Thinking Ahead: Prospection-Guided Retrieval of Memory with Language Models

Harshita Chopra[1] **Krishna Chintalapudi**[2] **Suman Nath**[2] **Ryen White**[2] **Chirag Shah**[1]

[1]University of Washington, Seattle, USA
[2]Microsoft Research, Redmond, USA
Email: hchopra@cs.washington.edu

github.com/harshita-chopra/PGR-mem Dataset: MemoryQuest

Abstract

Long-horizon personalization requires dialog assistants to retrieve user-specific facts from extended interaction histories. However, many relevant facts have low semantic similarity to the query under dense retrieval. Standard Retrieval-Augmented Generation (RAG) and GraphRAG systems remain largely retrospective: they rely on embedding similarity or fixed graph traversals, often missing relevant facts that lie far from the query in embedding space. Inspired by *prospection*, the human ability to use imagined futures as cues for recall, we introduce **Prospection-Guided Retrieval (PGR)**, which decouples retrieval from memory storage. Given a user query, PGR expands the goal into a linear chain or tree of thoughts (TOT) denoting plausible next steps, and uses these steps as retrieval probes instead of the original query alone. Retrieved facts are then used to personalize the next round of *prospection*, enabling PGR to uncover additional memories by grounding the simulation in the user's history. We also introduce **MemoryQuest**, a challenging multi-session benchmark where each query is annotated with 3-5 dated reference facts under a low query-reference similarity constraint. Across 1,625 queries spanning 185 user profiles from 3 public datasets, PGR-TOT substantially improves retrieval, achieving nearly 3× recall on MemoryQuest over baselines. In pairwise LLM-as-judge evaluations, PGR-generated responses are preferred on 89–98% of queries, with blinded human annotations showing a similar trend. Overall, the results show that explicit prospection yields large gains in long-horizon retrieval and response quality over similarity-only baselines.

1 Introduction

Human memory holds onto a vast amount of information and experiences, but retrieving those details is much more dynamic than a passive lookup process. Neuropsychology research has established that the brain uses *prospection*, the ability to simulate possible futures, to trigger the recall of relevant knowledge and experiences (Schacter et al., 2017; Addis et al., 2007). This mechanism helps surface memories that appear irrelevant in the current moment but become important for anticipating future states. In humans, this process depends on interaction between the prefrontal cortex and the hippocampus, which together support iterative simulation and memory retrieval. By contrast, modern AI systems remain largely anchored in retrospective lookup, rather than reasoning about future utility.

Decoupling Retrieval from Storage. Current Retrieval-Augmented Generation (RAG) (Guu et al., 2020; Lewis et al., 2020) and GraphRAG (Edge et al., 2024) systems tightly couple *storage structure* and *retrieval strategy*. In these paradigms, retrieval is largely dictated by the architecture: vector stores rely on similarity search, while graphs rely on traversal. We argue that retrieval should instead be treated as an independent policy. By decoupling how memories are stored from how they are retrieved, we can enable Large Language Model (LLM) agents to reason about informational needs in advance.

Inspired by anticipatory mechanisms in the human brain, we introduce ***Prospection-Guided Retrieval (PGR)***, a retrieval paradigm for personalized agentic memory. Unlike existing methods that retrieve retrospectively by matching a query to past records, PGR retrieves *prospectively*: given a user goal, it combines future reasoning and memory retrieval through an iterative reasoning loop:

$$\text{Simulate Future} \rightarrow \text{Retrieve Relevant Memories} \rightarrow \text{Refine Trajectory}$$

Crucially, PGR is a *retrieval policy*. It is orthogonal to the underlying memory representation and can be composed with vector stores, entity graphs, or episodic logs to transform static recall into anticipatory intelligence. Because simulation occurs in

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 46773–46794
October 24-29, 2026 ©2026 Association for Computational Linguistics

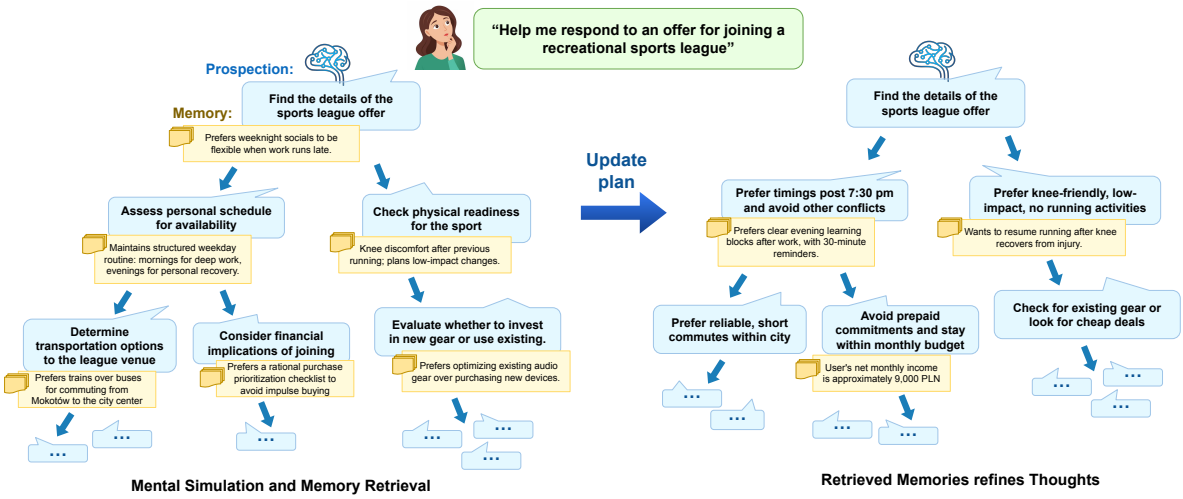


Figure 1: Prospection involves simulating possible futures and triggering recall of context-relevant past memories at each imagined step. These forward simulations enable anticipatory and contextually appropriate behavior. Retrieved memories modify the simulated plan accordingly.

real-time, PGR avoids the limitations of the rigid, pre-indexed structures like GraphRAG, allowing the system to dynamically adapt and surface latent memories that a static search would fail to reach.

Figure 1 illustrates an example PGR loop from our dataset. When a user considers joining a recreational sports league, the system executes a **Tree-of-Thought (ToT)** to simulate plausible future trajectories, such as assessing schedule availability or physical readiness. These simulated steps trigger retrieval of latent past experiences (e.g., knee discomfort from prior running or a preference for flexible schedules), which iteratively refine the simulation. For example, recalling a knee injury shifts the search toward lower-impact activities. By exploring multiple futures simultaneously, PGR can anticipate informational needs and surface indirect memories, such as transportation preferences for a city-center venue, that similarity-based retrieval would likely miss due to limited surface overlap with the original query.

To evaluate PGR, we introduce **MemoryQuest**, a novel benchmark designed for multi-session, multi-task human–agent interactions. Sessions are chronologically consistent, with events logically evolving over weeks to reflect realistic long-term dependencies. Unlike datasets such as *MSC* (Xu et al., 2022) or *LongMemEval* (Wu et al., 2025), which often focus on direct semantic matches, MemoryQuest enforces queries that require retrieval of multiple memories semantically dissimilar from the query scattered across multiple domains. MemoryQuest serves as a stress test for long-term personalization, exposing the failure modes of standard RAG that prospection-based methods are designed to overcome.

Our key contributions are:

- **Prospection-Guided Retrieval (PGR):** A brain-inspired paradigm that decouples retrieval policy from storage via an iterative Simulate → Retrieve → Refine loop. This enables agents to surface latent or counterfactual memories by anticipating future needs, overcoming the rigidity of pre-ordained structures like GraphRAG.
- **MemoryQuest Benchmark:** A multi-session dataset designed for long-term personalization. It requires retrieving 3–5 disparate memories linked by chronological and logical dependencies rather than surface semantic similarity, specifically targeting the "retrospective bottleneck" of current systems.
- **Empirical Evaluation:** A scalable evaluation framework using human-as-a-judge (MTurk) to ground and validate a large-scale LLM-as-a-judge analysis across 1,500+ queries on three benchmarks. Results demonstrate that PGR significantly outperforms baselines in retrieval recall and anticipatory response quality.

2 Related Work

Retrieval-Augmented Generation (RAG). Standard RAG (Lewis et al., 2020) integrates external

knowledge into the generation process to improve factual grounding. While subsequent hybrid variants (Izacard and Grave, 2021) and persistent memory stores (Park et al., 2023) improve scalability, they remain inherently *retrospective*. These systems retrieve based on surface similarity to the current query rather than goal-oriented anticipation, often failing to surface memories that are only indirectly relevant to future needs.

Graph-Based Retrieval and GraphRAG. To move beyond isolated text chunks, *GraphRAG* (Edge et al., 2024) leverages knowledge graphs to support multi-hop inference. However, these methods are constrained by a *pre-ordained structure*: retrieval is limited to traversing static edges established at construction time. Consequently, they cannot infer counterfactual relations or anticipate connections dynamically based on specific query.

Agentic Memory and Reasoners. Recent frameworks such as *ToT* (Yao et al., 2023a) and **ReAct** (Yao et al., 2023b) use explicit reasoning traces to enhance planning and action. Agentic memory systems such as Mem0^g (Chhikara et al., 2025) maintain persistent user memories and entity relations, but retrieval still relies on vector search and extracted graph structure. In contrast, *PGR* extends the principles of branching reasoning to the *retrieval domain*: simulated ToT or chain steps become retrieval probes, and retrieved facts refine subsequent prospection. This transforms retrieval into an active, iterative policy that is orthogonal to, and can be layered upon, any underlying storage architecture.

Cognitive neuroscience suggests that human memory is constructive rather than archival, supporting both recollection of the past and imagination of the future (Schacter et al., 2017; Addis et al., 2007). This *simulation* lets people recombine fragments of experience for anticipatory planning (Hassabis et al., 2007). PGR builds on this idea by aligning memory retrieval with the same kind of simulation-based processing seen in human cognition.

3 Methodology

Inspired by theories of human prospection, Prospection-Guided Retrieval (PGR) treats retrieval as an active reasoning process rather than a static lookup operation (Schacter et al., 2017). Given a user goal, PGR executes an iterative Simulate $\rightarrow$ Retrieve $\rightarrow$ Refine loop, using simulated future steps to guide memory retrieval and adapt subsequent reasoning. Unlike RAG and GraphRAG systems, where retrieval is tightly coupled to storage structure, PGR decouples memory storage from retrieval strategy, enabling the system to surface latent or indirectly relevant memories that may become useful later in the interaction.

3.1 System Architecture Overview

Figure 2 presents the overall PGR architecture. The system consists of three connected components: (1) a user interaction layer, (2) a long-term memory store, and (3) the PGR retrieval loop.

User Interaction Layer. User–assistant conversations are continuously processed to extract structured memory facts, which are stored in long-term memory. At inference time, a new user query is passed to PGR, which retrieves relevant memories and produces a prospection summary used by the assistant for response generation.

Long-Term Memory. The memory store maintains persistent knowledge about the user across sessions. PGR requires only a queryable memory interface and is agnostic to the underlying storage technology. The backend may consist of a vector index, a knowledge graph, or a hybrid retrieval system. In our implementation, memory consists of structured *facts* stored in a vector database together with *conversation logs* used for fact extraction and consolidation.

PGR Retrieval Loop. PGR performs retrieval through an iterative *simulate–retrieve–refine* process. Starting from a user query, the system first generates prospective reasoning steps (*Initial Prospection*), which are used as retrieval probes into memory. Retrieved facts are then used to condition a second round of personalized reasoning (*Personalized Prospection*), producing refined retrieval probes that surface additional latent memories. This loop repeats until the number of new facts falls below a threshold. The final retrieved facts and prospection summary are returned to the assistant for response generation.

3.2 Construction of Memory

Our memory system is built with the following components and methods:

Conversation Logs. Each user–assistant session is stored as a multi-turn conversation with a unique identifier and timestamp. These logs serve as the source for extracting memory facts but are not directly used during retrieval.

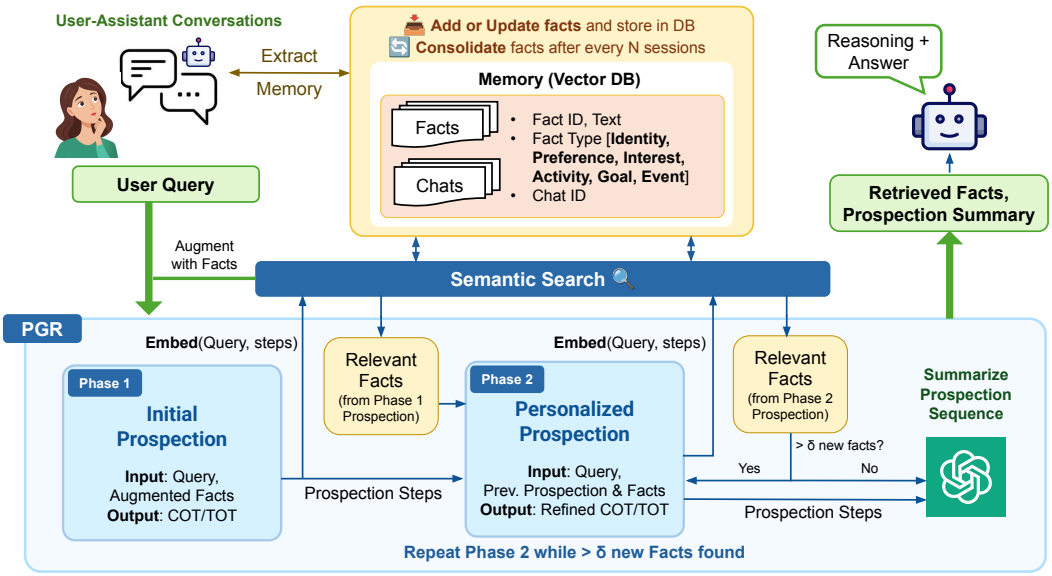


Figure 2: Overview of Prospection-Guided Retrieval (PGR) in a Conversational Interface.

Memory Facts. Facts are short atomic statements extracted from conversations, such as user demographics, preferences, plans, interests, and past events. Each fact is associated with a unique identifier, a semantic type, provenance metadata, and related entities. We use six fact categories: *Identity*, *Preference*, *Goal*, *Interest*, *Activity*, and *Event*. Fact extraction and updates are performed using an LLM, while memory writes occur periodically to reduce storage and inference costs.

Memory Upsert. During the first interaction, all extracted facts are inserted as new entries. For subsequent conversations, extracted facts are upserted: new facts are inserted while existing facts are updated when appropriate.

Memory Consolidation. To reduce redundancy, memory facts are periodically consolidated after a fixed number of new insertions. Facts are clustered using embedding similarity and temporal proximity, and an LLM merges semantically overlapping facts into a unified representation while preserving provenance information.

Retrieval Interface. At query time, retrieval operates over the memory facts using the underlying retrieval backend (e.g., vector similarity search). PGR builds prospective reasoning and iterative retrieval on top of this interface.

All prompts used for extraction, updates, and consolidation are provided in Appendix 7.

3.3 Prospection Guided Retrieval

In this section, we describe the design choices behind the proposed PGR mechanism.

Formal Setup. Let $\mathcal{M}$ denote the memory store. The retrieval function $\phi(x; \mathcal{M}, \Theta)$ returns facts from memory $\mathcal{M}$ relevant to a sub-query x, parameterized by Θ. In our implementation, $\Theta = \{K, \tau\}$ retrieves the top-K facts above embedding similarity threshold τ. For different memory architectures, Θ may instead encode traversal depth, temporal windows, or community hierarchy. R denotes the accumulated set of retrieved facts, initialized to $\emptyset$.

Query Augmentation. PGR first generates an augmented query q_A to resolve underspecified user queries by retrieving tightly relevant context:

$$q_A = \text{LLM}(P_A(q,\ \phi(q; \mathcal{M}, \Theta_o))) \tag{1}$$

where P_A instructs the model to disambiguate using context Θ_o (e.g., K_o, τ_o). This ensures the simulation begins with a high-fidelity representation of user intent.

Next, we employ two sequential phases of prospection that iteratively retrieve and refine relevant memories:

Phase 1: Initial Prospection. The LLM processes q_A to generate a broad *horizon* of potential future actions. This simulation can be operationalized as either a linear *Chain-of-Thought (CoT)* or a branching *Tree-of-Thought (ToT)* structure. Let's denote the prospection sequence as $S^0 = \{s_1^0, \ldots, s_L^0\}$. Formally,

$$S^0 = \text{LLM}(P_{\text{sim}}(q_A)) \tag{2}$$

Each hypothesized step $s_l^0 \in S^0$ serves as an independent **sub-query** to the memory store, potentially surfacing latent facts that the original query

alone would fail to trigger. These additional facts are merged with the initial query results:

$$R \leftarrow \phi(q; \mathcal{M}, \Theta) \cup \bigcup_{l=1}^{L} \phi(s_l^0; \mathcal{M}, \Theta) \quad (3)$$

Phase 2: Personalized Prospection. Using the accumulated knowledge (R) from Phase 1, the LLM iteratively refines the prospection sequence, making it personalized to the user. This personalization allows the steps to reflect user-specific constraints, helping the system reach semantically distant aspects of memory. Formally, we denote the updated sequence as:

$$S^i = \text{LLM}(P_{\text{ref}}(q_A, R, S^{i-1})) \quad (4)$$

For each sub-query $s_l^i \in S^i$, similar facts are retrieved. Δ^i denotes the set of unique, previously unseen facts discovered during iteration i:

$$\Delta^i = \bigcup_{l} \phi(s_l^i; \mathcal{M}, \Theta) \setminus R \quad (5)$$

The memory set is updated $R \leftarrow R \cup \Delta^i$, and the loop terminates when $|\Delta^i|$ falls below a convergence threshold δ or a maximum iteration $I_{\max}$ is reached. The final set R^* is returned alongside a *prospection reasoning summary* detailing how the simulation guided the memory recall.

Algorithm 1 Prospection-Guided Retrieval (PGR)

Require: Query q, memory store $\mathcal{M}$, abstract retrieval function ϕ, augmentation parameters Θ_o, retrieval parameters Θ, convergence threshold δ, max iterations $I_{\max}$
Ensure: Final retrieved fact set R^*

```
R ← ∅
                                  ▷ Query Augmentation
q_A ← LLM(P_A(q, φ(q; M, Θ_o)))
                          ▷ Phase 1: Initial Prospection
S^0 ← LLM(P_sim(q_A))
R ← φ(q; M, Θ) ∪ ⋃_{l=1}^{|S^0|} φ(s_l^0; M, Θ)
                  ▷ Phase 2: Personalized Prospection
for i = 1, 2, ..., I_max do
    S^i ← LLM(P_ref(q_A, R, S^{i-1}))
    Δ^i ← ⋃_{l=1}^{|S^i|} φ(s_l^i; M, Θ) \ R
    R ← R ∪ Δ^i
    if |Δ^i| < δ then break
    end if
end for
return R* ← R
```

4 Benchmark Dataset: MemoryQuest

Answering a query often requires recovering a thread of temporally distributed facts with weak lexical overlap to the query itself. Existing benchmarks primarily focus on conversational continuity, temporally local evidence, or semantically aligned retrieval. We introduce **MemoryQuest**, a dataset that evaluates long-horizon personalized retrieval in realistic assistant settings where user histories span months and multiple domains. It emphasizes retrieval of relevant but non-obvious memories distributed across long interaction histories.

4.1 Gap Analysis and Comparison

Prior datasets evaluate complementary aspects of long-term conversational memory, but do not jointly capture multi-session reasoning, temporal grounding, and low-similarity retrieval:

Conversational Continuity and Temporal Structure. MSC (Xu et al., 2022), LongMemEval (Wu et al., 2025), and LoCoMo (Maharana et al., 2024) study long-context or temporally structured memory, but they do not emphasize multi-session retrieval of low-overlap evidence.

Low-Similarity Retrieval. PersonaMem (Jiang et al., 2025) and ImplexConv (Li et al., 2025) evaluate personalization and implicit reasoning, but supporting evidence often remains semantically aligned with the query. MemoryQuest explicitly constructs low-overlap retrieval settings that challenge similarity-based retrieval.

Fine-Grained Evaluation. Each query is annotated with a small set of dated atomic reference facts, enabling direct evaluation of retrieval quality independently of downstream generation.

4.2 Automated Generation

MemoryQuest is built on *PersonaLens* profiles (Zhao et al., 2025), providing demographics across 20 domains. These profiles derive from the PRISM Alignment dataset (Kirk et al., 2024), which is real interaction data spanning 1,500 participants across 75 countries. The personas reflect realistic demographic diversity, and generating the surrounding multi-session content with an LLM follows the same accepted methodology as PersonaLens itself and other recent long-term memory benchmarks. Crucially, unlike open-ended generation, every query is annotated with dated, atomic reference facts, making its ground truth explicit and

directly checkable rather than a matter of subjective judgment. We generate *in-situ* queries involving planning and follow-through rather than standalone factoid prompts via a three-step pipeline:

Step 1: Query Generation. An LLM generates candidate queries, each including a query date, reasoning, and 3–5 **required references**. We retain candidates with average query–reference cosine similarity $\gamma \leq 0.3$ and keep at most $n_q \leq 15$ queries per user, prioritizing those with the lowest similarity scores.

Step 2: Timeline Synthesis. For each query, an LLM builds a chronologically ascending timeline of 5–10 events, starting shortly before the earliest required-reference date. Each required reference is embedded in exactly one timeline event, while remaining events are persona-grounded **filler** chats for a realistic multi-domain conversation history.

Step 3: Dialogue Expansion. Each timeline event is expanded into a multi-turn conversation (5–30 turns) via an LLM, conditioned on user demographics and domain summaries for logical continuity. A separate memory-extraction step (Section 3.2) converts these logs into the structured fact store used at test time.

Scope and Generalizability. MemoryQuest is intentionally designed as a challenging stress-test for semantically indirect long-term personalization. We therefore do not present MemoryQuest as representative of all retrieval workloads; rather, it targets an important subset of assistant interactions: underspecified requests that require recovering temporally distributed, personalized evidence from long histories. To assess whether PGR's benefits extend beyond this constructed setting, we also evaluate on ImplexConv (Li et al., 2025) and PersonaMem (Jiang et al., 2025), two independently published benchmarks with different construction procedures and task objectives.

5 Experiments

We evaluate PGR against multiple baselines on long-term, multi-session conversational tasks to test whether prospective retrieval improves personalization over retrospective retrieval.

5.1 Datasets

MemoryQuest (*Ours*) includes 50 users and **535 queries** across 20 task domains; each query requires 3–5 references, which serve as retrieval and answer ground truth. Users have an average of 77.6 sessions with 6.2 exchanges per session.

ImplexConv (Li et al., 2025) provides long multi-session dialogues with opposed and supportive implicit-reasoning scenarios; we use only the *opposed* setting, where relevant context is semantically distant and contradicts the surface persona. Each query has a single required reference; we evaluate **196 queries** across 85 users (sampled from 1,550 users in the full dataset), with an average of 110 sessions and 5.8 exchanges per session.

PersonaMem (Jiang et al., 2025) provides persona profiles and long conversation histories; we use the 32k chat-history variant and sample 50 random profiles, excluding `ask_to_forget` and `sensitive_info` queries to focus on preference-grounded personalization, yielding **894 queries** (sampled from 200 users and 3,441 queries in the full dataset); each user has a single long chat history averaging 116 exchanges (∼32k tokens).

Overall, we evaluate 1,625 queries across 185 user profiles. We release the MemoryQuest generation framework to support reproducibility and scaling to 1,500+ user profiles.

5.2 Baselines

We compare PGR against three other retrieval baselines. We use Azure OpenAI GPT-4o for simulation and answer generation, and `text-embedding-3-small` for embedding and retrieval (OpenAI, 2024). Additional experiments using PGR-TOT with DeepSeek-V3.2 as the generation model are reported in Appendix A.

Mem0^{g}. (Chhikara et al., 2025) utilizes a Neo4j-backed entity-relation graph to store memories. Retrieval combines vector search with graph relation extraction to test if explicit relational structures can compensate for the absence of prospection. We evaluate at $K = 20$.

GraphRAG. (Edge et al., 2024) indexes dialogue sessions into an entity graph and retrieves the top-$K = 5$ community reports via embedding similarity to evaluate structured graph summarization as a non-prospective alternative.

TaciTree. (Li et al., 2025) builds a hierarchical memory tree by clustering facts via UMAP and Gaussian Mixture Models (max $k = 6$ per cluster), summarizing each cluster into a leaf node with an LLM, and iterating until the root holds $L = 15$ clusters. At inference, an LLM prunes top-down to select the $K = 5$ most relevant leaf summaries.

Query-expansion baselines. Because PGR

mechanistically resembles iterative LLM query expansion, we additionally compare against the two most directly related single-pass query-expansion methods, both *budget-matched* to PGR-TOT (same embedder, generation LLM, and ∼35 retrieved facts). **Multi-query RAG** (Jagerman et al., 2023) prompts the LLM to reformulate the query into several probes and retrieves the union of their nearest facts. **HyDE** (Gao et al., 2023) generates hypothetical answer passages and averages their embeddings with the query for retrieval. We also report **Standard RAG** (Lewis et al., 2020) at a budget-matched K=36 (in addition to the K=20 setting of Figure 3) to isolate the effect of retrieval budget from that of prospection. All use embedding threshold $\tau = 0.3$.

Hyperparameters. All PGR variants use $K = 5$ facts per sub-query with embedding threshold $\tau = 0.3$. Query augmentation uses a stricter τ_o=0.6 with K_o=5 facts to seed Phase 1 with high-confidence context. The refinement loop runs for at most $I_{\max} = 10$ iterations and stops early when fewer than $\delta = 5$ new facts are retrieved.

5.3 Evaluation Metrics

We use Azure OpenAI GPT-5.2 for evaluating the responses. Prompts are provided in the Appendix.

Retrieval-level: For each method, we obtain the *retrieved context* (e.g., the set of facts or chat excerpts retrieved to generate the answer). The LLM is given the query, the list of required references, and the retrieved context, with a simple instruction to independently identify (binary, Yes/No) whether each reference is found explicitly or implicitly. We report **Recall** as the fraction of required references judged present, averaged over queries. For ImplexConv and PersonaMem, the judgment is binary. Since MemoryQuest has multiple references, we report the fraction as well as **Recall**$_{Exact}$, which is 1 if all references are identified, else 0.

Response-level: We also compare the *final answers* generated using PGR and each baseline using an LLM-based user simulator with access to the full user context. For each query, the simulator is shown both responses along with the required reference facts, and selects the response that is more useful and proactive (e.g., acknowledges past context, anticipates needs, and provides actionable guidance), or marks a tie. To reduce position bias, we repeat the evaluation with reversed response order and average the results. Prompts are provided in the Appendix. We report the **Win Rate** (%) of PGR against each baseline as the fraction of queries where the simulator prefers the PGR response.

We also conducted a blinded **human evaluation** on Amazon MTurk, to verify the efficacy of LLM-as-a-Judge on a smaller subset of queries, comprising approximately 4% of MemoryQuest, 5% of ImplexConv, and 2% of PersonaMem queries. Human evaluations were conducted on each PGR-baseline pair, and results are reported in Table 1. IRB exemption was received for this study.

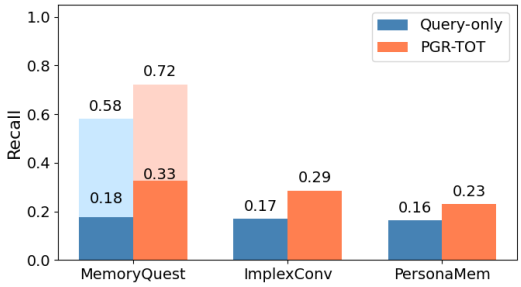


Figure 3: Results comparing Retrieval Recall using Query-only and PGR-TOT. For MemoryQuest, lighter and darker bars show Recall and Recall$_{Exact}$ respectively.

6 Results and Discussion

To establish the advantage of prospective retrieval over standard RAG, Figure 3 compares retrieval performance between a *Query-only* baseline (**Standard RAG**; using top $K = 20$ facts) and PGR-TOT on MemoryQuest. We decompose results into Recall$_{Exact}$ (dark regions), representing queries where the entire ground-truth set is recovered, and fractional recall (light regions). The visualization highlights the "prospection gap": while the Query-only baseline achieves a total recall of only 0.58, PGR-TOT reaches 0.723. This suggests that simulating future trajectories helps PGR bypass the semantic similarity bottleneck and recover the complete thread of chronologically distant facts required for complex personalization.

Results in Table 1, demonstrate that PGR outperforms traditional retrieval baselines across all evaluated datasets and metrics.

6.1 Retrieval Performance

On the MemoryQuest benchmark, PGR-TOT achieves the highest recall, nearly 3x the performance of the strongest baseline, Mem0, and vastly exceeding GraphRAG. This trend persists across ImplexConv and PersonaMem, where PGR consistently surfaces relevant context that passive seman-

Dataset →	MemoryQuest				ImplexConv			PersonaMem		
Method ↓	Recall	Recall_{Exact}	Win Rate$_{PGR}$		Recall	Win Rate$_{PGR}$		Recall	Win Rate$_{PGR}$	
			LLM	Humans*		LLM	Humans*		LLM	Humans*
GraphRAG	0.160	0.003	96.6	90.0	0.072	96.2	70.0	0.011	98.4	80.0
TaciTree	0.235	0.024	94.3	75.0	0.122	89.9	60.0	0.073	88.7	65.0
Mem0^g	0.256	0.015	94.7	95.0	0.050	93.5	70.0	0.121	92.9	65.0
PGR-TOT	**0.723**	**0.326**	-	-	**0.286**	-	-	**0.229**	-	-

Table 1: Average Recall and Win Rate(%) of PGR-TOT over baseline methods. Human* evaluations were done on a subset of queries. Higher is better. All methods use GPT-4o.

Method	MemoryQuest		ImplexConv
Standard RAG (K=20)	0.580 ,	0.176	0.168
Standard RAG (K=36)	0.583 ,	0.178	0.168
Multi-query	0.608 ,	0.206	0.199
HyDE	0.686 ,	0.301	0.230
PGR-TOT	**0.723** ,	**0.326**	**0.286**

Table 2: Budget-matched comparison against query-expansion baselines. Both Recall and Recall_{Exact} are reported for MemoryQuest (first two columns, respectively). All methods use GPT-4o.

Human \ LLM	PGR	Tie	Baseline
PGR	70.0	4.0	2.0
Tie	8.0	0.0	0.7
Baseline	12.0	2.0	1.3

Table 3: Aggregated comparison matrix between judgments of humans (rows) and LLM (columns). Values denote percentage (%).

tic search fails to identify. These gains are particularly pronounced in Recall_{Exact} metrics, indicating that PGR is capable of retrieving the complete set of facts required for complex reasoning.

Effect of Budget (K). Recall can grow with the amount of retrieved context; hence, we verify that PGR's advantage reflects prospection rather than a larger retrieval budget (Table 2). Increasing Standard RAG's budget by ∼80% (K=20 → 36) yields only a marginal recall gain (0.580 → 0.583). Since our retrieval also applies a semantic threshold ($\tau = 0.3$) that blocks very low-similarity matches, a larger K simply surfaces near-duplicates, capping effective retrieval at ∼23 facts. PGR-TOT retrieved an average of K=36 facts in MemoryQuest tasks and K=28 in ImplexConv tasks. It outperforms other query-expansion methods, such as Multi-query and HyDE, when they are run under the matched budget on both datasets.

In Appendix B, we report 95% bootstrap confidence intervals and paired Wilcoxon signed-rank tests against the strongest baseline, showing that PGR-TOT's improvement is large and statistically significant on every dataset.

6.2 Quality of Generated Responses

The improvement in retrieval translates directly to superior response quality. As shown in Table 1, PGR achieves dominant win rates in LLM-as-a-judge evaluations, ranging from **88.7% to 98.4%**. Human evaluations confirm this preference, with **annotators favoring PGR responses in 60–95%** of cases. Table 3 reports the aggregated Human-vs-LLM agreement matrix across all baseline pairs and datasets (total 150 queries). Humans and the LLM judge agree on preferring PGR in **70%** of all evaluated cases, with a chance-corrected agreement of Gwet's AC1 = 0.66. The human study functions as an independent validation of the automatic judge: even when the two disagree, human preference aligns with PGR's advantage, with humans favoring a baseline over PGR in only 2% of cases. Human annotators noted that PGR-based responses were significantly more proactive, correctly utilizing long-term personal history to provide actionable guidance. Because the evaluation rubric is applied blindly and symmetrically, any baseline retrieving the same facts could win. Furthermore, PGR's lead on the rubric-independent Recall metric confirms that these gains reflect actual retrieval quality rather than stylistic bias.

6.3 Ablation Study

We conduct ablations on MemoryQuest to isolate the contributions of iterative retrieval, the use of the prospection summary in answer generation, and the type of prospection strategy within PGR.

Iterative Prospection consistently improves memory recall and answer quality. As shown in Table 4, both PGR-COT and PGR-TOT achieve higher recall and win rates when sub-queries are retrieved and refined iteratively.

Prospection-Augmented Generation (PAG). Conditioning answer generation on the prospection summary further boosts performance. Table 5 shows that including PAG improves recall and win rate over using retrieved facts alone for both COT and TOT variants.

Prospection Strategy. Comparing PGR-TOT and PGR-COT (Table 6) reveals that branching tree-of-thought exploration slightly outperforms linear chain-of-thought in Recall, though win rate differences are modest.

The empirical results highlight a limitation of lookup-based memory systems: retrieval is largely driven by local similarity in the index. In contrast, PGR uses iterative retrieval to anticipate information needs during inference. This allows the system to surface relevant memories that are not directly aligned with the current turn but are necessary for completing long-horizon tasks. Overall, these results suggest that prospection is a useful complement to similarity-based retrieval in building more personalized conversational agents.

Method	Recall		Win Rate
	Base	Iterative	Iterative
PGR-TOT	0.677	0.723	70.13
PGR-COT	0.645	0.718	75.41

Table 4: Effect of iterative prospection.

Method	Recall		Win Rate
	PAG	∼PAG	PAG
PGR-TOT	0.722	0.703	75.73
PGR-COT	0.735	0.722	74.37

Table 5: Effect of prospection summary.

PGR Method	Recall	Win Rate (%)
TOT *vs* COT	0.723 > 0.718	55.67

Table 6: Effect of prospection strategies.

7 Conclusion

We introduce Prospection-Guided Retrieval (PGR), a retrieval policy independent of the underlying memory system. PGR replaces similarity-based lookup with an iterative *simulate* → *retrieve* → *refine* loop, surfacing relevant information that is difficult to recover using standard embedding- or graph-based search over long histories. This addresses a key limitation of existing RAG and GraphRAG systems, which often miss low-similarity or temporally distant evidence. We also introduce MemoryQuest, a multi-session benchmark requiring coordination of multiple dated references under low query–reference similarity, and evaluate PGR on MemoryQuest, ImplexConv, and PersonaMem. Across datasets, PGR improves recall, including full multi-reference recovery, and produces answers preferred by both LLM judges and humans. Overall, results show that prospection is effective for retrieving hard-to-access, dispersed memories, complementing nearest-neighbor retrieval in long-horizon personalized assistants.

Limitations

PGR's iterative process improves accuracy by carefully planning and exploring potential future trajectories, but this comes at the cost of additional LLM calls compared to single-pass retrieval. While PGR requires significantly fewer LLM calls than hierarchical tree-based alternatives (Appendix Table 10) and remains highly competitive with simpler baselines, its multi-phase prospection could introduce latency in strict real-time applications. However, because the underlying embedding-based retrieval sub-steps are fully parallelizable, the actual operational overhead is primarily bounded by a few sequential reasoning calls. Future work could reduce this overhead by training specialized lightweight retrievers on the generated prospection traces, enabling faster memory selection while retaining a large part of the anticipatory reasoning benefit.

References

Donna Rose Addis, Alana T. Wong, and Daniel L. Schacter. 2007. Remembering the past and imagining the future: Common and distinct neural substrates during event construction and elaboration. *Neuropsychologia*, 45(7):1363–1377.

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

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual*

Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1762–1777, Toronto, Canada. Association for Computational Linguistics.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Demis Hassabis, Dharshan Kumaran, Seralynne D. Vann, and Eleanor A. Maguire. 2007. Patients with hippocampal amnesia cannot imagine new experiences. *Proceedings of the National Academy of Sciences*, 104(5):1726–1731.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 874–880. Association for Computational Linguistics. (FiD: Fusion-in-Decoder approach).

Rolf Jagerman, Honglei Zhuang, Zhen Qin, Xuanhui Wang, and Michael Bendersky. 2023. Query expansion by prompting large language models. *arXiv preprint arXiv:2305.03653*.

Bowen Jiang, Yuan Yuan, Maohao Shen, Zhuoqun Hao, Zhangchen Xu, Zichen Chen, Ziyi Liu, Anvesh Rao Vijjini, Jiashu He, Hanchao Yu, and 1 others. 2025. Personamem-v2: Towards personalized intelligence via learning implicit user personas and agentic memory. *arXiv preprint arXiv:2512.06688*.

Hannah Rose Kirk, Alexander Whitefield, Paul Röttger, Andrew Michael Bean, Katerina Margatina, Rafael Mosquera, Juan Manuel Ciro, Max Bartolo, Adina Williams, He He, Bertie Vidgen, and Scott A. Hale. 2024. The PRISM alignment dataset: What participatory, representative and individualised human feedback reveals about the subjective and multicultural alignment of large language models. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Xintong Li, Jalend Bantupalli, Ria Dharmani, Yuwei Zhang, and Jingbo Shang. 2025. Toward multi-session personalized conversation: A large-scale dataset and hierarchical tree framework for implicit reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 11493–11506. Association for Computational Linguistics.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of llm agents. *Preprint*, arXiv:2402.17753.

OpenAI. 2024. New embedding models and api updates. https://openai.com/index/new-embedding-models-and-api-updates/. Refers to text-embedding-3-small.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22.

Daniel L. Schacter, Roland G. Benoit, and Karl K. Szpunar. 2017. Episodic future thinking: Mechanisms and functions. *Current Opinion in Behavioral Sciences*, 17:41–50.

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

Jing Xu, Arthur Szlam, and Jason Weston. 2022. Beyond goldfish memory: Long-term open-domain conversation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5180–5197, Dublin, Ireland. Association for Computational Linguistics.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023b. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

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

Appendix

A PGR with additional LLMs

To assess whether PGR's retrieval gains are robust to the choice of generation model, we run PGR-TOT with additional LLMs: DeepSeek-V3.2 and LLama-3.3-70B, for prospection and answer generation, while keeping the same retrieval pipeline,

embeddings, and hyperparameters. Table 7 reports retrieval Recall metrics for the base single-phase and iterative two-phase variants across all three datasets. Results are consistent with the main GPT-4o findings, confirming that PGR's gains generalize across generation models.

Dataset ↓ / PGR-method →	Recall	
	Base	**Iterative**
DeepSeek-V3.2		
MemoryQuest	0.671 [0.245]	**0.750** [**0.348**]
ImplexConv	0.308	**0.374**
PersonaMem	0.210	**0.229**
Llama-3.3-70B		
MemoryQuest	0.657 [0.239]	**0.707** [**0.299**]
ImplexConv	0.215	**0.236**
PersonaMem	0.185	**0.210**

Table 7: PGR-TOT with alternative LLMs. For MemoryQuest, [Recall$_{Exact}$] is reported in brackets.

B Significance Testing

To quantify uncertainty, we report recall with 95% confidence intervals (percentile bootstrap over per-query recall, matching the normal-approximation mean ± 1.96 SE) for every method across all three datasets in Table 9; the intervals for PGR-TOT are non-overlapping with those of all baselines. Because all methods are evaluated on the same queries, we further assess significance with a paired Wilcoxon signed-rank test on per-query recall differences against the strongest baseline per dataset (Table 8). PGR-TOT's improvement is large and highly significant on every dataset.

Dataset	vs	Δ Recall (95% CI)	p
MemoryQuest	Mem0^g	0.466 [0.440, 0.493]	<1e-70
ImplexConv	TaciTree	0.163 [0.092, 0.235]	<1e-5
PersonaMem	Mem0^g	0.109 [0.081, 0.137]	<1e-12

Table 8: Mean per-query recall improvement of PGR-TOT over the strongest baseline, with 95% bootstrap CIs and paired Wilcoxon signed-rank p-values.

C Computational Cost

Table 10 reports the average number of LLM calls and unique facts retrieved per query for PGR-TOT and each baseline. PGR-TOT requires approximately 4–5 LLM calls per query (one Phase-1 prospection + avg. Phase-2 refinements + one prospection summary + one answer generation), compared to 1 for GraphRAG and ∼2 for Mem0^g.

Dataset	Method	Recall (95% CI)
MemoryQuest	GraphRAG	0.160 [.141, .180]
	TaciTree	0.236 [.212, .260]
	Mem0^g	0.256 [.234, .278]
	PGR-TOT	**0.723** [.702, .744]
ImplexConv	GraphRAG	0.073 [.036, .109]
	TaciTree	0.122 [.077, .168]
	Mem0^g	0.050 [.017, .091]
	PGR-TOT	**0.286** [.224, .352]
PersonaMem	GraphRAG	0.011 [.004, .019]
	TaciTree	0.073 [.056, .091]
	Mem0^g	0.121 [.100, .143]
	PGR-TOT	**0.229** [.202, .257]

Table 9: Recall with 95% bootstrap confidence intervals for each method across all datasets.

TaciTree's higher count reflects its level-by-level LLM pruning at inference (offline tree-construction calls excluded). The embedding-based retrieval sub-steps are parallelizable and add negligible latency relative to the LLM calls.

In terms of tokens, PGR-TOT consumes on average ∼14K input and ∼3K output tokens per query on MemoryQuest, which corresponds to roughly **$0.07 per query** at current GPT-4o rates: a small absolute cost relative to the substantial gains in retrieval and response quality. Importantly, much of this input is not intrinsic to the method: a large share is fixed few-shot prompt scaffolding re-sent on each prospection call rather than query-specific content, and is readily reduced via shorter prompts or prompt caching. As noted in the Limitations, distilling prospection traces into a lightweight retriever offers a further route to lower inference cost while retaining most of the benefit.

D Example of End-to-End PGR

Figures 4 to 8: Complete PGR-TOT trace on a MemoryQuest query with four successfully retrieved references. Phase 1 (initial TOT) surfaces 15 facts, followed by two iterative refinement phases. Note how node constraints progressively evolve as retrieved facts ground the simulation in the user's specific context.

Dataset	GraphRAG	Mem0^g	TaciTree	PGR-TOT	PGR Ph-1 Facts	PGR Ph-2 New Facts
MemoryQuest	1.0	2.0	∼13.2	**4.79**	26.78	8.41
ImplexConv	1.0	2.0	∼15.9	**4.32**	22.41	3.85
PersonaMem	1.0	2.0	∼9.7	**4.41**	13.04	4.83

Table 10: Average query-time LLM calls and unique facts retrieved per query for PGR-TOT.

Query, Date, and Required References

```
Date:  2026-03-05
Query: "I want to pre-order a new game that just got announced.
        Check if it makes sense for me to do it now."

Required references (all 4 retrieved by PGR-TOT):
  [1] GPU overheating issue -- user plans to reduce overheating during gaming (2026-02-20).
  [2] March trip funds locked -- user frames trip money as 'already spent' (2026-02-23).
  [3] Upcoming online course expenses constraining GPU/hardware budget (2026-03-03).
  [4] Backlog of unfinished games -- user committed to completing RDR2 before any new
      title (2026-03-01).
```

Figure 4: Sample Query and its ground-truth references.

Phase 1 — Initial TOT (initial, pre-retrieval)

```
A1  Check the game's release date and pre-order availability
    constraints: official announcement, platform compatibility, region restrictions
    +-- A2  Look up gameplay trailers or developer previews
    |       constraints: video quality, gameplay mechanics, genre preference
    |       +-- A4  Read reviews from early access / beta testers
    |       |       constraints: reviewer credibility, performance feedback
    |       |       +-- A7  Decide if game aligns with personal gaming preferences
    |       |               constraints: time available, interest in story/mechanics
    |       |               +-- A9  [leaf] Pre-order or wait for post-release reviews
    |       |                       constraints: confidence in purchase, willingness to wait, FOMO
    |       +-- A5  [leaf] Compare with similar games in the genre
    |               constraints: gameplay depth, replay value, prior genre experience
    +-- A3  Check pre-order bonuses or exclusive content
            constraints: bonus relevance, edition types, availability
            +-- A6  Check the pre-order price and payment options
                    constraints: budget, refund policy, currency exchange rates
                    +-- A8  [leaf] Evaluate risk of bugs / incomplete features at launch
                            constraints: developer track record, past launch issues, patch history
```

Phase 1 — Retrieved Facts (all 15 facts passed to Iteration 1)

```
[2026-03-01] [Preference] User prefers finishing one main game at a time; avoids starting
                          new games until current one is completed or consciously dropped.
[2026-02-23] [Preference] 2-rule purchase system: under EUR 10 requires 24-hr wait;
                          over EUR 10 must fit fun money AND be played immediately.
                          User now muting sale notifications until after March trip.
[2026-02-23] [Preference] Frames trip money as 'already spent' to mentally lock it.
                          Rule: "Trip money is locked. Games from fun money only.
                          If fun money = 0 EUR, wishlist and wait."
[2026-02-23] [Goal]       Fun money for March: EUR 10 (tight) to EUR 15-20 (normal);
                          trip savings locked in separate pocket labelled "DO NOT TOUCH".
[2026-03-01] [Goal]       Prioritising RDR2 main story: 90-min capped sessions x2/week
                          (Wed afternoons). Witcher 3 parked guilt-free.
[2026-03-01] [Preference] Gaming sessions capped at 90 min to prevent late nights and
                          maintain focus for job applications and 9:00 AM alarm.
[2025-12-29] [Preference] Keyboard and mouse for gaming; mostly RPGs; prefers long sessions.
[2026-03-03] [Preference] Prefers maintenance over upgrades for GTX 1660 Super GPU due to
                          tight budget, locked travel savings, upcoming online course expenses.
[2026-02-21] [Goal]       Tracks fun money (EUR 15-20) weekly; Sunday 18:00 reminder
                          "Quick check: fun money (5 min)".
[2025-12-29] [Preference] Enjoys gaming evenings/weekends to save money but limits
                          bedtime shifts to avoid disrupting weekday routines.
[2026-01-29] [Goal]       EUR 50 bill as visual safety net for next 3 days -- keep untouched
                          as a measure of financial control.
[2026-02-14] [Preference] Prefers fun sci-fi or action movies for casual friend hangouts.
[2025-12-27] [Preference] Budget-friendly charger replacements; uses Amazon.es, PcComponentes.
[2026-01-16] [Goal]       Calendar note for Friday interviews: "Possible transport disruption
                          -- message recruiter by 7:30 if issues".
[2026-02-03] [Goal]       Recurring 'Available for interviews' note protecting early afternoons.
```

Figure 5: Phase 1: Initial Tree-of-Thought (TOT).

Iteration 1 — Personalized TOT (9 nodes; constraints updated to reflect retrieved facts)

```
A1  Check the game's release date and pre-order availability
    constraints: official announcement, platform compatibility, region restrictions
    +-- A2  Look up gameplay trailers or developer previews
    |       constraints: genre preference, [+relevance to current gaming interests, RPGs]
    |       +-- A4  Read reviews from early access / beta testers
    |       |       constraints: reviewer credibility, [+alignment with gaming preferences]
    |       |       +-- A7  Decide if game aligns with personal gaming preferences
    |       |               constraints: time available, [+current RDR2 commitment --
    |       |                            must finish before starting new games]
    |       |               +-- A9  [leaf] Pre-order or wait
    |       |                       constraints: FOMO, [+fun money only; current RDR2 focus]
    |       +-- A5  [leaf] Compare with similar games
    |               constraints: gameplay depth, [+preference for RPGs and long sessions]
    +-- A3  Check pre-order bonuses
            +-- A6  Check pre-order price and payment options
                    constraints: budget, [+fun money only; >EUR 10 requires immediate
                                 playability; refund policy]
                    +-- A8  [leaf] Evaluate risk of bugs at launch
                            constraints: track record, [+preference for maintenance over
                                         upgrades; tight budget and online course expenses]

New facts retrieved in Iteration 1 (5 new facts):
  [2026-02-21] [Goal]  Track fun money (EUR 15-20) weekly; Sunday 18:00 reminder.
  [2026-01-29] [Goal]  EUR 50 bill as visual safety net -- keep untouched.
  [2026-02-20] [Goal]  Plans to open PC case side panel during gaming sessions to
                       test airflow and reduce GPU overheating.
  [2025-12-29] [Event] Eye strain during late-night gaming; uses blue light filters,
                       reduced brightness, warm lighting to mitigate fatigue.
  [2026-02-23] [Goal]  Weekly Sunday reminder "March trip fund still locked" to
                       reinforce saving discipline.
```

Figure 6: Phase 2 (Iteration I): Personalized refinement with retrieved facts.

Iteration 2 — Further Refined TOT (9 nodes; constraints tightened)

```
A1  Check the game's release date and pre-order availability
    constraints: [+release date relevance to March trip AND RDR2 completion goal]
    +-- A2  Look up gameplay trailers or developer previews
    |       constraints: genre preference, RPGs, [+alignment with 90-min session cap]
    |       +-- A4  Read reviews from early access / beta testers
    |       |       constraints: reviewer credibility, [+focus on avoiding games with
    |       |                    launch bugs or incomplete features]
    |       |       +-- A7  Decide if game aligns with personal gaming preferences
    |       |               constraints: [+RDR2 in progress; 90-min sessions x2/week;
    |       |                            starting new game violates current commitment]
    |       |               +-- A9  [leaf] Pre-order or wait
    |       |                       constraints: [+fun money EUR 15-20 for March;
    |       |                                     trip money non-negotiable; FOMO low]
    |       +-- A5  [leaf] Compare with similar games
    |               constraints: [+RDR2/Witcher 3 backlog already present]
    +-- A3  Check pre-order bonuses
            +-- A6  Check pre-order price and payment options
                    constraints: [+fun money capped EUR 15-20; >EUR 10 requires
                                 immediate playability -- not possible while on RDR2]
                    +-- A8  [leaf] Evaluate risk of bugs at launch
                            constraints: [+GPU overheating issue; prefers maintenance;
                                         online course costs in March]

New facts retrieved in Iteration 2 (2 new facts):
  [2026-02-07] [Goal]  Flag 'bad-weather mornings' in weekly reviews to avoid early
                       outdoor plans unexpectedly.
  [2026-01-27] [Goal]  Weekly co-op gaming with friends Thursdays 9:00--11:30 PM;
                       marked Busy to avoid interview conflicts; wrap-up reminder
                       at 11:45 PM for early Friday wake-up.
```

Figure 7: Phase 2 (Iteration II): Further refinement with tighter constraints.

Prospection Summary (passed to answer generation)

```
The user may initially focus on confirming release date, platform compatibility, and
pre-order availability, while exploring trailers and bonuses to gauge interest.
Considerations then shift to gameplay mechanics, genre alignment, and budget constraints.
Over time, the decision is further refined by comparing the game with similar titles,
assessing early reviews for launch issues, and ensuring it fits the current gaming
schedule (finishing RDR2 before starting a new game). By the final stage, the decision
hinges on specific factors: release timing relative to the March trip, strict fun-money
budget constraints, and a preference for 90-minute capped sessions -- ultimately pointing
toward adding the game to the wishlist rather than pre-ordering now.
```

Reasoning & Final Answer

```
Reasoning:
  - Trip money is mentally locked ('already spent'); fun money is the only game budget.
  - Fun money for March: EUR 15-20 depending on stress and cash availability.
  - 2-rule system: >EUR 10 must fit fun money AND be something played immediately --
    this game cannot be played immediately; RDR2 is in progress.
  - Muting sale notifications on gaming platforms until after the March trip.
  - Currently committed to finishing RDR2 before starting any new game.
  - GPU overheating issue flagged (2026-02-20); prefers maintenance over upgrades.
  - Upcoming online course expenses further constrain discretionary spending in March.
  - Backlog already includes Witcher 3 (parked); adding another title is premature.
  - Pre-ordering now would violate the 2-rule system (can't play immediately).
  - Query date is 2026-03-05; trip savings locked and non-negotiable.

Answer:
  "Pre-ordering now likely isn't the right move. You're mid-way through RDR2 and your
   2-rule system requires any purchase over EUR 10 to be something you'll play
   immediately -- that's not the case here. Your March trip fund is locked, and fun
   money is limited this month. Add it to your wishlist, mute the notifications (as
   you planned), and revisit after the trip or once RDR2 is wrapped up."
```

Figure 8: Prospection summary and final answer.

E Prompt Templates

Phase 1: COT Prospection Generation Prompt

You are simulating how a person would think via prospection: mentally projecting themselves into the future to anticipate actions, needs, and outcomes.
Instructions:

- Given a user query, break it down into a sequence of steps a human would take, notice, or experience in that situation.
- Consider likely future developments, emerging needs, and hidden objectives.
- Avoid abstract planning or meta-cognition style phrases (DO NOT start with words like imagine, anticipate, think, plan, decide, evaluate, review, verify, etc.).
- Steps must be concrete actions in real time or situational questions starting like "how...","whether...","what..." etc.
- Each step should focus on a distinct dimension of the scenario (e.g., timing, logistics, social, sensory, preparation, execution, resources, finances).
- Avoid redundancy or lexical overlap between steps. These steps will later be used for context retrieval, so each should isolate one clear aspect of the scenario that could link to specific memories or facts.

For each step, include:

- `action`: A specific step or observation a person would make, experience, or mentally simulate in this scenario
- `constraints`: The factors or aspects that influence that action (preferences, limits, timing, logistics, social cues, costs, sensory issues, etc.)

Output a chronological sequence of prospection steps in a JSON list.
Examples (only for illustration):
`3 Input/Output pairs`
—
Date: `query_date`

User query: " `query` " `main_query_context`
Output Steps [json]:

Phase 2: COT Prospection Generation Prompt

You are simulating how a person would think via *prospection* — mentally projecting themselves into the future to anticipate actions, needs, and outcomes.
Given a user query and their personal context, transform the initial thinking steps into a personalized sequence.
Date: `query_date`

User Query: " `query` "

Initial Thinking Steps: `initial_steps_json`

Personal Context: `retrieved_context`
Instructions:

- Given the specific user facts (preferences, constraints, habits, events), update the initial prospection sequence. Dates (if provided) correspond to when the fact was created or updated after a conversation. Pay attention to infer the temporal relevance of a fact to the query based on dates.
- Revise, refine, or reorganize the steps as needed to make it precisely tailored and applicable to the user.
- Consider likely future developments, emerging needs, and hidden objectives.
- Avoid abstract planning or meta-cognition style phrases (DO NOT start with words like imagine, anticipate, think, plan, decide, evaluate, review, verify, etc.).
- Steps must be concrete actions in real time or situational questions starting like "how...","whether...","what..." etc.
- Each step should focus on a distinct dimension of the scenario (e.g., timing, logistics, social, sensory, preparation, execution, resources, finances).
- Avoid redundancy or lexical overlap between steps. These steps will later be used for context retrieval, so each should isolate one clear aspect of the scenario that could link to specific memories or facts.

Output a JSON list where each step contains:

- "action": A clear, context-aware step or observation the user would take, experience, or mentally simulate in this scenario, reflecting personalization and possible adjustments.
- "constraints": User-related factors or aspects that influence this action (preferences, limits, timing, logistics, social cues, costs, other issues, etc.).

Example output format (only for illustration):
`3 Input/Output pairs`
Action and constraints should be descriptive and specific to enable effective information retrieval.
Output Updated Steps [json]:

Phase 1: TOT Prospection Generation Prompt

You are simulating how a person would think via *prospection* — mentally projecting themselves into possible future situations and outcomes. Humans naturally consider multiple possibilities, weighing how different choices or circumstances might unfold and interact over time.
Generate a structured representation of this process, similar to a Tree of Thoughts, capturing multiple possible futures, their sub-goals, and information needs that could later guide memory retrieval or action.
Instructions:

- Given a user query, imagine how a person would naturally think about alternative possibilities and sequential outcomes.
- Each node represents a concrete situational action or observation in real time. Avoid abstract planning or meta-cognition style phrases (DO NOT start with words like imagine, anticipate, think, plan, decide, evaluate, review, verify, etc.)
- Capture branching possibilities — e.g., different options, conditions, or paths the person could take.
- Each action-constraint pair should focus on a distinct dimension (logistical, temporal, physical, sensory, social, financial, etc.)
- Avoid redundancy or lexical overlap between nodes, as they will be later used for context retrieval. Hence, each should isolate one clear aspect of the scenario that could link to specific memories or facts.
- Each node includes:
  - "action_id": unique ID (A1, A2, ...)
  - "action": A specific step or observation a person would make, experience, or mentally simulate in this scenario
  - "constraints": The factors or aspects that influence that action (preferences, limits, timing, logistics, social cues, costs, sensory issues, etc.)
  - "parent": ID of the node it follows from (null for the first)
  - "children": IDs of next possible actions branching from it

Example output format (only for illustration):
`3 Input/Output pairs`
—
Given the input below, return the Prospection Tree, with each node as a distinct dimension of possible future situations, as valid JSON without any additional text or explanation.
Date: `query_date`

User query: " `query` " `main_query_context`
Output Prospection Tree [json]:

Phase 2: TOT Prospection Generation Prompt

You are simulating how a person would think via *prospection* — mentally projecting themselves into possible future situations and outcomes. Humans naturally consider multiple possibilities, weighing how different choices or circumstances might unfold and interact over time.
Instructions:

- You are given an initial "Tree of Thoughts Prospection" for a user query. Each node represents a concrete situational action or observation with its constraints and branching possibilities.
- Using the provided user-specific facts as additional context, **revise, refine, prune, or expand nodes** to make the Prospection Tree more personalized and realistic. Dates (if provided) correspond to when the fact was created or updated after a conversation. Pay attention to infer the temporal relevance of a fact to the query based on dates.
- Each node should remain a concrete action or situational observation (avoid abstract planning/meta-cognition phrasing). DO NOT start with words like imagine, anticipate, think, plan, decide, evaluate, review, verify, etc.
- Keep nodes coherent, distinct, and avoid redundancy. Each node's action-constraint pair should focus on one clear dimension (logistics, temporal, social, sensory, financial, etc.), but now adjusted to the user's context if relevant.
- Branching is allowed: nodes can be added, removed, or reconnected to better reflect likely decisions, paths, or experiences for this particular user.
- Maintain the structure:
  - "action_id": unique ID (can reuse existing IDs or assign new ones)
  - "action": specific step or observation a person would make, experience, or mentally simulate in this scenario
  - "constraints": factors or aspects that influence that action (preferences, limits, timing, logistics, social cues, costs, sensory issues, etc.)
  - "parent": ID of the node it follows from (null for the root)
  - "children": IDs of next possible actions

Example (only for illustration):
`1 Input/Output pair`
—
Given a user query and their personal context, transform the initial Prospection Tree into a personalized sequence, following the above Instructions.
Date: `query_date`

User Query: " `query` "

Initial Prospection Tree: `initial_steps_json`
Personal Context:
`retrieved_context`
Output Updated Prospection Tree [json]:

Answer Generation Prompt

You are a helpful AI Assistant. Given the user's query and your memory of the user's personal context, craft a brief, personalized response.
First, reason over the provided user context to decide what is logically, temporally (if dates are provided), or situationally relevant to the query.

- List down (as bullet points) all the information that could matter into a set of atomic, one line facts.
- Each line should capture the essence of one or more provided context items. You may merge or rephrase facts for conciseness, but do not introduce new information. Always mention key entities or nouns.
- Consider how past situations, constraints, or ongoing circumstances could affect what the user is likely to do next given the intent behind the query.
- Infer and include facts that could plausibly affect planning, tradeoffs, timing, or feasibility of requested task.
- Aim to surface as many usable facts as possible, while keeping each line minimal and precise.

In the response text, proactively make explicit references to the provided facts and reasoning details you used for personalization. The response text should be crisp and easy to read. You must use markdown formatting to make it legible.
Return a valid JSON object in the following format: {"reasoning": "...", "answer": "..."}
User's Personal Context: `context`

`optional_date`

User Query: " `query` "

`optional_simulation_context`

Output Reasoning and Answer to User Query [JSON]:

Retrieval Evaluation Prompt

You are evaluating whether required references are present in a retrieved context (facts or conversation excerpts).
INPUTS:

- Query: `user_query`
- Required references (numbered; find out if these are mentioned in the retrieved context): `refs_list`
- Retrieved context (facts or chat excerpts that were retrieved for this query): `retrieved_context`

TASK:
For each required reference, indicate with Yes/No whether it is acknowledged or mentioned in the retrieved context (explicitly or implicitly). Output valid JSON with a single key "ack" whose value is the list: one item per reference, format as:

{ "ack" : ["1. Yes. <one-line reason>", "2. No. <one-line reason>", ...]}.

Numbering must match the input list. Only mark Yes if the retrieved context clearly contains or implies that reference.
OUTPUT [JSON]:

Pairwise Comparison Prompt

You are evaluating two AI-generated responses from the perspective of a specific user. User's context is provided to help judge which response is more useful.
`persona_text`

User Context (Profile/Facts): `facts_text`
—
Compare the two candidate responses. Use the following signals to make a choice.

- A strong response:
  - Acknowledges different or additional past experiences appropriately.
  - Connects them to the current or future situation in a logical way.
  - Identifies implications, anticipates follow-ups, and flags potential issues appropriately.
  - Provides concrete, actionable next steps or structured guidance.
- A weak response:
  - Is generic or non-personalized.
  - Reacts only to the surface wording of the query.
  - Provides vague advice that is hard to execute.
  - Misses clear opportunities to guide or anticipate needs.

Return a valid JSON with exactly two keys:
{
"reasoning": "Your brief comparison of the two responses, whether or not they are different, additional details, proactivity, or if they have no meaningful difference, why, etc.",
"choice": "q6cg" or "j52d" or "tie"
}
— Inputs: —
`optional_date`

User (Your) Query: " `query` "

`ground_truth_section`
—
Response `first_label` : `first_response`

Response `second_label` : `second_response`
—
Output [JSON]:

Memory Creation Prompt

Your task is to update a list of short, atomic facts about the user from a conversation given below. Each fact should be concise, self-contained, *non-redundant*, and useful for building the user profile for personalization.
Facts can include the user's:

- Personal details (e.g., name, age, location, relationships), important dates, and user's health/wellness information.
- Preferences and lifestyle choices, including likes, dislikes, and habits across food, activities, travel, entertainment, and services.
- Plans, goals, upcoming events, career details, and professional information.
- Miscellaneous favorites like books, movies, and brands and any relevant context that can help personalize future interactions.
- Situations or experiences the user has had or is currently going through.

Each fact should be a dictionary with these keys:

- "fid": Sequential ID / Unique identifier for the fact
- "info": The fact retrieved from given conversation
- "type": One of ['"Identity"', '"Preference"', '"Goal"', '"Interest"', '"Activity"', '"Event"']
  - Identity: Personal details like name, age, location, profession, qualifications, education, achievements or relationships. This reflects the core identity of the user.
  - Preference: Likes, dislikes, and habits, including food, entertainment, or lifestyle choices
  - Goal: Stated intentions, plans, or upcoming events in the conversation
  - Interest: Topics, hobbies, or subjects the user cares about
  - Activity: Actions the user does or has done, such as commute, attending events, visits, sending emails, or completing tasks
  - Event: Any situations, events, transitions, milestones, or life changes — positive or negative — including emotional, social, financial, materialistic, or health-related circumstances that had or will shape the user's choices and actions.
- "frequency": How many times a similar fact has appeared in previous conversations. Set to 1 for new facts, or increment by 1 for updating existing facts.
- "related_entities": List of keywords or concepts related to the fact (entities, topics, themes)
- "state": One of ['"update"', '"add"']
  - "update": Existing fact with incremented frequency
  - "add": New fact from this conversation

Example/Format:

Example/Format

[{"fid": "c2f10", "info": "Prefers vegan food and healthy lifestyle", "type": "Preference", "frequency": 3, "related_entities": ["food", "vegan", "health"], "state": "update"}, {"fid": "c3f5", "info": "Received a Masters in Genomics from UCL", "type": "Identity", "frequency": 2, "related_entities": ["education", "Masters degree", "Genomics"], "state": "update"}, {"fid": "c3f21", "info": "Inquiring about strategy and revenue of business companies X, Y and Z", "type": "Interest", "frequency": 2, "related_entities": ["business", "X", "Y", "Z", "revenue"], "state": "update"}, {"fid": "c4f6", "info": "Drafted a message to follow up with Ross on travel plans", "type": "Activity", "frequency": 3, "related_entities": ["message", "Ross", "travel", "planning"], "state": "update"}, {"fid": "NEW_1", "info": "Experiencing vericose veins in the legs due to long hours of standing", "type": "Event", "frequency": 1, "related_entities": ["health", "vericose veins", "standing"], "state": "add"}, {"fid": "NEW_2", "info": "User has hit their credit card limit and are being financially sensitive", "type": "Event", "frequency": 1, "related_entities": ["credit card", "limit", "financial", "sensitive"], "state": "add"}, {"fid": "NEW_3", "info": "Name is Harry", "type": "Identity", "frequency": 1, "related_entities": ["name", "personal_info"], "state": "add"}, {"fid": "NEW_4", "info": "Tax filing in California due on April 15", "type": "Goal", "frequency": 1, "related_entities": ["taxes", "California", "finance", "deadline"], "state": "add"}, {"fid": "NEW_5", "info": "Visited London for one week to attend Coldplay's concert", "type": "Activity", "frequency": 1, "related_entities": ["concert", "Coldplay", "London"], "state": "add"}]

Do not get biased from example facts. You must return user-specific facts strictly from the context given below.
CONTEXT:
Existing Facts: The following facts were extracted from this user's previous conversations:
`curr_facts`
New Conversation:
`content`
Now, based on the conversation above:

- If it reveals any information that closely matches, overlaps, repeats or contradicts an existing fact, create an UPDATED fact (modify "info" appropriately detailing the change if the fact evolved) under the same "fid", increment its "frequency", and mark its "state": "update".

- For any new information, create a NEW fact with the appropriate "fid" (starting with "NEW_") and other attributes. Mark its "state": "add".
- REMEMBER: Each fact should independently convey a unique, non-redundant, and meaningful observation. We aim for a minimal number of well-crafted facts that capture the maximum information about the user.
- Do not return existing facts that remain unchanged. Only return a list of updated or newly added facts.
- If the given conversation does not yield any new or updated facts, return empty list [].

Follow the provided format. Do not include any explanations, formatting, or extra text in the output.
Output facts [list]:

Memory Consolidation Prompt

You are given a list of similar facts or pieces of information about a user that need to be merged into a single coherent fact. These facts have been identified as highly similar based on their content and should be combined to reduce redundancy while preserving all important information.
Your task is to merge these facts into ONE comprehensive fact that:

- Combines all the information from the input facts
- Uses the most appropriate fact "type" from the input facts
- Creates a coherent "info" text that captures all the details
- Has a compact set of important related_entities based on info
- Combines all conversation IDs (removing duplicates)
- Preserves the frequency information appropriately

The merged fact should follow the given structure:

```
{
   "fid": <highest_fid_from_input_facts>,
   "info": "<comprehensive_info_
      combining_all_facts>",
   "type": "<most_appropriate_type_
      from_input_facts>",
   "frequency": <sum_of_all_frequencies>,
   "related_entities": [<compact_set_
      of_all_related_entities>],
   "conversation_ids": [<union_of_
      all_conversation_ids>],
   "merged_fids": [<list_of_all_
      input_fids>],
   "state": "add"
}
```

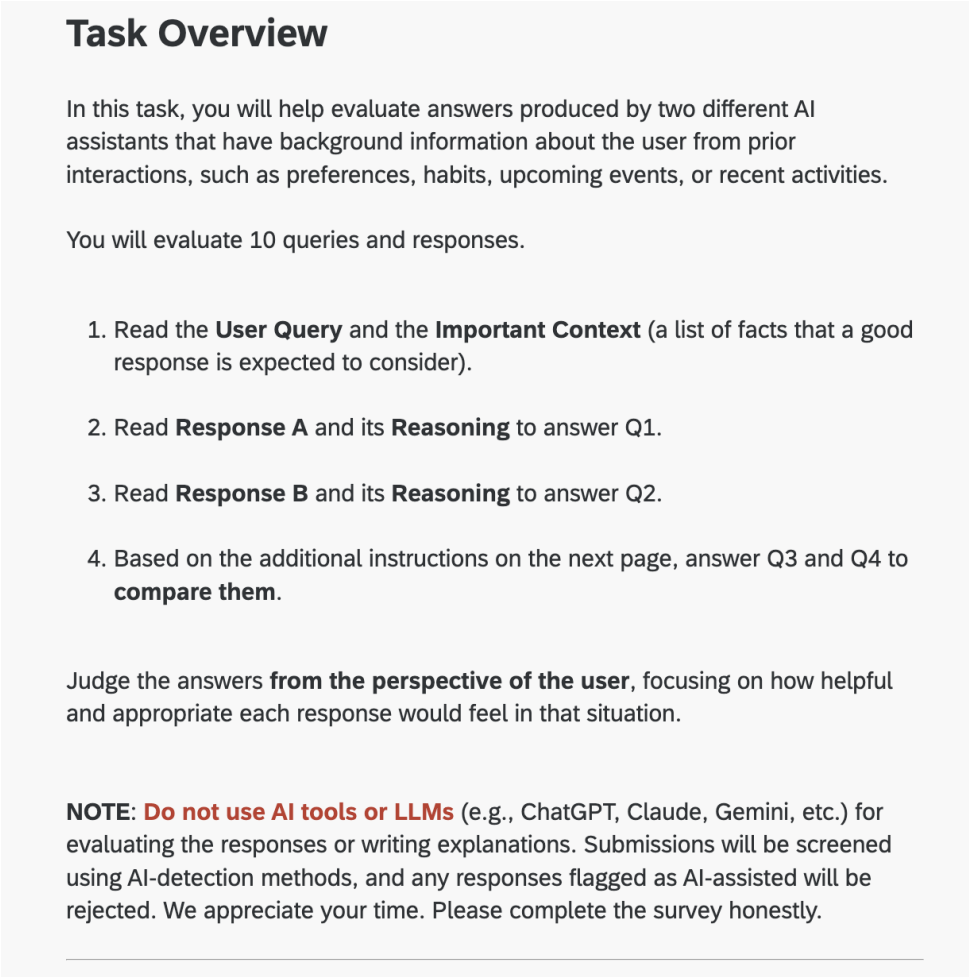
Guidelines:

- The "info" should be a single coherent sentence that captures the essence of all input facts.
- List of related entities should be concise, including only the important ones.
- The "merged_fids" field should contain ALL the original fact IDs that were combined

Return only the single merged fact dictionary. Do not include any explanations, formatting, or extra text.
Input facts to merge: `facts_to_merge`
Output [Merged fact]:

F Human Evaluation Survey

Human evaluations were conducted on Amazon Mechanical Turk, where workers were paid $8 per task comprising 10 pairwise response comparisons, with each comparison taking approximately 1–2 minutes, yielding an effective hourly rate of $24–48. Each query was annotated by 5 workers. The filter on annotators' country was set to USA, and the payment per task was hence adequately determined to be on par with the standard rates. IRB exemption was received for this study.



Task Overview

In this task, you will help evaluate answers produced by two different AI assistants that have background information about the user from prior interactions, such as preferences, habits, upcoming events, or recent activities.

You will evaluate 10 queries and responses.

1. Read the **User Query** and the **Important Context** (a list of facts that a good response is expected to consider).
2. Read **Response A** and its **Reasoning** to answer Q1.
3. Read **Response B** and its **Reasoning** to answer Q2.
4. Based on the additional instructions on the next page, answer Q3 and Q4 to **compare them**.

Judge the answers **from the perspective of the user**, focusing on how helpful and appropriate each response would feel in that situation.

NOTE: **Do not use AI tools or LLMs** (e.g., ChatGPT, Claude, Gemini, etc.) for evaluating the responses or writing explanations. Submissions will be screened using AI-detection methods, and any responses flagged as AI-assisted will be rejected. We appreciate your time. Please complete the survey honestly.

Figure 9: Human Evaluations: Survey Preview

▾ Instructions (Read carefully)

Your task is to evaluate an AI assistant's **Proactive Reasoning Capability** to answer a user's query.
It measures whether an AI assistant recognizes relevant constraints from the past, figures out likely consequences and anticipates future needs to provide a useful and actionable answer.

A Strong Response:

- Acknowledges important factors or constraints correctly.
- Anticipates what might happen next or what the user might need.
- Provides useful and actionable next steps.
- Uses temporally valid context (draws useful links from past experiences to guide present or future planning).

A Weak Response:

- Is generic or non-personalized.
- Reacts only to the surface wording of the query.
- Provides vague advice that is hard to execute.
- Misses clear opportunities to guide or anticipate.

You will be given:

- A **User Query**
- **Important Context:** These are facts, constraints, or needs that a good response is expected to acknowledge or consider in order to be useful.
- **Two candidate responses:** Response A and Response B
- A brief **Reasoning**, which is an explanation of what information or facts each assistant used to generate its response. Some important context can also be found explicitly in the reasoning.

Note:

- Q1 and Q2 must be answered **independently** to evaluate Response A and Response B respectively.
- 'Important Context' is expected as a **minimum requirement.**
- Responses **may include additional** helpful details **beyond** the provided minimum requirement. Use them as **tie-breakers** to compare the responses.

Figure 10: Human Evaluations: Survey Instructions