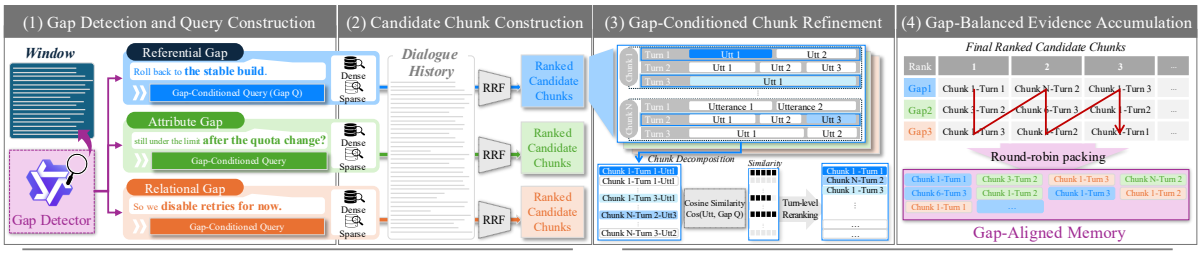
Table 1: Human validation of benchmark annotations, where IAA denotes inter-annotator agreement measured using Krippendorff's α.

Quality Validation. We validate the annotation pipeline on 200 gap labels (100 gap, 100 non-gap) and 200 gap-evidence links (100 resolving, 100 non-resolving), each judged independently by three annotators. As shown in Table 1, the majority human judgment agrees with the LLM pipeline annotations in 93% of gap-label-validity cases and 95% of gap-evidence-linking cases, indicating that the benchmark's core annotations largely reflect independent human judgments. Appendix B.1 provides further details on the evaluation protocol.

3.3 Evaluation Axes

Summary quality alone is insufficient to evaluate streaming dialogue summarization, as it cannot distinguish memory-grounded gap resolution from confabulated context. We therefore evaluate two axes: memory quality via *memory recall* and summary quality measured by *faithfulness*, *conciseness* and *completeness* (Song et al., 2024; Min et al., 2025). To isolate the effect of memory on summarization, completeness is decomposed into *window* completeness and *gap-resolution* completeness. The former targets memory-independent content; the latter targets memory-dependent content, conditioned on successful recovery.

Memory quality is assessed independently of the generated summary, as evidence recovery and summary generation are distinct failure modes. A summary can omit a resolution that memory did supply, and it can assert a resolution that memory never supplied. Measuring only the summary conflates the two. All five metrics are defined in Section 5.1 and formalized in Appendix C.



(a) Gap-Conditioned Evidence Retrieval (b) Gap-Conditioned Chunk Refinement

Figure 2: Overview of ReMEMBER. A gap detector reads only the current window and emits an evidence-seeking query per contextual gap (1); each query retrieves candidate chunks from the dialogue history (2), which are refined into gap-relevant turns (3) and packed across gaps under the memory budget into a gap-aligned memory (4).

4 Methodology: ReMEMBER

The quality of $\mathcal{M}_t$ constrains whether the summarizer receives gap-resolving evidence. Summarization-based memory can dilute such evidence through global compression, while retrieval with window-level queries preserves detail but favors window-overlapping history over gap-resolving evidence. ReMEMBER addresses this mismatch by constructing memory around contextual gaps, retrieving missing evidence with gap-specific queries, and refining retrieved chunks into a compact $\mathcal{M}_t$ under the memory budget (see Figure 2).

4.1 Gap-Conditioned Evidence Retrieval

Retrieval in streaming dialogue summarization suffers from a *history-window mismatch*. A query derived from $\mathcal{W}_t$ surfaces visible window content and retrieves history that repeats or paraphrases it. Yet the needed evidence is the presupposed referent, state, rationale, or causal link, which need not share surface form with any token in $\mathcal{W}_t$. ReMEMBER addresses this mismatch by conditioning retrieval on detected contextual gaps rather than the full $\mathcal{W}_t$.

Gap Detection and Query Construction. ReMEMBER first identifies which utterances are both summary-worthy and gap-bearing, then constructs an evidence query for each such utterance without hypothesizing its resolution. An utterance is *summary-worthy* when it conveys information that should be preserved in a self-contained summary, such as an action, decision, plan, or status update. It is *gap-bearing* when its interpretation depends on prior dialogue, such as an absent referent, rationale, prior state, or causal dependency. These criteria define the retrieval trigger at inference time. They are applied only to $\mathcal{W}_t$, without benchmark annotations, resolving evidence, or gap-type labels.

Detecting such utterances requires discourse-level reasoning beyond lexical overlap (Elgohary et al., 2019; Du et al., 2026). ReMEMBER implements this detection with a compact LLM that reads $\mathcal{W}_t$ alone, so gap detection cost stays constant as $\mathcal{H}_t$ grows. Each detected gap is represented as a structured query $g = (x, \rho, \epsilon, a)$, collected into the gap set $\mathcal{G}_t$, where x is the gap-bearing span, ρ states why x is not self-contained, ϵ is an evidence-seeking request naming what earlier content must be retrieved, and a collects lexical anchors drawn from $\mathcal{W}_t$. This avoids anchoring retrieval to a hypothetical answer that may not exist in $\mathcal{H}_t$, reducing retrieval drift.

Candidate Chunk Construction. Candidate construction follows three design choices. First, ReMEMBER segments $\mathcal{H}_t$ into overlapping chunks $\mathcal{C}_t$ of L tokens. This granularity keeps each chunk small enough to isolate specific evidence while preserving sufficient turn context for retrieval. Second, each gap query g is issued to a sparse and a dense retriever, which capture complementary aspects of relevance: the sparse retriever matches the anchors a, while the dense retriever embeds all of g, so ρ and ϵ contribute signal the anchors alone lack. Each returns its top-N chunks. Third, the two ranked lists are fused with reciprocal rank fusion, $s_g(c) = \frac{1}{\lambda + r^{\text{sparse}}(c)} + \frac{1}{\lambda + r^{\text{dense}}(c)}$, where $r^{\bullet}(c)$ is the rank of chunk c under each retriever and λ is the fusion constant (Cormack et al., 2009). Fusion aggregates ranks without calibrating scores across the two systems. The fused ranking gives the candidate set for gap g:$\mathcal{C}_g = \text{top}_K(s_g, \mathcal{C}_t), K = \lfloor B/L \rfloor$, where B is the memory budget. The retrieval depth K therefore follows from the budget rather than tuning, matching the depth the retrieval baselines pack under the same budget. Section 4.2 refines $\mathcal{C}_g$ into turn-level evidence under B.

4.2 Gap-Conditioned Chunk Refinement

Retrieved chunks introduce a second failure mode, *chunk-evidence mismatch*. A chunk of L tokens

provides sufficient context for retrieval. Yet it inevitably carries gap-irrelevant content that occupies memory space without contributing to gap resolution (Pan et al., 2025; Song et al., 2026). ReMEMBER therefore refines $\mathcal{C}_g$ through two steps, gap-conditioned turn extraction and gap-balanced evidence accumulation.

Gap-Conditioned Turn Extraction. Scoring at the chunk level dilutes the gap-resolution signal, since the required clue is often concentrated in a single utterance. ReMEMBER decomposes each chunk $c_i \in \mathcal{C}_g$ into constituent utterances. Each utterance u and the query g are embedded with the same encoder $e(\cdot)$ used for dense retrieval, and scored as $\mathrm{rel}_g(u) = \cos(e(u), e(g))$. Since g already encodes the specific evidence need that triggered retrieval, this cosine score serves as a sufficient gap-conditioned relevance signal without additional supervision. Utterances are thus re-ranked per gap, pushing gap-irrelevant content down. The complete turn containing each high-scoring utterance is then extracted, including its speaker information, as a turn-level evidence unit. This converts $\mathcal{C}_g$ into turns $\tau_{g,d}$ indexed by decreasing rel_g.

Gap-Balanced Evidence Accumulation. Turns are accumulated across gaps by Pack_B in rank-wise rounds. In round d, Pack_B appends $\tau_{g,d}$ for each $g \in \mathcal{G}_t$, skips turns already selected, and stops once the next turn would exceed B. This yields $\mathcal{M}_t = \{\tau_{g,d} \mid g \in \mathcal{G}_t,\ d = 1, \ldots, D\}$, where D is the last round begun within B; the final round may be partial. Selected turns are then restored to chronological order. Round-robin allocation prevents any single gap from monopolizing $\mathcal{M}_t$, so every unresolved dependency in $\mathcal{W}_t$ retains representation; score-ordered packing would instead starve gaps with sparse or lexically distant evidence. The resulting $\mathcal{M}_t$ is thus not a general compression of $\mathcal{H}_t$ but a gap-aligned collection of turn-level evidence, enabling the summary generator to produce a contextually self-contained summary of $\mathcal{W}_t$ without access to the full $\mathcal{H}_t$.

4.3 Integration with Summary Generation

ReMEMBER is designed as a gap-resolving memory module. The constructed memory $\mathcal{M}_t$ can be integrated with any standard summary generator. In our experiments, generation is held fixed across memory strategies. The generator receives only $\mathcal{M}_t$ and the current window $\mathcal{W}_t$ as input. Implementation details are provided in Section 5.1 and the gap-detection and summary-generation prompts in Figures 4 and 5.

5 Evaluation

In this section, we evaluate whether ReMEMBER constructs gap-resolving memory and improves streaming dialogue summaries under a fixed memory budget. Section 5.1 introduces metrics and baselines. Sections 5.2–5.3 evaluate memory recall and summary quality against memory construction baselines. Sections 5.4–5.7 further analyze recall breakdowns by gap type, component-level contributions, memory budget effects and computational cost.

5.1 Experimental Setup

The experiment isolates the memory construction effect by holding the summarizer fixed across all methods. Each method receives the same window $\mathcal{W}_t$ and constructs $\mathcal{M}_t$ from prior dialogue.

Metrics. Following the axes of Section 3.3, we report one memory metric and four summary metrics; full formulations appear in Appendix C.

Memory quality is measured by *memory recall*, the fraction of annotated gaps whose resolving information appears in $\mathcal{M}_t$. Summary quality is measured by *window completeness*, the fraction of salient window utterances whose core referent and relation are recoverable from the summary; *gap-resolution completeness*, the fraction of memory-resolved gaps whose minimal resolving fact is explicitly expressed in the summary; *conciseness*, the fraction of summary sentences that cover a salient utterance or express a resolved gap; and *faithfulness*, the fraction of atomic facts in the summary verified as factually grounded. We additionally report *composite score*, the arithmetic mean across the four summary dimensions, as an overall measure of summary quality.

All metrics are scored by Qwen3.6-27B (temperature 0.0); formal definitions are provided in Appendix C, and scoring criteria and prompts in Figures 6-11. We further conduct a human meta-evaluation confirming that the LLM-based evaluation is consistent with human judgments across all metrics (Appendix B.2).

Baselines. We compare ReMEMBER against two groups. The first group consists of *diagnostic references* that control memory availability and quality. The second group consists of *memory construction baselines* that build memory under the same capacity constraint.

(i) Diagnostic references isolate how summary quality changes as a function of memory availability and organization, independently of any construction method. `Ideal Memory` supplies the reference gap-resolving evidence as memory, serving as an upper-bound diagnostic for evidence availability. `No Memory` summarizes $\mathcal{W}_t$ alone, measuring the summary quality achievable without any historical context. `Full Memory` supplies the entire $\mathcal{H}_t$ as memory, revealing whether the bottleneck lies in evidence availability or in how evidence is organized for generation. When $\mathcal{H}_t$ exceeds the model context limit, the oldest turns are truncated first.

(ii) Memory construction baselines represent three representative strategies for converting $\mathcal{H}_t$ into a bounded memory. `Recency-based Memory` (`Recent Memory`) retains the most recent turns immediately preceding $\mathcal{W}_t$, under the rationale that temporally proximate turns are most likely to resolve current gaps. `Summarization-based Memory` (`Inc. Summary`, `Hier. Summary`) instead compresses the entire $\mathcal{H}_t$ into a fixed-length memory, covering the full history at the cost of abstractive compression. `Inc. Summary` updates a running summary at each window, while `Hier. Summary` aggregates segment-level summaries bottom-up (prompt in Figure 12). The compression operates over 4,096-token segments with a 1,024-token overlap, applied recursively. `Retrieval-based Memory` (`Sparse`, `Dense`, `Hybrid`) retrieves chunks from $\mathcal{H}_t$ using $\mathcal{W}_t$ as a single composite query via lexical, semantic and hybrid matching, respectively, as window-aware retrieval better captures the evidence signal than compression. Sparse retrieval uses BM25 (Robertson and Zaragoza, 2009) and packs the top-8 chunks to stay within the memory budget; dense retrieval uses Qwen3-Embedding-0.6B (Zhang et al., 2025) and packs the top-8 chunks in the same manner; hybrid retrieval packs the top-4 chunks from each ranking.

Implementation Details. Both the memory capacity B and window size are fixed at 1,024 tokens. This ensures that memory does not dominate the generation context while providing sufficient capacity for historical evidence. Under this setting, `ReMEMBER` segments $\mathcal{H}_t$ into $L = 128$ token chunks with 32-token overlap. Each gap query retrieves the top $N = 30$ chunks per gap from each retriever. The two ranked lists are fused with $\lambda = 60$, following the standard setting for reciprocal rank fusion (Cormack et al., 2009). This yields the top $K = \lfloor B/L \rfloor = 8$ fused chunks per gap as candidates for chunk refinement. Gap detection uses Qwen3.5-4B (Qwen Team, 2026a).[1] BM25 (Robertson and Zaragoza, 2009) and Qwen3-Embedding-0.6B (Zhang et al., 2025) serve as the sparse and dense retrievers, with the latter also used as the utterance encoder $e(\cdot)$. All memory-based methods use the same summary-generation instruction (Figure 5). `No Memory`, which requires no memory context, uses a separate prompt (Figure 13). Summary evaluation results are averaged across three summarizers (Qwen3.5-4B, Qwen3.5-9B and Gemma-4-E2B-it at temperature 0.7). Further details are in Appendix E.

Memory Type	Memory Recall	Referential Gap	Attribute Gap	Relational Gap
Ideal Memory	1.0000	0.3251	0.5421	0.1328
Full Memory	0.8445	0.2410	0.4252	0.0935
Recent Memory	0.4126	0.1756	0.2130	0.0441
Inc. Summary	0.3581	0.1355	0.1983	0.0340
Hier. Summary	0.3212	0.1095	0.2196	0.0214
Sparse	0.5130	0.2029	0.2617	0.0621
Dense	0.4897	0.1969	0.2490	0.0561
Hybrid	0.5412	0.2183	0.2777	0.0614
ReMEMBER	0.6984	0.2457	0.3558	0.1008

Table 2: Memory recall and its breakdown by gap type, where each type-level score is the fraction of gaps included in $\mathcal{M}_t$ out of all annotated gaps of that type. Domain-wise results are in Appendix D.2.

5.2 Gap-Resolving Evidence Recovery

Table 2 reveals **memory recall depends on how evidence is selected, not on how much history is accessed.** `ReMEMBER` achieves a memory recall of 0.6984, outperforming the best memory construction baseline, `Hybrid`, by 0.1572. `Inc. Summary` and `Hier. Summary` access the full history yet fall below even `Recent Memory`. This indicates that indiscriminate compression actively discards task-relevant evidence. `Sparse`, `Dense` and `Hybrid` recover more evidence but remain bounded by surface similarity to the current window, leaving implicitly connected evidence unrecovered regardless of retrieval budget.

5.3 Impact on Streaming Summaries

Table 3 reports summary quality across dialogue length bins. `ReMEMBER` achieves the best composite score among memory construction baselines in

[1] Ablations with Qwen3.5-9B and Gemma-4-E2B-it (DeepMind, 2026) show no consistent gain over the Qwen3.5-4B (Appendix D.1).

Memory Type	Win-Comp			Gap-Comp			Conc			Faith			Composite		
	-32K	32K-64K	64K-	-32K	32K-64K	64K-	-32K	32K-64K	64K-	-32K	32K-64K	64K-	-32K	32K-64K	64K-
Ideal Memory	0.73	0.71	0.70	0.73	0.76	0.77	0.65	0.71	0.73	0.88	0.88	0.90	0.75	0.77	0.77
No Memory	0.70	0.65	0.65	N/A	N/A	N/A	0.61	0.66	0.68	0.92	0.91	0.91	0.55	0.56	0.56
Full Memory	0.39	0.21	0.13	0.45	0.42	0.24	0.31	0.22	0.16	0.76	0.80	0.80	0.48	0.41	0.33
Recent Memory	0.68	0.68	0.67	0.21	0.24	0.40	0.52	0.60	0.63	0.88	0.89	0.90	0.57	0.60	0.65
Inc. Summary	0.72	0.73	0.72	0.27	0.17	0.17	0.54	0.63	0.65	0.74	0.74	0.75	0.57	0.57	0.57
Hier. Summary	0.74	0.73	0.71	0.30	0.17	0.05	0.55	0.61	0.62	0.73	0.71	0.72	0.58	0.56	0.52
Sparse	0.69	0.69	0.70	0.28	0.31	0.43	0.54	0.63	0.67	0.87	0.88	0.89	0.60	0.63	0.67
Dense	0.70	0.66	0.67	0.32	0.32	0.36	0.56	0.62	0.67	0.87	0.87	0.88	0.62	0.62	0.64
Hybrid	0.70	0.68	0.69	0.31	0.31	0.45	0.55	0.63	0.67	0.88	0.90	0.90	0.61	0.63	0.68
ReMEMBER	0.73	0.72	0.72	0.41	0.48	0.47	0.58	0.66	0.68	0.88	0.88	0.89	0.65	0.68	0.69

Table 3: Summary quality across dialogue length bins (-32K, 32K-64K, 64K-). Win-Comp, Gap-Comp, Conc, and Faith denote window completeness, gap-resolution completeness, conciseness, and faithfulness, respectively. For No Memory, the unavailable Gap-Comp is counted as 0 as no historical evidence is supplied. Scores are averaged over three summarizers, and higher scores indicate better performance. Model-wise results are in Appendix D.3.

Memory Type	Referential Gap		Attribute Gap		Relational Gap	
	Mem	Sum	Mem	Sum	Mem	Sum
Ideal Memory	100.0	80.34	100.0	73.73	100.0	72.96
Full Memory	78.80	35.73	83.79	37.11	74.03	21.45
Recent Memory	51.15	37.31	39.10	26.90	33.79	22.71
Inc. Summary	42.24	26.96	35.98	21.01	25.14	12.89
Hier. Summary	33.24	21.21	36.38	19.34	16.02	7.00
Sparse	60.33	42.77	48.2	31.96	45.95	28.58
Dense	58.61	43.82	44.99	30.30	42.17	26.00
Hybrid	64.32	46.34	51.29	34.46	46.13	25.68
ReMEMBER	75.26	52.47	66.15	41.91	76.89	46.90

Table 4: Percentage of annotated gaps per type whose resolving evidence is included in memory (Mem) and reflected in the generated summary (Sum).

Variations	Memory Recall	Gap-Comp	Win-Comp	Conc	Faith
(i) w/o Gap-Conditioned Retrieval	0.60	0.43	0.63	0.62	0.82
(ii) w/o Chunk Refinement	0.66	0.46	0.64	0.66	0.82
ReMEMBER	0.70	0.50	0.68	0.66	0.81

Table 5: Ablation results using Qwen3.5-4B as the gap detection model and summarizer.

every length bin. This advantage is driven mainly by gap-resolution completeness, where ReMEMBER improves over Hybrid by up to 0.17, while maintaining strong window completeness, conciseness and faithfulness. This pattern indicates that the summary gains come from resolving missing contextual dependencies without sacrificing current window coverage or reliability. In contrast, Full Memory exposes substantially more history, but achieves lower window completeness and conciseness than Retrieval-based Memory. The diagnostic references show that **the central challenge is not evidence availability alone, but organizing historical evidence into a form that the summarizer can reliably use.**

5.4 Gap-Level Evidence Use

Table 4 separates evidence inclusion in memory from evidence reflection in the summary. Relational gaps are hardest, requiring dispersed causal or discourse evidence with little surface overlap with $\mathcal{W}_t$. ReMEMBER improves most on this type by reasoning beyond lexical overlap, raising both memory inclusion and summary coverage over memory construction baselines. Full Memory nonetheless buries gap-resolving evidence among irrelevant context across all gap types, producing a needle-in-a-haystack failure (Lee et al., 2025b). Even Ideal Memory does not guarantee full evidence reflection, showing that generation imposes a separate ceiling. **The persistent gap between memory inclusion and summary use suggests that full resolution requires co-design of memory construction and summary generation.**

5.5 Ablation Study

Table 5 isolates the contribution of each component in ReMEMBER. *(i) w/o Gap-Conditioned Retrieval* retrieves with the current window as a single query and keeps the same memory budget; *(ii) w/o Chunk Refinement* stores retrieved chunks without refinement. Removing gap-conditioned retrieval causes the larger drop. Window-level retrieval tends to recover history overlapping with $\mathcal{W}_t$, rather than evidence that resolves its implicit dependencies. Removing chunk refinement also degrades performance, indicating that raw chunks contain gap-irrelevant context that consumes the fixed memory budget. These results show that ReMEMBER improves memory construction by targeting missing evidence needs and densifying retrieved evidence.

Memory Budget	Memory Recall		Gap-Comp	
	Hybrid	ReMEMBER	Hybrid	ReMEMBER
512	0.42	0.54 (+0.12)	0.30	0.38 (+0.08)
1,024	0.54	0.70 (+0.16)	0.38	0.50 (+0.12)
2,048	0.68	0.79 (+0.11)	0.46	0.54 (+0.08)
3,072	0.70	0.82 (+0.12)	0.50	0.56 (+0.06)
4,096	0.76	0.83 (+0.07)	0.51	0.56 (+0.05)

Table 6: Performance under varying memory budgets, where Gap-Comp denotes Gap-Resolution Completeness. Parentheses denote gains over Hybrid.

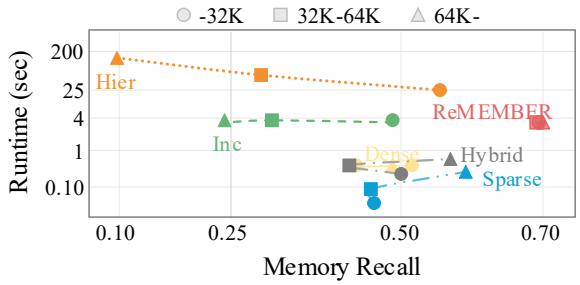


Figure 3: Memory recall versus memory build runtime (log scale, in seconds) across three dialogue length bins.

5.6 Effect of Memory Budget

Table 6 compares ReMEMBER with Hybrid, the strongest baseline in Table 2, across budgets from 512 to 4,096 tokens. **ReMEMBER is not only more accurate but also more budget-efficient, matching Hybrid's recall at roughly one-third of its budget.** The margin remains positive throughout, peaking at our default 1,024-token setting. Gap-resolution completeness, however, shows diminishing returns well before recall does, indicating that the bottleneck shifts from memory construction to the summarizer as budget grows.

5.7 Latency–Memory Recall Trade-off

Figure 3 compares memory recall and memory-construction runtime across three dialogue-length bins.[2] **ReMEMBER consistently achieves the highest memory recall at a construction latency of approximately four seconds, regardless of dialogue length.** As this cost is incurred once per summary request rather than once per turn, it remains practical despite exceeding retrieval-based latency. Hier. Summary, in contrast, incurs the highest runtime of any method while yielding the lowest recall, indicating that repeated compression is both costly and lossy under a fixed memory budget.

[2]We test on a single GPU and CPU thread.

6 Conclusion

We present streaming dialogue summarization as a new task with a benchmark and evaluation metrics. Our analysis shows that existing memory paradigms retrieve evidence aligned with what the current window states, not what it leaves implicit. To address this mismatch, ReMEMBER constructs memory around detected contextual gaps rather than surface similarity. Under the same budget, ReMEMBER improves memory recall and gap-resolution completeness across histories up to 160K tokens. Even with reference gap-resolving evidence, a persistent gap remains between memory recall and summary use, indicating that summary generation independently limits gap resolution.

Limitations

The scope of our claims is limited in two respects. First, our experiments focus on long-form chit-chat and workplace dialogues. These settings contain rich cross-turn dependencies and are well-suited to studying streaming dialogue summarization. However, other settings, such as task-oriented support, educational tutoring, multilingual dialogue, or multimodal interaction, may exhibit different dependency patterns. Extending the benchmark to such settings would further test the generality of the findings. Second, ReMEMBER performs additional inference for gap-conditioned memory construction. This improves memory recall under a fixed budget, but adds construction overhead relative to simple retrieval baselines. The overhead does not grow with dialogue length and is incurred once per summary request rather than once per turn. However, it remains a real cost: the method is most appropriate when improved memory quality is worth the added overhead, and less suited to settings that regenerate summaries every turn at interactive latency.

Ethics Statement

Our human studies were conducted with crowdsourced annotators on Amazon Mechanical Turk, who were compensated at or above the U.S. federal minimum wage. We collect no personally identifiable information from annotators. Further details are provided in Appendix B.

Acknowledgment

This work was supported by Institute of Information & Communications Technology Planning &

Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2025-25464461, AI's Vision of Harmony: A Fair and Transparent Multimodal Agentic Platform for Conflict Mediation). Additionally, this work was supported by the National IT Industry Promotion Agency (NIPA) grant funded by the Korea government (MSIT) (No. RS-2026-25621604).

References

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, and 1 others. 2025. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *ACL*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Yi-Pei Chen, Noriki Nishida, Hideki Nakayama, and Yuji Matsumoto. 2025. Post persona alignment for multi-session dialogue generation. In *EMNLP*.

Gordon V Cormack, Charles LA Clarke, and Stefan Buettcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *SIGIR*.

Google DeepMind. 2026. Gemma 4. `https://huggingface.co/google/gemma-4-31b-it`.

Kaustubh Deshpande, Ved Sirdeshmukh, Johannes Baptist Mols, Lifeng Jin, Ed-Yeremai Hernandez-Cardona, Dean Lee, Jeremy Kritz, Willow E Primack, Summer Yue, and Chen Xing. 2025. Multichallenge: A realistic multi-turn conversation evaluation benchmark challenging to frontier llms. In *ACL*.

Yiming Du, Bingbing Wang, Yang He, Bin Liang, Baojun Wang, Zhongyang Li, Lin Gui, Jeff Z Pan, Ruifeng Xu, and Kam-Fai Wong. 2026. Memguide: intent-driven memory selection for goal-oriented multi-session llm agents. In *AAAI*.

Ahmed Elgohary, Denis Peskov, and Jordan Boyd-Graber. 2019. Can you unpack that? learning to rewrite questions-in-context. In *EMNLP-IJCNLP*.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023a. Precise zero-shot dense retrieval without relevance labels. In *ACL*.

Shen Gao, Xin Cheng, Mingzhe Li, Xiuying Chen, Jinpeng Li, Dongyan Zhao, and Rui Yan. 2023b. Dialogue summarization with static-dynamic structure fusion graph. In *ACL*.

Mohamed Imed Eddine Ghebriout, Gaël Guibon, Ivan Lerner, and Emmanuel Vincent. 2025. Quartz: Qa-based unsupervised abstractive refinement for task-oriented dialogue summarization. In *EMNLP*.

Chuanrui Hu, Tong Li, Xingze Gao, Hongda Chen, Yi Bai, Dannong Xu, Tianwei Lin, Xiaohong Li, Yunyun Han, Jian Pei, and 1 others. 2026. Evaluating long-horizon memory for multi-party collaborative dialogues. In *KDD*.

Eunjeong Hwang, Yichao Zhou, James Bradley Wendt, Beliz Gunel, Nguyen Vo, Jing Xie, and Sandeep Tata. 2024. Enhancing incremental summarization with structured representations. In *EMNLP*.

Qi Jia, Yizhu Liu, Siyu Ren, and Kenny Q Zhu. 2023. Taxonomy of abstractive dialogue summarization: Scenarios, approaches, and future directions. *ACM Computing Surveys*, 56(3):1–38.

Keyan Jin, Yapeng Wang, Leonel Santos, Tao Fang, Xu Yang, Sio Kei Im, and Hugo Gonçalo Oliveira. 2026. Reasoning or not? a comprehensive evaluation of reasoning llms for dialogue summarization. *Expert Systems with Applications*, 299:129831.

Hyuntak Kim and Byung-Hak Kim. 2025. Nexussum: Hierarchical llm agents for long-form narrative summarization. In *ACL*.

Jiho Kim, Woosog Chay, Hyeonji Hwang, Daeun Kyung, Hyunseung Chung, Eunbyeol Cho, Yeonsu Kwon, Yohan Jo, and Edward Choi. 2024a. Dialsim: A dialogue simulator for evaluating long-term multi-party dialogue understanding of conversational agents. *arXiv preprint arXiv:2406.13144*.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. 2024b. Prometheus 2: An open source language model specialized in evaluating other language models. In *EMNLP*.

Frederic Kirstein, Jan Philip Wahle, Bela Gipp, and Terry Ruas. 2025. Cads: A systematic literature review on the challenges of abstractive dialogue summarization. *Journal of Artificial Intelligence Research*, 82:313–365.

Frederic Kirstein, Jan Philip Wahle, Terry Ruas, and Bela Gipp. 2024. What's under the hood: Investigating automatic metrics on meeting summarization. In *EMNLP*.

Dong-Ho Lee, Adyasha Maharana, Jay Pujara, Xiang Ren, and Francesco Barbieri. 2025a. Realtalk: A 21-day real-world dataset for long-term conversation. *arXiv preprint arXiv:2502.13270*.

Yuho Lee, Jiaqi Deng, Nicole Hee-Yeon Kim, Hyangsuk Min, Taewon Yun, Minjeong Ban, Kim Yul, and Hwanjun Song. 2025b. Towards a holistic and automated evaluation framework for multi-level comprehension of llms in book-length contexts. In *EMNLP*.

Yuho Lee, Taewon Yun, Jason Cai, Hang Su, and Hwanjun Song. 2024. Unisumeval: Towards unified, fine-grained, multi-dimensional summarization evaluation for llms. In *EMNLP*.

Daniel Li, Thomas Chen, Albert Tung, and Lydia B Chilton. 2021. Hierarchical summarization for long-form spoken dialog. In *UIST*, pages 582–597.

Hao Li, Chenghao Yang, An Zhang, Yang Deng, Xiang Wang, and Tat-Seng Chua. 2025a. Hello again! llm-powered personalized agent for long-term dialogue. In *NAACL*.

Raymond Li, Chuyuan Li, Gabriel Murray, and Giuseppe Carenini. 2025b. Hierarchical attention adapter for abstractive dialogue summarization. In *NewSum*.

Haitao Lin, Junnan Zhu, Lu Xiang, Feifei Zhai, Yu Zhou, Jiajun Zhang, and Chengqing Zong. 2023. Topic-oriented dialogue summarization. *TASLP*, 31:1797–1810.

Dongshuo Liu, Zhijing Wu, Dandan Song, and He-Yan Huang. 2025. A persona-aware llm-enhanced framework for multi-session personalized dialogue generation. In *ACL*.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024a. Lost in the middle: How language models use long contexts. *TACL*, 12:157–173.

Shengjie Liu, Jing Wu, Jingyuan Bao, Wenyi Wang, Naira Hovakimyan, and Christopher G Healey. 2024b. Towards a robust retrieval-based summarization system. *arXiv preprint arXiv:2403.19889*.

Zhengyuan Liu, Ke Shi, and Nancy Chen. 2021. Coreference-aware dialogue summarization. In *SIGDIAL*.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of llm agents. In *ACL*.

Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. 2021. Generation-augmented retrieval for open-domain question answering. In *ACL-IJCNLP*.

Hyangsuk Min, Yuho Lee, Minjeong Ban, Jiaqi Deng, Nicole Hee-Yeon Kim, Taewon Yun, Hang Su, Jason Cai, and Hwanjun Song. 2025. Towards multi-dimensional evaluation of llm summarization across domains and languages. In *ACL*.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. Factscore: Fine-grained atomic evaluation of factual precision in long form text generation. In *EMNLP*.

Kai Tzu-iunn Ong, Namyoung Kim, Minju Gwak, Hyungjoo Chae, Taeyoon Kwon, Yohan Jo, Seungwon Hwang, Dongha Lee, and Jinyoung Yeo. 2025. Towards lifelong dialogue agents via timeline-based memory management. In *NAACL*.

OpenAI. 2022. tiktoken. https://github.com/openai/tiktoken.

Litu Ou and Mirella Lapata. 2025. Context-aware hierarchical merging for long document summarization. In *ACL*.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Xufang Luo, Hao Cheng, Dongsheng Li, Yuqing Yang, Chin-Yew Lin, H Vicky Zhao, Lili Qiu, and 1 others. 2025. Secom: On memory construction and retrieval for personalized conversational agents. In *ICLR*.

Qwen Team. 2026a. Qwen3.5: Towards native multimodal agents.

Qwen Team. 2026b. Qwen3.6-27B: Flagship-level coding in a 27B dense model.

Mathieu Ravaut, Aixin Sun, Nancy Chen, and Shafiq Joty. 2024. On context utilization in summarization with large language models. In *ACL*.

Stephen Robertson and Hugo Zaragoza. 2009. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc.

Hwanjun Song, Jeonghwan Choi, and Minseok Kim. 2026. Aligning extraction and generation for robust retrieval-augmented generation. In *WSDM*.

Hwanjun Song, Hang Su, Igor Shalyminov, Jason Cai, and Saab Mansour. 2024. Finesure: Fine-grained summarization evaluation using llms. In *ACL*.

Li Sun, Liu He, Shuyue Jia, Yangfan He, and Chenyu You. 2025. Docagent: An agentic framework for multi-modal long-context document understanding. In *EMNLP*.

Zhen Tan, Jun Yan, I-Hung Hsu, Rujun Han, Zifeng Wang, Long Le, Yiwen Song, Yanfei Chen, Hamid Palangi, George Lee, and 1 others. 2025. In prospect and retrospect: Reflective memory management for long-term personalized dialogue agents. In *ACL*.

Xiangru Tang, Arjun Nair, Borui Wang, Bingyao Wang, Jai Desai, Aaron Wade, Haoran Li, Asli Celikyilmaz, Yashar Mehdad, and Dragomir Radev. 2022. Confit: Toward faithful dialogue summarization with linguistically-informed contrastive fine-tuning. In *NAACL*.

GLM Team, Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, Kedong Wang, Lucen Zhong, Mingdao Liu, Rui Lu, Shulin Cao, Xiaohan Zhang, Xuancheng Huang, Yao Wei, and 152 others. 2025. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models. *arXiv preprint arXiv:2508.06471*, arXiv:2508.06471.

Aman Singh Thakur, Kartik Choudhary, Venkat Srinik Ramayapally, Sankaran Vaidyanathan, and Dieuwke Hupkes. 2025. Judging the judges: Evaluating alignment and vulnerabilities in llms-as-judges. In *GEM*.

Bin Wang, Zhengyuan Liu, and Nancy Chen. 2023a. Instructive dialogue summarization with query aggregations. In *EMNLP*.

Bingbing Wang, Jing Li, and Ruifeng Xu. 2026. Proactive memory for ad-hoc recall over streaming dialogues. *arXiv preprint arXiv:2603.04885*.

Liang Wang, Nan Yang, and Furu Wei. 2023b. Query2doc: Query expansion with large language models. In *EMNLP*.

Lifang Wang, Meng Zhao, Hongru Ji, Zejun Jiang, Ronghan Li, Zhongtian Hu, and Xinyu Lu. 2024. Dialogue summarization enhanced response generation for multi-domain task-oriented dialogue systems. *Information Processing & Management*, 61(3):103668.

Qingyue Wang, Yanhe Fu, Yanan Cao, Shuai Wang, Zhiliang Tian, and Liang Ding. 2025. Recursively summarizing enables long-term dialogue memory in large language models. *Neurocomputing*, 639.

Jerry Wei, Chengrun Yang, Xinying Song, Yifeng Lu, Nathan Hu, Jie Huang, Dustin Tran, Daiyi Peng, Ruibo Liu, Da Huang, and 1 others. 2024. Long-form factuality in large language models. In *NIPS*.

Yisha Wu, Cen Zhao, Yuanpei Cao, Xiaoqing Xu, Yashar Mehdad, Mindy Ji, and Claire Na Cheng. 2025. Incremental summarization for customer support via progressive note-taking and agent feedback. In *EMNLP*.

Jianfei Xiao, Yancan Chen, Yimin Ou, Hanyi Yu, Kai Shu, and Yiyong Xiao. 2024. Baichuan2-sum: Instruction finetune baichuan2-7b model for dialogue summarization. In *IJCNN*.

Jing Xu, Arthur Szlam, and Jason Weston. 2022. Beyond goldfish memory: Long-term open-domain conversation. In *ACL*.

Shannan Yan, Jingchen Ni, Leqi Zheng, Jiajun Zhang, Peixi Wu, Dacheng Yin, Jing Lyu, Chun Yuan, and Fengyun Rao. 2026. Adamem: Adaptive user-centric memory for long-horizon dialogue agents. *arXiv preprint arXiv:2603.16496*.

Chengyuan Yang, Zequn Sun, Wei Wei, and Wei Hu. 2026. Beyond static summarization: Proactive memory extraction for llm agents. *arXiv preprint arXiv:2601.04463*.

Zihao Yi, Jiarui Ouyang, Zhe Xu, Yuwen Liu, Tianhao Liao, Haohao Luo, and Ying Shen. 2025. A survey on recent advances in llm-based multi-turn dialogue systems. *ACM Computing Surveys*, 58(6):1–38.

Chanwoong Yoon, Gangwoo Kim, Byeongguk Jeon, Sungdong Kim, Yohan Jo, and Jaewoo Kang. 2025. Ask optimal questions: Aligning large language models with retriever's preference in conversation. In *NAACL*.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, and 1 others. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv:2506.05176*.

Yusen Zhang, Ansong Ni, Tao Yu, Rui Zhang, Chenguang Zhu, Budhaditya Deb, Asli Celikyilmaz, Ahmed Hassan Awadallah, and Dragomir Radev. 2021. An exploratory study on long dialogue summarization: What works and what's next. In *EMNLP*.

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, and 1 others. 2021. Qmsum: A new benchmark for query-based multi-domain meeting summarization. In *NAACL*.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. Memorybank: Enhancing large language models with long-term memory. In *AAAI*.

Changtai Zhu, Siyin Wang, Ruijun Feng, Kai Song, and Xipeng Qiu. 2025a. Convsearch-r1: Enhancing query reformulation for conversational search with reasoning via reinforcement learning. In *EMNLP*.

Chenguang Zhu, Ruochen Xu, Michael Zeng, and Xuedong Huang. 2020. A hierarchical network for abstractive meeting summarization with cross-domain pretraining. In *EMNLP*.

Rongxin Zhu, Jey Han Lau, and Jianzhong Qi. 2025b. Factual dialogue summarization via learning from large language models. In *COLING*.

Rongxin Zhu, Jianzhong Qi, and Jey Han Lau. 2023. Annotating and detecting fine-grained factual errors for dialogue summarization. In *ACL*.

A Benchmark Construction Detail

A.1 Benchmark Statistics

This section provides additional details on the benchmark dataset described in Section 3.2. Table 7 reports the source datasets and benchmark statistics by domain. The benchmark contains 900 window-level instances sampled from 35 source dialogues (10 LoCoMo (Maharana et al., 2024), 10 REALTALK (Lee et al., 2025a) and 15 EverMemBench (Hu et al., 2026)). Chit-chat instances are drawn from LoCoMo and REALTALK, while workplace instances are drawn from EverMemBench. Dialogue length is measured at the benchmark-instance level as $|\mathcal{H}_t| + |\mathcal{W}_t|$, with the current window fixed to 1,024 tokens, rather than as the length of the original source dialogue. Table 13 lists the original dataset repositories used to obtain the raw data before preprocessing.

Domain	Source	# Samples	Avg. Dialogue Length	Referential Gap	Attribute Gap	Relational Gap	Avg. Sal.	Avg. Gap
Chit-Chat	LoCoMo, REALTALK	300	20,446 (8,024-43,608)	205	188	71	4.70	1.55
Workplace	EverMemBench	600	80,574 (8,024-159,576)	282	624	128	9.49	1.73
All	–	900	60,532 (8,024-159,576)	487	812	199	7.90	1.67

Table 7: Benchmark statistics by domain. Dialogue length reports the average number of tokens in each benchmark instance, including both the accumulated history and the current window, with the minimum–maximum range in parentheses. Referential, Attribute, and Relational gaps denote the number of annotated contextual gaps by type. Avg. Sal. and Avg. Gap denote the average number of salient utterances and contextual gaps per instance.

A.2 Gap Taxonomy

This section provides detailed definitions and examples for the contextual gap types introduced in Section 3.2. Building on prior work on faithfulness error types in dialogue summarization (Liu et al., 2021; Tang et al., 2022; Zhu et al., 2023; Kirstein et al., 2024), we reinterpret the underlying discourse phenomena as pre-generation contextual losses in streaming dialogue summarization. Prior work analyzes faithfulness failures after generation, including missing information, incorrect entity or circumstance details and unsupported links in generated summaries. Our setting instead focuses on the corresponding omissions before generation. The current window may omit the antecedent needed to resolve a reference, the attribute needed to specify an entity or event, or the relation needed to understand why an utterance follows. We therefore annotate history-dependent incompleteness as a contextual gap that can be resolved by evidence from $\mathcal{H}_t$, rather than as an error introduced by the summarizer.

We organize contextual gaps into three history-resolvable dependency types according to what is omitted from the current window. Prior context may supply a missing referent, a missing attribute, or a missing relation. These roles correspond to referential, attribute and relational gaps, respectively. Table 8 provides annotated examples across all three types.

• **Referential gaps** arise when an utterance contains an expression whose antecedent or referent is absent from $\mathcal{W}_t$. Such expressions include pronouns, shorthand references, vague noun phrases, demonstratives and discourse references. These gaps ask which entity, event, decision, request, or prior statement the utterance refers to.

• **Attribute gaps** arise when the relevant entity or event is identifiable, but a necessary property, parameter, state, preference, constraint, scope, role, assignment, or specification is absent from $\mathcal{W}_t$. These gaps ask what attribute or specification is needed to make the utterance self-contained.

• **Relational gaps** arise when the current utterance depends on an omitted causal, temporal, logical, or discourse relation. The missing relation may involve prior reasoning, motivation, agreement, comparison, condition, or constraint. These gaps ask why the utterance follows, how it is justified, or how it connects to prior dialogue.

If a salient utterance contains multiple distinct missing dependencies, they are identified separately during annotation. For evaluation, each history-dependent salient utterance is associated with one representative gap and one or more pieces of evidence from $\mathcal{H}_t$ that explicitly resolve the missing referent, attribute, or relation.

A.3 Gap Annotation Pipeline Details

We provide full implementation details for the five-stage annotation pipeline introduced in Section 3.2. Unless noted otherwise, each stage is run with three LLMs from distinct families, Qwen3.6-27B (Qwen Team, 2026b), Gemma-4-31B-it (DeepMind, 2026) and GLM-4.7-Flash (Team et al., 2025), at temperature 0.0. In all stages, a candidate is retained only when at least two of the three LLMs agree; disagreements are resolved by the majority label without further adjudication.

(i) Salient Utterance Identification. We select utterances from $\mathcal{W}_t$ that contain a concrete decision, request, action item, or status update. Simple acknowledgments, greetings, praise and emotional reactions are excluded. When consecutive utterances form a single coherent update or decision, they are merged into one entry.

(ii) Contextual Dependence Judgment. For each salient utterance, we assess whether it meets the standard of self-contained summarizability, defined as the ability to be summarized clearly and stably from $\mathcal{W}_t$ alone without recourse to earlier dialogue. Each utterance is examined for unresolvable pronouns, vague noun phrases, underspecified attributes and unstated reasoning that would prevent

Gap Type	Gap Utterance	Evidence	Gap Reason	Resolution
Referential Gap	During testing, be sure to have the other party simulate a few requests with incorrect signatures and non-standard data formats.	We are officially starting a new development task today to implement a Webhook interface to receive order notifications generated by external partner channels.	The phrase "the other party" cannot be stably resolved from the current utterance alone.	The evidence identifies the relevant counterpart as the external partner channels that generate order notifications.
Attribute Gap	I plan to reuse the distributed lock solution used when creating orders previously to ensure the idempotency of callback processing.	I'm focusing on testing the idempotency logic for payment callbacks, especially the solution you emphasized, which uses Redis distributed locks combined with a unique index for the database payment transaction number.	The target mentions a prior distributed lock solution, but omits its technical specification.	The evidence specifies that the solution combines Redis distributed locks with a unique database index.
Relational Gap	@All members Synchronizing the final decision on the selection logic of emission factors.	Technically, @Ruiqing Jiang, when designing the algorithm, you need to reserve an interface to prioritize user-defined factors, and if none exist, then fall back to our built-in database.	The target announces a final decision, but the decision logic itself is not stated.	The evidence supplies the relation that defines the decision, namely prioritizing user-defined factors before falling back to the built-in database.

Table 8: Examples of contextual gap types. The gap utterance is from the current window $\mathcal{W}_t$, while the evidence is from the prior history $\mathcal{H}_t$.

independent summarization. A gap is assigned only when the missing information is not recoverable from nearby utterances within $\mathcal{W}_t$; additional detail that would merely enrich the summary does not qualify. Each gap is recorded under one of three types following the taxonomy in Appendix A.2: referential, attribute, or relational. An utterance is marked as history-dependent only when at least two LLMs agree on the same gap type. Each history-dependent utterance is assigned at most one representative gap, selected as the dependency whose absence most severely impairs summarizability.

Stages (i) and (ii) are implemented jointly in a single prompt, provided in Figure 14.

(iii) Candidate History Filtering. We screen all chunks in $\mathcal{H}_t$ for relevance to $\mathcal{W}_t$ and retain only those warranting further examination in Stage (iv). Each chunk receives a binary label independently. A chunk is labeled relevant when it contains prior content that is necessary or directly useful for understanding or summarizing $\mathcal{W}_t$, such as a decision or plan explicitly continued in the window, information needed to resolve references, or earlier steps of an ongoing task. Topical similarity alone does not qualify; a chunk is discarded when it is only loosely related or when connecting it to the window requires inference beyond what the dialogue supports. The prompt is provided in Figure 15.

(iv) Evidence and Resolution Annotation. For each history-dependent utterance, we audit every sentence within the retained candidate chunks to identify those that explicitly supply the missing information. Each sentence is evaluated independently through a two-step protocol. The judge first writes a reasoning trace specifying what missing content the sentence provides or fails to provide with respect to the identified gap, then assigns a binary label of *resolve* or *not_resolve* based solely on that trace. Topical overlap, shared entities, discourse proximity and broad background context do not constitute resolution; a sentence qualifies only when it explicitly provides the exact referent, attribute, prior decision, or commitment that the target utterance presupposes. The reasoning traces of all positively labeled sentences are subsequently aggregated into a single resolution statement for the gap. Given the fine-grained reasoning demands of this stage, we run inference with extended thinking enabled on Gemma-4-31B-it; the prompt is provided in Figures 16 and 17.

(v) Annotation Validation. We apply four joint acceptance criteria to remove annotations that passed earlier stages but fail closer scrutiny. The evidence must be sufficient, in that the resolving sentence supplies the exact missing information defined by the gap type. It must be non-inferential, in that no additional guessing or assumption is required to close the gap. It must support independence, in that the target utterance becomes unambiguously self-contained once the resolving information is incorporated. It must be non-redundant, in that the resolving content is meaningfully distinct from what is already present in $\mathcal{W}_t$. This step is specifically designed to catch two recurring failure modes, namely cases in which resolving evidence is in fact present in $\mathcal{W}_t$ but was overlooked in Stage (ii),

and cases in which the resolution is too vague or indirect to constitute a verifiable resolution. The prompt is provided in Figure 18.

B Human Validation

We conduct two human studies to validate our evaluation setup. We first examine whether our LLM-assisted pipeline produces reliable benchmark annotations (Appendix B.1). We then examine whether our LLM-based metrics induce system comparisons consistent with human judgment (Appendix B.2). We also describe the annotator recruitment shared by both studies (Appendix B.3).

B.1 Validation of Benchmark Annotations

We describe the evaluation protocol in detail here; the corresponding results are reported in Section 3.2.

Each study draws a balanced sample of positive and negative cases, so that agreement cannot be inflated by a skewed label distribution. For gap labels, we randomly sample 100 gap-annotated and 100 non-gap-annotated utterances from all salient utterances across the 900 benchmark instances. For gap-evidence links, we pair 100 evidence utterances retained by majority agreement with 100 candidates rejected during the same voting procedure; since the rejected candidates had already passed earlier filtering stages, they serve as hard negatives rather than trivially unrelated controls.

Annotators judge each item from the information relevant to the corresponding pipeline decision. For gap labels, they see only the current window and decide whether the target utterance is interpretable from that window alone. For gap-evidence links, they additionally see the annotated gap and its description alongside the candidate historical evidence, and decide whether that evidence supplies the missing information. Three annotators judge each item independently. We report accuracy, the proportion of items whose majority human judgment matches the benchmark annotation, together with Krippendorff's α over the annotators' binary judgments as inter-annotator agreement.

B.2 Human Meta-Evaluation of LLM-based Metrics

We provide the evaluation results and protocol for our LLM-based metrics.

Evaluation Results. Table 9 reports LLM–Human Agreement, the proportion of pairwise comparisons on which the outcome implied by the metric scores coincides with the majority human judgment. Averaged over the five metrics, the LLM-based evaluation agrees with the majority human judgment on 83% of comparisons, with a mean inter-annotator agreement of 0.61. Agreement is highest for Faithfulness at 0.93, where the judgment reduces to checking whether each claim is supported, while the remaining four metrics fall between 0.80 and 0.82. Conciseness shows the weakest inter-annotator agreement at 0.39, indicating that annotators themselves diverge on how much detail counts as redundant. Across all five metrics, the comparison outcomes remain consistent with human evaluators, supporting the reliability of the system comparisons reported in the main results.

Evaluation Protocol. Since absolute scores are hard to calibrate across annotators, we validate the metrics through pairwise comparison. For each metric, we collect 300 comparisons between `ReMEMBER` and the seven systems (`No Memory` and the six memory construction baselines; the diagnostic references `Ideal Memory` and `Full Memory` are excluded as they are not standalone construction methods), evenly distributed across baselines and dialogue-history length ranges, yielding 1,500 comparisons across the five metrics in total. Memory Recall excludes `No Memory`, which constructs no memory to evaluate, leaving six systems.

Each comparison presents the two outputs anonymized and in randomized order. Annotators receive the metric definition given in Section 3.3 and the same inputs as the LLM judge. Three annotators independently choose the winner or mark a tie, and their majority vote becomes the human outcome. The LLM outcome follows the metric scores, where the higher-scoring system wins and equal scores yield a tie. LLM–Human Agreement is the proportion of comparisons on which the two outcomes match and inter-annotator agreement is measured by Krippendorff's α.

B.3 Annotator Recruitment

We conduct both studies on Amazon Mechanical Turk with U.S.-based annotators who have at least 100 approved HITs and an approval rate of at least 95%. We include one or two attention checks in each HIT, and exclude and replace responses that fail a check. Annotators are compensated at or above the U.S. federal minimum wage.

Metric	Agreement	IAA
Window Completeness	0.80	0.63
Gap-Resolution Completeness	0.82	0.71
Faithfulness	0.93	0.68
Conciseness	0.80	0.39
Memory Recall	0.80	0.62
Average	0.83	0.61

Table 9: Human validation of the LLM-based evaluation. Agreement and IAA denote LLM–Human Agreement and inter-annotator agreement.

C Evaluation Metric Formulations

In this section, we formalize the five metrics motivated in Section 3.3 and instantiated in Section 5.1. All metric labels are produced by Qwen3.6-27B at temperature 0.0.

Memory Recall. Let $\mathcal{G}_t = \{g_1, \ldots, g_{|\mathcal{G}_t|}\}$ denote the set of annotated contextual gaps for step t, and let $\mathcal{G}_t^+ \subseteq \mathcal{G}_t$ denote the subset for which $\mathcal{M}_t$ explicitly contains the resolving information, verified either verbatim or as a close semantic paraphrase of the supporting evidence. Memory recall is defined as

$$\mathrm{MemRecall}(\mathcal{M}_t, \mathcal{G}_t) = \frac{|\mathcal{G}_t^+|}{|\mathcal{G}_t|}.$$

This metric measures how completely the memory module preserves the historical evidence required for self-contained summarization, independently of the generated summary. The prompt is provided in Figure 6.

Window Completeness. Let $\mathcal{U}_t^{\mathrm{sal}} = \{u_1, \ldots, u_{|\mathcal{U}_t^{\mathrm{sal}}|}\}$ denote the set of salient utterances in $\mathcal{W}_t$, and let $\mathcal{U}_t^+ \subseteq \mathcal{U}_t^{\mathrm{sal}}$ denote the subset whose primary claim, comprising both the core referent and the essential relation, is recoverable from S_t. Window completeness is defined as

$$\mathrm{WinComp}(\mathcal{U}_t^{\mathrm{sal}}, S_t) = \frac{|\mathcal{U}_t^+|}{|\mathcal{U}_t^{\mathrm{sal}}|}.$$

Narrative coverage of the same topic without recovering the specific referent and relation does not qualify. See Figure 7 for the prompt.

Gap-Resolution Completeness. Let $\mathcal{G}_t^{++} \subseteq \mathcal{G}_t^+$ denote the subset of memory-resolved gaps whose minimal gap-resolving fact, including both the specific referent and the essential relation, is explicitly expressed in S_t. Gap-resolution completeness is defined as

$$\mathrm{GapComp}(\mathcal{G}_t^+, S_t) = \frac{|\mathcal{G}_t^{++}|}{|\mathcal{G}_t^+|}.$$

This metric measures how well the model utilizes recovered evidence in generation, conditioned on successful memory retrieval. The prompt is provided in Figures 8 and 9.

Conciseness. Let $\mathcal{S}_t^{\mathrm{win}} \subseteq S_t$ denote the set of summary sentences that cover at least one salient utterance, as determined by $\mathcal{U}_t^+$, and let $\mathcal{S}_t^{\mathrm{gap}} \subseteq S_t$ denote the set of summary sentences that explicitly recover at least one gap-resolving fact, as determined by $\mathcal{G}_t^{++}$. Conciseness is defined as

$$\mathrm{Conciseness}(S_t) = \frac{|\mathcal{S}_t^{\mathrm{win}} \cup \mathcal{S}_t^{\mathrm{gap}}|}{|S_t|}.$$

This metric measures the proportion of summary sentences that serve a demonstrable communicative function, penalizing content that neither reflects salient window utterances nor expresses resolved contextual gaps. Conciseness shares the evaluation outputs of window completeness and gap-resolution completeness and requires no additional model call.

Faithfulness. Let $S_t = \{s_1, \ldots, s_N\}$ be the generated summary of N sentences. Unlike completeness and conciseness, which are measured at the sentence level, faithfulness is evaluated at the atomic fact level. Summaries of $\mathcal{W}_t$ that incorporate resolved contextual gaps tend to produce long sentences that conflate multiple verifiable claims; sentence-level verification is therefore too coarse to detect partial hallucinations within a single sentence. Following Min et al. (2023) and Wei et al. (2024), each sentence s_n is first decomposed into a set of atomic facts $\mathcal{F}_n = \{f_{n,1}, \ldots, f_{n,|\mathcal{F}_n|}\}$, each expressing a single self-contained verifiable claim. Each atomic fact is then verified following the fact-checking protocol of Song et al. (2024), extended to support multi-section dialogue transcripts. $\mathcal{F}_n^* \subseteq \mathcal{F}_n$ denotes the subset verified as factually grounded, and faithfulness is defined as

$$\mathrm{Faithfulness}(S_t) = \frac{\sum_{n=1}^{N} |\mathcal{F}_n^*|}{\sum_{n=1}^{N} |\mathcal{F}_n|}.$$

The verification source differs by method type. For memory-based methods, each atomic fact is verified against $\mathcal{M}_t \cup \mathcal{W}_t$. For full-context methods, $\mathcal{H}_t$ is too long to serve directly as a grounding document; we therefore apply sparse retrieval (Robertson and Zaragoza, 2009) over $\mathcal{H}_t$ to select the top-50 chunks most relevant to each atomic fact, and verify against this retrieved set using the

Model	Chit-chat		Workplace	
	MR	Gap-Comp	MR	Gap-Comp
Qwen3.5-9B	0.6810	0.4006	0.6322	0.4581
Gemma-4-E2B-it	0.6509	0.4041	0.6361	0.4652
Qwen3.5-4B	0.6918	0.4559	0.7018	0.5165

Table 10: Gap detection model variation. Domain-wise memory recall and Gap-Comp scores are reported. Summaries are generated with Qwen3.5-4B across all configurations. MR denotes Memory Recall, and Gap-Comp denotes Gap-Resolution Completeness.

Memory Type	Chit-Chat	Workplace
Recent Memory	0.3019	0.4679
Inc. Summary	0.4859	0.2942
Hier. Summary	0.5519	0.2059
Sparse	0.4057	0.5666
Dense	0.4983	0.4854
Hybrid	0.4798	0.5719
ReMEMBER	0.6918	0.7018

Table 11: Domain-wise Memory Recall scores across memory construction methods, reported separately for Chit-Chat and Workplace domains.

same prompt. The decomposition and verification prompts are provided in Figures 10 and 11.

Composite Score. Composite score is the arithmetic mean of window completeness, gap-resolution completeness, conciseness and faithfulness:

$$\text{Composite}(S_t) = \tfrac{1}{4}\big(\text{WinComp} + \text{GapComp} + \text{Conciseness} + \text{Faithfulness}\big).$$

Memory recall is excluded, as it evaluates $\mathcal{M}_t$ rather than S_t.

D Additional Analysis of ReMEMBER

We provide additional analyses of ReMEMBER beyond the main results. We examine whether the gains are consistent across different summarization models and dialogue domains, whether memory recall improvements hold separately in chit-chat and workplace settings, and whether gap detection depends on the choice of compact LLM. Together, these analyses test whether the benefits of gap-conditioned memory construction are robust to generator choice, domain characteristics and the model used for detecting unresolved contextual gaps.

D.1 Gap Detection Model Ablation

Table 10 reports the effect of varying the gap detection model across two domains. The 4B detector matches or exceeds both larger alternatives on every metric, indicating that gap detection does not require model scale in our setting; we therefore use Qwen3.5-4B throughout.

D.2 Memory Recall Across Domains

Table 11 reports Memory Recall separately for the chit-chat and workplace domains. Across both domains, ReMEMBER achieves the highest recall among memory construction baselines, reaching 0.6918 in chit-chat and 0.7018 in workplace. This indicates that gap-conditioned memory construction improves evidence recovery across both dialogue settings. The domain-wise results show different baseline behavior. In chit-chat, summarization-based memory is relatively strong, with `Hier. Summary` reaching 0.5519 recall and `Inc. Summary` reaching 0.4859. This is likely because chit-chat dialogues are shorter on average, making compression less lossy than in longer workplace histories. In workplace dialogues, summarization-based memory degrades sharply, while retrieval-based memory is stronger, with `Hybrid` reaching 0.5719 recall. This suggests that as dialogue histories become longer, repeated compression loses fine-grained evidence, whereas retrieval better preserves localized historical evidence. Despite these domain differences, ReMEMBER consistently improves over the strongest baseline in each domain. It improves over `Hier. Summary` by 0.1399 in chit-chat and over `Hybrid` by 0.1299 in workplace. These results show that explicitly querying unresolved contextual gaps recovers evidence that both compression-based and window-level retrieval baselines miss.

D.3 Performance Across Summarizers and Domains

Table 12 reports summary quality across three summarization models and two dialogue domains. Across all summarizers, ReMEMBER outperforms the memory construction baselines in both domains, achieving the highest composite score in every setting, indicating the best overall balance among window completeness, gap-resolution completeness, conciseness and faithfulness. The gains are mainly driven by gap-resolution completeness. ReMEMBER consistently improves Gap-Comp over recency-, summarization- and retrieval-based memory, showing that the recovered evidence is not only present in memory but also usable by different downstream summarizers.

The results also show that the benefit of ReMEMBER is not tied to a particular generator. With Qwen3.5-4B (Qwen Team, 2026a),

Summ. Model	Memory Type	Chit-Chat					Workplace				
		Win-Comp	Gap-Comp	Conc	Faith	Composite	Win-Comp	Gap-Comp	Conc	Faith	Composite
Qwen3.5-4B	Ideal Memory	0.7185	0.8141	0.7032	0.7857	0.7554	0.6514	0.8375	0.7576	0.8350	0.7712
	No Memory	0.7186	N/A	0.6433	0.8707	0.5582	0.6517	N/A	0.6900	0.8819	0.5559
	Full Memory	0.3135	0.4566	0.2691	0.6607	0.4250	0.1811	0.3378	0.2124	0.8076	0.3843
	Recent Memory	0.6280	0.1989	0.5076	0.7895	0.5310	0.6334	0.3526	0.6093	0.8460	0.6103
	Inc. Summary	0.6551	0.2681	0.5084	0.6568	0.5221	0.6789	0.2093	0.6391	0.6721	0.5499
	Hier. Summary	0.6989	0.3321	0.5307	0.6381	0.5499	0.6599	0.1367	0.6168	0.6124	0.5064
	Sparse	0.6482	0.2500	0.5374	0.7734	0.5518	0.6464	0.4201	0.6606	0.8189	0.6363
	Dense	0.6527	0.3033	0.5471	0.7790	0.5705	0.6182	0.3634	0.6604	0.8002	0.6106
	Hybrid	0.6404	0.2884	0.5494	0.8099	0.5720	0.6331	0.4163	0.6629	0.8569	0.6423
	ReMEMBER	0.6913	0.4559	0.6038	0.7853	0.6341	0.6684	0.5165	0.6900	0.8255	0.6751
Qwen3.5-9B	Ideal Memory	0.7521	0.7773	0.7322	0.8685	0.7825	0.7004	0.7877	0.8066	0.8830	0.7946
	No Memory	0.7033	N/A	0.6889	0.9043	0.5741	0.6408	N/A	0.7306	0.8963	0.5669
	Full Memory	0.4421	0.4429	0.4063	0.7071	0.4996	0.2176	0.3259	0.2968	0.7674	0.4019
	Recent Memory	0.7147	0.1937	0.6013	0.8668	0.5941	0.6784	0.3382	0.7252	0.8831	0.6562
	Inc. Summary	0.7451	0.3076	0.6436	0.7072	0.6009	0.7085	0.1939	0.7378	0.6969	0.5843
	Hier. Summary	0.7810	0.3092	0.6461	0.6968	0.6083	0.7016	0.1229	0.7125	0.6674	0.5511
	Sparse	0.7373	0.2695	0.6358	0.8621	0.6262	0.6892	0.4086	0.7571	0.8774	0.6831
	Dense	0.7508	0.3401	0.6678	0.8812	0.6600	0.6777	0.3583	0.7399	0.8750	0.6627
	Hybrid	0.7439	0.3192	0.6277	0.8600	0.6377	0.6944	0.3986	0.7614	0.8890	0.6859
	ReMEMBER	0.7501	0.4122	0.6757	0.8624	0.6751	0.7030	0.4762	0.7717	0.8857	0.7091
Gemma-4-E2B-it	Ideal Memory	0.7845	0.6078	0.4653	0.9419	0.6999	0.7381	0.6654	0.5891	0.9682	0.7402
	No Memory	0.7265	N/A	0.4813	0.9521	0.5400	0.6413	N/A	0.5752	0.9741	0.5476
	Full Memory	0.5029	0.4002	0.2322	0.8215	0.4892	0.1178	0.2918	0.0872	0.8492	0.3365
	Recent Memory	0.7179	0.1609	0.3615	0.9387	0.5445	0.6858	0.3278	0.5235	0.9673	0.6261
	Inc. Summary	0.8031	0.2236	0.3870	0.8238	0.5593	0.7704	0.1569	0.5400	0.8810	0.5871
	Hier. Summary	0.8103	0.2286	0.3908	0.8313	0.5653	0.7584	0.1012	0.5266	0.8847	0.5677
	Sparse	0.7478	0.1998	0.3846	0.9419	0.5689	0.7094	0.3676	0.5536	0.9707	0.6503
	Dense	0.7555	0.2806	0.3992	0.9415	0.5942	0.6887	0.3360	0.5506	0.9657	0.6353
	Hybrid	0.7566	0.2394	0.4076	0.9381	0.5854	0.7079	0.3986	0.5512	0.9689	0.6566
	ReMEMBER	0.7854	0.3133	0.4089	0.9401	0.6119	0.7529	0.4230	0.5632	0.9681	0.6768

Table 12: Summary quality by domain and summarization model across memory construction methods. Win-Comp, Gap-Comp, Conc, Faith, and Composite denote window completeness, gap-resolution completeness, conciseness, faithfulness and their mean, respectively.

Type	Name / Usage	Source / ID	Params	Hardware	VRAM	Prec.	Avg. GPU h / call	Avg. sec. / call
LLM	Qwen3.5-4B	Qwen/Qwen3.5-4B	4B	NVIDIA H200 ×1	30GB	FP16	0.000436	1.57
	Qwen3.5-9B	Qwen/Qwen3.5-9B	9B	NVIDIA H200 ×1	60GB	FP16	0.000392	1.41
	Qwen3.6-27B	Qwen/Qwen3.6-27B	27B	NVIDIA H200 ×1	90GB	FP16	0.001060	3.81
	Gemma-4-E2B-it	google/gemma-4-E2B-it	5B	NVIDIA H200 ×1	30GB	FP16	0.000387	1.39
	Gemma-4-31B-it	google/gemma-4-31B-it	31B	NVIDIA H200 ×1	90GB	FP16	0.000458 (0.02000)	1.65 (72.00)
	GLM-4.7-Flash	zai-org/GLM-4.7-Flash	30B	NVIDIA H200 ×1	90GB	FP16	0.000162	0.58
Embed.	Qwen3-Embedding-0.6B	Qwen/Qwen3-Embedding-0.6B	0.6B	NVIDIA H200 ×1	3GB	FP16	–	–
Preproc.	Sentence segmentation	NLTK punkt (Bird et al., 2009)	–	–	–	–	–	–
	Token counting	tiktoken cl100k_base	–	–	–	–	–	–
Datasets	EverMemBench	EverMind-AI/EverMemBench-Dynamic	–	–	–	–	–	–
	LoCoMo	snap-research/locomo	–	–	–	–	–	–
	REALTALK	danny911kr/REALTALK	–	–	–	–	–	–

Table 13: Model, embedding, preprocessing, and dataset implementation details. All model runs use a single NVIDIA H200 GPU. VRAM denotes the approximate peak allocated GPU memory. GPU hours and seconds report the average cost per model call, computed over all calls made with each model. Parenthesized values for Gemma-4-31B-it indicate thinking-mode runs.

Qwen3.5-9B and Gemma-4-E2B-it (DeepMind, 2026), ReMEMBER yields the strongest overall performance across both domains while maintaining competitive window completeness and faithfulness. Retrieval-based methods (Sparse, Dense, Hybrid) often preserve faithfulness, but their Gap-Comp remains lower because window-level retrieval misses evidence required to resolve underspecified current-window utterances. Summarization-based memory (Inc. Summary, Hier. Summary) sometimes improves window completeness, but its Gap-Comp remains limited, indicating that repeated compression discards fine-grained historical evidence.

The diagnostic references further clarify the bottleneck. No Memory preserves high faithfulness but cannot resolve history-dependent gaps, while Full Memory often lowers window completeness and conciseness, especially in the workplace domain.

`Ideal Memory` provides a reference-evidence upper bound. Its gap-resolution scores remain below perfect, indicating that even when relevant evidence is supplied, summary generation can still fail to fully express the resolved information. Overall, these results suggest that gap-conditioned memory construction improves overall summary quality consistently across summarization models and dialogue domains.

E Model and Hardware Details

Table 13 summarizes the implementation resources used in our experiments, including model and embedding identifiers, dataset sources, hardware, precision, peak allocated VRAM and measured average runtime cost. We use Qwen3.5-4B (Qwen Team, 2026a), Qwen3.5-9B (Qwen Team, 2026a) and Gemma-4-E2B-it (DeepMind, 2026) for gap detection and summary generation. We use Qwen3.6-27B (Qwen Team, 2026b), Gemma-4-31B-it (DeepMind, 2026) and GLM-4.7-Flash (Team et al., 2025) for benchmark construction and evaluation. For Gemma-4-31B-it, we report non-thinking mode by default, with thinking-mode costs shown in parentheses. We use NLTK (Bird et al., 2009) `punkt` for sentence segmentation and `tiktoken` (OpenAI, 2022) with the `cl100k_base` encoding for approximate token counting when constructing windows, chunks and memory budgets. All model runs use a single NVIDIA H200 GPU. All results are reported from a single experimental run.

F Scientific Artifacts

All experiments use existing research datasets and open-weight language models. The model checkpoints are publicly accessible through Hugging Face. Benchmark statistics and implementation details are provided in Appendix A.1 and Appendix E.

G Use of Generative AI

We used AI assistants for coding assistance (Codex) and for grammar checking (Claude) during the preparation of this paper.

Prompt: Gap Detection in ReMEMBER

You are building retrieval queries for long-dialogue memory.

Input is ONLY the current conversation window. Earlier history is hidden.

Task:
Find current-window utterances that a concise summary would mention, but that cannot be stated faithfully without earlier dialogue.
Do not answer the missing context. Convert each missing slot into a retrieval need for earlier dialogue.

Select an utterance when:

- it is a concrete action, plan, decision, result, request, question, issue, artifact, task, review, test, fix, status update, owner assignment, or personal update;
- it would likely appear in a summary of the current window;
- a specific referent, artifact, owner, project, scope, version, rationale, requirement, decision, result, deadline, or prior state is missing from the current window;
- the missing context is likely stated in earlier dialogue.

Do not select greetings, thanks, small talk, generic reactions, self-contained utterances, image-only references, or merely nice-to-know details.

For each gap:

- quote the exact current-window utterance or the shortest complete span;
- gap_reason must name the missing slot and why the quote is not self-contained;
- evidence_need_query starts with "retrieve earlier dialogue ..." and uses only visible words from the current window plus the missing-slot type;
- keywords are stable anchor terms copied from the quote or surrounding topic;
- never guess the answer.

Prefer recall for real summary-worthy gaps. If several related utterances refer to the same missing context, select the most informative one.

Output up to `{max_gaps}` gaps. Return compact JSON only:

```
{
  "gaps": [
    {
      "utterance_index": 0,
      "quote": "exact quote, <= 220 chars",
      "gap_reason": "missing slot and why earlier context is needed, <= 220 chars",
      "evidence_need_query": "retrieve earlier dialogue identifying/stating/explaining ...",
      "keywords": ["up", "to", "8", "anchors"]
    }
  ]
}
```

Positive examples:

Utterance: "Please help review it again to see if the logic is sound."
Output gap_reason: "The artifact referred to by 'it' and the logic being reviewed are not identified in the current window."
Output evidence_need_query: "retrieve earlier dialogue identifying the artifact to review and the logic that should be checked"

Utterance: "Received, I will immediately arrange test cases, and formal testing will begin tomorrow."
Output gap_reason: "The feature or task being tested is not identified in the current window."
Output evidence_need_query: "retrieve earlier dialogue identifying the feature or task for the test cases and formal testing"

Utterance: "This confirms we should not change the current allocation."
Output gap_reason: "The allocation and confirming rationale are not stated in the current window."
Output evidence_need_query: "retrieve earlier dialogue stating the current allocation and why it should remain unchanged"

Negative examples:

Utterance: "Thanks, that sounds great!"
Do not output: not summary-worthy and no required earlier evidence.

Utterance: "The API documentation was updated today."
Do not output if the current window already identifies which API or the summary can faithfully state the update without earlier context.

Current window:

`{window}`

Figure 4: Prompt template for gap detection in ReMEMBER.

Prompt: Memory-Augmented Dialogue Summary Generation

Your task is to generate a 'Reinforced Dialogue Summary' by synthesizing current [Window] with [Memory] in a fact-centered manner.

Instructions:

1. Fully understand [Window] then carefully read [Memory].
2. Map all entities in [Window] to their specific Identity, Role, or Context defined in [Memory]. Replace all vague terms with these precise identifiers.
3. Enrich the content of [Window] by integrating missing context such as coreference, causality, logical continuity, and technical details from [Memory] where applicable.
4. Generate a concise, high-density, and contextually reinforced summary **centered on the content in [Window]**. The result should be a cohesive narrative that reflects the core insights and the logical evolution of the situation.

Provide your output as JSON. No additional explanation.

```
{{"summary": "Your high-density, contextually reinforced summary here."}}
```

[Window]

```
{window}
```

[Memory]

```
{memory}
```

Figure 5: Prompt template for memory-augmented dialogue summary generation.

Prompt: Memory Recall Evaluation

You are a strict memory coverage evaluator.
Your task is to judge whether MEMORY explicitly contains the information needed to resolve each GAP.
For this task, the information needed to resolve a GAP is already defined in the GAP ITEM. You must not redefine it.

- supporting_sentences provide the information that resolves the GAP.
- supporting_reasoning explains why that information resolves the GAP.

Do NOT decide that the GAP needs additional information beyond the provided supporting_sentences and supporting_reasoning. Your job is not to redesign the gap resolution. Your job is only to check whether MEMORY contains that key information explicitly, either verbatim or as a close semantic paraphrase.

For each GAP:

1. Identify the key information expressed by supporting_sentences, using supporting_reasoning only to understand why that information resolves the GAP.
2. Check whether that same key information is present in MEMORY.
3. Decide 1 if MEMORY contains that information explicitly, either verbatim or as a close semantic paraphrase.
4. Decide 0 only if MEMORY does not contain the key information from the supporting_sentences, contains only broad topical overlap, or contradicts it.

Important:

- The coverage criterion is fixed by supporting_sentences and supporting_reasoning, not by your own idea of what else might be useful to resolve the salient_utterance.
- Do NOT introduce new required information.
- Do NOT mark 0 because some additional background, definition, detail, or rationale would be helpful if it is not part of supporting_sentences or supporting_reasoning.
- Use DIALOGUE only to understand the salient_utterance context. Do NOT derive additional required information from DIALOGUE.
- The final decision must be based only on MEMORY, but the target information to look for is defined by supporting_sentences and supporting_reasoning.
- MEMORY does not need to quote supporting_sentences verbatim. A close semantic paraphrase is enough, but broad topical similarity is not enough.
- Do not use outside knowledge or common sense.
- If MEMORY contains the key information from supporting_sentences, choose 1.

Gap-type guide:

- Coreference (C): MEMORY must contain the information that identifies the referent.
- Attribute (A): MEMORY must contain the information for the missing attribute, state, role, scope, constraint, or specification.
- Logic (L): MEMORY must contain the information for the rationale, causal link, motivation, dependency, or justification.

Return JSONL only:

```
{{
  "gap_id": "...",
  "gap_type": "C/A/L",
  "target_information": [
    "Atomic key information from supporting_sentences, interpreted with supporting_reasoning"
  ],
  "evidence_from_memory": [
    {{
      "target_item": "...",
      "memory_index": "M# or null",
      "text": "Exact or semantically equivalent supporting span from MEMORY"
    }}
  ],
  "missing_target_information": [
    "Key target information not found in MEMORY"
  ],
  "decision": "1 or 0",
  "reasoning": "One concise sentence explaining whether MEMORY contains the target information."
}}
```

[DIALOGUE]
`{window}`

[MEMORY]
`{memory}`

[GAP ITEMS]
`{gaps}`

Figure 6: Prompt template for memory recall evaluation.

Prompt: Window Completeness Evaluation

For each salient utterance (U#), follow these steps:

1. Identify the single most important claim in the utterance. Decompose it into two parts:
   (a) the core referent — the specific entity, event, decision, or quantity at the center of the claim;
   (b) the essential relation — the role this referent plays (who did what, when, why, or to whom) that makes the utterance salient.
2. Decide whether a reader who sees only the summary could recover the claim with both (a) the correct referent and (b) the correct relation.
   - Paraphrase is acceptable for the referent (synonyms, role-based references, alternate phrasings) as long as it uniquely identifies the same thing in (a).
   - Narrative coverage of the same topic, project, or domain — without naming the specific referent or stating the specific relation — does NOT count. Score 0.0 in that case.
   - A summary that mentions the referent but only in a different context (different action, different time, different participants) does NOT count for that relation.
   - Do not use outside knowledge or commonsense inference. The claim must be directly recoverable from the summary text.

Scores (binary):

- 1.0: The summary states both (a) the specific referent and (b) the specific relation. A reader recovers the complete claim from the summary alone.
- 0.0: Anything else — the referent is missing or only described by a generic role; the relation is generalized, vague, or missing; the summary only mentions the broader topic/narrative; the summary contradicts the claim; or the claim is only implied through commonsense. Partial coverage does NOT earn credit.

Output JSON only. No explanation.

```
[
  {{
    "utterance_num": "U0",
    "score": 1.0,
    "line_number": ["S1", "S3"]
  }},
  ...
]
```

{N} Summary Sentences (S#):

`{summary}`

{M} Salient Utterances (U#):

`{salient_utterances}`

Figure 7: Prompt template for window completeness evaluation.

Prompt: Gap-Resolution Completeness Evaluation, Part I

For each gap case (G#), evaluate whether the summary recovers the missing context.

Goal:
Reward the summary only if it contains the specific missing context from relevant_chunk that is absent from salient_utterance. Do NOT reward the summary for merely covering the same event, topic, relationship, preference, plan, project, document, task, or broad narrative.

For each gap case (G#), follow these steps:

1. Identify the exact gap in the salient_utterance.
   Write one concrete question Q that asks for the missing antecedent, referent, source, target, object, decision, time, quantity, or relation. Q must be answerable only by using relevant_chunk, not by using salient_utterance alone.
2. Select exactly one gap-resolving fact F from relevant_chunk.
   F must be the minimal fact that directly answers Q. F must satisfy ALL of the following:
   - (a) Directness: F resolves the exact missing information in salient_utterance. Do not choose a related fact just because it appears in the summary.
   - (b) Core referent: F includes the specific entity, event, decision, document, person, date, quantity, or object that answers Q.
   - (c) Essential relation: F includes the role this referent plays (who did what, when, why, to whom, about what, or based on what).
   - (d) Novelty: F is not already lexically or semantically present in salient_utterance. If the summary could be written using only salient_utterance without reading relevant_chunk, score 0.0.
   - (e) Minimality: F should not be a broad summary of relevant_chunk. It should be the smallest fact needed to fill the gap.

   If relevant_chunk contains multiple possible facts, choose the one that most directly fills the missing slot in salient_utterance. Do not choose an alternative fact merely because the summary mentions it. Do not use "OR" facts. Pick one F.
3. Decide whether the summary alone contains F.
   The reader sees only the summary, not relevant_chunk and not salient_utterance.

Apply ALL tests. Any single failure -> score 0.0.

T1. Exact gap coverage:
Does the summary answer Q with the same F selected from relevant_chunk?

T2. Referent + relation:
Does at least one cited summary sentence explicitly contain both the core referent and the essential relation? If the summary mentions only the referent, or only a related action, score 0.0.

T3. Fidelity:
The summary's answer to Q must match relevant_chunk. If it gives a different plausible answer, different participant, different object, different time, or different relation, score 0.0.

T4. Anti-echo:
If the summary only restates the salient_utterance's topic, category, or already-known referent without adding F from relevant_chunk, score 0.0.

T5. Anti-narrative:
If the summary only states that an event, conversation, relationship, preference, routine, trip, health issue, project, task, meeting, document, decision, or implementation happened, but does not recover the missing concrete context required by Q, score 0.0.

T6. Specificity:
Generic role mentions such as "they", "someone", "the friend", "the family member", "the place", "the thing", "the plan", "the issue", "the team", "the project", "the document", "the solution", or "the requirements" do not count unless they uniquely identify the same concrete referent and relation as F.

T7. No outside knowledge:
Do not use commonsense, world knowledge, or assumptions to bridge missing information. F must be directly recoverable from summary text.

Scores (binary):

- 1.0: The summary explicitly recovers the selected F, including both the specific referent and essential relation, and the cited summary sentence(s) support it.
- 0.0: Any test fails, no valid F exists, the summary is silent on Q, the summary only covers the broad narrative, the summary only echoes salient_utterance, or the summary contradicts F.

Figure 8: Prompt template for gap-resolution completeness evaluation, Part I.

Prompt: Gap-Resolution Completeness Evaluation, Part II

Examples (illustrative, do NOT copy verbatim):

Example A — score 1.0 (personal plan)
salient_utterance: "I finally booked it for next Friday."
relevant_chunk: "Maya said she wanted to celebrate her birthday at the small Thai place near the river because it has vegan options."
Q: What was booked for next Friday?
F: "Maya's birthday celebration at the small Thai place near the river"
summary contains: "the speaker booked Maya's birthday dinner at the riverside Thai restaurant for next Friday"
score: 1.0

Example B — score 0.0 (wrong answer)
salient_utterance: "He said that was the reason he stopped going."
relevant_chunk: "Daniel stopped going to the climbing gym after he hurt his wrist during a bouldering class."
Q: Why did Daniel stop going?
F: "Daniel stopped going to the climbing gym because he hurt his wrist during a bouldering class"
summary contains: "Daniel stopped going because the membership became too expensive"
score: 0.0

Example C — score 0.0 (salient echo)
salient_utterance: "I made sure to avoid it when I cooked for her."
relevant_chunk: "Nina is allergic to walnuts, but almonds and cashews are fine."
Q: What did the speaker avoid when cooking for Nina?
F: "walnuts, because Nina is allergic to them"
summary contains: "the speaker cooked carefully for Nina and avoided an ingredient"
score: 0.0

Example D — score 0.0 (broad narrative)
salient_utterance: "That place is why I want to go back in June."
relevant_chunk: "When we visited Kyoto last spring, the quiet garden behind Nanzen-ji was Elena's favorite stop."
Q: Which place made the speaker want to go back in June?
F: "the quiet garden behind Nanzen-ji in Kyoto"
summary contains: "the conversation described a memorable previous trip and a desire to return"
score: 0.0

Example E — score 1.0 (work/domain case)
salient_utterance: "This task can be closed today."
relevant_chunk: "Today is the last day for the platform's microservice architecture diagram review."
Q: Which task can be closed today?
F: "the platform's microservice architecture diagram review can be closed today"
summary contains: "the platform's microservice architecture diagram review was completed and closed today"
score: 1.0

Output JSON only. No explanation.

```
[
  {{
    "gap_num": "G0",
    "utterance_num": "U0",
    "question": "What concrete missing information is required?",
    "selected_F": "The minimal fact from relevant_chunk that directly answers the question.",
    "score": 1.0,
    "line_number": ["S1", "S4"]
  }},
  ...
]
```

{N} Summary Sentences (S#):
`{summary}`

{M} Gap Cases (G#):
`{gap_cases}`

Figure 9: Prompt template for gap-resolution completeness evaluation, Part II.

Prompt: Summary Fact Decomposition for Faithfulness

You are decomposing a generated summary into self-contained evaluable facts for factuality evaluation.

For each summary sentence (S#), follow these steps:

1. Identify all verifiable claims in the sentence, not only the main claim.
   A claim is verifiable if it can be checked against the dialogue context as true or false.
2. Decompose each claim into a self-contained evaluable fact. For each fact, preserve:
   (a) the core referent — the specific entity, event, decision, state, quantity, or issue at the center of the claim;
   (b) the essential relation — what happened to the referent, who did what, when, why, how, or to whom;
   (c) support-relevant details — details that affect verification, such as time, location, quantity, frequency, negation, comparison, condition, cause, or consequence.
3. Split coordinated or compressed claims when they express different verifiable information.
   - Split claims connected by "and", "but", "while", "because", "so", "after", "before", or similar relations when each part can be verified separately.
   - If a causal, temporal, contrastive, or decision relation is itself asserted, preserve it as a separate evaluable fact.
   - Do not drop details that change the truth condition of the claim.
4. Make each fact self-contained.
   - Resolve pronouns and vague references using only the generated summary.
   - You may use surrounding summary sentences only to resolve references.
   - Do not use the source dialogue, retrieved memory, external knowledge, or commonsense inference.
   - Do not add information that is not explicitly stated in the generated summary.
5. Exclude non-verifiable or subjective statements.
   - Do not extract vague judgments such as "the discussion was productive" unless the summary explicitly states verifiable evidence for them.
   - Do not create trivial facts such as "the team existed", "the issue existed", or "the box existed."
   - Each fact must be traceable to a specific source span in the generated summary.

Output JSON only. No explanation.
Return a compact JSON array. Each object must contain exactly these keys:

- "sentence_num": the source sentence id such as "S0"
- "fact_id": a stable id such as "S0-F0"
- "fact": one self-contained evaluable fact

Return [] if there are no verifiable facts.
After the closing] stop immediately.

```
[
  {{
    "sentence_num": "S0",
    "fact_id": "S0-F0",
    "fact": "A self-contained evaluable fact."
  }}
]
```

{N} Summary Sentences (S#):

```
{summary}
```

Figure 10: Prompt template for decomposing summary sentences into atomic facts.

Prompt: Faithfulness Evaluation

You will receive a transcript followed by a corresponding summary. Your task is to assess the factuality of each summary sentence across nine categories:

- no error: the statement is factually consistent with the transcript. This includes paraphrase, generalization, aggregation, and reasonable inference that is entailed by or follows naturally from the transcript, even if it is not stated verbatim. Do not penalize a sentence merely because the wording differs or because it synthesizes information across multiple turns.
- out-of-context error: the statement asserts a specific fact that is neither stated in nor reasonably inferable from the transcript (genuine fabrication). Do NOT use this category for legitimate paraphrase, summary-level abstraction, or inference that the transcript supports.
- entity error: the primary arguments (or their attributes) of the predicate are wrong.
- predicate error: the predicate in the summary statement is inconsistent with the transcript.
- circumstantial error: the additional information (like location or time) specifying the circumstance around a predicate is wrong.
- grammatical error: the grammar of the sentence is so wrong that it becomes meaningless.
- coreference error: a pronoun or reference with wrong or non-existing antecedent.
- linking error: error in how multiple statements are linked together in the discourse (for example temporal ordering or causal link).
- other error: the statement contains any factuality error which is not defined here.

The transcript may be split into several labeled sections (for example prior context / memory and the current dialogue window, or retrieved passages). Treat all sections together as a single body of ground-truth evidence: a summary sentence is supported as long as ANY section supports it. Never flag a sentence merely because its supporting evidence sits in the prior-context/memory portion rather than the current window.

Instruction:
First, compare each summary sentence with the transcript. Judge whether the claim is contradicted by or unsupportable from the transcript — not whether it is restated word-for-word. Reasonable inference, paraphrase, and synthesis across turns are acceptable and should be labeled "no error". Only assign an error category when the sentence states something that conflicts with the transcript or asserts a specific fact the transcript does not support.
Second, provide a single sentence explaining which factuality error the sentence has.
Third, answer the classified error category for each sentence in the summary.

Provide your answer in JSON format. The answer should be a list of dictionaries whose keys are "sentence", "reason", and "category". No additional information:

```
[ {{"sentence": "first sentence", "reason": "your reason", "category": "no error"}}, {{"sentence": "second sentence", "reason": "your reason", "category": "out-of-context error"}}, {{"sentence": "third sentence", "reason": "your reason", "category": "entity error"}}, ]
```

Transcript:
`{transcript}`

Summary with {N} sentences:
`{summary}`

Figure 11: Prompt template for faithfulness evaluation.

Prompt (a): Incremental Summarization — Initial Segment Summary

Generate a fact-centered summary for the following dialogue segment.

[Instructions]

1. Introduce all participants and entities with their exact names and roles as stated.
2. Preserve all proper nouns, technical terms, and numerical data verbatim.
3. Record specific content — not that a topic was raised, but exactly what was said.

[Input] **Dialogue Segment:** `{chunk}`

Prompt (b): Incremental Summarization — Running Summary Update (Incremental)

Update the running summary by integrating Segment `{i}`.

[Instructions]

1. Preserve all proper nouns, technical terms, and numerical data from both sources verbatim.
2. Record specific content — not that a topic was raised, but exactly what was said.
3. Integrate new information seamlessly into the existing summary.

[Input] **Running Summary:** `{running_summary}` **New Segment:** `{chunk}`

Prompt (c): Hierarchical Summarization — Level 0 Segment Summarization

Summarize Segment `{i}` with full factual fidelity. Preserve all proper nouns, technical terms, numerical data, and speaker attributions. Do not generalize.

[Input] **Dialogue Segment:** `{chunk}`

Prompt (d): Hierarchical Summarization — Intermediate Merge

Merge these summaries into one cohesive summary. Preserve all proper nouns, technical terms, and factual details.

[Input] `{merged_input}`

Prompt (e): Hierarchical Summarization — Final Merge

Merge the following summaries into a single cohesive summary. Strict limit: under `{max_new_tokens}` tokens. Preserve all proper nouns, technical terms, and factual details.

[Input] `{merged_input}`

Figure 12: Prompt templates for summarization-based memory construction. **Incremental summarization**: (a) initialization generates the first summary from a new segment; (b) update integrates a new segment into an existing running summary. **Hierarchical summarization**: (c) level-0 summarizes each segment individually; (d) intermediate merge combines groups of segment summaries; (e) final merge produces the top-level summary under a strict token budget.

Prompt: Memory-Free Dialogue Summary Generation

Your task is to generate a 'Dialogue Summary' of current [Window] in a fact-centered manner.

Instructions:

1. Fully understand [Window].
2. Generate a concise, high-density summary of [Window]. The result should be a cohesive narrative that reflects the core insights and the logical evolution of the situation.

Provide your output as JSON. No additional explanation.

```
{{"summary": "Your high-density summary here."}}
```

[Window]

```
{window}
```

Figure 13: Prompt template for dialogue summary generation.

Prompt: (i) Salient Utterance Identification and (ii) Contextual Dependence Judgment

You are a Linguistic and Dialogue Context Analyst specializing in discourse coherence and pragmatics. Your task is to analyze the provided dialogue excerpt to identify Salient Utterances and categorize their Contextual Gaps based ONLY on the provided text.

 # Instructions

1. Select sentences that contain a concrete decision, request, action item, or status update.
   - Merge Rule: If consecutive sentences form one coherent update/decision, merge them into a single entry.
   - Exclude: Simple acknowledgments ("Okay"), greetings, praise ("Great job"), or emotional reactions.
2. For each salient utterance, judge whether it is contextually self-contained for summarization:
   - Identify pronouns, vague entities, underspecified concepts, missing attributes, and unstated reasoning that may prevent the utterance from being summarized independently.
   - Check whether the information needed to interpret the utterance in a stable and unambiguous way is explicitly available in the excerpt.
   - Assign a gap if the utterance is not sufficiently self-contained to support a contextually self-contained summary based on the current window alone.
3. For each salient utterance, assign a gap type:
   - Coreference Resolution (C): a pronoun, shorthand reference, or vague noun phrase cannot be stably resolved from the excerpt.
   - Attribute/State Supply (A): a key concept, entity property, parameter, state, scope, or specification is missing, making the utterance insufficiently self-contained for summary writing.
   - Logic/Causal Connection (L): the utterance depends on prior reasoning, motivation, agreement, comparison, or constraint that is not given in the excerpt, so the conclusion or action is not fully supported within the window.
   - No Gap (N): the utterance is sufficiently self-contained to be summarized clearly and stably from the excerpt alone.
4. Rules:
   - Use the standard of self-contained summarizability, not mere rough interpretability.
   - Mark N only when the utterance can be summarized from the current window without needing earlier dialogue to resolve its main meaning, reference, or rationale.
   - If nearby sentences fully resolve the missing information, do not mark a gap.
   - Do not mark a gap just because additional detail could make the utterance richer.
   - Commonly understood domain terms are not gaps by themselves, but project-specific concepts may be gaps if they remain underspecified for summary writing.
   - Clarification questions or examples can still have gaps if they rely on missing prior context.

Provide your answer in JSONL format. No additional explanation.

```
{{'sentence_numbers': [number, number], 'sentences': ['text', 'text'], 'reason': 'concise gap analysis', 'gap_type': 'N | C | A | L'}}
{{'sentence_numbers': [number], 'sentences': ['text'], 'reason': 'concise gap analysis', 'gap_type': 'N | C | A | L'}}
```

—

 # [Input Dialogue]

```
{window}
```

Figure 14: Prompt template for salient utterance identification and contextual dependence judgment.

Prompt: (iii) Candidate History Filtering

You are a Context Retriever specializing in conversation threading. Your task is to determine, for EACH chunk in [Dialogue Context], whether the chunk is relevant to the [Dialogue Window]. [Dialogue Context] contains multiple past-dialogue chunks. You must evaluate each chunk independently and output one binary label per chunk.

A label should be:

- 1: The chunk contains prior dialogue that is necessary or directly useful for understanding, continuing, or summarizing the current [Dialogue Window].
- 0: The chunk is unrelated, only loosely topically similar, or not needed to maintain the conversational thread.

A chunk is relevant if it contains at least one of the following:

1. A topic, entity, event, plan, task, or decision that is explicitly continued in the Dialogue Window.
2. Prior information needed to resolve references in the Dialogue Window, such as "this," "that," "it," "the previous one," "the method," or similar expressions.
3. User preferences, constraints, assumptions, or prior decisions that directly affect the current window.
4. Earlier steps of the same ongoing task, project, or discussion.

A chunk is NOT relevant if:

1. It discusses a different topic or a separate session.
2. It is only generally similar but not needed for the current window.
3. It contains background information that does not change the interpretation of the current window.
4. It requires guessing or inferring a connection that is not clearly supported.

Important:

- Evaluate each chunk separately.
- Process chunks in 0-based index order, from the first chunk to the last chunk.
- Do NOT skip any chunk, including the final chunk.
- Before final output, verify that the list contains one label for every chunk index from 0 to N-1.
- Do NOT assign one label to the entire Dialogue Context.
- Do NOT rank the chunks.
- Do NOT explain your reasoning.
- Output only a Python-style list of 0s and 1s.
- If there are N chunks in [Dialogue Context], your output must contain exactly N labels. Any output other than a single list of binary labels is invalid.

–

 # [Dialogue Window]

```
{Dialogue_Window_Content}
```

 # [Dialogue Context, 0-based index]

```
{Dialogue_Context_Content}
```

–

 # Output Format,

```
[0,1,0,...]
```

Figure 15: Prompt template for candidate history filtering.

Prompt: (iv) Evidence and Resolution Annotation, Part I

You are a Contextual Gap Resolver. Your task is to audit EACH sentence in the [CURRENT SEARCH CHUNK] and decide whether the sentence provides explicit evidence that fills a concrete missing information slot in the [TARGET].

Gap Type

- Coreference Resolution (C): a pronoun, shorthand reference, or vague noun phrase cannot be stably resolved from the excerpt.
- Attribute/State Supply (A): a key concept, entity property, parameter, state, scope, or specification is missing, making the utterance insufficiently self-contained for summary writing.
- Logic/Causal Connection (L): the utterance depends on prior reasoning, motivation, agreement, comparison, logic, cause, or constraint that is not given in the excerpt, so the conclusion or action is not fully supported within the window.

You must act as a strict logic gate.

- Use only explicit evidence in the [CURRENT SEARCH CHUNK].
- Do not use outside knowledge.
- Do not infer unstated links.
- Do not mark a sentence as resolving the gap merely because it shares the same topic, entity, meeting, or general context.
- A sentence resolves the gap **only if it explicitly provides information that the target depends on for independent understanding**, such as:

A valid resolving sentence must explicitly provide at least one of the following:

- the exact referent of a vague or ambiguous expression
- a missing attribute, definition, state, scope, role, parameter, or specification
- a necessary prior decision, rationale, agreement, comparison, condition, or event
- an earlier action, request, or commitment that the target continues, refers back to, or presupposes

Do not accept a sentence just because:

- it is about the same topic
- it mentions the same people, team, or project
- it appears nearby in the discourse or event flow
- it provides only broad background or loosely related context

Figure 16: Prompt template for evidence and resolution annotation, Part I.

Prompt: (iv) Evidence and Resolution Annotation, Part II

 # Strict Audit Protocol

1. Read the [REFERENCE WINDOW], [TARGET], [Gap Type], and [Gap Reason], and determine the exact missing content needed to resolve the gap.
   - The missing content must be specific and concrete.
   - Do not define it as a broad topic, general context, or abstract category.
2. Audit every sentence in the [CURRENT SEARCH CHUNK], from the first sentence.
3. For each sentence, think step-by-step:
   - Judge the sentence only with respect to the missing content implied by the [Gap Type] and [Gap Reason].
   - State whether the sentence provides explicit content that fills a concrete target gap.
   - Focus on what exact content the sentence provides or fails to provide, not on whether it is generally related.
4. Write your judgment in `resolution_reasoning`.
   - `resolution_reasoning` must explain the sentence strictly in terms of its gap-resolution role.
   - If the sentence resolves the gap, briefly and concretely state what missing content it provides and how that helps make the target independently understandable.
   - If the sentence does not resolve the gap, state what the sentence does contain, then explain why that is still insufficient to resolve the gap.
   - Make clear whether the sentence is only topically related, background-only, or missing the actual dependency needed by the target.
5. Only after writing `resolution_reasoning`, decide the final status.
   - Mark the sentence as "resolve" only if it explicitly provides the missing content needed by the target.
   - Otherwise mark it as "not_resolve".

Additional constraints for `resolution_reasoning`

- Do not say only that the sentence is "related," "relevant," "same context," or "helpful."
- Do not justify using topic overlap, shared entities, or discourse proximity alone.
- Do not restate the sentence without explaining its gap-resolution role.
- Do not use vague phrases such as "this helps understand the context" unless you specify exactly what missing content it provides.
- Keep it brief, but include the exact reason for resolve vs. not_resolve.

Provide your answer in JSONL format. No additional explanation.

```
{{"source_id":"T0-S0","sentence":"...","reasoning":"...","status":"resolve"}}
{{"source_id":"T0-S1","sentence":"...","reasoning":"...","status":"not_resolve"}}
```

[REFERENCE WINDOW] (Contextual Background)
NOTE: This data is for contextual background only. Do NOT attempt to resolve the gap using this section, as it has been pre-verified that the required information is NOT present here.
`{window}`

[TARGET]
[Sentence]: `{target_sentence}`
[Gap Type]: `{gap_type}`
[Gap Reason]: `{gap_reason}`

[CURRENT SEARCH CHUNK]
`{search_chunk}`

Figure 17: Prompt template for evidence and resolution annotation, Part II.

Prompt: Critical Logic Audit for Contextual Gap Resolution

You are a Critical Logic Auditor. Your task is to perform a final validation of a "Contextual Gap Resolution" result. You must determine if the provided [RESOLVING SENTENCE] truly and sufficiently fills the [GAP] in the [TARGET SENTENCE] as claimed in the [REASONING].

 # Validation Criteria:

1. Sufficiency: Does the resolving sentence provide the *exact* missing information (referent, attribute, or logic) defined in the Gap Reason?
2. Non-Inference: Is the resolution explicit? If you still need to "guess" or "assume" anything to make the target clear, the resolution is INVALID.
3. Independence: After incorporating the information from the resolving sentence, can the target sentence now be summarized as a standalone, unambiguous fact?
4. Redundancy: Is the information in the resolving sentence meaningfully distinct from what is already present in the [REFERENCE WINDOW]? If the resolving sentence merely restates or duplicates content already available in the window, the resolution is INVALID — it provides no new contextual value beyond what was already accessible.

 # Evaluation Rules:

- 1: The evidence is explicit, directly addresses the gap, and makes the target self-contained.
- 0: The reasoning is based on topical overlap, the evidence is too vague, or the resolution requires additional unstated context.

 # Instructions:

- Audit each entry in the [CANDIDATE LIST] independently.
- Identify the "source_id" for every candidate that receives a score of **1**.
- Return the results **strictly** as a JSON list of strings containing only the successful "source_id"s.
- No preamble, no explanation, no additional text.

 # Output format:

```
["Tn-Sm", "Tn-Sm", ...]
```

[REFERENCE WINDOW] (Contextual Background)
NOTE: This data is for contextual background only. Do NOT attempt to resolve the gap using this section, as it has been pre-verified that the required information is NOT present here.

`{window}`

[TARGET]
[SENTENCE] `{target_sentence}`
[GAP TYPE] `{gap_type}`
[GAP REASON] `{gap_reason}`

[CANDIDATE LIST]
`{candidate}`

Figure 18: Prompt template for final validation of contextual gap resolution annotations.