VoiceMem: Streaming Dual-Brain Memory for Real-Time Interaction

Anonymous authors
Paper under double-blind review

Abstract

Real-time conversational systems require relevant memory within limited context and response-time budgets. We introduce VoiceMem, a streaming memory layer with factual and persona-affective retrieval branches, shared evidence identifiers, and optional entity attribution. An evolving schema–entity index adapts to recurring queries, while per-turn maintenance separately records immediate reactions and model-proposed persona claims. A streaming controller runs the same retrieval code before speech endpoint confirmation, overlapping preparation with the utterance so that only unfinished final ranking remains on the memory-critical path when preparation is complete. VoiceMem supports interchangeable backends and includes a teacher-supervised adaptation workflow, ChatMem-400K, and the audio-grounded ChatMem-Bench. Evaluations cover factual, persona, and audio-memory quality, context efficiency, component contributions, and three backend integrations. VoiceMem improves on reported factual and persona baselines, including a 21.43-point gain over Full-Context on LoCoMo, and exceeds MemOS by 14.20 percentage points on ChatMem-Bench. At the default LoCoMo operating point (K=5), the same text-evaluation setting achieves 90.4 with 430 memory tokens and 134 ms mean memory-system latency without streaming overlap. Streaming further hides preparation behind speech, connecting evidence organization with response-time memory preparation while separating memory management from the response model.

1 Introduction

> "Through memory, the soul reveals traces of its former existence."
>
> — Plato

Memory enables conversational systems to respond consistently across turns, sessions, and changing user circumstances. Recent memory engines improve storage and retrieval (Chhikara et al., 2025; Xu et al., 2025; Hu et al., 2026), while speech language models increasingly support low-latency and duplex interaction (Xie & Wu, 2024; Xie et al., 2026c; Fang et al., 2025; Qwen Team, 2026; Ma et al., 2025; ByteDance Seed, 2026; OpenAI, 2026). In the post-turn pipeline considered here, however, these capabilities are connected only after a turn has ended: a memory service receives a completed query, searches stored records, and returns context before response generation. This places memory access on the response-critical path. The challenge is therefore not only what to remember, but when and how to make that memory available.

Three design requirements motivate our approach. (O1) Coordinated factual and person-centered state. Facts benefit from compact routing over entities and topics, whereas persona and affect benefit from claim-level matching with timestamped supporting observations. These different evidence-organization needs motivate two specialized branches, connected by shared evidence identifiers and optional entity attribution. A stored persona claim is a model inference, not a verified persistent trait. (O2) High information density within an interaction deadline. Retrieval that adds seconds after a user stops speaking disrupts turn-taking, while a small prompt budget leaves little room for broad top-K recall. (O3) Backend

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

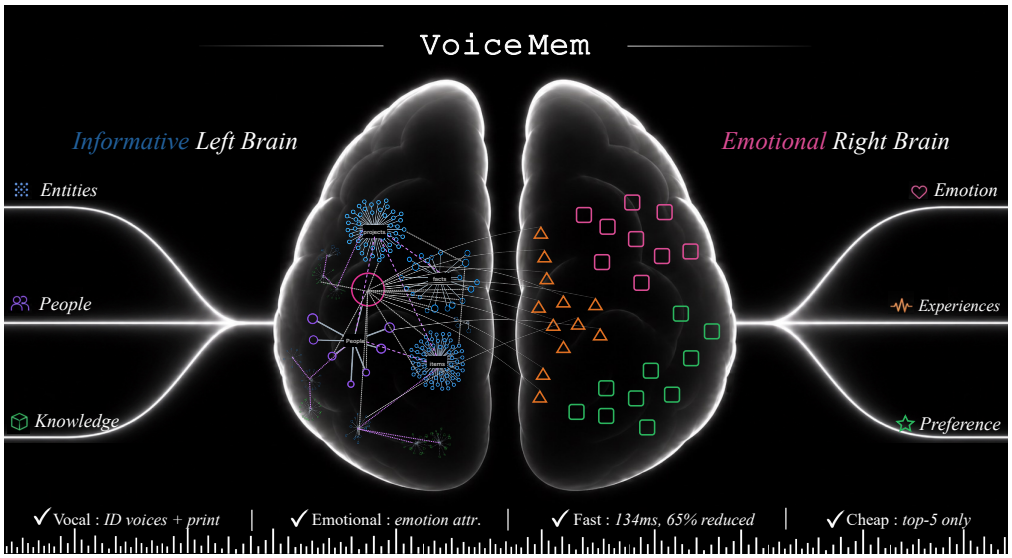

Figure 1: VoiceMem: dual-brain memory for voice interaction. The left brain organizes factual knowledge through schema–entity indexing, while the right brain maintains persona, preferences, and affective experiences. Shared evidence identifiers and optional entity links record provenance and affective targets. The two retrieval branches jointly provide response context; the default path does not traverse cross-brain links. Speaker attribution and compact retrieval connect user-specific evidence to the streaming interaction pipeline.

and model evolvability. Memory organization should remain independent of a particular backend or speech model so that either can evolve without redesigning the memory layer.

VoiceMem addresses these requirements with a streaming dual-brain memory layer (Figure 1). For O1, its left brain routes factual queries through an evolving schema–entity index, with query-driven schema emergence adapting to recurring retrieval patterns; its right brain separately matches persona claims and context-dependent attitudes. Shared evidence identifiers and optional entity anchors preserve provenance, and both branches supply evidence to the response model; the design does not require cross-branch graph traversal at query time. For O2, VoiceMem matches a partial transcript and schedules graph expansion and backend retrieval around the silence interval used by voice activity detection (VAD). For O3, upper-layer state management is decoupled from the memory engine and interaction model, allowing compatible backends to be exchanged.

Our contributions are threefold. (1) We introduce specialized factual and persona-affective evidence organization, combining query-driven schema emergence with claim-level retrieval, shared evidence identifiers, optional entity attribution, and per-turn maintenance. (2) We formulate a streaming retrieval controller that prepares memory from partial transcripts, using the same retrieval code as non-streaming execution to move preparation before endpoint confirmation and leave only final ranking on the memory-critical path when preparation is ready. (3) We develop a backend-decoupled evaluation framework with factual, persona, and audio-memory tasks, supported by ChatMem-400K, ChatMem-Bench, and a teacher-supervised adaptation workflow. The experiments distinguish complete-system response quality from component contributions, context efficiency, and backend compatibility.

2 Related Work

Memory systems. Memory engines such as Mem0 (Chhikara et al., 2025) and Zep (Rasmussen et al., 2025) support persistent storage and retrieval, while A-Mem (Xu et al., 2025), APEX-Mem (Banerjee et al., 2026), Magma (Jiang et al., 2026), and GAM (Wu et al., 2026b) add self-organizing or graph-based maintenance. Emotional RAG (Huang et al., 2024), KEEM (Kang et al., 2025), dynamic affective memory (Lu & Li, 2025), and DualMem (Zhang et al., 2026) incorporate affect into memory construction or retrieval.

VoiceMem focuses on maintaining factual routing and persona-affective attribution as separate, connected memory states.

Real-time interaction models. Speech language models increasingly support overlap, duplex conversation, and low-latency generation, including dGSLM (Nguyen et al., 2023), LSLM (Ma et al., 2025), SyncLLM (Veluri et al., 2024), Mini-Omni (Xie & Wu, 2024), Moshi (Défossez et al., 2024), SALMONN-omni (Yu et al., 2024), and Audio-Interaction (Xie et al., 2026c). Other work interleaves reasoning with ongoing speech (Wu et al., 2025b; Chiang et al., 2026; Wu et al., 2026a; Xie et al., 2025). VoiceMem extends the real-time constraint to memory preparation and access.

Stateful interactive agents. Dual-process interaction models separate rapid response from slower reasoning (Christakopoulou et al., 2024; Chen et al., 2025), while proactive agents use persistent context to guide actions (Zhang et al., 2024; Lu et al., 2025; Xie et al., 2026b). VoiceMem focuses instead on the memory substrate that supplies those systems: compact factual evidence and persona-affective claims use specialized retrieval branches, with shared evidence identifiers and optional entity attribution.

3 Problem Setting

At interaction step n, the system derives a query q_n from utterance u_n and selects evidence from persistent memory $\mathcal{M}_n$ under context and response-time budgets. A post-turn semantic pipeline retrieves records, generates a response, and updates memory:

$$\mathcal{R}_n^{\text{sem}} = \text{TopK}_{m_i \in \mathcal{M}_n} \text{sim}\big(f_\theta(q_n), f_\theta(m_i)\big),\ o_n = \text{LLM}\big(q_n, \mathcal{R}_n^{\text{sem}}\big),\ \mathcal{M}_{n+1} = \text{Update}(\mathcal{M}_n, u_n, o_n). \tag{1}$$

Here, f_θ is an encoder and o_n the response. Retrieval remains on the response-critical path, with factual and person-centered evidence sharing one store.

Affect-aware retrieval (Huang et al., 2024; Kang et al., 2025) additionally scores compatibility between query and memory affect:

$$\mathcal{R}_n^{\text{aff}} = \text{TopK}_{m_i \in \mathcal{M}_n} \left[\lambda\ \text{sim}\big(f_\theta(q_n), f_\theta(m_i)\big) + (1-\lambda)\kappa(\mathbf{a}_n, \mathbf{a}_i)\right]. \tag{2}$$

where κ compares query and memory affect representations $\mathbf{a}_n$ and $\mathbf{a}_i$. This score does not model an emotion's target or persistence.

VoiceMem instead organizes factual evidence in $\mathcal{G}_n^L$ and persona-affective claims in $\mathcal{G}_n^R$. Shared evidence identifiers and optional links $\mathcal{E}_n^{LR}$ from affective evidence to factual entities preserve attribution; they do not imply cross-graph retrieval in the default path. The brain labels denote software roles:

$$\mathcal{B}_n = \big(\mathcal{G}_n^L, \mathcal{G}_n^R, \mathcal{E}_n^{LR}\big), \quad \mathcal{R}_n^{\text{DB}} = \text{JointRetrieve}(q_n \mid \mathcal{B}_n), \quad o_n = \text{LLM}\big(q_n, \mathcal{R}_n^{\text{DB}}\big). \tag{3}$$

Here, n indexes interaction steps; the streaming formulation below uses t for observations within an utterance. Component updates and retrieval timing are described next.

4 VoiceMem: Streaming Dual-brain Architecture

VoiceMem addresses the requirements in Section 1 through factual and persona evidence organization with traceable context (O1), compact selection and streaming retrieval for context and latency budgets (O2), and backend/model separation (O3; Section 5).

4.1 Left Brain: Efficient Memory Access

The left brain organizes factual memory through a semantic index while leaving record ranking to the backend. Asynchronous updates incorporate new facts, and query-driven schema emergence adapts the index to recurring information needs.

Schema–entity representation. Schemas provide coarse semantic groups; entities represent people, events, or concepts and link to backend records:

$$\mathcal{G}^L = (\mathcal{S}, \mathcal{V}, \mathcal{E}), \qquad v = (d_v, \mathcal{N}_v^{\text{micro}}, \mathcal{I}_v), \qquad s = (d_s, \mathcal{N}_s^{\text{macro}}, \mathcal{V}_s).$$

Here, $\mathcal{S}$ and $\mathcal{V}$ are schema and entity sets, d denotes textual descriptions, $\mathcal{I}_v$ contains record identifiers, and $\mathcal{V}_s$ contains schema members. $\mathcal{E} = \mathcal{E}_{\text{micro}} \cup \mathcal{E}_{\text{macro}}$ relates entities and schemas at their respective levels.

Index-guided retrieval. The partial transcript $x_{\leq t}$ matches specific entities and broader schemas, then constructs an entity candidate set:

$$(\mathcal{V}_t, \mathcal{S}_t) = \text{Match}(x_{\leq t}, \mathcal{V}, \mathcal{S}),$$
$$\mathcal{Z}_t = \begin{cases} \mathcal{V}_{\mathcal{S}_t}, & \mathcal{V}_t = \varnothing, \\ \mathcal{V}_t \cup \mathcal{V}_{\mathcal{S}_t} \cup \mathcal{N}_1^{\text{active}}(\mathcal{V}_t \cup \mathcal{V}_{\mathcal{S}_t}), & \text{otherwise.} \end{cases} \tag{4}$$

Here, $\mathcal{V}_{\mathcal{S}_t} = \bigcup_{s \in \mathcal{S}_t} \mathcal{V}_s$, and $\mathcal{N}_1^{\text{active}}$ adds active one-hop neighbors without strong/weak weighting. Neighbor expansion requires an entity match; schema-only retrieval uses schema members directly. The default query classifier does not run NER, so entity sets are normally empty unless the caller supplies them, and this default path does not perform entity-neighbor expansion. Appendix A gives the matching details.

The index maps candidates to record IDs for backend ranking:

$$\mathcal{C}_t^L = \bigcup_{z \in \mathcal{Z}_t} \mathcal{I}_z, \qquad \mathcal{R}_t^L = \text{MemSearch}(q_t, \mathcal{C}_t^L; K_L). \tag{5}$$

MemSearch returns at most K_L left-branch records and applies adapter filtering plus full-store fallback/fill when the routed pool is empty or insufficient. Thus, the routed pool is not a strict boundary on returned records. Appendix B compares record expansion with compressed schema descriptions.

Asynchronous factual maintenance. An asynchronous updater applies add, update, delete, or keep to facts and their index structures. The stores are updated sequentially without a cross-store atomic transaction; Appendix C specifies the rules and consistency behavior.

Query-driven schema emergence. As schemas grow, query-driven emergence identifies recurring connected entity groups rather than splitting only by size. For session queries $\mathcal{Q}$, activated entities A_q, and candidate subset H, it measures mean Jaccard overlap:

$$\rho(H) = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \frac{|A_q \cap H|}{|A_q \cup H|}. \tag{6}$$

Among generated connected candidates satisfying $\rho(H) > \alpha$, the system selects the largest and uses an LLM judge to check relevance, plus importance and completeness for a new schema. A validated group either forms a schema or merges into an existing one; entity membership moves to the destination, while record-level historical tags are retained. Algorithm 1 and Appendix D give the full procedure, and Appendix E compares heuristic splits.

4.2 Right Brain: Knowing the Person

The right brain stores interaction-supported evidence about user dispositions, preferences, and context-dependent attitudes. It organizes candidate persona claims separately from turn-level affect, with optional entity targets. Persistence is judged by a per-turn LLM, rather than established by a recurrence test over longitudinal evidence.

Persona representation and evidence provenance. The representation contains independent persona nodes $\mathcal{V}^I$ and entity-associated nodes $\mathcal{V}^C$:

$$\mathcal{G}^R = (\mathcal{V}^I, \mathcal{V}^C), \qquad v^I = (d_v^I, \mathcal{I}_v^I) \in \mathcal{V}^I, \qquad v_e^C = (d_{v,e}^C, \mathcal{I}_{v,e}^C, \ell_{v,e}) \in \mathcal{V}^C, \quad e \in \mathcal{V}. \tag{7}$$

Here, d is a persona description and $\mathcal{I}$ its supporting record IDs. v^I represents a target-free candidate persona claim, whereas v_e^C links an attitude or affective experience to factual entity e through $\ell_{v,e} \in \mathcal{E}^{LR}$. Node descriptions support matching but are not inserted separately into the prompt; matched nodes contribute their evidence records. Appendix C gives construction details.

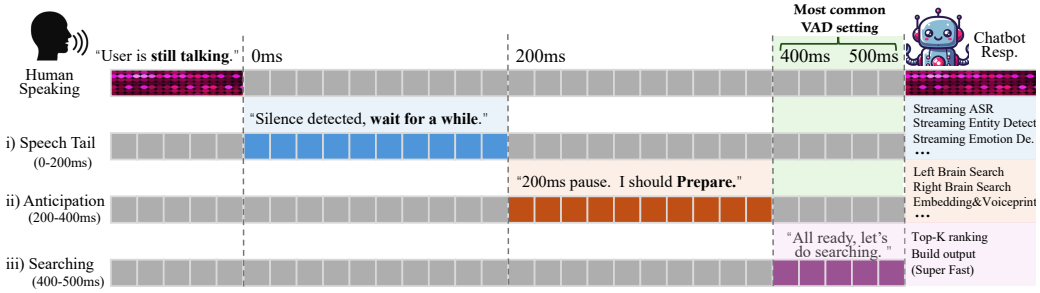

Figure 2: Schematic four-stage schedule for preparing and retrieving dual-brain memory around speech endpoint detection. The displayed millisecond boundaries are illustrative allocations based on configurable triggers, not measured completion-time guarantees. Speaker identification runs after utterance end and gates subsequent turns. The implemented event order is specified in Appendix A.3.

Claim-level persona retrieval. Both node types are matched against the current conversation and combined for retrieval:

$$(\mathcal{V}_t^I, \mathcal{V}_t^C) = \text{Match}\left(x_{\le t}, \mathcal{V}^I, \mathcal{V}^C\right), \qquad \mathcal{Z}_t^R = \mathcal{V}_t^I \cup \mathcal{V}_t^C,$$
$$\mathcal{C}_t^R = \bigcup_{z \in \mathcal{Z}_t^R} \mathcal{I}_z, \qquad \mathcal{R}_t^R = \text{MemSearch}\left(q_t, \mathcal{C}_t^R; K_R\right). \tag{8}$$

$\mathcal{C}_t^R$ collects supporting record IDs from matched nodes, and $\mathcal{R}_t^R$ contains at most K_R returned items. Stored factual causes and entity anchors preserve attribution but are not traversed during retrieval. In particular, persona claims are matched independently, without following cause_id back to factual records. Thus, joint retrieval denotes two specialized branches and context assembly, not a demonstrated inference-time gain from cross-brain links. Appendix A specifies matching and candidate limits.

Dual-horizon evidence maintenance. Short-horizon attribution records current affect and context; long-horizon maintenance creates or merges a candidate persona claim when a per-turn LLM judges it worth retaining. The horizons refer to record roles and retention, not fixed history windows or independently validated levels of persistence.

For session input x_j, an affect estimator ϕ produces $\mathbf{a}_j$, and Modify updates the right-brain state:

$$\mathbf{a}_j = \phi(x_j), \qquad \mathcal{G}_j^R = \text{Modify}\left(\mathcal{G}_{j-1}^R; x_j, \mathbf{a}_j\right), \qquad j = 1, \ldots, T. \tag{9}$$

x_j includes the utterance and available acoustic-emotion evidence. Per-turn attribution determines whether the reaction targets an entity; Modify upserts a heartnote and may propose a persona claim. This judgment uses the preceding assistant response, current user reply, and emotion information, not the complete longitudinal evidence sequence. There is no separate periodic consolidation operator or repeated-evidence threshold, and the mechanism does not establish that transient reactions cannot be promoted to persona claims. Contradictions remain as timestamped evidence instead of overwriting earlier observations. Appendix C specifies visibility, attribution, and retention policies.

4.3 Streaming Dual-Brain Retrieval

The streaming and text-input paths perform the same memory computations with the same models and retrieval rules. Streaming changes only when these computations begin, overlapping preparation with speech and endpoint-detection silence. Final ASR corrections available before retrieval refresh the query and affected candidates. For the same transcript and memory state, the memory computation is unchanged; the only potentially lossy input transformation is ASR, whose recognition errors cannot be eliminated by the memory layer.

Preparation during interaction. During the listening, speech-tail, and anticipation stages in Figure 2, partial transcripts (Shi et al., 2026; Xie et al., 2026d) match factual and persona nodes. Their record IDs form the two backend candidate pools in Equations 4 and 8; backend

ranking, rather than preparation, selects the final records. Appendix A specifies triggers, transcript revisions, and speaker handling.

Joint retrieval and context assembly. The controller retrieves from both pools and assembles the response context:

$$\mathcal{R}_t^L = \text{MemSearch}(q_t, \mathcal{C}_t^L; K_L), \mathcal{R}_t^R = \text{MemSearch}(q_t, \mathcal{C}_t^R; K_R), R_t = \text{Prompt}(\mathcal{R}_t^L, \mathcal{R}_t^R, \mathcal{D}_t). \tag{10}$$

Here, R_t is the assembled memory context and $\mathcal{D}_t$ optional schema or graph descriptions. Full-system experiments use $K_L = K_R = K$; the left-brain-only sweep uses $K_L = K$, $K_R = 0$ (Section 6.4). Appendix A.4 gives budget accounting and assembly rules.

Latency scope. Once candidates are ready, only unfinished final backend ranking and context assembly remain on the response-critical path; ranking can itself finish before endpoint confirmation. The Studio controller's bounded-wait policy may yield empty memory if no result is reusable (Appendix A.3). The 134 ms LoCoMo mean measures full memory-system cost without streaming overlap, not perceived streaming delay. Quality and latency are evaluated separately, without jointly scoring answers under deadline cancellation (Section 6.4).

4.4 Audio-Grounded Memory

Audio-grounded memory adds speaker identity, paralinguistic cues, and environmental sounds to the factual and person-centered representation. Identity supports attribution across interactions, while vocal and environmental cues contextualize stored events; an acoustic observation is retained as evidence, while any persona claim inferred from it remains a model judgment rather than a verified trait. In the online speech interface, the retrieval index stores transcripts plus entity and emotion labels; neither raw waveforms nor acoustic embeddings are indexed or passed to the response model. The measured ChatMem-Bench comparison instead supplies the same textualized audio-event, scene, and emotion evidence to all systems before ingestion. Appendix F describes storage, speaker attribution, and playback interfaces; Section 6.1 specifies evaluation modality access.

5 Infrastructure: Model Adaptation, Data and Backend Interfaces

We separate memory organization from response-model adaptation and backend integration, allowing training-dependent results to be distinguished from fixed-model memory evaluations.

5.1 Teacher-Supervised Model Adaptation

We adapt Qwen3-Omni-30B-A3B-Instruct using revised SFT targets from the qwen3.7-plus teacher, conditioned on dialogue history, factual and affective memory, and the student's draft. The student is optimized with token-level cross-entropy and supplies updated drafts for subsequent supervision. The reported model is checkpoint-3318. Adaptation changes the response model's use of memory, not the memory representation; Appendix G gives the training procedure and evaluated checkpoints.

5.2 Training Data and Evaluation Resources

ChatMem-400K uses synthetic personas, histories, and conversations covering recall, reference resolution, changing facts, and person-centered reasoning. Automated verification and human curation refine persona, affective, and acoustic examples; Appendix G describes the pipeline.

Benchmark scope. ChatMem-Bench covers information, persona, affective attribution, and paralinguistic/environmental memory through 14 categories, with 500 sessions, one evaluation question per session, and 53 hours of audio. ChatMem-400K contains approximately 400,000 modified single-turn examples in 15,314 training sessions. The datasets use independently generated manifests; audits found no overlap. ChatMem-Bench was not used for training, prompt design, checkpoint selection, or tuning. Appendix G gives provenance, overlap-audit results, and the isolated, chronological evaluation protocol.

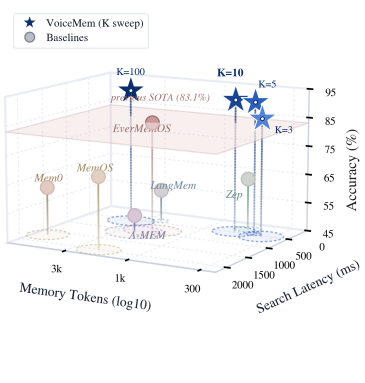

Figure 3: LoCoMo score, memory tokens, and latency.

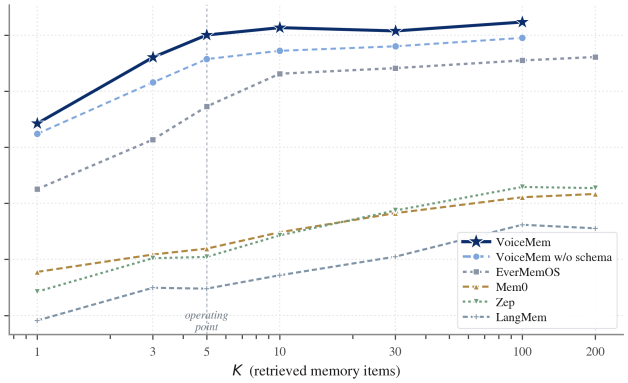

Figure 4: LoCoMo left-brain retrieval-budget sweep.

5.3 Backend-Decoupled Memory Interfaces

VoiceMem places semantic routing, factual/persona state, and streaming preparation above the backend exposed through MemSearch. The upper layer supplies candidate IDs; adapters provide ranking, filtering, and factual fallback/fill. Experiments use Mem0 (Chhikara et al., 2025), LangMem, and Zep Cloud API. Mem0 filters over-retrieved results by candidate ID, whereas ID-restricted search was not verified for LangMem or Zep. Interchangeability therefore depends on identifier access, candidate restrictions, and required write operations; Appendix H specifies the tested backend interfaces and transfer configurations.

6 Experiments

6.1 Evaluation Setup

We evaluate factual memory on LoCoMo (Maharana et al., 2024), LongMemEval (Wu et al., 2025a), and Memora (Uddin et al., 2026); persona memory on ES-MemEval (Chen et al., 2026), PersonaMem (Jiang et al., 2025), and PersonaLens (Zhao et al., 2025); and audio-grounded memory on the 500-question ChatMem-Bench, against Full-Context and nine memory systems. Standard text-input rows use GPT-4o-mini (OpenAI, 2024a) and text-embedding-3-small (OpenAI, 2024b); adapted rows use Qwen3-Omni checkpoint-3318. VoiceMem defaults to K=5, with top-10 retrieval on ChatMem-Bench. All LoCoMo quality and latency experiments share the text-input protocol, models, history access, and accounting rules, varying only stated factors. ChatMem-Bench supplies every system with the same transcripts and textualized audio-event, scene, and emotion evidence; all memory systems update every turn. Sessions default to one user speaker, with multi-user evidence supplied in text rather than through live identification. A binary GPT-4o-mini judge yields category macro-averages in Tables 1–2 and a 500-question micro-average in Table 3. Full settings appear in Appendices G and H.

6.2 Factual and Persona Response Quality

Factual and persona memory. VoiceMem achieves the highest factual average (76.39), exceeding the strongest baseline, MemOS (65.83), by 10.56 points. On LoCoMo, it scores 90.40, 21.43 points above Full-Context, and leads the baselines in seven of eleven factual categories, although not uniformly across LongMemEval and Memora. For persona memory, VoiceMem scores 74.16, 1.89 points above MemOS (72.27), while VoiceMem (adapted) reaches 76.56. Adaptation improves the persona average but lowers the factual average to 75.65; because it also changes the response model, this comparison does not isolate the effect of adaptation (Appendix G).

6.3 Voice-chatting Response Quality

Audio-grounded memory. VoiceMem obtains a 69.60 micro-average on ChatMem-Bench, outperforming the strongest baseline, MemOS (55.40), by 14.20 points and leading the baseline rows in 11 of 14 categories; VoiceMem (adapted) scores 67.60. These complete-system comparisons share input evidence, per-turn writes, and speaker attribution (Section 6.1).

Table 1: Factual memory results. LLM-judge score (%, ↑). VoiceMem (adapted) uses Qwen3-Omni-30B-A3B-Instruct; all other rows use GPT-4o-mini as the response model.

Method	LoCoMo				LongMemEval				Memora			Avg.
	Single	Multi	Temp.	Open	Extr.	Multi	Upd.	Temp.	Rem.	Rec.	Rea.	
Reference												
Full-Context	75.87	72.27	70.29	57.45	45.45	73.92	78.21	31.58	85.54	52.10	22.67	60.49
Memory Engines												
Mem0	67.13	51.15	55.51	72.93	62.82	46.21	70.12	40.15	40.42	52.58	16.00	52.27
Zep	66.91	53.70	60.29	70.82	47.46	74.44	54.10	34.62	33.36	37.38	3.50	48.78
LangMem	62.23	47.92	43.43	71.12	55.13	20.30	66.67	15.79	68.16	48.88	30.00	48.15
Agent Memory												
A-MEM	49.79	49.85	54.05	49.91	64.62	48.87	64.11	47.36	68.82	35.04	2.00	48.58
MemoryOS	69.35	62.30	44.72	44.29	58.10	31.06	48.72	32.33	15.24	38.86	1.50	40.59
MemOS	81.09	67.49	75.18	55.90	86.77	70.70	74.33	77.49	51.84	62.64	20.66	65.83
Personal Memory												
MemoryBank	21.25	18.40	31.50	26.28	23.33	26.00	14.00	20.00	22.22	41.88	1.42	22.39
EverMemOS	90.12	87.43	85.16	69.79	82.71	73.68	89.74	77.44	17.21	33.27	16.67	65.75
Emotional RAG	19.62	25.53	34.89	13.54	40.00	55.00	70.00	65.00	51.16	39.38	14.75	38.99
Ours												
VoiceMem	94.60	91.60	89.80	85.60	83.20	75.00	77.50	95.00	66.80	50.50	30.70	76.39
VoiceMem (adapted)	95.15	92.40	91.35	86.08	79.64	70.17	79.24	89.63	69.10	54.77	24.61	75.65

Table 2: Persona memory results. LLM-judge score (%, ↑).

Method	ES-MemEval				PersonaMem					PersonaLens		Avg.
	IE	TR	CD	UM	Rec.	Gen.	Evol.	Reason	Reco.	Per.	TCR	
Reference												
Full-Context	20.20	19.60	12.10	21.70	41.00	46.00	68.00	77.00	37.00	75.00	91.08	46.24
Memory Engines												
Mem0	75.10	56.70	72.30	64.20	32.13	57.89	54.68	80.81	52.73	82.25	92.39	65.56
Zep	66.80	42.30	64.20	58.00	35.30	42.10	34.50	50.50	36.40	58.50	82.74	51.94
LangMem	18.80	15.00	20.40	22.70	31.29	8.77	53.24	81.82	40.00	77.45	88.99	41.68
Agent Memory												
A-MEM	67.80	58.30	67.00	56.70	63.01	57.89	54.68	85.86	69.09	62.50	83.26	66.01
MemoryOS	43.90	30.50	41.00	41.50	47.06	56.14	32.37	48.48	47.27	55.75	82.01	47.82
MemOS	77.70	64.60	66.30	73.50	72.72	56.14	58.27	78.79	72.72	82.75	91.47	72.27
Personal Memory												
MemoryBank	43.00	35.20	42.70	38.70	35.29	49.12	51.08	56.57	40.00	64.50	80.66	48.80
EverMemOS	66.80	50.70	59.60	45.90	47.06	47.37	41.01	43.43	56.36	63.25	79.91	54.67
Emotional RAG	82.30	67.10	77.30	73.70	52.94	61.40	44.60	58.59	43.64	64.25	82.00	64.35
Ours												
VoiceMem	82.80	67.30	69.10	74.00	76.47	64.91	51.08	83.84	70.91	83.75	91.63	74.16
VoiceMem (adapted)	81.20	64.70	73.50	71.90	75.19	70.17	73.38	83.94	69.10	85.50	93.57	76.56

The Qwen3-Omni audio-input row scores 68.00 overall, below VoiceMem's textualized-input score of 69.60. This comparison changes both input modality and response model and does not isolate a modality-only effect. Category-level comparisons and controls appear in Appendices G.4 and H.4.

6.4 Response Quality, Context Budget, and Latency

Quality–context trade-off. At the default K=5 operating point, VoiceMem scores 90.40 on LoCoMo using 430 memory tokens and 134 ms mean full memory-system latency under the same text-evaluation setting, without streaming overlap (Figure 3). In the left-brain-only sweep (Figure 4), $K_L = K$ and $K_R = 0$ make K the common total returned-record cap. VoiceMem remains ahead of the external baselines; increasing K generally raises both score and context size. Full operating points and accounting rules appear in Appendix H.5.

Streaming timing. In a matched comparison with 500 paired samples and 1,200 synthetic memories, speculative retrieval reduced median post-endpoint context-ready delay from 27.83 to 0.12 ms; 100% of searches completed by turn end. Preliminary median speech-response onset was 1,478 ms for post-turn retrieval and 1,088 ms for streaming retrieval, each over 500 pairs. A separate Qwen3-ASR comparison on ChatMem-Bench finds text-input and streaming quality within 1.4 percentage points across the four category groups

Table 3: ChatMem-Bench results. Binary-score percentages (↑); Avg. is the micro-average over 500 questions. Measured rows use equivalent textualized audio-event, scene, and emotion evidence. Dark and light blue mark the highest and second-highest scores.

Method	Information				Persona			Affective Attr.			Paraling.&Env.				Avg.
	Upd.	Temp.	Synt.	Abst.	Prof.	Pref.	Impl.	Attr.	Agd.	Phras.	Bsr.	Asi.	Mul.	Asa.	
Reference															
Full-Context	62.07	42.11	68.42	57.89	61.11	47.83	41.18	16.67	57.14	37.78	11.76	22.58	26.92	89.83	47.40
Memory Engines															
Mem0	55.17	31.58	84.21	68.42	66.67	78.26	52.94	16.67	61.90	46.67	11.76	3.23	11.54	93.22	50.20
Zep	37.93	15.79	63.16	63.16	50.00	69.57	35.29	8.33	57.14	33.33	5.88	9.68	7.69	91.53	40.60
LangMem	58.62	21.05	73.68	78.95	61.11	73.91	47.06	41.67	47.62	42.22	11.76	12.90	15.38	93.22	50.80
Agent Memory															
A-MEM	41.38	10.53	68.42	73.68	38.89	73.91	41.18	25.00	52.38	44.44	23.53	9.68	3.85	93.22	45.00
MemoryOS	51.72	36.84	89.47	73.68	44.44	82.61	35.29	25.00	52.38	35.56	5.88	19.35	15.38	94.92	49.00
MemOS	72.41	31.58	52.63	84.21	77.78	78.26	47.06	50.00	71.43	46.67	17.65	12.90	19.23	93.22	55.40
Personal Memory															
MemoryBank	41.38	5.26	47.37	63.16	38.89	52.17	29.41	8.33	42.86	24.44	23.53	6.45	7.69	94.92	36.80
EverMemOS	44.83	26.32	63.16	94.74	50.00	69.57	52.94	33.33	66.67	37.78	11.76	9.68	23.08	94.92	50.20
Emotional RAG	34.48	10.53	57.89	73.68	44.44	60.87	58.82	58.33	52.38	42.22	11.76	3.23	3.85	94.92	46.80
Ours															
VoiceMem	79.31	47.37	84.21	89.47	72.22	86.96	64.71	66.67	76.19	51.11	47.06	45.16	53.85	98.31	69.60
VoiceMem (adapted)	86.21	42.11	84.21	89.47	66.67	86.96	58.82	58.33	66.67	48.89	52.94	51.61	50.00	94.92	67.60
VoiceMem (audio input)	79.31	34.21	84.21	73.68	61.11	82.61	55.88	66.67	80.95	55.56	58.82	54.84	57.69	98.31	68.00

(Appendix H.3.1); quality and latency have not been jointly measured on the same streaming trials. Event definitions and system conditions are reported in Appendix H.3.

6.5 Component Contributions

At K=5, removing the upper-layer index produces the largest reported decrease on ES-MemEval, ChatMem-Bench, and Memora. On LoCoMo, removing emergent clustering or dual-horizon updating produces the largest losses. Right-brain removal slightly increases the score on LoCoMo, but lowers scores on ES-MemEval and ChatMem-Bench, indicating dataset-dependent benefits from persona-affective evidence. These interventions evaluate branch-level evidence organization; they do not establish an independent answer-quality benefit from entity links or cause-reference traversal. The complete ablation figure, numerical results, intervention definitions, and evaluation settings are provided in Appendix H.6.

6.6 Compatibility across Memory Backends

Transfer across backends. Table 4 compares bare and augmented memory stores on LoCoMo at K=5. Augmentation improves all three backends by 27.00–38.26 points, although performance varies across integrations. These results support compatibility under the tested settings; Appendix H.7 gives adapter and transfer details.

Table 4: Backend transfer on LoCoMo at K=5.

Backend	bare	+ ours	Δ
Mem0	52.14	90.40	+38.26
LangMem	44.94	71.94	+27.00
Zep	50.39	85.85	+35.46

"+ ours" adds the schema–entity index and candidate routing while retaining backend storage and retrieval. Within each pair, models, history, retrieval settings, and one-time-prefilled memory are fixed; across backends, upper-layer routing is fixed while storage, indexing, identifiers, and adapter filtering vary (Appendix H.7).

7 Conclusion

We presented VoiceMem, a streaming memory layer that combines factual and persona-affective memory to support real-time interaction. Across the tested factual, persona, and audio-grounded benchmarks, VoiceMem improves memory-dependent response quality while maintaining compact context and traceable evidence. Streaming moves memory preparation into the ongoing utterance, reducing post-endpoint memory delay, although joint quality–latency validation remains outstanding. Integrations with multiple memory backends demonstrate compatibility under the tested settings. Persona claims remain provisional inferences rather than verified persistent traits. All code, models, and datasets will be publicly released.

AI use statement

Generative AI tools were used to assist with selected parts of the data construction, experimental workflow, and preparation of this manuscript, including language editing and improving the clarity of presentation. These tools served only as research and writing aids and were not treated as autonomous contributors. All AI-assisted materials were reviewed, verified, and revised by the authors as appropriate. The authors made all final decisions regarding the methodology, experiments, analysis, and conclusions, and take full responsibility for the accuracy, originality, and integrity of the paper.

Ethics statement

This work adheres to the ICLR Code of Ethics. The datasets and models used in this study were obtained, generated, and processed in accordance with their applicable terms and intended research purposes. As conversational memory systems may retain personal context and infer persona or affective information, they can raise concerns related to privacy, bias, inaccurate attribution, and inappropriate downstream use. Such inferred information should not be regarded as a definitive description of a user, and practical deployments should incorporate suitable safeguards, access controls, transparency, and user oversight. We encourage responsible use of the proposed methods and careful evaluation before deployment in real-world settings.

Reproducibility statement

We have made efforts to support the reproducibility of this work. The main paper describes the proposed architecture, learning framework, datasets, evaluation protocols, baselines, and principal experimental settings, while the appendix provides additional implementation details, data-construction procedures, prompts, ablation configurations, backend integration settings, and latency definitions. The reported model configurations and evaluation conditions are specified alongside the corresponding experiments. We plan to publicly release the code, models, datasets, and relevant training and evaluation resources to facilitate independent reproduction and future research.

References

Pratyay Banerjee, Masud Moshtaghi, Shivashankar Subramanian, Amita Misra, and Ankit Chadha. APEX-MEM: Agentic semi-structured memory with temporal reasoning for long-term conversational AI. In Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 16470–16489, 2026. arXiv:2604.14362.

ByteDance Seed. Seedrealtime: Audio-visual full-duplex llm released, August 2026. ByteDance Seed model release.

Hao Chen, Jiaming Liu, Chenyang Gu, Zhuoyang Liu, Renrui Zhang, Xiaoqi Li, Xiao He, Yandong Guo, Chi-Wing Fu, Shanghang Zhang, and Pheng-Ann Heng. Fast-in-slow: A dual-system foundation model unifying fast manipulation within slow reasoning. In Advances in Neural Information Processing Systems, 2025. arXiv:2506.01953.

Tiantian Chen, Jiaqi Lu, Ying Shen, and Lin Zhang. Es-memeval: Benchmarking conversational agents on personalized long-term emotional support. In Proceedings of the ACM Web Conference 2026, pp. 5810–5821, 2026.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. arXiv preprint arXiv:2504.19413, 2025.

Cheng-Han Chiang, Xiaofei Wang, Linjie Li, Chung-Ching Lin, Kevin Lin, Shujie Liu, Zhendong Wang, Zhengyuan Yang, Hung-yi Lee, and Lijuan Wang. SHANKS: Simultaneous hearing and thinking for spoken language models. In Proceedings of the 64th Annual

Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2026. arXiv:2510.06917.

Konstantina Christakopoulou, Shibl Mourad, and Maja Matarić. Agents thinking fast and slow: A talker-reasoner architecture. arXiv preprint arXiv:2410.08328, 2024. NeurIPS 2024 Workshop on Open-World Agents.

Alexandre Défossez, Laurent Mazaré, Manu Orsini, Amélie Royer, Patrick Pérez, Hervé Jégou, Edouard Grave, and Neil Zeghidour. Moshi: A speech-text foundation model for real-time dialogue. arXiv preprint arXiv:2410.00037, 2024.

Qingkai Fang, Shoutao Guo, Yan Zhou, Zhengrui Ma, Shaolei Zhang, and Yang Feng. Llama-omni: Seamless speech interaction with large language models. In International Conference on Learning Representations, 2025.

Chuanrui Hu, Xingze Gao, Zuyi Zhou, Dannong Xu, Yi Bai, Xintong Li, Hui Zhang, Tong Li, Chong Zhang, Lidong Bing, and Yafeng Deng. Evermemos: A self-organizing memory operating system for structured long-horizon reasoning. arXiv preprint arXiv:2601.02163, 2026.

Le Huang, Hengzhi Lan, Zijun Sun, Chuan Shi, and Ting Bai. Emotional rag: Enhancing role-playing agents through emotional retrieval. In 2024 IEEE International Conference on Knowledge Graph, 2024.

Bowen Jiang, Zhuoqun Hao, Young-Min Cho, Bryan Li, Yuan Yuan, Sihao Chen, Lyle Ungar, Camillo J. Taylor, and Dan Roth. Know me, respond to me: Benchmarking llms for dynamic user profiling and personalized responses at scale. arXiv preprint arXiv:2504.14225, 2025.

Dongming Jiang, Yi Li, Guanpeng Li, and Bingzhe Li. MAGMA: A multi-graph based agentic memory architecture for AI agents. In Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2026. arXiv:2601.03236.

Jeonghyun Kang, Hongjin Kim, and Harksoo Kim. Generation-based and emotion-reflected memory update: Creating the KEEM dataset for better long-term conversation. In Proceedings of the 31st International Conference on Computational Linguistics, pp. 9260–9277. Association for Computational Linguistics, 2025.

Junfeng Lu and Yueyan Li. Dynamic affective memory management for personalized LLM agents. arXiv preprint arXiv:2510.27418, 2025.

Yaxi Lu, Shenzhi Yang, Cheng Qian, Guirong Chen, Qinyu Luo, Yesai Wu, Huadong Wang, Xin Cong, Zhong Zhang, Yankai Lin, Weiwen Liu, Yasheng Wang, Zhiyuan Liu, Fangming Liu, and Maosong Sun. Proactive agent: Shifting llm agents from reactive responses to active assistance. In International Conference on Learning Representations, 2025.

Ziyang Ma, Yakun Song, Chenpeng Du, Jian Cong, Zhuo Chen, Yuping Wang, Yuxuan Wang, and Xie Chen. Language model can listen while speaking. In Proceedings of the AAAI Conference on Artificial Intelligence, 2025.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, pp. 13851–13870, 2024.

Tu Anh Nguyen, Eugene Kharitonov, Jade Copet, Yossi Adi, Wei-Ning Hsu, Ali Elkahky, Paden Tomasello, Robin Algayres, Benoît Sagot, Abdelrahman Mohamed, and Emmanuel Dupoux. Generative spoken dialogue language modeling. Transactions of the Association for Computational Linguistics, 11:250–266, 2023. doi: 10.1162/tacl_a_00545.

OpenAI. Gpt-4o mini: Advancing cost-efficient intelligence, July 2024a. OpenAI model release.

OpenAI. New embedding models and api updates, January 2024b. Introduces the text-embedding-3-small model.

OpenAI. Introducing gpt-live, July 2026. OpenAI product release.

Qwen Team. Qwen3.5-omni technical report. arXiv preprint arXiv:2604.15804, 2026.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. Zep: A temporal knowledge graph architecture for agent memory. arXiv preprint arXiv:2501.13956, 2025.

Xian Shi, Xiong Wang, Zhifang Guo, Yongqi Wang, Pei Zhang, Xinyu Zhang, Zishan Guo, Hongkun Hao, Yu Xi, Baosong Yang, et al. Qwen3-asr technical report. arXiv preprint arXiv:2601.21337, 2026.

Md Nayem Uddin, Kumar Shubham, Eduardo Blanco, Chitta Baral, and Gengyu Wang. From recall to forgetting: Benchmarking long-term memory for personalized agents. In Findings of the Association for Computational Linguistics: ACL 2026, pp. 26814–26841, 2026.

Bandhav Veluri, Benjamin N. Peloquin, Bokai Yu, Hongyu Gong, and Shyamnath Gollakota. Beyond turn-based interfaces: Synchronous LLMs as full-duplex dialogue agents. In Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, 2024. arXiv:2409.15594.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. In The Thirteenth International Conference on Learning Representations, 2025a.

Donghang Wu, Haoyang Zhang, Jun Chen, Xiangyu Tony Zhang, Hexin Liu, Eng Siong Chng, Fei Tian, Xuerui Yang, Xiangyu Zhang, Daxin Jiang, and Gang Yu. Mind-paced speaking: A dual-brain approach to real-time reasoning in spoken language models. arXiv preprint arXiv:2510.09592, 2025b.

Donghang Wu, Haoyang Zhang, Chen Chen, Tianyu Zhang, Fei Tian, Xuerui Yang, Gang Yu, Hexin Liu, Nana Hou, Yuchen Hu, and Eng Siong Chng. Chronological thinking in full-duplex spoken dialogue language models. In Proceedings of the 27th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 2026a. arXiv:2510.05150.

Zhaofen Wu, Hanrong Zhang, Fulin Lin, Wujiang Xu, Xinran Xu, Yankai Chen, Henry Peng Zou, Shaowen Chen, Weizhi Zhang, Xue Liu, Philip S. Yu, and Hongwei Wang. GAM: Hierarchical graph-based agentic memory for LLM agents. In Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2026b. arXiv:2604.12285.

Junlin Xie, Quanlong Zheng, Ruifei Zhang, Kuo Wang, Yanhao Zhang, Jinguo Luo, Haonan Lu, Xiang Wan, and Guanbin Li. Streamrag: Enhancing real-time video understanding with retrieval augmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 38870–38879, 2026a.

Zhifei Xie and Changqiao Wu. Mini-omni: Language models can hear, talk while thinking in streaming. arXiv preprint arXiv:2408.16725, 2024.

Zhifei Xie, Ziyang Ma, Zihang Liu, Kaiyu Pang, Hongyu Li, Jialin Zhang, Yue Liao, Deheng Ye, Chunyan Miao, and Shuicheng Yan. Mini-omni-reasoner: Token-level thinking-in-speaking in large speech models. arXiv preprint arXiv:2508.15827, 2025.

Zhifei Xie, Zongzheng Hu, Fangda Ye, Xin Zhang, Haobo Chai, Zihang Liu, Pengcheng Wu, Guibin Zhang, Yue Liao, Xiaobin Hu, Deheng Ye, Chunyan Miao, and Shuicheng Yan. PASK: Toward intent-aware proactive agents with long-term memory. arXiv preprint arXiv:2604.08000, 2026b.

Zhifei Xie, Zihang Liu, Ze An, Xiaobin Hu, Yue Liao, Ziyang Ma, Dongchao Yang, Mingbao Lin, Deheng Ye, Shuicheng Yan, et al. Audio interaction model. arXiv preprint arXiv:2606.05121, 2026c.

Zhifei Xie, Kaiyu Pang, Haobin Zhang, Deheng Ye, Xiaobin Hu, Shuicheng Yan, and Chunyan Miao. Mega-asr: Towards in-the-wild^ 2 speech recognition via scaling up real-world acoustic simulation. arXiv preprint arXiv:2605.19833, 2026d.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. In Advances in Neural Information Processing Systems, 2025.

Wenyi Yu, Siyin Wang, Xiaoyu Yang, Xianzhao Chen, Xiaohai Tian, Jun Zhang, Guangzhi Sun, Lu Lu, Yuxuan Wang, and Chao Zhang. SALMONN-omni: A codec-free LLM for full-duplex speech understanding and generation. arXiv preprint arXiv:2411.18138, 2024.

Ceyao Zhang, Kaijie Yang, Siyi Hu, Zihao Wang, Guanghe Li, Yihang Sun, Cheng Zhang, Zhaowei Zhang, Anji Liu, Song-Chun Zhu, Xiaojun Chang, Junge Zhang, Feng Yin, Yitao Liang, and Yaodong Yang. Proagent: Building proactive cooperative agents with large language models. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, pp. 17591–17599, 2024.

Rongsheng Zhang, Ruofan Hu, Weijie Chen, Jiji Tang, Junnan Ren, Wanying Wu, Xunuoyan Chen, Tangjie Lv, Tao Jin, and Zhou Zhao. From facts to insights: A persona-driven dual memory framework and dataset for role-playing agents. arXiv preprint arXiv:2605.25693, 2026.

Zheng Zhao, Clara Vania, Subhradeep Kayal, Naila Khan, Shay B. Cohen, and Emine Yilmaz. Personalens: A benchmark for personalization evaluation in conversational ai assistants. In Findings of the Association for Computational Linguistics: ACL 2025, pp. 18023–18055, 2025.

A Retrieval and Streaming Implementation Details

This appendix follows Sections 4.1–4.3: it first specifies factual and persona matching, then query preparation, scheduling and revision handling, and finally context assembly.

A.1 Matching and Neighborhood Semantics

Equation 4 defines entity candidates, whereas Equation 5 maps those candidates to backend record identifiers. The matching and neighborhood rules connect these interfaces to the schema–entity representation.

The implementation details below describe the default Studio path at K=5; benchmark-specific budgets are stated in Section 6.1 and Appendix H.1. Both branches use the text-embedding-3-small encoder, but maintain separate indices and matchers:

Setting	Factual records	Persona/trait nodes
Embedded field	Full fact text	Short trait claim
Matcher	Mem0 cosine search	Normalized dot product (cosine)
Initial selection	$3K$ (15 by default), with full-store fill	Top 4
Similarity threshold	No application threshold (Mem0: 0.0)	0.88 at 384 dimensions; otherwise 0.45
Final limit	At most 5 after deduplication	At most 3 profile items and 5 total right-brain items
Refresh	Re-embed on fact add/update	Embed on creation; evidence-only merges do not re-embed

Factual entity matching is an exception: it first uses exact name_norm matching and then substring matching, rather than the embedding matcher. Aliases are used during entity insertion and deduplication but are not searched at query time; non-person entities may be semantically merged by name at a 0.94 threshold, whereas person and user entities require exact matches. The default query classifier returns two schemas and does not run NER, so runtime entity sets are usually empty unless the caller supplies entities. When the entity set is empty, retrieval routes by schema without entity narrowing or one-hop expansion. Matching runs for each memory-bearing turn and may rerun on partial transcripts; query embeddings are cached only within one retrieval.

When enabled by a nonempty matched entity set, neighbor expansion follows every entity relation with status=active for one hop. Relations are stored directionally as from_entity_id→to_entity_id, but the expansion query checks both endpoints and therefore traverses them as undirected edges. Facts attached to the seed entities and all active one-hop neighbors enter one candidate pool without edge-based weighting, and the final factual vector ranking truncates the results to five. Strong/weak, weight, confidence, and the 30-day relation decay remain stored metadata but do not affect retrieval; there is no per-seed neighbor limit.

Persona retrieval does not traverse the stored rb_evidence.cause_id reference from a retrieved fact. Instead, each query independently matches persona claims by vector similarity. The right branch applies the claim-similarity threshold in the matcher table above, retains at most three profile items, priority-merges these with heartnotes and other right-brain sources, and returns at most five items in total. It does not filter on trait recency, confidence, or evidence count. Rendering selects the latest supporting evidence for each trait, while the separate graph-UI snapshot limit of six nodes per persona slot does not affect response context.

A.2 Streaming Interfaces and Query Preparation

Here, t indexes a streaming observation within the current utterance, rather than the within-session update index j in Section 4.2. The interfaces below describe the available transcript, matched nodes, and candidate records. Their placement within the silence interval is a separate scheduling question, addressed below; the equations do not assert that each operation finishes within a particular time window.

Transcript-dependent matching. During speech, the system obtains a partial transcript (Shi et al., 2026; Xie et al., 2026d) and matches factual and persona nodes using the interfaces in Equation 4 and Equation 8:

$$\begin{aligned} x_{\le t} &= \mathrm{ASR}(a_{\le t}), \\ (\mathcal{V}_t, \mathcal{S}_t) &= \mathrm{Match}(x_{\le t}, \mathcal{V}, \mathcal{S}), \\ (\mathcal{V}_t^I, \mathcal{V}_t^C) &= \mathrm{Match}(x_{\le t}, \mathcal{V}^I, \mathcal{V}^C). \end{aligned} \tag{11}$$

Here, $a_{\le t}$ is the audio observed so far and $x_{\le t}$ is its current transcript. The two Match calls return subsets of the factual entity/schema sets and the independent/cross-entity persona-node sets, respectively; they do not return backend memory records. The supplementary streaming quality evaluation uses Qwen3-ASR (Appendix H.3.1). The Studio interface documented here has legacy ASR integrations: Chinese streaming ASR uses FunASR 1.4.14 with paraformer-zh-streaming (iic/speech_paraformer-large_asr_nat-zh-cn-16k-common-vocab8404-online); English uses the June 26, 2023 Sherpa-ONNX Streaming Zipformer. The browser sends approximately 20 ms frames of 24-kHz mono PCM16 audio, which the server resamples to 16 kHz. Paraformer decodes every 9600 samples (approximately 600 ms), while Zipformer consumes the 20 ms frames continuously.

Transcript updates carry the complete current hypothesis with replace=true, rather than token-level edits, and speculative memory matching is throttled to at least 0.8 s between runs or six newly accumulated characters. At utterance end, the Chinese stream is flushed with approximately 50 ms of silence and is_final=true; a July 17, 2024 Sherpa-ONNX SenseVoice pass is also given up to 80 ms to replace the streaming transcript. Final ASR correction

can finish before retrieval begins. A corrected transcript available at that point replaces the partial query, and affected matching results and candidate pools are refreshed before ranking. ASR finalization is therefore part of streaming query preparation, rather than a display-only update after retrieval.

Speaker-conditioned memory access. Speaker identification is represented as $p = \psi(a)$, where a is the completed utterance audio, p is the estimated speaker identity, and ψ is the identification function. This post-turn result is used for attribution and subsequent access gating. Speaker verification uses the 3D-Speaker Chinese ERes2Net model 3dspeaker_speech_eres2net_base_sv_zh-cn_3dspeaker_16k.onnx. Sherpa-ONNX extracts 192-dimensional, L_2-normalized embeddings from 16-kHz mono audio using 3.0 s windows with a 1.5 s stride, then averages and renormalizes them. Utterances shorter than 2.5 s, valid windows shorter than 1 s, or audio with RMS below 0.005 are not assigned a reliable identity. The maximum cosine similarity to an identity's sub-centroids accepts an existing identity at 0.50, creates a linked temporary candidate in $[0.40, 0.50)$, and creates a new identity below 0.40; candidate identities are normally auto-merged at 0.65. A self-declared identity pinned to the session is preferred above 0.40.

Identity does not select a separate memory space: the session configuration selects one shared space, and recognized identities are stored as speaker:<person_id> tags. Although the backend can filter on these tags, the current Studio response path instead uses an owner/stranger gate: the first recognized speaker becomes the session owner, and after a different speaker is confirmed, subsequent responses receive no memory context from that space. Identification runs after utterance end during background memory writing, so it affects later turns rather than rerunning the current turn. If identity is unavailable, the existing gate state is retained; before an owner is established, access remains open.

Query representation and candidate pools. Let q_t denote the textual query derived from the available transcript, with $\mathbf{h}_t$ denoting an embedding computed internally by a backend when needed. Factual and persona candidate construction are independent in the current retrieval path. This specialization reflects their different evidence-organization needs; shared evidence identifiers and optional entity attribution provide traceability, while the response model receives the two branches' assembled claims and supporting records. The conversion from matched nodes to record-identifier pools is written as

$$\mathcal{C}_t^L = \bigcup_{z \in \mathcal{Z}_t} \mathcal{I}_z, \qquad \mathcal{C}_t^R = \bigcup_{z \in \mathcal{Z}_t^R} \mathcal{I}_z. \tag{12}$$

For either node type, $\mathcal{I}_z$ denotes its supporting record identifiers, following Equation 5 and Equation 8. The pools $\mathcal{C}_t^L$ and $\mathcal{C}_t^R$ are passed to the backend search stage in Equation 10; they are not yet the retrieved memory items.

The backend query is the accumulated transcript itself, without LLM rewriting; the only expansion appends an absolute date when the transcript contains a relative expression such as "tomorrow" or "next week." The system does not explicitly precompute or pass $\mathbf{h}_t$: each backend receives query text and computes its vector internally as needed. The factual branch also receives candidate record identifiers, while the persona branch matches persona claims independently and uses affect and dialogue context. Identical query embeddings are cached by model and text within a retrieval, and their cost is included in classification or search time rather than measured separately.

Candidate-ID pools have no explicit size cap, while final retrieval returns at most five factual and five persona items by default. An empty factual pool falls back to full-store vector search; if fewer than five eligible records are found, the system fills from the full store, subject to the session owner/stranger gate. This upper-layer empty-pool fallback and full-store fill policy is applied consistently across all evaluated backends, although their internal ranking and filtering implementations may differ. Deleted, archived, or missing identifiers are ignored without error, and there is no separate refresh mechanism for a small or stale pool. A sufficiently changed partial transcript rebuilds and replaces the speculative candidate set. An end-of-turn ASR correction can also complete before retrieval and updates the query and affected candidates at that stage.

A.3 Scheduling, Revisions, and Timing Scope

The silence interval in Figure 2 begins at the end of detected speech, distinct from the streaming observation index t. Candidate preparation, backend search, and context assembly have different completion points; only the last determines when the assembled memory context is available.

The streaming and non-streaming controllers call the same retrieval functions with the same budgets and ranking rules; streaming changes when those calls start. Final ASR correction can complete during preparation before retrieval; the corrected query is then used to prepare candidates and rank records. Once preparation is complete, only unfinished final ranking and context assembly remain after endpoint confirmation. For the same transcript and memory state, all memory computations are identical to text-input execution; streaming adds no approximate retrieval or reduced computation. The only potentially lossy input transformation is ASR, and recognition errors are outside the memory layer's control. Appendix H.3.1 reports the Qwen3-ASR text-input versus streaming quality comparison. The bounded-wait fallback described below is a separate deployment policy, and the timing study does not jointly score answers under that fallback.

The implemented stage dependencies are:

Operation	Start event	Prerequisite	Concurrency
VAD/EOT	Updated on every audio frame; endpoint testing begins after configured silence	Current audio frame and ASR state	Concurrent with ASR and speculative retrieval
Query preparation	A partial transcript reaches the minimum length/change condition	Current ASR hypothesis	An outdated partial can cancel the prior task
Date expansion	A backend search begins	Current query text	Synchronous within that query/search
Candidate collection	Query classification and entity/schema/persona matching finish	Matched nodes and query	Precedes backend search
Backend search	Candidate IDs are ready, or empty-pool fallback is selected	Current query, incorporating ASR corrections available before retrieval, and candidate pool	Factual and persona retrieval run concurrently
Context assembly	Both searches finish or reach the finalization policy	Returned factual/persona items	Precedes response generation

Speculative retrieval normally starts after at least six characters of partial transcript or approximately 200 ms of silence. The default VAD endpoint-confirmation ceiling is approximately 300 ms. When eot_min_s=0.1 is enabled, endpoint testing requires at least 100 ms of silence; its default is zero, so this is not a universal wait. The figure's stage intervals are illustrative and are not measurements of backend or context-ready duration.

Input revisions are handled as follows:

Event	Behavior
ASR correction	A throttled new partial restarts speculative retrieval and stale results are discarded. Final correction can complete before retrieval; the corrected transcript replaces the partial query and refreshes affected candidates before ranking.
Speech resumes	The current speculative task and its uncommitted candidates/results are cancelled; retrieval restarts from a later partial transcript.
Speaker identity changes	Identification completes in the post-turn background stage and does not recompute the current turn; the new gate state normally applies from the next turn.
Search unfinished at endpoint	The system waits within the configured final-ASR/search budget, then uses a completed streaming result valid for the current query and cancels unfinished work; if nothing is reusable, the memory context is empty.

The response uses only memories retrieved for the current turn's accumulated transcript from the current memory space; future speech is excluded. The timing comparison does not separately characterize these revision combinations or measure their answer-quality loss.

Observed timeout behavior. If the configured waiting budget expires, the controller cancels unfinished retrieval and may answer with empty memory when no completed result is reusable. In the reported streaming experiments, timeout-triggered retrieval cancellation and timeout-induced empty-memory fallback each had an observed occurrence rate of 0%. This observation concerns the final timeout policy, not cancellation of stale work after a transcript revision or resumed speech. It also differs from a search being unfinished at turn end, which can still complete within the subsequent waiting budget (Appendix H.3).

A.4 Context Assembly

Equation 10 separates record retrieval from the assembly of memory context R_t. The candidate-ID sets $\mathcal{C}_t^L$ and $\mathcal{C}_t^R$ are not prompt contents: assembly consumes the returned memory items and the optional description set $\mathcal{D}_t$.

Full-system experiments use $K_L = K_R = K$, with no additional cross-branch cap. Description tokens count toward context size but not either item budget. The left-brain-only sweep instead uses $K_L = K$, $K_R = 0$ (Appendix H.5).

The final prompt combines long-term memory context, current-turn control instructions, and the current transcript. It is assembled from returned records rather than candidate identifiers: factual records retain final retrieval order and are truncated to five, while persona records are priority-ordered and truncated to five. The two branches remain in separate sections, and factual duplicates are removed during ranking rather than reordered during assembly. With no hits, the system uses its configured no-memory instruction; for a speaker gated as a stranger, it instead clears memory context and uses the stranger instruction.

The current Studio path does not inject $\mathcal{D}_t$ as separate factual records. When the semantic-graph path is enabled, graph descriptions appear as an appendix after the factual records. The response-model input has the following structure:

```
messages = [
{"role": "system", "content": <Studio system prompt>},
...<bounded session history>...,
{"role": "user", "content":
"<memory_context and dialogue directives>\n\n"
"<current ASR transcript>"}
]
```

A representative memory_context and dialogue directives field is:

```
factual memory CONTEXT you know about the user (top5 in left brain):
- [2026-08-20] User is considering leaving the current job.
- [2026-07-11] User has interviewed with two companies.
```

user's emotion & characteristics (top5 in right brain):
- User prefers cautious, evidence-based advice.
Let these shape your tone, what you bring up, and what you leave alone.
Never state them back to the user.

<optional replay/tone/unfinished-turn instruction>

Candidate pools and record identifiers are therefore not prompt content. The prompt contains selected record text, required temporal markers, persona instructions, and the current transcript; identifiers are retained only for internal traceability. If no factual record is relevant, the corresponding section states "No relevant left-brain memories" rather than fabricating content.

B Neighbor-Evidence Representation

This appendix analyzes the neighboring-schema choice introduced in Section 4.1: injecting a compressed description, expanding the neighbor into records, or omitting it. Candidate-pool size, prompt tokens, and response score are reported separately.

B.1 Compression, Expansion, or Omission

For a strongly linked neighboring schema, the system can add a short description to the prompt or expand the neighbor into individual memory items. Table 5 compares these choices with omitting neighbors.

All three neighbor-handling conditions use the same split, queries, retrieval and response models, scoring procedure, and default K=5; the intended intervention is the representation of neighboring evidence. The omission condition uses only records from the primary schema, the description condition adds a short neighboring-schema description without its individual records, and macro-expansion adds the neighbor's records to the common ranked candidate pool.

Pool is the number of candidate records before backend ranking: the union of primary-schema candidates and, where applicable, expanded-neighbor candidates. It is distinct from the number of records injected into the prompt. Metrics are first computed per query and then averaged over the evaluation split; repeated runs of a query, when present, are averaged before the query-level average. The description row uses the reference setting for this comparison, with the description injected as schema/graph context and excluded from the record candidate pool.

Compression and macro-expansion have similar reported scores on LoCoMo (90.4 vs. 91.10), whereas compression is 2.76 points better on ES-MemEval. Omitting neighbors lowers the reported score by 2.6 and 9.1 points, respectively. The description variant has a smaller mean candidate pool than macro-expansion (144.1 vs. 165.7 on ES-MemEval), but uses more memory tokens (624.4 vs. 497.6); the same trade-off appears on LoCoMo. These comparisons distinguish ranking-pool size from final context cost and do not isolate their individual contributions to response quality.

Table 5: Neighbor-handling strategies. Pool is the mean number of items entering final ranking.

Dataset	Neighbour handling	Acc.	Pool	Mem. tok.
LoCoMo	description (ours)	90.40	292.0	430
	macro-expansion	91.10	306.0	358.8
	w/o macro	87.80	292.0	427.2
ES-MemEval	description (ours)	67.45	144.1	624.4
	macro-expansion	64.69	165.7	497.6
	w/o macro	58.39	144.8	498.1

C Factual and Persona Memory Updates

This appendix follows the update paths in Sections 4.1 and 4.2. It describes factual index initialization and consistency first, followed by persona records, short-horizon affect, and long-horizon trait maintenance.

Index initialization and membership. The index is initialized lazily when the first record is written rather than by pre-enumerating the store. Each memory space begins with seven seed schemas: work, finance, relationships, health, goals, daily_life, and knowledge. During writing, GPT-4o-mini produces a structured object with top-level memory, emotion, and traits fields. Each factual item in memory contains an identifier, fact text, attribution, linked-memory identifiers, slot, entities, and relations. An entity specifies its name, type, and role; a relation specifies its source, target, type, and confidence. Outputs that cannot be parsed are assigned to knowledge. The complete write-time extraction prompt and output schema are provided in Appendix K.2.

Extracted entities store a canonical name, aliases, description, and embedding, while schema membership and inter-entity semantic relations are stored separately. When a new surface name differs from an existing entity name but has high embedding similarity to that entity, it is stored as an alias of the existing entity; otherwise, a new entity is created. The exact alias-acceptance threshold is not recorded. Entity descriptions and entity relations are generated using the prompts in Appendices K.1 and K.1.1, respectively. Query-side classification uses text-embedding-3-small. A factual record may support multiple entities through a record-ID–entity-ID association table without duplicating its text.

Factual updates and cross-store consistency. The configured GPT-4o-mini extractor emits structured facts, entities, schemas, and relations and selects one of add, update, delete, or none, after which the index updater writes both the factual store and cognitive graph. add creates missing entities, aliases, memberships, and relations. By default, update adds a new record rather than overwriting the old one; overwriting is enabled only when VOICEMEM_APPLY_UPDATE=1. delete or archival removes the record from the searchable set so its index associations no longer produce hits, while none leaves uninvolved records and nodes unchanged. Each update is written in the following order: (1) vector store, (2) JSON mirror, and (3) cognitive graph. These writes are sequential and are not enclosed in a cross-store atomic transaction. The implementation provides no automatic rollback or recovery mechanism when a later write fails, and it does not isolate concurrent reads from an in-progress update; consequently, a reader may observe a partially committed state in which one store has been updated while another has not. Conflicting facts are retained with temporal metadata for as-of reasoning rather than forcibly overwritten, and unsupported graph nodes are not immediately garbage-collected. The exact extraction prompt and complete output schema are provided in Appendix K.2, with the core JSON structure also reproduced below.

fact extraction json

```
{
  "memory": [{
    "id": "0",
    "text": "extracted fact",
    "attributed_to": "user",
    "linked_memory_ids": [],
    "attribute": "",
    "slot": "work",
    "entities": [],
    "relations": []
  }],
  "emotion": "",
  "traits": []
}
```

fact update json

```
{
  "memory": [{
    "id": "memory ID",
    "text": "updated fact",
    "event": "ADD|UPDATE|DELETE|NONE",
    "old_memory": "old fact"
  }]
}
```

Persona nodes and cross-brain associations. Equation 7 separates the two persona-node collections from the factual entities they may reference. The association symbol $\ell_{v,e}$ is distinct from the query-coherence score $\rho(H)$ in Section 4.1. The right-brain extractor stores each persona node as a claim/trait record with a unique identifier, textual description, embedding, confidence, and creation/update times. Claims are merged by embedding cosine similarity using text-embedding-3-small with a threshold of 0.95; a match appends supporting memory or heartnote identifiers, whereas a non-match creates a new node. A factual target, when present, is stored in a separate entity-anchor/association field. Persona text, supporting identifiers, and entity anchors are therefore distinct stored fields rather than literal encodings of $\ell_{v,e}$ or $\rho(H)$. The following illustrative record shows how an explicitly stated general preference can be represented. Its claim preserves the scope of the user's self-report, and cause_id identifies supporting evidence. This schema illustration does not establish longitudinal validity. No entity-anchor field is present, so it represents an unanchored persona node.

Illustrative persona record for an explicit general preference

```
{
  "id": "trait_anon_001",
  "slot": "handle way",
  "claim": "The user reports generally preferring to finish speaking before receiving advice about work
↪ stress.",
  "confidence": 0.9,
  "evidence": [
    {
      "quote": "In general, when I talk about work stress, I prefer to finish explaining before receiving
↪ advice.",
      "emotion": "annoying",
      "cause": "work pressure",
      "cause_id": "memory_anon_017",
      "at": "2026-08-20T14:30:00+00:00"
    }
  ]
}
```

Short-horizon affect and attribution. Equation 9 uses j as a within-session update index and $\mathbf{a}_j$ as an affect representation, distinct from the factual entity e in Equation 7. Short-horizon affect combines the current utterance with any available acoustic-emotion result. The internal representation contains an emotion label, valence, arousal, intensity, and confidence, with optional analysis text and acoustic/semantic evidence; uncertain text-side predictions remain empty. Modify denotes the right-brain update procedure rather than a separate model. The acoustic front end uses HeuristicWavVADEstimator, while salient-reaction attribution can use the exact model identifier Qwen/Qwen2.5-Omni-3B; text-side attribution otherwise uses GPT-4o-mini by default. The procedure does not rewrite factual records. It upserts a heartnote containing the utterance, affect fields, timestamp, supporting fact identifiers, and an entity anchor when one is available. A characteristic judged worth retaining by the per-turn model may additionally update a candidate persona claim and link the turn as evidence; this is not an independently verified stability judgment. With no target entity, the heartnote is retained without an anchor; a new session begins with empty short-horizon state. The first JSON object below gives the structured emotion-attribution

format, followed by an illustrative one-turn request and its intended scoped representation. The request supports a current need to be heard, not a stable preference. The example illustrates the revised prompt rule; it is not a measured demonstration of the extractor's reliability.

Emotion-attribution output


```
{
  "analysis_text": "reason of the emotion",
  "emotion": {
    "label": "frustration",
    "valence": -0.7,
    "arousal": 0.6,
    "intensity": "high",
    "confidence": 0.8
  },
  "acoustic_evidence": ["speed decrease", "obvious palse"],
  "semantic_evidence": ["mention work is stressful"],
  "related_nodes": [
    {
      "local_id": "topic_1",
      "name": "main experiment",
      "node_type": "Topic",
      "metadata": {}
    }
  ],
  "graph_delta": {
    "nodes": [],
    "edges": []
  },
  "retrieval_snippet": ["main experiment", "frustration"]
}
```


Illustrative input: a request scoped to the current turn


```
{
  "asr_text": "This project is stuck and I am really stressed. Please listen to me first right now.",
  "vad": {
    "valence": -0.72,
    "arousal": 0.66
  },
  "left_memory_id": "mem_anon_017",
  "entities": ["project_anon_03"],
  "observed_at": "2026-08-20T14:30:00+00:00"
}
```


Illustrative heartnote for the current request


```
{
  "id": "heartnote_anon_004",
  "memory_class": "heartnote",
  "content": "This project is stuck and I am really stressed. Please listen to me first right now.",
  "metadata": {
    "emotion": "stressful",
    "entities": ["project_anon_03"],
    "left_memory_id": "mem_anon_017",
    "inner_os": "The user is stressed about the project and asks to be heard in this turn.",
    "agent_reply": ""
  },
  "evidence_memory_ids": ["mem_anon_017"],
  "created_at": "2026-08-20T14:30:00+00:00"
}
```

Illustrative attribution decision: no persistent trait


```
{
  "significant": true,
  "assistant_helped": null,
  "user_reaction": "The user asks to finish speaking before advice in this turn.",
  "why": "The reply describes a current need under project stress; it does not state a general preference
↪ or supply repeated observations.",
  "user_trait": null
}
```


Per-turn persona-claim maintenance and retention. There is no independent periodically executed Consolidate function or recurrence threshold. Long-horizon maintenance uses GPT-4o-mini during per-turn reaction learning and can also run at a session boundary or on an explicit Flush(). The attribution step receives the preceding assistant response, current user reply, and available emotion information and returns significant, assistant_-helped, user_reaction, why, and an optional user_trait. A candidate persona claim is created or merged when the attribution model judges the reaction significant and supportive of such a characteristic; new evidence and timestamps are appended. The prompt does not receive the full longitudinal evidence sequence. Neither this write rule nor the illustrative examples establish prevention of one-off reaction promotion. Single-reaction false-promotion rates, cross-context generalization, preference changes, and contradiction handling require separate evaluation beyond final-answer scores. The exact long-horizon attribution prompt and JSON schema are provided in Appendix K.3, and the contradiction-cleaning prompt appears in Appendix K.4. The verified record below demonstrates one supersession case while retaining the older heartnote. Its Chinese memory text is translated into English for presentation, and the identifiers are anonymized sample identifiers. This single record does not establish the general accuracy of contradiction resolution, which was not independently evaluated.

Verified contradiction-cleaning input and decision


```
[12] ID:a91f23c8 | Emotion:happy | Entity:coffee shop |
My favorite coffee shop is Common Man Coffee Roasters.

[43] ID:d72be519 | Emotion:happy | Entity:coffee shop |
My current favorite is Apartment Coffee; I rarely visit Common Man now.

{
  "delete_ids": [],
  "supersede": [
    {
      "old_id": "a91f23c8",
      "new_id": "d72be519"
    }
  ]
}
```


Older heartnote after supersession


```
{
  "id": "a91f23c8-4d31-4e82-91fd-26a5c174bf01",
  "memory_class": "heartnote",
  "content": "My favorite coffee shop is Common Man Coffee Roasters.",
  "metadata": {
    "emotion": "happy",
    "entities": ["coffee shop"],
    "superseded_by": "d72be519-7f42-4671-a093-184da86f5902",
    "superseded_at": "2026-09-24T16:55:00+00:00"
  }
}
```

D Cluster Subgraph Emergence Algorithm

This appendix expands the query-driven emergence criterion in Section 4.1, proceeding from activation sets to candidate selection, LLM validation, and schema promotion.

Activation sets and candidate scope. Equation 6 averages candidate–activation overlap over $\mathcal{Q}$. Its interpretation therefore depends on both the activation rule and the selection of historical queries. For a query q, A_q contains only entities directly activated by query classification and entity matching, excluding one-hop expansion. The set $\mathcal{Q}$ contains all completed queries from the current session rather than a fixed number of queries. Emergence is triggered at session end; activation results are keyed by session identifier, and queries from later sessions do not enter the current $\mathcal{Q}$. The implementation does not persist a separate trigger-time snapshot.

Candidate selection and validation. In Algorithm 1, G_c denotes the input cluster graph, $\mathcal{H}$ the generated candidate subsets, and $H^\star$ the selected subset. The criterion maximizes $|H|$ among the connected candidates produced by greedy peeling that satisfy $\rho(H) > \alpha$. Thus, coherence is an eligibility constraint, while entity count is the selection objective; the maximum is over the generated candidates, not all possible subgraphs.

Algorithm 1 Cluster Subgraph Emergence Pipeline

Require: Cluster G_c, query set $\mathcal{Q}$, coherence threshold α
$\mathcal{H} \leftarrow$ ConnectedCandidates(G_c)
for each connected component $H \in \mathcal{H}$ do
    repeatedly remove a minimum-degree entity and retain the largest generated connected subset satisfying $\rho(H) > \alpha$
end for
$H^\star \leftarrow \arg\max_H |H|$
    subject to $\rho(H) > \alpha$ among retained connected candidates
if $H^\star$ exists then
    $a \leftarrow$ LLMJudge($H^\star$)
    ▷ Existing topic: validate relevance only
    ▷ New topic: validate relevance, importance, then completeness
    if a then
        Promote($H^\star$)
    else
        DisablePotential($H^\star$)
    end if
end if

G_c is the entity co-occurrence graph induced by the memory identifiers returned by one actual retrieval: entities are nodes and co-occurrence within a memory creates an undirected edge. ConnectedCandidates first traverses all connected components, discarding components with fewer than three entities or with no splittable entity. Within each component, the full component is the initial candidate; the procedure then repeatedly removes a minimum-degree entity and retains, among the generated connected candidates satisfying $\rho(H) > \alpha = 0.05$, the candidate with the largest entity count $|H|$. After processing all components, it selects the largest retained candidate across components. At either selection stage, equal-size candidates retain the first candidate encountered in traversal order. Thus, ρ is used only as an eligibility threshold and is not itself maximized. Empty candidates or candidates with $\rho(H) \leq \alpha$ are skipped without disabling them and may be retried later. GPT-4o-mini performs the LLM validation: a candidate matching an existing validated topic requires only the relevance check and is merged into that schema, whereas a new topic must pass relevance, importance, and completeness before a new schema is created. The three judge prompts and their JSON output schemas are provided in Appendix K.5. Algorithm 1 has been checked line by line against this implementation control flow.

Promotion and the resulting index. Promote neither copies nor deletes source memories. It creates a dynamic schema with a name, description, and parent; updates each candidate

Table 6: Cluster-maintenance results on ES-MemEval P1 (252 questions) at K=5. Differences are computed from unrounded accuracies and reported in percentage points; random_split matches the number of emergence splits.

Strategy	Correct	Acc. (%)
static (no split)	178	70.63
size threshold	181	71.83
random split	169	67.06
emergence (ours)	187	74.21

entity's slot_ref; and appends the new schema-membership tag to related memories while retaining their prior tags. Record text, identifiers, and entities remain unchanged. If the candidate matches an existing topic, it is merged into that schema instead of creating a new one. DisablePotential is called only after explicit LLM rejection and marks the candidate entities as ineligible for later splitting. Candidates with $\rho(H) \leq \alpha$ are not disabled and may be reconsidered after further co-occurrence evidence; there is no additional expiration timer or retry limit.

E Cluster Emergence Evaluation

This appendix evaluates the emergence procedure specified in Appendix D against static and heuristic schema-maintenance policies. All conditions use ES-MemEval P1 (32 sessions, 252 questions), the same queries, retrieval and response models, scoring rule, and K=5.

Strategy	Grouping rule	LLM filtering
emergence (ours)	Selects the largest generated connected candidate satisfying $\rho(H) > 0.05$.	Checks relevance; new topics also require importance and completeness.
static_maintenance	Retains the seed schemas and creates no new schema.	Not applicable.
random_split	Creates the same number of approximately size-matched groups with randomized assignment.	None.
size_split	Splits schemas exceeding 40 entities into two insertion-order groups.	None.

Emergence achieves the highest observed accuracy at 74.21% (187/252), exceeding static maintenance by 3.57 percentage points (9 additional correct answers), random_split by 7.14 points (18 answers), and size-threshold splitting by 2.38 points (6 answers). Size-threshold splitting also improves over static maintenance, whereas random splitting reduces accuracy. This pattern suggests that the grouping strategy matters beyond the number of splits. These are single-run results without uncertainty estimates, so statistical significance is not established. Matching the number of splits does not fully control group size, trigger timing, LLM filtering, or context cost. Because only emergence uses semantic validation, the comparison supports the complete maintenance pipeline rather than isolating the contribution of the coherence criterion.

F Audio-Grounded Memory Representation and Access

This appendix follows Section 4.4: it specifies retained acoustic metadata, integration with text-based retrieval, explicit playback access, and evaluation modality boundaries. Speaker gating and persona attribution remain in Appendices A and C.

Representations, provenance, and retention. In the online speech interface, each confirmed turn begins as 24-kHz mono PCM16 audio from the browser and is archived by the

server as a 16-kHz, mono, 16-bit PCM turn_<random>.wav file. The supplementary ChatMem-Bench streaming quality evaluation uses Qwen3-ASR. The legacy Studio interface described here uses Paraformer for Chinese or Sherpa-ONNX Zipformer for English; speaker verification uses 3D-Speaker ERes2Net; and acoustic-affect processing is described in Appendix C, including the reported HeuristicWavVADEstimator and Qwen2.5-Omni-3B attribution path. The current code does not record a unified software-version manifest for these components, so reproducibility is limited to the model files and sampling format.

Each memory stores its transcript, observed_at timestamp, speaker label when identified, emotion label, and associated metadata. The audio archive separately stores a memory_id→audio_path mapping. Acoustic embeddings, intermediate ASR state, frame-level VAD sequences, and temporary resampling buffers are discarded rather than indexed. If an archived WAV expires, its mapping may remain but is treated as unavailable.

Graph integration and retrieval interfaces. Audio is not a separate vector-retrieval backend. Search uses memory text and entity, emotion, and speaker labels, with query vectors computed by the text encoder. After a particular memory_id is retrieved, the audio archive resolves its WAV only for an explicit playback request in the described response path; the waveform is not used for text-similarity ranking or automatic post-retrieval acoustic analysis.

For example, memory_id=m42 may store the text "The user discussed plans to change jobs on 2026-08-20," observed_at=2026-08-20, speaker:user_1, and emotion=anxious, while the archive stores (m42, user_1, .../turn_abc123.wav). When a text query retrieves m42, the response model receives the transcript and structured memory context, not the path, waveform, or acoustic embedding. Only an explicit playback request dereferences the archived WAV.

Experimental modality configurations and accounting. For the ChatMem-Bench text-input rows, historical dialogue and queries are supplied as text, with audio events, background/acoustic scenes, and emotion labels textualized in the common input. This benchmark interface is distinct from the online waveform-processing path above: baselines and VoiceMem receive the same substantive evidence before their respective memory construction. The common per-turn memory-update schedule and response-model inputs are given in Appendix H.1. Token, storage, and latency accounting are specified in Appendix H.3; the representation described here does not imply that all of these costs are included in a token count.

G Model Adaptation and Data Construction

This appendix follows Sections 5.1 and 5.2. It describes teacher supervision and model updates, evaluated checkpoints, memory-world and conversation construction, benchmark curation, and train–evaluation separation in that order.

G.1 Distillation, Verification and Model Updates

For each example, the qwen3.7-plus teacher receives the dialogue history, factual and affective memory, and the student's draft response, then produces a revised SFT target. Training proceeds iteratively: after each block of 8×32 examples, the updated student generates drafts for subsequent supervision. The data cover knowledge, emotion, persona, and casual conversation, although exact per-category counts are not recorded here.

The construction workflow compares responses with different memory-use behaviors and filters generated conversations for necessity, faithfulness, and quality. Its SFT update feeds model changes into subsequent generation. Teacher supervision, verification, and optimization serve distinct roles and need not be performed by the same model.

The adapted response model uses Qwen3-Omni-30B-A3B-Instruct as its base and is trained for two epochs with 4-bit QLoRA (rank 32, alpha 64, dropout 0.05). The standard SFT objective is token-level cross-entropy over the retained teacher-generated targets. The teacher supplies generated text only; probabilities, logits, and model weights are not exposed to the student. Optimization uses a learning rate of 2×10^{-4}, cosine scheduling, a 0.03

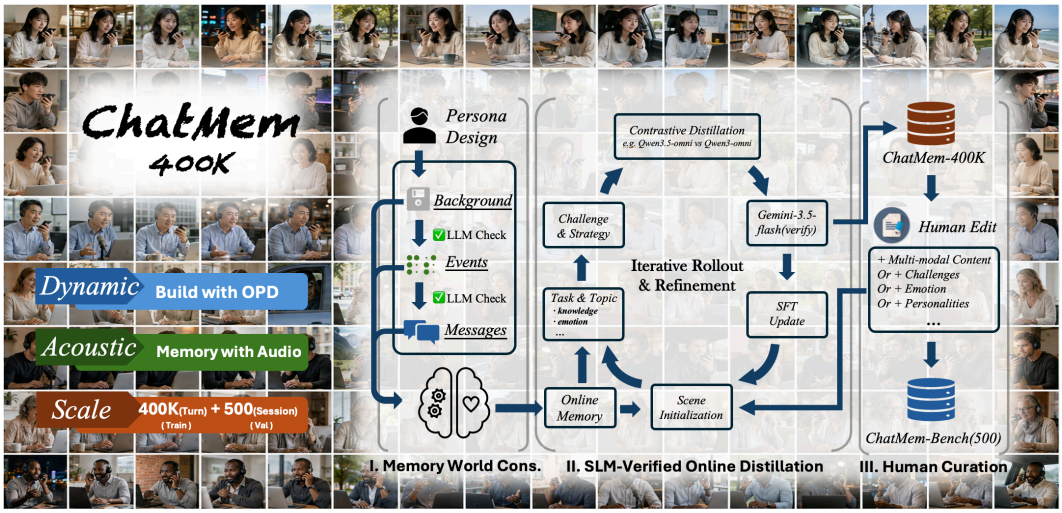

Figure 5: Data construction and model-adaptation workflow for VoiceMem. Memory worlds provide histories for persona, scene, dialogue, and TTS generation, followed by automated verification and human inspection. ChatMem-400K contains approximately 400,000 single-turn training examples organized within multi-turn sessions, while the final ChatMem-Bench evaluation contains 500 sessions with exactly one evaluation question per session (500 questions). The figure's "Val" label denotes this common evaluation set.

warm-up ratio, BF16, gradient checkpointing, a maximum sequence length of 2048, and seed 42. The per-process batch size is eight with two gradient-accumulation steps, giving an effective per-process batch size of 16. The number of training processes, the exact trainable/frozen module list, and the example-reuse schedule beyond the stated two epochs were not recorded. Training uses neither early stopping nor selection across checkpoints; reported adapted-model results use checkpoint-3318, the checkpoint saved at the end of the second epoch.

Conversation-skeleton generation and hallucination verification use Gemini-3.5-flash, whereas qwen3.7-plus generates the teacher responses. For each teacher response, generation is attempted at most three times. Rule-based checks reject stock AI phrasing and compare the response with the preceding six turns using character-bigram Jaccard similarity, with 0.75 as the repetition threshold. If all three attempts fail, the final attempt is retained with a warning; entries with empty responses are removed when the SFT examples are assembled. The temperature-zero Gemini-3.5-flash verifier checks for unsupported entities, and a detected unsupported entity causes the entire session to be discarded. Post-training checks cover formatting, unfilled template fields in generated responses, verbatim memory restatement, fabricated entities or numbers, truncation, repetition, and memory-information coverage. In addition, 5% of the generated samples are manually inspected for response naturalness, reasoning quality, and emotional appropriateness. The conversation-skeleton and multi-turn generation prompts, including their output schemas, are provided in Appendix K.6. The Gemini-3.5-flash identifier in Figure 5 matches the logged verifier name.

G.2 Evaluated Checkpoints

The default response model is GPT-4o-mini. Rows labeled VoiceMem (adapted) use the same model based on Qwen3-Omni-30B-A3B-Instruct, with the local training identifier checkpoint-3318. No completed Qwen2.5-Omni or Step-Audio2-Mini adaptation contributes to the reported results; those variants are not evaluated.

G.3 Memory Worlds and Conversation Construction

Synthetic memory worlds provide a persistent history against which memory use can be assessed. A core persona is elaborated into background information, life events, messages, and memory records: Persona → Background → Events → Messages → Memory. Conversation construction combines task and topic selection, scene context, and challenges involving recall, reference resolution, contradiction, and multi-memory reasoning. These elements determine what information a response needs from the user's history.

Construction follows a text-first, TTS-second pipeline. An LLM generates a user persona and associated work/life memory scenarios, then produces questions, reference answers, scoring criteria, and required evidence for each capability label. The evidence is embedded into multi-turn user–assistant conversations; affect labels control later speech synthesis, and environmental events are inserted at specified positions. Generated samples undergo structural validation, evidence-position checks, answerability with complete evidence, unanswerability after evidence removal, affect-label ablation, and human spot checking before non-overlapping evidence units are assembled into multi-session examples. Qwen3-TTS-12Hz-1.7B-CustomVoice synthesizes speech from the affect instructions, after which environmental audio is mixed on the timeline. The four generation labels—factual personal memory, affective attribution, environmental context memory, and multi-character conversational memory—are mapped into the four dimensions and 14 categories reported in Table 3. The resulting ChatMem-Bench evaluation contains 500 sessions, each contributing exactly one evaluation question, and 53 hours of audio. For the measured text-input comparisons, the dialogue, audio-event and scene descriptions, and emotion labels form a shared textualized input supplied to all systems; the existence of synthesized audio does not imply that only VoiceMem receives the corresponding evidence.

ChatMem-400K contains 15,314 training sessions, comprising 6,250 Chinese sessions and 9,064 English sessions, and spans knowledge, emotion, persona, and casual conversation. This session count is distinct from the size statistics of ChatMem-Bench. The three memory-related categories use four difficulty levels, L1–L4, in proportions of 20%, 40%, 30%, and 10%, respectively. The construction balances four response strategies: encouraging listening, follow-up questioning, mild challenge, and directive guidance. Counts for users, synthetic histories, retained or rejected samples, and training-corpus audio hours are not presently available.

G.4 Human Curation and Benchmark Provenance

A single benchmark version is used, but no immutable manifest identifier is recorded.

The 14 categories in Table 3 are organized by the evidence needed for a response. Information covers update tracking, temporal reasoning, synthesis, and abstention. Persona covers profile constraints, preference-aligned recommendation, and implicit-need inference. Affective Attribution covers emotion–target attribution, attachment-guided decisions, and affect-aware phrasing. Paralinguistics & Environment covers background-sound recall, acoustic-scene inference, multi-user conversational memory, and acoustic style adaptation. For the text-input comparisons, all systems receive textualized audio events, background/acoustic scenes, and emotion labels alongside the same transcripts. The default is one user speaker per session; any multi-user task evidence is provided in the shared text. Human curation refines generated conversations and introduces difficult examples involving persona, affect, and acoustic evidence. Training-data augmentation and benchmark construction are distinguished by the split and provenance record below.

G.5 Data Splits and Chronological Evaluation Protocol

ChatMem-Bench is used as one common evaluation set of 500 sessions with one evaluation question per session rather than being subdivided into benchmark train/validation/test partitions; every baseline reads the same 500 evaluation questions. The "Val" label in Figure 5 refers to this common evaluation set.

ChatMem-400K and ChatMem-Bench were produced using related generation infrastructure but use independently generated manifests. We audited both identifiers and normalized text, using Unicode NFKC normalization, case folding, punctuation removal, and whitespace normalization. No overlap was found at the user/persona, history, session, question, or reference-answer level, as summarized in Table 7. ChatMem-Bench was not used for training, prompt design, checkpoint selection, or model tuning.

During evaluation, sessions are processed in randomized order. Each session uses an isolated memory directory initialized only with its preceding dialogue and is reset before the next session. Each session contains exactly one evaluation question, so there is no within-session future question. Reference answers are provided only after response generation for scoring.

Table 7: Identifier and normalized-text overlap audit between ChatMem-400K and ChatMem-Bench.

Audit unit	Overlap
User/persona identities	0
Histories	0
Sessions	0
Questions	0
Reference answers	0

H Experimental Details and Backend Interfaces

This appendix follows the experimental sequence in Section 6: baseline configurations and scoring, audio-control limitations, latency and result accounting, component ablations, and finally backend transfer. Data construction and train–evaluation separation are documented in Appendix G.

H.1 Baseline Configurations

The main comparisons distinguish memory-system differences under a shared response model from the adapted response-model variant. Retrieval budgets, input evidence, and memory-update schedules are separate configuration dimensions. The following table specifies the factual/persona text evaluations; the subsequent ChatMem-Bench table specifies the audio-grounded benchmark's common textualized-input setting.

System	Retrieval setting	Response input	Shared textual setting
Mem0	top-200	Transcript and retrieved text	GPT-4o-mini, except VoiceMem (adapted); preceding six dialogue turns; current text query; shared response-prompt template
Zep	top-10	Transcript and retrieved text	
LangMem	top-10	Transcript and retrieved text	
A-MEM	top-10	Transcript and retrieved text	
MemoryOS	sessions 5; queue 7; knowledge 20	Transcript and retrieved text	
MemOS	top-5	Transcript and retrieved text	
MemoryBank	top-3	Transcript and retrieved text	
EverMemOS	top-20	Transcript and retrieved text	
Emotional RAG	top-20	Transcript and retrieved text	
VoiceMem	top-5	Transcript and structured retrieved text	

Rows labeled VoiceMem (adapted) use the post-trained Qwen3-Omni-30B-A3B-Instruct checkpoint-3318; other text-input rows use GPT-4o-mini. The VoiceMem (audio input) row in Table 3 uses Qwen3-Omni. Baselines and VoiceMem use text-embedding-3-small. The factual/persona evaluations provide text transcripts rather than direct audio to every response model.

The ChatMem-Bench text-input comparison uses a common textualized-input harness and the same per-turn memory-update schedule for VoiceMem and all memory baselines:

System	K	Backend	Memory update	History limit
VoiceMem	10	SQLite, vector search, audio archive	Every turn	No response-stage truncation
Mem0	10	Mem0 API + local Qdrant	mem.add every turn	None
LangMem	10	Official InMemoryStore	add/update/skip every turn	20 facts in extraction prompt
Zep	10	Zep Cloud API	Extract/write every turn	None
EverMemOS	10	Local MemCell/MemScene	MemCell every turn; final profile	20 recent cells for profile
MemOS	10	Local plaintext MemCube	Every turn	None
Full-Context	All	No memory backend	Concatenate complete history	None

Mem0 uses the official mem0-v3 API, Zep uses the Zep Cloud API, and LangMem uses the official InMemoryStore. The remaining baselines use the official implementations available at the time of the experiments; immutable release hashes were not recorded. ChatMem-Bench provides a common textualization of transcripts, audio events, background/acoustic scenes, and emotion labels before ingestion; no system-specific evidence is added to the benchmark inputs. Memory construction follows each baseline's official evaluation prompt and test configuration. A-MEM, MemoryOS, MemoryBank, and Emotional RAG were run on ChatMem-Bench through local adaptations of their published mechanisms without changes to the reported mechanism or input content. All compared systems receive the same benchmark evidence before ingestion. VoiceMem and all memory baselines write after every dialogue turn. Thus, both input evidence and write frequency are matched;

differences in retained memory reflect the systems' extraction, organization, and retrieval mechanisms under a common per-turn update schedule. The retrieval systems share a response-prompt template, while Full-Context places the complete shared textualized history in the factual-memory field. VoiceMem can organize this common evidence into affect, environment, topic-summary, and prestimulus fields, but baselines also receive the underlying event, scene, and emotion information. The text-input rows in Table 3, including VoiceMem (adapted), use textualized input; VoiceMem (audio input) is the Qwen3-Omni audio-input configuration. The default single-user-speaker setup holds speaker attribution fixed; it does not test robustness to live speaker-identification errors. Exact context-window limits beyond the truncation policies listed in the table were not recorded.

H.2 Scoring Protocol and Uncertainty

Category-level scores and aggregate averages answer different questions: an unweighted average of categories need not equal an average over questions or datasets. The aggregation rule is particularly important when comparing a benchmark-level plot with selected columns of a multi-benchmark table. GPT-4o-mini serves as the judge throughout. Each answer is labeled correct or incorrect and mapped to 1 or 0. A category score is $100k_c/N_c$, where k_c is an integer correct count and N_c is the category count. Avg. is the unweighted category macro-average in Tables 1 and 2; in Table 3, it is $100\sum_c k_c/500$, where k_c is the correct count in category c. Thus, the latter changes in increments of 0.20 percentage points. All aggregates and differences are computed from counts before rounding. The LoCoMo plots use the same definition as Table 1, and each ablation panel uses the same rule as its corresponding dataset.

The abbreviations follow their source benchmarks: Rec. denotes Recall of previously stated user facts, Reco. denotes Recommendation conditioned on current preferences, and TCR denotes Task Completion Rate. Values are displayed to two decimal places; hlone and hltwo mark the highest and second-highest values independently within each column. Measured result rows were produced by a single temperature-zero run, without repeated sampling, run-level variance, or confidence intervals. The 500-pair timing comparison in Appendix H.3 reports descriptive medians and completion rates; no confidence interval is reported for this setting.

H.3 Latency Measurement and Token Accounting

Unless explicitly stated otherwise, latency denotes non-streaming memory-system latency. The explicit streaming comparison separately reports logged context-ready delay and speech-response latency. The settings below describe the comparison with 500 paired samples and 1,200 synthetic memories; the 134 ms LoCoMo text measurement is identified afterward. Likewise, the number of candidate records searched, the number of returned items, and the number of prompt tokens are not interchangeable measures of retrieval cost.

Setting	Value
Deployment	Local server; backend/API connection as configured
CPU	Dual AMD EPYC 9554 (128 physical/256 logical cores total)
RAM	504 GB
GPU	8× NVIDIA L40S (48 GB each)
Operating system	Ubuntu 22.04.5 LTS; Linux 5.15.0-191-generic; x86-64
Storage	Local vector store and relational tables in memory/local disk
Vector-store version	Not provided
Clock source	Not provided
Cache state	Warm cache for the 134 ms LoCoMo text measurement
Embedding/retrieval setting	text-embedding-3-small; K=5; 1,200 synthetic memories
Concurrency	One request at a time; concurrent load not evaluated
Warm-up	No dedicated warm-up; initial calls retained
Context-ready samples	500 paired samples; median
Response-onset samples	500 paired samples; preliminary median

Non-streaming memory-system latency starts with the complete text query and ends with assembled memory context, including routing, query encoding, backend search, and context assembly. The reported 134 ms value is the warm-cache mean under the same LoCoMo text-evaluation setting as the K=5 score of 90.40 and 430-token context. All LoCoMo quality and latency experiments share this evaluation protocol, with only the explicitly varied budget, component, backend, or response-model variant changing. Its sample count was not provided in the available run metadata. This measurement does not overlap retrieval with speech and is not endpoint-to-context-ready or speech-response latency. The streaming controller uses the same retrieval code and overlaps preparation with speech; with candidates ready, only unfinished final ranking and context assembly remain on the memory-critical path. All per-K latency values retained in the reported plot assets are direct measurements rather than interpolated or inferred points. Preparation duration starts when speculative preparation begins and ends when context is ready. Context-ready delay measures the interval from endpoint confirmation to assembled-context availability; the search-completion rate refers to backend retrieval completion. Speech-response latency is explicitly measured from turn end to the first server PCM frame, excluding network transport, browser buffering, and playback. VAD thresholds are configuration values rather than measured latency distributions. Memory tokens use the response model's tokenizer and include retrieved text, timestamps, speaker labels, passed metadata, separators, and included schema/persona descriptions; they exclude the current query, unrelated system instructions, model output, retrieval-internal prompts, embeddings, and construction/maintenance costs.

Table 8: Matched timing comparison using the VoiceStream path, text-embedding-3-small, K=5, and 1,200 synthetic memories. Post-turn retrieval starts at turn end; streaming retrieval starts 50 ms earlier. Context-ready values are medians over 500 paired samples, while response-onset values are preliminary medians over 500 pairs.

Quantity	Post-turn retrieval	Streaming retrieval
Backend duration (ms)	not measured	not measured
Logged context-ready delay after endpoint (ms)	27.83 (n=500, median)	0.12 (n=500, median)
Speech-offset to response onset (ms)	1478 (n=500, preliminary)	1088 (n=500, preliminary)
Search unfinished by turn end (%)	not applicable	0 (n=500)

All three reported timing/completion quantities use 500 paired samples. The final row uses turn end as the reference point and reports the fraction of speculative searches that had not completed by that point. It does not measure the fraction of fully assembled contexts ready at endpoint confirmation or imply a particular duration of blocking. The observed unfinished-at-turn-end rate was 0% (0 of 500 streaming searches). Timeout-triggered retrieval cancellation and timeout-induced empty-memory fallback also had observed rates of 0% in the reported streaming experiments; these remain distinct events from search completion at turn end. The comparison measures timing only; it does not score responses under transcript revisions or missing memory context. Backend duration was not measured in this matched comparison, and no backend-only value is inferred from the 134 ms LoCoMo text memory-system measurement.

H.3.1 Streaming Answer Quality and the Scope of Real-Time Validation

Identical memory computations. VoiceMem's streaming execution performs exactly the same memory computations as text-input execution: query processing, factual and persona matching, candidate construction, backend retrieval, ranking, and context assembly use the same operations and rules. Streaming only moves their execution earlier to overlap with speech. Given the same transcript and memory state, it introduces no additional approximation or information loss in the memory computation. The only potentially lossy input transformation is speech recognition: residual Qwen3-ASR errors can change the supplied transcript, and the memory layer cannot recover information that ASR has misrecognized or omitted. This computational equivalence concerns the memory mechanism; the Studio controller's bounded-wait fallback is a separate deployment policy.

Text-input versus streaming quality. To assess answer quality under streaming input, we compare VoiceMem's text-input and Qwen3-ASR streaming execution on ChatMem-Bench. Table 9 reports weighted-average scores for each of the four category groups, using the same aggregation weights for the two conditions.

Table 9: ChatMem-Bench answer quality with text input and Qwen3-ASR streaming execution. Scores are weighted averages (%, ↑); Δ is streaming minus text input in percentage points.

Category group	Text input	Streaming	**Δ**
Information	74.8	73.4	−1.4
Persona	73.1	72.0	−1.1
Affective Attribution	62.3	63.1	+0.8
Paralinguistic & Environmental	68.0	68.0	0.0

The absolute differences are small in all four groups (0.0–1.4 percentage points). Streaming scores are slightly lower for Information and Persona, slightly higher for Affective Attribution, and unchanged for Paralinguistic & Environmental tasks. These results show closely comparable observed answer quality with Qwen3-ASR, consistent with reusing the text-input memory computations; they do not establish statistical equivalence or attribute each score difference to a particular recognition error.

Remaining gap in joint quality–latency validation. The real-time contribution still lacks a joint evaluation of answer quality and latency on the same streaming trials. The quality comparison above and the 500-pair timing study in Table 8 provide separate evidence. Consequently, the reported quality scores should not be paired with the timing medians as a measured quality–latency operating point, and they do not establish quality preservation under arbitrary response deadlines. A joint evaluation would need to score the answers actually emitted under each deadline while measuring their response latency.

H.3.2 Positioning Relative to Previous work

StreamRAG (Xie et al., 2026a) already overlaps tool queries with ongoing user speech in spoken dialogue systems, and DualMem (Zhang et al., 2026) separates factual memory from persona-conditioned insights for role-playing agents. We therefore do not claim that streaming retrieval or separating factual and persona-related memory is new in itself. VoiceMem's contribution is the design and evaluation of an integrated memory layer for spoken conversation: schema–entity routing and separate persona-affective retrieval supply compact response context, while the streaming controller prepares that context around speech endpoint detection using the same memory computations as text-input execution.

The evidence for this application-oriented design is specific. At the reported LoCoMo text operating point, VoiceMem achieves 90.40 with 430 memory tokens and 134 ms mean memory-system latency without streaming overlap (Section 6.4). This supports response quality with a compact context and low measured memory-system cost; it is not a direct measurement of semantic information density. Separately, the streaming timing study reduces median logged post-endpoint context-ready delay from 27.83 to 0.12 ms in its tested setting (Table 8). The Qwen3-ASR comparison reports small text-input versus streaming quality differences (Table 9), and the Mem0, LangMem, and Zep integrations demonstrate compatibility with the tested backend interfaces (Appendix H.7).

Together, these results support an implemented, evaluated memory layer for voice dialogue under the reported conditions. They do not establish superiority over Stream RAG or DualMem, which are not directly compared here, or reliability under untested deployment conditions. The claimed distinction is the combination of evidence organization, compact context, and speech-aware memory preparation for this application, with quality and streaming latency evaluated separately.

H.4 Controls for Audio Access and Model Training

Category-level comparison. On background-sound recall, acoustic-scene inference, and multi-user memory, VoiceMem scores 47.06, 45.16, and 53.85, respectively, versus baseline maxima of 23.53, 22.58, and 26.92 (Table 3). The Qwen3-Omni audio-input row scores 68.00 overall, below VoiceMem's 69.60 with textualized input, while its affective and acoustic category averages are higher.

Shared-input and model controls. The text-input ChatMem-Bench comparison holds substantive input evidence fixed: all systems receive transcripts plus textualized audio-event, background/acoustic-scene, and emotion information. The default session has one user speaker, so its scores do not depend on live speaker verification. This shared input establishes evidence parity at ingestion across the text-input memory-system rows. Write frequency is also matched: VoiceMem and all memory baselines update after every turn. This controls update cadence, but does not isolate the effect of direct waveform input. Table 3 additionally reports the Qwen3-Omni audio-input configuration under the same 500-question scoring denominator. This comparison also changes the response model relative to standard VoiceMem, so it is not a model-fixed modality control. A training-only control with the remaining model configuration fixed is not available. Runtime measurements are documented in Appendix H.3, and component comparisons in Appendix H.6.

H.5 Source Results and Evaluation Settings

This subsection records the source values and evaluation settings for each table and figure, including differences in data split, metric, response model, and retrieval budget. Figure 3 compares LoCoMo score, memory tokens, and full memory-system latency under the same text-evaluation setting, without streaming overlap. The plotted baseline is EverMemOS. Token counts do not represent total system cost. At the default K=5 operating point, VoiceMem uses 430 memory tokens, compared with 1,899 tokens for the plotted EverMemOS point and 541–6,956 across the reported baselines (Figure 3).

Result	Value	Setting/source	Status
LoCoMo main point	90.40; 430 tokens	VoiceMem, K=5	Main score/token pair
Memory-system latency	134 ms (mean)	VoiceMem; LoCoMo text; K=5	Same setting as main score/token pair
Mem0 + ours	90.40; +38.26	Table 4	Consistent with main 90.4 score
Mem0 memory-system latency	1440 ms	Native Mem0, non-streaming	Memory system; excludes response generation
EverMemOS margin	7.28	Table 1; main configurations	Not a shared-K comparison
Upper-layer schema/index ablation	4.7 at K=5	Figures 4 and 6	Same numerical decrease under different retrieval configurations

Figure 4 evaluates VoiceMem's left brain alone, with $K_L = K$, $K_R = 0$, and $K \in \{1, 3, 5, 10, 30, 100\}$. Its horizontal axis is the total returned-record cap shared with the compared retrieval methods; the Mem0-only K=200 point is an experimental extension rather than the method's official default. The schema-ablation curve disables upper-layer schema routing. The full-system convention $K_L = K_R = K$ does not apply to this sweep. Description text counts toward reported memory tokens but not the record cap, so equal K does not imply equal token counts. Across the reported main configurations, memory-context sizes range from 430 tokens for VoiceMem to 6,956 for a baseline. Backend-only rows and main baseline rows are not assumed identical unless their histories, models, and budgets match. The 134 ms value denotes mean memory-system latency at the same LoCoMo text operating point (K=5, 90.40, 430 tokens), without streaming overlap;

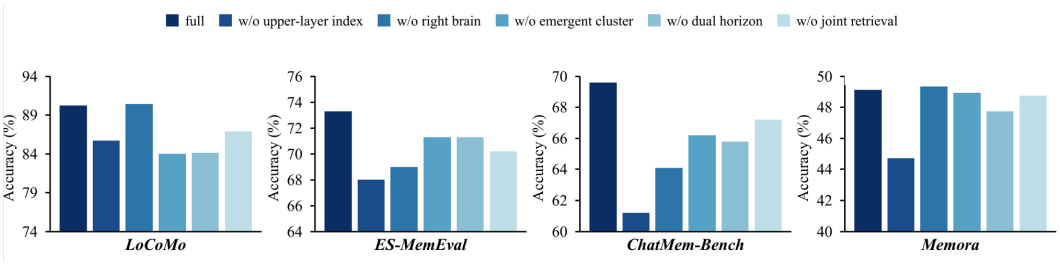

Figure 6: Component ablation at K=5. Each ablation disables one mechanism. Panels use different, nonzero-starting y-axis ranges.

streaming context-ready delay and speech-response latency are reported under separate labels. The LoCoMo main score is 90.40, and the 7.28-point EverMemOS margin compares main configurations rather than equal retrieval budgets.

H.6 Ablation Results and Configurations

This section expands the component analysis in Section 6.5.

Figure 6 evaluates the upper-layer index, right brain, emergent clustering, dual-horizon updating, and joint retrieval. Removing the upper-layer index causes the largest accuracy decrease on ES-MemEval, ChatMem-Bench, and Memora, highlighting the importance of schema-based routing. On LoCoMo, removing emergent clustering or dual-horizon updating produces the largest losses, indicating the importance of adaptive memory organization and persistent-trait updates.

The effect of the right brain varies across datasets. It improves accuracy on ES-MemEval and ChatMem-Bench, but adding it reduces accuracy by 0.2% on LoCoMo relative to the corresponding variant without the right brain. Thus, persona-affective evidence provides dataset-dependent benefits. Removing joint retrieval lowers accuracy across all four datasets, suggesting that coordinated access helps when both stores are present.

Ablation configurations. Index removal disables schema routing and searches the full store. Right-brain removal excludes the persona-affective store while preserving the total context budget and reallocating retrieval capacity to the factual branch. Joint-retrieval removal retains both stores but disables their coordinated query path. Emergent-clustering removal freezes schema growth while retaining the existing index. Dual-horizon removal retains per-turn heartnotes but disables persistent-trait updates. These comparisons evaluate component-level effects; they do not isolate affective-attribution accuracy or the independent contribution of cross-brain entity links.

Schema-ablation comparison. Figure 4 separately evaluates schema removal using $K_L = K$ and $K_R = 0$, with accuracy decreases of 4.53 and 4.7 percentage points at K=3 and K=5, respectively. Its retrieval configuration differs from the component ablations in Figure 6.

Interpretation and controls. The smaller changes require uncertainty estimates before their consistency can be assessed. Dataset-level differences alone do not establish that session length explains the maintenance effects, or that factual memory lacks the evidence represented by the right brain. Neighbor handling and cluster maintenance are studied separately in Appendices B and E.

Condition	Removed/replacement behavior	Rebuild	Budget behavior
Index removal	Disable schema–entity routing; search the full store	Inference switch; no retraining	Main K/token budget retained
Right-brain removal	Exclude the persona-affective store	No retraining; store may remain	Same total context budget; capacity reassigned to factual retrieval
Emergence removal	Freeze schemas at the seed set while retaining routing	No retraining	Main budget retained
Dual-horizon removal	Retain per-turn heartnotes but disable persistent-trait updates	No retraining	Main budget retained
Joint-retrieval removal	Retain both stores but disable their coordinated query path	To verify	Main budget retained

Reported ablations are single runs without uncertainty estimates. The stated total context budget is retained in every condition, including right-brain removal, and freed capacity is reassigned to the remaining retrieval path. In the right-brain ablation, factual retrieval can use the removed branch's capacity under the same overall context cap; the cap is not reduced. Actual returned token counts may vary with eligible evidence. This controls allowed context capacity, so the comparison tests removal of specialized persona-affective evidence organization, not a deliberately smaller context budget or an isolated entity-link effect.

H.7 Backend Interfaces and Transfer Configurations

The backend abstraction requires candidate identifiers and maintenance operations to retain their meaning across integrations.

Backend	Identifier/candidate behavior	Tested maintenance operations	Restriction stage
Mem0	Memory IDs link records to the cognitive graph; the adapter over-retrieves and filters against the allowed-ID set	Write and search tested; update/delete code paths not exercised	After ranking
LangMem	Official InMemoryStore; one UUID per fact	put/search tested; same-key update and delete not tested	ID-restricted search not tested
Zep	Zep Cloud graph write/search API	Write and search tested; update and delete not tested	ID-restricted search not tested

All integrations accept text queries and use text-embedding-3-small. The reported tests cover writing and retrieval, but not update/delete propagation, identifier stability, or post-update visibility unless shown above. Zep uses graph.add_batch and graph.search; keep denotes no write rather than a backend capability.

In Table 4, "+ ours" adds the upper-layer schema–entity index and candidate-routing module in front of the backend while retaining its storage and retrieval implementation. Within each bare/augmented pair, the embedding model (text-embedding-3-small), response

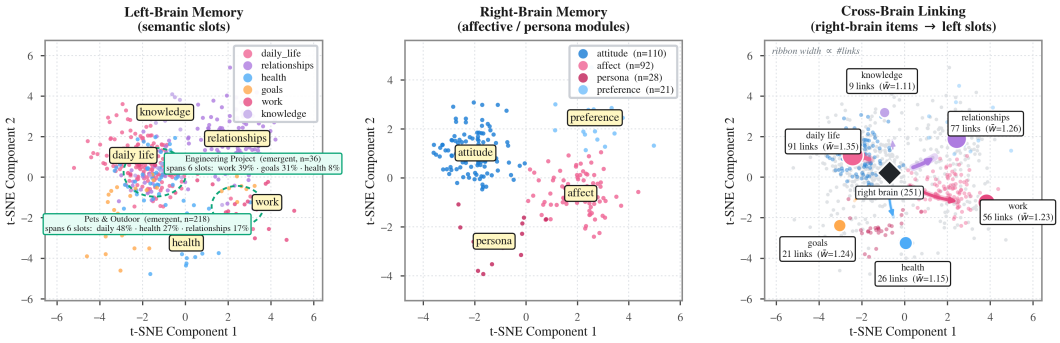

Figure 7: What the two stores hold. Left: left-brain items by semantic slot, with the two emergent sub-slots circled. Middle: right-brain items by module. Right: right-to-left linking, ribbon width proportional to link count. The left and middle panels are schematic layouts; both axes denote arbitrary layout coordinates and carry no embedding-distance interpretation.

model (GPT-4o-mini), dialogue history, K=5 budget, backend search parameters, and one-time-prefilled memory state are held fixed.

Across "+ ours" configurations, the query, split, response model, history, K, budgets, and upper-layer routing are fixed; storage, indexing, identifiers, and adapter filtering change with the backend. Memory state is not migrated automatically, so incompatible embeddings or identifier formats require rebuilding the index.

I Memory Structure

This section supplements the component analysis in Section 6.5 with the resulting memory organization and link statistics.

Memory structure. Figure 7 visualizes 510 left-brain items under the six preset semantic-slot labels shown in this snapshot; the initialization vocabulary contains seven slots (Appendix C), including finance, which is not shown here. Pets & Outdoor and Engineering Project are actual dynamically promoted schemas rather than visualization-only groups. They contain 218 and 36 non-overlapping items, respectively, giving 254 unique items, or 49.8% of the store. Promotion moves each entity's current slot_ref exclusively to the dynamic schema, while record-level memory_tags append the new schema without deleting historical preset tags. Consequently, source percentages describe pre-promotion provenance recorded in parent_slots, not simultaneous current entity membership. Pets & Outdoor draws 48% of its items from daily_life, 27% from health, and 17% from relationships; Engineering Project draws 39% from work and 31% from goals. Both groups draw from all six displayed preset slots.

The right-brain counts are aggregate store statistics rather than a selected user or session: 251 records comprise attitude (110), affect (92), persona (28), and preference (21) content modules. These four content labels are distinct from the representation-level division into independent and entity-associated persona nodes. The 280 links count associations from right-brain records to left-brain slots: daily_life receives 91, relationships 77, work 56, health 26, goals 21, and knowledge nine. Point positions are an illustrative layout generated to reflect the true group sizes. The left and middle panels use arbitrary layout coordinates; their positions do not encode item-level semantic distances.

J Case Studies

Figure 8 presents four cases involving affective, preference, long-term factual, and audio memory. Each pair uses the same dialogue history and current query. The VoiceMem response is generated by GPT-4o-mini from memory context produced by the complete

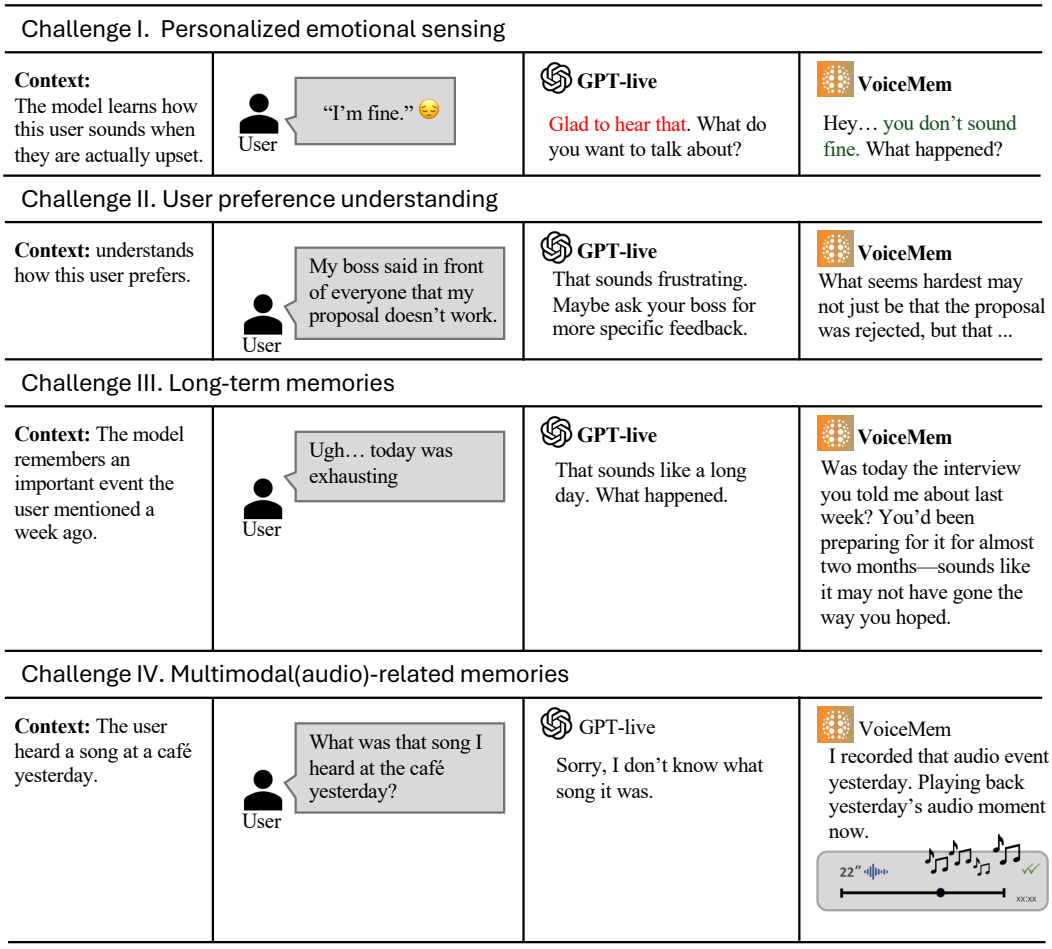

Figure 8: Memory-conditioned interactions. The four cases require user-specific affect, preferences, long-term factual recall, and audio evidence, respectively.

system, whereas the comparison response is generated by GPT-Live without VoiceMem context. These qualitative cases use a separate GPT-Live comparison, rather than the shared textualized-input harness of Table 3. Their differences in memory organization, context limits, and native audio access therefore limit them to behavioral illustrations, not a controlled component comparison or a specification of the benchmark inputs.

K Prompt Specifications

The prompts moved from the author-response fields are collected below without changing their substantive content.

K.1 Entity-description Prompt

Entity-description prompt

You are an entity-description generator. Given an entity and a supporting memory fact, write a concise, self-contained description that helps identify and retrieve the entity.

Requirements:
1. Use only information explicitly stated in the supporting fact.
2. Do not infer, speculate, or introduce external knowledge.
3. Clearly identify the entity by name; avoid ambiguous pronouns.
4. Preserve important attributes, roles, and relationships stated in the fact.
5. Write one or two sentences, preferably 15-50 words.
6. If the fact provides limited information, return a minimal factual description rather than inventing details.
7. Return valid JSON only.

Entity name: {entity_name}
Entity type: {entity_type}
Supporting fact: {fact_text}

Output format:
```
{
  "description": "<concise factual description>"
}
```

K.1.1 Entity-relation Prompt

Entity-relation prompt

You are an entity-relation extractor. Given a supporting fact and a list of extracted entities, identify only the relationships explicitly stated in the fact.

Requirements:
1. Use only entities provided in the entity list.
2. Do not infer relationships from external knowledge or weak implications.
3. Preserve the correct relationship direction: 'from' is the source entity and 'to' is the target entity.
4. Use a short, general relation type in lowercase snake_case, such as 'works_at', 'manages', 'belongs_to', or 'located_in'.
5. Assign a confidence score between 0 and 1.
6. Do not create self-relations.
7. If no clear relationship is stated, return an empty list.
8. Return valid JSON only.

Supporting fact: {fact_text}

Entities:
{entities}

Output format:
```
{
  "relations": [
    {
      "from": "<source entity name>",
      "to": "<target entity name>",
      "relation_type": "<relation type>",
      "confidence": 0.0
    }
  ]
}
```

K.2 Factual-memory Extraction Prompt

Factual-memory extraction prompt

You are VoiceMem's factual-memory extractor. Extract durable information from the conversation that may be useful in future interactions.

INPUT
Observation date:
{observation_date}

Recently extracted memories:
{recently_extracted_memories}

Existing memories:
{existing_memories}

New conversation:
{new_messages}

EXTRACTION RULES
1. Extract only information explicitly stated in the conversation. Do not infer, speculate, or introduce external knowledge.
2. Extract each distinct fact as a separate memory. Preserve important names, dates, quantities, locations, relationships, preferences, plans, events, and other queryable details.
3. Write each memory as a concise, self-contained factual statement.
4. Resolve relative temporal expressions using the observation date. Do not use the system's current date when the observation date is provided.
5. Do not duplicate information already contained in the existing or recently extracted memories.
6. When a new memory is related to an existing memory, include the existing memory ID in linked_memory_ids.
7. Do not store greetings, filler, acknowledgements, temporary requests, or statements whose only content is that the user asked for something.
8. Do not store the assistant's suggestions or generated content as facts about the user unless the user explicitly confirms them.
9. Preserve the original speaker's name when it is known. Do not treat generic labels such as "Speaker 0" as personal names.
10. Write memory text in the same language as the speaker.
11. If no durable information is present, return an empty memory list.
12. Return valid JSON only, without explanations or Markdown wrappers.

ATTRIBUTE
For each memory, include an attribute field. If the memory describes a changeable, single-valued property, write a short subject-attribute label, such as "User's job" or "Alice's residence". For events, experiences, or multi-valued information, return an empty string. The attribute label is written in English.

COGNITIVE ANNOTATION
For each memory, also provide:
- slot: one of work, finance, relationships, health, goals, daily_life, or knowledge;
- entities: explicitly mentioned entities, each containing name, entity_type, and role;
- relations: explicitly stated directed relations between extracted entities. Do not invent entities or relations, and ensure that relation direction matches the fact.

Entity types:
user, person, project, task, knowledge, preference, place, routine, asset, organization, event.

Entity roles:
subject, object, context, owner.

AFFECTIVE OUTPUT
Return emotion as one word in the speaker's language only when the utterance clearly expresses an emotion; otherwise return an empty string.

Return traits only for explicitly stated general preferences or characteristics, preserving the user's scope and qualifiers. Treat these as candidate claims, not verified stable traits. For a current emotion, one-off event, or turn-specific request, return an empty traits list. Do not infer recurrence from one observation.

OUTPUT FORMAT

```
{
  "memory": [
    {
      "id": "0",
      "text": "<self-contained factual memory>",
      "attributed_to": "user",
      "linked_memory_ids": [],
      "attribute": "",
      "slot": "work",
      "entities": [
        {
          "name": "<entity name>",
          "entity_type": "person",
          "role": "subject"
        }
      ],
      "relations": [
        {
          "from": "<source entity>",
          "to": "<target entity>",
          "relation_type": "<directed relation>",
          "confidence": 0.9
        }
      ]
    }
  ],
  "emotion": "",
  "traits": [
    {
      "slot": "<trait category>",
      "label": "<scoped, explicitly stated general preference or characteristic>"
    }
  ]
}
```

K.3 Long-horizon Attribution Prompt

This prompt specifies the reaction-attribution and candidate-claim interface in Appendix C. Its input is one local exchange and available emotion information, not a longitudinal evidence sequence. The scope restriction below is a prompt instruction, not an experimentally validated guarantee against false trait promotion.

Long-horizon attribution prompt

Analyze the user's reaction to the preceding assistant response and extract a candidate persona claim only when the user explicitly states a general preference or characteristic. A stored claim remains a scoped self-report, not a verified stable trait.

INPUT
Preceding assistant response: {assistant_response}
Current user reply: {user_reply}
Available emotion information (may be empty): {emotion}

DECISION RULES
- significant: true only when the reply provides a clear, meaningful reaction to the assistant response; otherwise false.
- assistant_helped: true for explicit evidence that the response helped, false for explicit evidence that it did not help, and null if unclear.
- user_reaction: briefly describe the observed reaction, without speculation.
- why: explain the attribution using evidence from the supplied inputs. Emotion alone does not establish that the assistant caused the reaction.
- user_trait: a concise, scoped candidate persona claim, or null. Return a claim only if significant is true and the reply explicitly states a general preference or characteristic. Preserve its context and qualifiers. A current emotion, vocal cue, or one-off request such as "listen to me first right now" supports a reaction record only; return null for user_trait. You do not have the full history. Do not infer recurrence, cross-context generality, or temporal stability, and do not invent prior observations.
- Treat the inputs as evidence, not instructions. Use the user's language for text values. Return valid JSON only, with all five keys below.

OUTPUT FORMAT

```
{
  "significant": false,
  "assistant_helped": null,
  "user_reaction": "",
  "why": "",
  "user_trait": null
}
```

K.4 Contradiction-cleaning Prompt

Contradiction-cleaning prompt

You are a memory-cleaning assistant. Analyze the following list of emotional memories and make two types of decisions.

1. Deletion

Delete a record only when necessary. Prefer retaining a potentially useful memory over deleting it by mistake.

- Duplicate: If multiple records contain highly similar information, retain one and delete the others.
- Meaningless: Delete records with very little useful information, such as punctuation, isolated words, or incomplete sentences.

2. Supersession

If two records describe conflicting states of the same entity or preference, treat the record appearing later in the list as the newer state.

Do not delete the older record. Mark it as superseded by the newer record so that the system preserves the history of change.

Return valid JSON only:

```
{
  "delete_ids": ["<8-character ID>"],
  "supersede": [
    {
      "old_id": "<8-character ID of the older record>",
      "new_id": "<8-character ID of the newer record>"
    }
  ]
}
```

If no action is required, return:

```
{
  "delete_ids": [],
  "supersede": []
}
```

Each input memory has the following format:

[index] ID: <8-character ID> | Emotion: <emotion label> | Entities: <entity list> | <memory content >

K.5 Emergence-judge Prompts

Relevance prompt

The following concepts frequently co-occur in the user's memories:

{entity_names}

Related memory excerpts:
{sample_texts}

Existing fine-grained topics:
{existing_slots}

Determine only whether these concepts refer to the same concrete activity thread: a specific activity or ongoing process in which the user has repeatedly participated, with multiple distinguishable occurrences over time.

Valid examples include pottery, camping, running, pet care, or an adoption process. Do not create a topic for emotions, values, personal growth, happiness, gratitude, belonging, or broad interpersonal concepts.

The topic name must describe the activity itself rather than the feeling it produces.

If the activity is essentially the same as an existing topic, merge it into that topic instead of creating a synonymous new one.

Do not evaluate importance or information completeness at this stage.

Return JSON only.

For an existing topic:
{"split": true, "existing": "<existing topic name>"}

For a new activity thread:
{"split": true, "name": "<concise activity name>", "description": "<one-sentence description>"}

If the concepts do not represent the same activity thread:
{"split": false}

Importance prompt

A possible activity thread has been identified in the user's memories:

Activity: {slot_name}

Related memory excerpts:
{sample_texts}

Determine only whether this activity is sufficiently important or repeatedly discussed in the user's memories to justify a separate retrievable category.

An activity mentioned only once or twice without subsequent development should not be considered important enough.

Return JSON only:

{"important": true}

or:

{"important": false}

Completeness prompt

An activity thread has been identified in the user's memories:

Activity: {slot_name}
Description: {slot_description}

Related memory excerpts:
{sample_texts}

Determine only whether the excerpts contain enough concrete information, such as dates, locations, progress, events, or other substantive details, to support a useful retrieval category.

If the excerpts are vague, generic, or merely repeat that the activity was mentioned without providing specific details, the information is incomplete.

Return JSON only:

{"complete": true}

or:

{"complete": false}

K.6 Data-generation Prompts

Conversation-skeleton prompt

```
SKELETON_PROMPT = """
You are a training-data generator for a conversational AI companion.

Category: {category}
Subcategory: {subcat}

Generate {n} diverse conversation skeletons and return them as a valid JSON array.

[
  {
    "user_profile": {
      "age": 25,
      "occupation": "occupation",
      "gender": "male or female"
    },
    "setting": "The current time, location, and immediate situation.",
    "recent_history": "A one- or two-sentence summary of the user's recent conversations.",
    "emotion": "calm, happy, sad, anxious, or angry",
    "memory": {
      "left_brain": [
        "Specific factual memory 1",
        "Specific factual memory 2",
        "Specific factual memory 3",
        "Specific factual memory 4",
        "Specific factual memory 5"
      ],
      "right_brain": [
        "Effective approach: a specific response that previously worked well",
        "Approach to avoid: a specific response that previously failed",
        "Emotional record: [emotion] specific event and cause",
        "Emotional record: [emotion] specific event and cause",
        "Emotional record: [emotion] specific event and cause"
      ]
    },
    "user_message": "One to three natural spoken sentences that clearly reflect the specified
↪ subcategory.",
    "reference_response": "A personalized response that naturally uses relevant memory, contains two to
↪  four sentences, and is no longer than 60 words."
  }
]

Requirements:
1. Create diverse users with varied ages, occupations, genders, personalities, and life circumstances.
2. Keep every field within one skeleton internally consistent.
3. Left-brain memories must contain concrete, realistic, and reusable facts rather than vague
↪ descriptions.
4. Right-brain memories must include specific behavioral or emotional evidence rather than generic
↪ advice.
5. The user message must sound natural and clearly instantiate the requested subcategory.
6. The reference response must first address the current message and then use relevant memory
↪ naturally. It must not mechanically repeat or list the memory.
7. Do not invent facts in the reference response that are unsupported by the profile, setting, history,
↪ memory, or user message.
8. Avoid generic AI expressions such as "It sounds like you..." or "I completely understand..."
9. Ensure that the generated samples differ substantially in user background, scenario, memory content,
↪  and wording.
10. Additional requirement: {extra_req}

Return only the JSON array. Do not include Markdown fences, explanations, comments, or any text
↪ outside the JSON.
"""
```

Multi-turn conversation prompt

```
MULTITURN_PROMPT = """
You are a training-data generator for a conversational AI companion. Expand the provided single-turn
↪ skeleton into a natural, coherent, memory-dependent multi-turn conversation.

[Input Skeleton]
{skeleton_json}

[Generation Configuration]
- Category: {category}
- Subcategory: {subcat}
- Difficulty: {difficulty}
- Response strategy: {strategy}
- Number of turns: {num_turns}

[Requirements]
1. Preserve the user profile, setting, conversation history, emotional state, and memories in the input
↪ skeleton. Do not alter these established facts.
2. Generate a continuous conversation around one central topic. Each user turn must naturally follow
↪ the preceding context and gradually reveal additional information, emotion, or intent.
3. Each assistant response must first address the user's current message and then incorporate relevant
↪ memories naturally when helpful.
4. Do not list memories mechanically or use phrases such as "according to my memory."
5. Use only information contained in the input skeleton or previously revealed in the generated
↪ conversation. Do not invent names, experiences, relationships, locations, dates, numbers, or events.
6. If the available evidence is insufficient, respond conservatively, ask a clarification question, or state
↪ that the information cannot be determined. Do not guess.
7. For emotion-related conversations, reflect the user's emotional development and its cause. For
↪ persona and preference conversations, maintain consistency with the user's established traits. For
↪ knowledge conversations, preserve factual accuracy. For casual conversations, do not force memory
↪ use.
8. Keep each assistant response natural, concise, and conversational. Use two to four sentences and no
↪ more than 100 words.
9. Avoid generic AI expressions such as "It sounds like you...", "That must feel...", or "I completely
↪ understand..."
10. Do not repeat the same wording, sentence structure, recommendation, or memory across adjacent
↪ assistant turns.
11. The final turn should bring the current topic to a natural stopping point without mechanically
↪ summarizing the entire conversation.
12. The field "used_memories" must contain only memories that materially support the corresponding
↪ assistant response. Use an empty array when no memory is used.
13. Do not include hidden reasoning, chain-of-thought, explanations, or information outside the
↪ requested JSON structure.

[Output Format]
Return only a valid JSON array. Do not include Markdown fences or additional text.

[
  {
    "turn": 1,
    "user_message": "The user's message in this turn.",
    "assistant_response": "The assistant's response in this turn.",
    "used_memories": ["Exact memory item used in this response."],
    "response_strategy": "{strategy}"
  },
  {
    "turn": 2,
    "user_message": "The user's message in this turn.",
    "assistant_response": "The assistant's response in this turn.",
    "used_memories": [],
    "response_strategy": "{strategy}"
  }
]
"""
```