PROACTIVE RECALL FOR LLMS: TOWARDS RELIABLE AND TIMELY TASK EXECUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in LLM-based dialogue systems have substantially improved their ability in task execution. However, lacking the ability to proactively recall necessary information for pending tasks, they misjudge whether to execute: acting with fabricated values when information is insufficient or staying inactive despite sufficient information already being available in the interaction history. To address this limitation, we formalize **Proactive Task Execution (PTE)** and introduce PTE-BENCH, a benchmark for trustworthy and timely LLM task execution. PTE-BENCH encompasses 2,268 long conversations and 10,639 evaluation instances across four conversation scenarios, providing a comprehensive testbed for probing whether an LLM can proactively recall task-relevant information to decide whether to execute, and to identify what is missing when it cannot. We further propose **ARDSE**, **A**ctive **R**ecall through **D**ynamic **S**tate **E**volution, which supports proactive recall by dynamically monitoring pending task states through a Progressive Task-State Graph and using evidence-conditioned activation to determine which task to recall in multi-task conversations. Extensive experiments on four LLM backbones and five representative baselines show that ARDSE achieves the best performance, demonstrating the effectiveness of persistent task-state evolution for proactive task execution. Data is available at `https://anonymous.4open.science/r/active_recall-65ED`.

1 INTRODUCTION

Large language models (LLMs) are increasingly deployed as assistant agents to execute tasks for humans. Recent advances in memory systems (Packer et al., 2023; Zhong et al., 2024; Xu et al., 2025; Tan et al., 2025; Ong et al., 2025) and proactive systems (Lu et al., 2025; Liu et al., 2025; Dong et al., 2025) provide important foundations for LLM agents to evolve into systems that execute tasks through multi-turn interaction by leveraging past interactions and act proactively. As LLM agents take on increasingly long and interactive task sessions, their ability to recall information scattered across a conversation and judge whether it is sufficient to execute has become a critical capability, determining whether an agent acts reliably and efficiently.

Yet current LLMs still struggle with this capability. On the one hand, due to the lack of the ability to actively recall the conditions required for a task, existing LLMs often struggle to correlate historical information promptly to determine that the task has met the completion criteria and actively execute the task without the need for further interaction. As illustrated in Figure 1 (a), a vanilla LLM may remain focused on the local context and overlook a pending task even after its final requirement becomes available; in contrast, an ideal proactive-recall system should recognize it and execute the task without an explicit reminder.

On the other hand, when required information is missing, most LLMs either fabricate a plausible value and proceed, or repeatedly ask the user for information that may already be available in the conversation history, rather than actively judging whether the task can be executed and, if not, what is missing. We conduct a preliminary study on 50 STAR conversations (Mosig et al., 2020). Figure 1 (b) shows that removing the explicit task statement from the query reduces average task completion by 17.0% and information coverage by 16.7%, indicating weak proactive recall capabilities: models capable of executing explicitly specified tasks often fail to recall them without a reminder. Figure 1 (c) shows that models correctly recognize only 7.0% of incomplete task states, while over-

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.

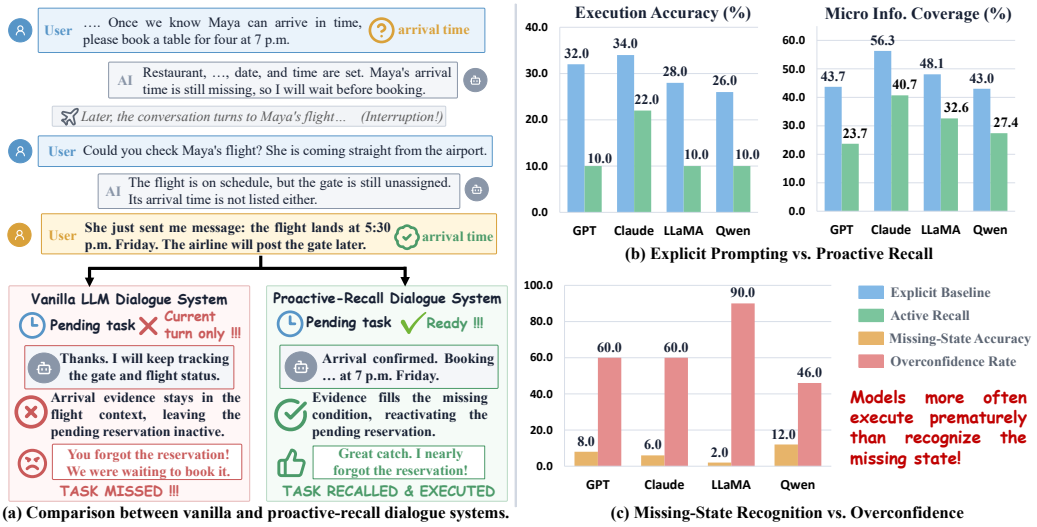

Figure 1: The left shows the comparison between vanilla dialogue systems and proactive-recall dialogue systems. The right presents the results of preliminary studies.

confidently executing 64.0% of incomplete tasks. These results highlight the limited ability of LLMs to identify missing information; instead, they fabricate plausible values and execute prematurely. As research in cognitive psychology suggests, the intention to complete a task persists in memory but requires relevant information to be reactivated proactively (Patalano & Seifert, 1997; Rummel & Kvavilashvili, 2023). LLM agents face a structurally similar challenge: a pending task should be advanced once the conversation provides sufficient information, yet the relevant information may already be present in the history while the LLM lacks the ability to proactively recall it.

Therefore, in this paper, we introduce the concept of *Proactive Recall* and propose PTE-BENCH for **Proactive Task Execution (PTE)**, which provides a comprehensive and objective benchmark for measuring and advancing the capability of trustworthy and timely task execution in LLMs. PTE-BENCH, constructed from SGD (Rastogi et al., 2020) and MultiWOZ (Zang et al., 2020), transforms seed conversations into task-state trajectories that record information acquisition, revision, and supersession. We interleave coherent off-task exchanges while preserving the underlying task state and further construct cross-session and conditional scenarios. For each conversation, we identify the first turn at which all gold values of required information become available to evaluate proactive task execution, while earlier turns evaluate incomplete-state recognition and missing-information identification. It contains 2,268 conversations and 10,639 evaluation instances.

To realize proactive recall in task execution, a system must overcome three challenges: inferring when to recall without an explicit reminder, tracking task states that evolve as information is added or revised, and coupling recall with execution so that the outcome of recall directly determines whether to act. However, existing dialogue agents primarily retain and retrieve interaction context rather than actively recalling pending tasks. They maintain no task state and leave the decision to act until the user raises it. Addressing these requires a system that continuously maintains task states, activates them at the right moment, and verifies their completeness before acting.

Therefore, we propose **ARDSE**, **A**ctive **R**ecall through **D**ynamic **S**tate **E**volution, a framework that supports proactive recall through dynamic task-state monitoring. To track pending tasks across task transitions and information changes, we design a Progressive Task-State Graph to connect each task with its target operation, required information, and evidence. Progressive updates allow the graph to track each task's current state separately as the conversation evolves. Furthermore, for conversations involving multiple tasks, we introduce evidence-conditioned activation that uses the conversational context and the evolution of these task states to determine which pending task to recall. Together, the progressive graph and activation mechanism allow later evidence to recall an earlier task without an explicit reminder, and the task state determines execution or identifying missing information. Experiments with four LLMs and five representative baselines show that ARDSE achieves the strongest

overall performance on PTE-BENCH, demonstrating the effectiveness of dynamic task-state monitoring and evidence-conditioned activation for proactive task execution.

Overall, our contributions are summarized as follows:

- We are the first to formalize **Proactive Task Execution (PTE)**, systematically exploring whether LLMs have the ability to proactively recall task-related information to achieve more reliable and timely LLM task responses.
- We propose **PTE-BENCH**, which consists of 2,268 conversations and 10,639 evaluation instances. PTE-BENCH's Active Recall and Task State Recognition tests jointly evaluate autonomous task execution, required-information usage, and missing-state identification, providing a comprehensive testbed for evaluating the proactive execution ability of LLMs.
- We propose **ARDSE**, a novel framework for proactive task execution that dynamically monitors pending task states through a Progressive Task-State Graph and uses evidence-conditioned activation to determine which task to recall. Experiments with four LLMs and five representative baselines demonstrate its effectiveness on PTE-BENCH.

2 RELATED WORK

Long-term memory enables LLM-based conversational systems to organize, retrieve, update, and act on information from historical interactions (Packer et al., 2023; Zhong et al., 2024; Xu et al., 2025; Tan et al., 2025; Ong et al., 2025), with recent approaches further exploring structured storage and memory management (Chhikara et al., 2025; Kang et al., 2025; Yan et al., 2026). Besides, benchmarks such as LoCoMo (Maharana et al., 2024), LongMemEval (Wu et al., 2025b), and LOCCO (Jia et al., 2025) evaluate information retrieval and reasoning over long-term conversations. Furthermore, proactive dialogue systems investigate user-need anticipation, intervention timing, and goal-directed guidance (Lu et al., 2025; Liu et al., 2025; Dong et al., 2025; Liu et al., 2026), including topic shifts informed by conversational memory (Wu et al., 2025a). Moreover, task-oriented systems track evolving user goals to guide responses and actions (Wu et al., 2019; Andreas et al., 2020; Zhao et al., 2023; Xu et al., 2024), while ATOD extends evaluation to interleaved goals, dependencies, and asynchronous execution (Zhang et al., 2026). Meanwhile, research on prospective memory examines how delayed intentions are retained and recalled when relevant circumstances arise for humans (Patalano & Seifert, 1997; Rummel & Kvavilashvili, 2023). Complementing these directions, we investigate proactive recall of pending tasks when information becomes sufficient and identification of missing information when it remains insufficient.

3 PRELIMINARY STUDY

We conduct a preliminary study on 50 conversations from STAR (Mosig et al., 2020) to examine whether LLMs can actively recall pending tasks and show the results in Figure 1 (b) and (c). Each conversation contains a target task followed by intervening tasks. We construct three settings: **Explicit Baseline**, which restates the target task in the query; **Active Recall**, which removes this explicit statement; and **Missing-State**, which retains the statement but removes one required information item. We evaluate GPT-4o Mini, Claude Sonnet 4.6, Llama-3.1-8B-Instruct, and Qwen3-14B. Explicit Baseline and Active Recall require both the correct operation and complete required information, while Micro Information Coverage measures information use. Missing-State requires identifying all missing information without execution, and Overconfidence Rate measures premature execution. Prompts are detailed in the Tables 9 and 10. Appendix A presents additional details.

Current LLMs struggle to recall pending tasks proactively. Removing the explicit task statement reduces average execution accuracy by 17.0% and information coverage by 16.7%. The decline reflects a failure of proactive recall rather than task execution itself. For example, a conversation shifts from a restaurant reservation to a banking task, and GPT-4o Mini correctly completes the reservation when explicitly prompted. Without this prompt, however, it remains focused on banking even when the final turn provides the remaining information required to complete the reservation. Similar failures involve following the most recent task, producing generic replies, or asking the user to restate the task, indicating that retaining task-relevant information does not ensure timely reactivation of a pending task. **Current LLMs cannot reliably verify whether a task is executable.** Only 7.0%

Table 1: Statistics of PTE-BENCH and comparison with related benchmarks on their test sets.

Subset	Dialogues	Avg. Turns	Avg. Req. Info	Active Recall	State Recognition	Total
SGD	1,710	61.7	6.97	1,710	6,348	8,058
MultiWOZ	118	74.9	6.75	118	602	720
Cross-Session	300	285.6	5.80	300	1,001	1,301
If–Then	140	236.8	6.53	140	420	560
Total	**2,268**	**102.8**	**6.78**	**2,268**	**8,371**	**10,639**

Dataset	Conv./ Scenarios	Eval. Items	Cross-Session	Info. Evolution	Proactivity Eval.	Deferred Execution	Missing-Set Eval.
MultiWOZ 2.2 (Zang et al., 2020)	1,000	1,000	✗	✓	✗	✗	✗
SGD (Rastogi et al., 2020)	4,201	4,201	✗	✓	✗	✗	✗
LoCoMo (Maharana et al., 2024)	10	1,986	✓	✓	✗	✗	✗
LongMemEval (Wu et al., 2025b)	500	500	✓	✓	✗	✗	✗
Mem2ActBench (Shen et al., 2026)	2,029	400	✓	✓	✗	✗	✗
ProactiveBench (Lu et al., 2025)	12	233	–	✗	✓	✗	✗
PM-Bench (Liu & Gabriel, 2026)	1	81	–	✓	✓	✓	✗
PTE-BENCH (OURS)	**2,268**	**10,639**	✓	✓	✓	✓	✓

of Missing-State cases are handled correctly on average, while 64.0% result in overconfident execution. Meanwhile, none of these overconfident executions produce the correct response. Models therefore often treat an explicit task request as sufficient for execution without validating its state. For instance, after the party size is removed from a restaurant-reservation case, Llama still claims that the reservation at Cactus Club for 5 pm is confirmed. Failures involve inventing missing values, using information from intervening tasks, or executing with incomplete states. These results show that proactive recall requires not only task execution capability, but also identifying missing information and executing only when its state is complete.

4 BENCHMARK CONSTRUCTION

This section introduces our PTE-BENCH for **Proactive Task Execution (PTE)**. A task-related conversation D_i contains a latent task $\tau_i = (o_i, R_i)$, where o_i is the target operation and $R_i = \{(r_{ij}, v^*_{ij})\}_{j=1}^{K_i}$ denotes the required information field-value pairs. Let t^*_i be the first turn at which R_i is complete. Earlier evaluation instances constitute the **Task State Recognition Test** and are labeled with missing information $\mathcal{M}_{i,t} \subseteq R_i$; the instance at t^*_i constitutes the **Active Recall Test** and is labeled with the gold operation and complete state. Prompts are detailed in Tables 11-18.

Task-oriented data construction. Seed conversations are derived from SGD (Rastogi et al., 2020) and MultiWOZ 2.2 (Zang et al., 2020). We select seven SGD services spanning restaurant, hotel, flight, rental-car, and calendar domains, and retain the longest MultiWOZ conversations with at least two required information fields, selecting one main task from each. We then identify turns that introduce, confirm, revise, or supersede task-relevant information and group information from the same interaction into the same stage. An LLM rewrites each stage as a natural user-assistant conversation while preserving the original information. In the conversation, the user primarily supplies information. To construct long-context conversations, an LLM inserts a topic-guided off-task exchange and a natural transition back to the task between adjacent stages. Each contains at least 20 turns and scales with the adjacent-stage length. It preserves coherence and speaker alternation, and neither reveals future required information, advances or executes the operation, nor changes the task state.

Cross-session and conditional subsets. The **Cross-Session** subset distributes a validated trajectory with at least four task stages and four required information fields across distinct task conversations, yielding 300 conversations over seven tasks. The **If-Then** subset extends beyond slot filling through seven conditional-operation scenarios with 20 variants each. In each variant, task arguments are distributed across the conversation, while the execution condition remains unresolved until the final instance. The pathway, information allocation, speaker roles, and surface realization vary while preserving the condition, yielding 140 conversations in total.

Annotation and validation. Each required information field is aligned with the stage establishing its gold value, and the earliest executable stage is identified. GPT-4o mini performs two-pass validation, requiring each conversation to cover all fields while remaining incomplete until its final task stage. It rewrites the final stage when evidence is missing and consolidates final stages when executability occurs prematurely. After expansion, we annotate the minimal supporting utterances for each required information item, excluding tentative, superseded, and redundant mentions, and three NLP researchers inspect random samples for annotation quality.

Statistics and evaluation. Table 1 summarizes the benchmark and compares it with related benchmarks. It contains 2,268 conversations and 10,639 evaluation instances: 2,268 for active recall and 8,371 for task state recognition. We define Active Recall Accuracy **(ARAcc)**, Required Information Coverage **(RIC)**, State Recognition Accuracy **(StateAcc)**, and Missing Information Coverage **(MIC)**. Let $\hat{o}_i$ denote the predicted operation, $\widehat{R}_i$ the correctly used required information, $\widehat{\mathcal{M}}_i$ the predicted missing set, and N_{AR} and N_{SR} the respective instance counts. The metrics are defined as $\mathrm{ARAcc} = \frac{1}{N_{\mathrm{AR}}} \sum_i \mathbb{I}[\hat{o}_i = o_i^* \wedge \widehat{R}_i = R_i]$, $\mathrm{RIC} = \frac{\sum_i |\widehat{R}_i|}{\sum_i |R_i|}$, $\mathrm{StateAcc} = \frac{1}{N_{\mathrm{SR}}} \sum_i \mathbb{I}[\hat{o}_i = \text{incomplete} \wedge \widehat{\mathcal{M}}_i = \mathcal{M}_i]$, and $\mathrm{MIC} = \frac{\sum_i |\widehat{\mathcal{M}}_i \cap \mathcal{M}_i|}{\sum_i |\mathcal{M}_i|}$. Appendix C and D present additional details.

5 ARDSE: Active Recall through Dynamic State Evolution

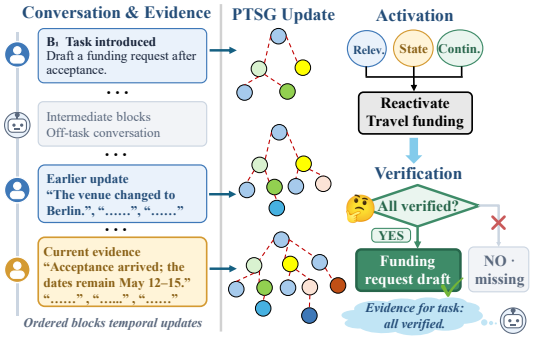

Figure 2: The overall framework of ARDSE.

We propose **ARDSE**, a framework for proactive task execution that dynamically monitors pending tasks and proactively recalls tasks and their supporting evidence to determine whether execution is possible. The overall framework of our **ARDSE** is illustrated in Figure 2. Each task is represented as $\tau_i = (o_i, R_i)$, where o_i is the target operation and $R_i = \{r_{i1}, \ldots, r_{iK_i}\}$ contains the K_i required-information items needed for execution. Given a conversation prefix $C_{1:t} = (x_1, \ldots, x_t)$, ARDSE monitors the task states, integrates new evidence into states, selects the task made relevant by recent updates, and verifies information sufficiency before execution. Prompts are detailed in Tables 19-22.

Progressive task-state graph. ARDSE represents pending tasks using a Progressive Task-State Graph (PTSG), denoted as $G_b = (V_b, E_b)$ at processing step b, where $V_b = V_b^\tau \cup V_b^o \cup V_b^R \cup V_b^e$ comprises task, operation, required-information, and conversational-evidence nodes. The edges encode their semantic, evidential, and temporal relations. For example, REQUIRES connects an operation to its required information, SUPPORTS grounds a value in conversational evidence, UPDATES records subsequent revisions, and BELONGS-TO associates graph elements with the corresponding task. The state of an item r_{ij} is represented as $q_{ij}^{(b)} = (r_{ij}, v_{ij}^{(b)}, s_{ij}^{(b)}, \mathcal{E}_{ij}^{(b)})$, where $v_{ij}^{(b)}$ is its current value, $\mathcal{E}_{ij}^{(b)}$ is its supporting evidence, and $s_{ij}^{(b)} \in \{\text{filled}, \text{missing}, \text{uncertain}, \text{conflicting}\}$ is its status. This records the source of each value and prevents evidence from different tasks from being conflated.

Progressive state update. ARDSE processes the conversation chronologically in blocks $B_1, \ldots, B_L$, where $B_b = C_{t_{b-1}+1:t_b}$, and updates the graph through

$$\Delta_b = f_{\mathrm{upd}}(B_b, G_{b-1}), \qquad G_b = \mathcal{U}(G_{b-1}, \Delta_b). \tag{1}$$

The updater f_{upd} extracts newly introduced tasks, required-information values, revisions, cancellations, conflicts, and supporting evidence, while $\mathcal{U}$ incorporates them into the inherited graph. An item is marked as filled only when evidence explicitly supports its value for the corresponding task. Repeated evidence strengthens its evidence set, an explicit correction supersedes the previous value, and contradictory or ambiguous evidence changes its status accordingly. When a block contains no relevant evidence, the previous state is retained. The updater may also introduce a new task or expand R_i when additional execution requirements emerge, allowing the graph to evolve without discarding unresolved tasks across topic transitions.

Evidence-conditioned task activation. Let $\mathcal{T}_b$ be the pending tasks in G_b. For each $\tau_i \in \mathcal{T}_b$, ARDSE computes semantic relevance $\rho_i^{(b)} = \cos(\phi(C_b^{\text{rec}}), \phi(D_i^{(b)}))$, recent state change $\delta_i^{(b)} = |\Delta R_i^{(b)}|/|R_i|$, and task continuity $\kappa_i^{(b)} = 1/(1 + t_b - \ell_i)$. Here, ϕ is a semantic encoder, C_b^{rec} is the recent conversational context, $D_i^{(b)}$ describes the operation and current state of τ_i, $\Delta R_i^{(b)}$ contains the items receiving new evidence in B_b, and ℓ_i is the most recent turn associated with the task. Each signal $x \in \{\rho, \delta, \kappa\}$ is min–max normalized across $\mathcal{T}_b$, and the activation score is

$$\mathcal{N}_b(x_i^{(b)}) = \frac{x_i^{(b)} - x_{\min}^{(b)}}{x_{\max}^{(b)} - x_{\min}^{(b)}}, \qquad A_b(\tau_i) = \frac{\mathcal{N}_b(\rho_i^{(b)}) + \mathcal{N}_b(\delta_i^{(b)}) + \mathcal{N}_b(\kappa_i^{(b)})}{3}. \tag{2}$$

We select the task with the highest activation score as τ_b^*. This mechanism allows later evidence to reactivate an earlier task without requiring its operation to be explicitly restated in the current turn.

Operation-conditioned state verification. Let the activated task be $\tau_b^* = (o_b^*, R_b^*)$. Before execution, ARDSE refines its required-information set as $\overline{R}_b = \Gamma(o_b^*, R_b^*, G_b[\tau_b^*])$, where Γ decomposes overly broad requirements and supplements requirements omitted during progressive updating. Each $r \in \overline{R}_b$ is then realigned with evidence from $C_{1:t_b}$, producing a verified state $\widehat{S}_b = \{\widehat{q}_r^{(b)} \mid r \in \overline{R}_b\}$. A value is verified as filled only when it is explicitly supported and correctly associated with τ_b^*; ambiguous references remain uncertain, conflicting values remain conflicting, and later corrections supersede earlier values. ARDSE defines the missing set $M_b = \{r \in \overline{R}_b \mid \widehat{s}_r^{(b)} \neq \text{filled}\}$ and deems the task executable when $z_b = \mathbb{I}[M_b = \varnothing] = 1$.

State-conditioned execution. Given the complete conversation, task τ_b^*, state $\widehat{S}_b$, and missing set M_b, ARDSE generates a response conditioned on information sufficiency. It executes o_b^* using verified values when $z_b = 1$; otherwise, it withholds execution and identifies the missing information.

6 Experiments

6.1 Experimental Settings

Models and baselines. We evaluate our method against five baselines on PTE-Bench. **Full-Context Prompting** directly inputs the complete conversation to LLMs. **A-MEM** (Xu et al., 2025) provides a memory-based baseline that constructs, evolves, and retrieves memory notes. We also include two proactive-dialogue baselines: **Proactive Agent**, which predicts a potential task before response generation (Lu et al., 2025), and **Inner Thoughts**, which forms and evaluates latent thoughts before deciding whether to participate (Liu et al., 2025). Finally, **AutoTOD** represents autonomous task-oriented dialogue through instruction-guided reasoning (Xu et al., 2024). We evaluate four backbone LLMs: GPT-4o Mini, Claude Sonnet 4.6, Llama-3.1-8B-Instruct, and Qwen3-14B.

Input settings. We evaluate each method under two settings. In **With Task**, the target operation and required-information schema are specified once at the beginning of the conversation. In **Without Task**, only the conversation is provided, requiring the model to recover the latent task and reconstruct its state. The task is never repeated at the evaluation instance, and the method-specific memory, retrieval, planning, and generation procedures remain unchanged between the two settings.

Implementation and evaluation. We set the temperature to 0 for all methods. Tables 23-26 present the baseline prompts. ARDSE processes conversations in 12-turn blocks and computes task activation from the 6 most recent turns. GPT-4o Mini serves as an auxiliary LLM judge, assessing operation correctness, required-information use, and semantic alignment with the gold missing-information set. Table 27 presents the evaluation prompt. We further manually review to validate evaluation quality. We report ARAcc as the main metric for Active Recall Test, and StateAcc as the main metric for the Task State Recognition Test. Appendix E presents additional details.

6.2 Main Results

Main results are shown in Table 2. Our **ARDSE** yields consistent improvements across all backbones, outperforming the baseline models in 26 of 32 comparisons. More importantly, our ARDSE performs consistently better than all baselines across all backbones in the accuracy comparisons. Under the With Task setting, it improves average ARAcc by 9.04% and average StateAcc by 17.77%

Table 2: Main results on PTE-BENCH. The best and second-best results for each backbone and input setting are shown in bold and underlined, respectively.

		With Task				Without Task			
Backbone	Method	ARAcc	RIC	StateAcc	MIC	ARAcc	RIC	StateAcc	MIC
GPT-4o Mini	Full-Context	39.68	69.45	37.46	**79.98**	4.63	16.35	9.43	19.15
	A-MEM	16.53	58.47	34.31	47.08	8.42	42.39	7.99	14.77
	Proactive Agent	13.45	27.34	29.41	46.86	7.41	23.77	7.31	19.28
	Inner Thoughts	5.34	24.08	18.22	34.13	4.76	23.37	5.11	13.76
	AutoTOD	13.27	25.57	27.46	43.94	7.76	18.72	10.33	**21.08**
	ARDSE (ours)	**61.11**	**84.66**	**56.52**	79.07	**21.43**	**57.73**	**10.92**	18.74
Claude Sonnet 4.6	Full-Context	56.13	84.09	41.12	80.39	12.37	26.10	9.77	26.88
	A-MEM	13.89	46.30	38.79	48.19	9.88	38.80	7.35	22.21
	Proactive Agent	21.21	40.97	38.37	59.98	14.86	34.61	9.01	26.71
	Inner Thoughts	5.25	28.75	26.01	48.26	5.60	27.73	10.57	30.24
	AutoTOD	18.87	46.22	40.43	60.17	12.08	39.60	8.89	**31.29**
	ARDSE (ours)	**67.06**	**91.74**	**55.02**	**80.77**	**27.73**	**56.69**	**11.27**	22.70
Llama-3.1-8B	Full-Context	39.64	**71.17**	16.99	49.54	4.10	18.62	9.14	17.38
	A-MEM	7.98	28.06	20.86	28.57	5.47	18.63	2.20	7.86
	Proactive Agent	8.55	24.93	20.74	35.07	5.29	18.62	9.80	**19.80**
	Inner Thoughts	4.84	26.85	10.69	22.73	4.52	25.46	2.29	9.66
	AutoTOD	8.77	19.95	13.40	20.09	5.31	14.98	8.04	14.57
	ARDSE (ours)	**41.18**	70.78	**47.01**	**73.51**	**18.39**	**61.17**	**11.00**	19.41
Qwen3-14B	Full-Context	51.90	77.94	26.63	57.63	8.73	26.73	2.28	8.79
	A-MEM	8.91	24.47	22.49	31.93	6.44	22.61	2.15	5.20
	Proactive Agent	16.62	32.90	15.47	25.85	9.89	27.06	2.01	11.89
	Inner Thoughts	5.50	21.42	18.48	37.41	5.06	20.80	2.22	15.07
	AutoTOD	16.80	31.59	27.81	41.14	10.32	25.05	3.26	**28.17**
	ARDSE (ours)	**54.14**	**78.94**	**34.72**	**60.60**	**15.12**	**52.73**	**3.74**	27.62

over the strongest corresponding baselines. Under the more challenging Without Task setting, the improvements are 11.31% in ARAcc and 1.57% in StateAcc. These improvements demonstrate consistent gains in both executing pending tasks when sufficient information is available and identifying missing information without premature execution when it is not. ARDSE supports these complementary decisions by associating distributed conversational evidence with dynamic task states and verifying whether their execution requirements are satisfied. The results highlight the role of proactive recall in PTE: recovering a task and its supporting information enables the system to assess execution readiness and act reliably without restating the task.

Existing baselines are weak in recalling pending tasks. Full-Context Prompting retains the complete conversation but suffers an average ARAcc decline of 39.38% when the initial task specification is removed. This result shows that available historical evidence does not ensure that a model will recall a pending intention after the local conversation has shifted to another topic. Proactive Agent and Inner Thoughts are designed to determine whether and when an intervention would be useful, but they do not persistently track the execution requirements of an unresolved task across the conversation. The interaction policy of AutoTOD is designed to advance a locally active task, whereas PTE requires proactively recalling a task and determining its executability after long off-task interaction. Consequently, these methods can respond appropriately to the current conversational context while still failing to resume a pending task once it becomes executable.

Task-relevant evidence does not guarantee proactive recall. A-MEM makes this distinction particularly clear. It achieves 42.39% RIC without task information with GPT-4o-mini, while its corresponding ARAcc remains only 8.42%. Memory retrieval can surface an individual fact because it is semantically related to the current context, but it does not necessarily identify the unresolved task to which the fact belongs, or whether the remaining execution conditions have been fulfilled. This limitation is especially important when task-relevant evidence is provided incidentally during an unrelated topic. In such cases, relevant information is accessible but is not incorporated into

Table 3: Ablation results comparing with four ablated variants.

Backbone	Variant	With Task				Without Task			
		ARAcc	RIC	StateAcc	MIC	ARAcc	RIC	StateAcc	MIC
GPT	Full	**61.11**	**84.66**	**56.52**	**79.07**	**21.43**	**57.73**	10.92	**18.74**
	w/o Prog.	48.41	71.95	44.49	70.10	18.52	49.32	**11.07**	18.67
	w/o Graph	54.23	77.89	55.37	78.29	16.71	49.11	10.70	18.63
	w/o Act.	–	–	–	–	16.84	49.31	10.87	18.65
	w/o Evi. Veri.	53.57	77.91	34.83	56.71	17.15	49.76	9.51	14.00
Claude	Full	**67.06**	**91.74**	**55.02**	**80.77**	**27.73**	**56.69**	**11.27**	22.70
	w/o Prog.	53.44	82.81	53.00	78.58	18.43	40.75	9.95	24.58
	w/o Graph	61.29	84.73	51.82	78.23	17.59	40.81	9.22	23.55
	w/o Act.	–	–	–	–	18.39	47.25	10.13	**24.72**
	w/o Evi. Veri.	55.56	82.91	50.23	77.25	18.43	46.93	7.10	20.67
LLaMA	Full	**41.18**	**70.78**	47.01	73.51	**18.39**	**61.17**	11.00	19.41
	w/o Prog.	35.49	64.47	46.63	73.12	13.58	50.98	11.30	**19.45**
	w/o Graph	35.89	64.79	46.90	73.33	13.40	50.60	10.91	18.84
	w/o Act.	–	–	–	–	13.40	50.63	10.98	19.40
	w/o Evi. Veri.	35.85	64.56	**48.62**	**74.33**	13.76	50.94	**11.77**	18.20
Qwen	Full	**54.14**	**78.94**	34.72	60.60	**15.12**	**52.73**	**3.74**	**27.62**
	w/o Prog.	53.84	78.77	42.73	67.84	14.37	50.82	3.72	23.01
	w/o Graph	53.79	78.72	42.99	68.14	12.57	45.91	2.96	22.52
	w/o Act.	–	–	–	–	12.52	45.81	3.02	22.08
	w/o Evi. Veri.	53.70	78.54	**43.93**	**68.70**	13.62	48.52	2.68	10.12

the pending task, leading models to omit required information and execute prematurely. ARDSE addresses this gap by explicitly updating task-requirement-evidence relationships and verifying the reconstructed task state before proceeding to execution. Appendix F reports additional details.

6.3 Ablation Study

We compare Full ARDSE with four variants and report the results in Table 3. **w/o Progressive Update** processes conversation blocks independently without considering current task states; **w/o Graph** replaces the Progressive Task-State Graph with a flat information representation; **w/o Task Activation** selects the most recently mentioned task; and **w/o Evidence Verification** determines execution sufficiency directly from the current state without revalidation. w/o Task Activation is not evaluated under the With Task setting, as the target task is explicitly specified. Full ARDSE exhibits the strongest overall performance, ranking first in 24 of the 32 comparisons and among the top two in 28 of them. Notably, it achieves the highest ARAcc and RIC across all comparisons, demonstrating the complementary contributions of these modules to proactive recall for Proactive Task Execution. More detailed results are also provided in the Appendix F.

Removing progressive updates reduces average ARAcc by 8.08% with task and 4.44% without it, and RIC by 7.03% and 9.11%, respectively, showing the importance of progressive state evolution across topic transitions. Removing the task-state graph reduces ARAcc by 5.60% and RIC by 10.47% without task, indicating that active recall requires preserving structural relations between evidence, task requirements, and evolving states. Removing evidence-conditioned activation lowers average ARAcc by 5.38% and RIC by 8.83%, indicating that new evidence can make a latent task, rather than the most recent one, executable. Removing evidence verification most strongly affects missing-information coverage, decreasing average MIC by 4.24% with the task and 6.37% without it. This highlights the importance of re-validating the state to filter unsupported values and avoid premature execution. Removing a component may improve state recognition in some settings (e.g., in Qwen), but these gains come at the cost of lower ARAcc and RIC. The complete framework provides the strongest overall support for PTE across executable and incomplete task states.

Figure 3 reports the active recall performance of ARDSE across task-graph sizes, measured by the number of required-information nodes. ARAcc remains relatively stable in the common range, and RIC demonstrates consistently robust task-state tracking capability. For example, Qwen3-14B and GPT-4o mini achieve over 80% RIC at 5-6 and 9-10 nodes, respectively; moreover, even with 14 or

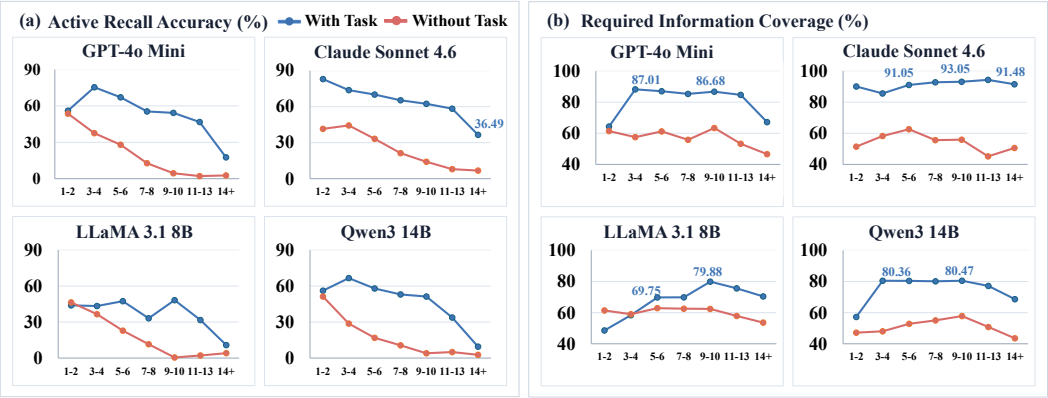

Figure 3: ARDSE performance across different task-graph sizes.

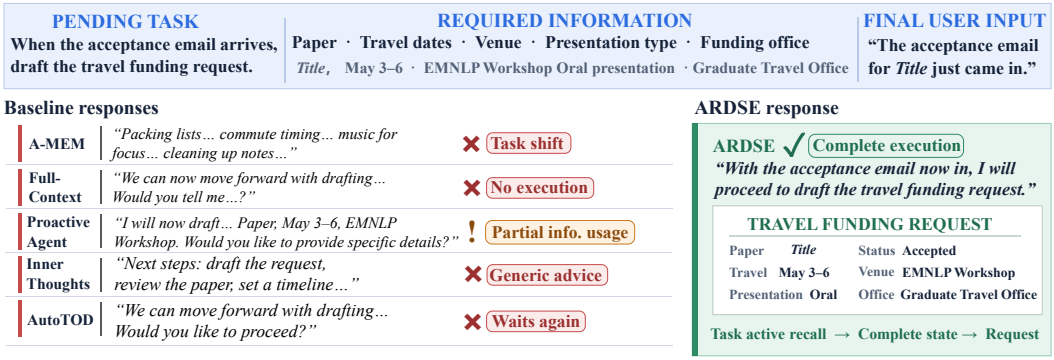

Figure 4: Case study comparing responses when the user's input makes a pending task executable.

more nodes, Claude retains 91.48% RIC. These results suggest that ARDSE preserves substantial information recall as graph size increases, although exact task execution becomes more challenging.

6.4 Case Study

Figure 4 presents a case with GPT-4o Mini, where task information is interleaved with multiple off-task discussions. The user requests a travel funding application once the paper is accepted. Upon confirming acceptance, ARDSE uses all information to draft the application correctly. In contrast, Full-Context Prompting and AutoTOD recognize task readiness but still request information, while Proactive Agent partially uses provided information. Inner Thoughts offers only generic planning suggestions, and A-MEM remains focused on intervening topics. This case illustrates that recognizing task readiness or retrieving relevant evidence alone is insufficient, and proactive recall requires integrating distributed evidence with the pending task. Appendix G presents another case study.

7 Conclusion

We are the first to formalize Proactive Task Execution (PTE) and systematically investigate whether LLMs can actively recall task-relevant information to support more reliable and timely task execution. Our experiments indicate that current LLMs exhibit weak proactive recall of suspended tasks and limited ability to recognize incomplete task states. We further introduce PTE-Bench, which evaluates both active task recall and incomplete-state recognition in long conversations. We propose ARDSE, which monitors pending tasks as dynamically evolving, evidence-grounded states, enabling proactive recall to determine whether and when to execute rather than merely retrieve historical information. Experiments across four LLM backbones and five representative baselines demonstrate the effectiveness of ARDSE for reliable and timely proactive task execution.

AI USE STATEMENT

In this work, we used generative AI tools for the following tasks: generate synthetic data sets, help develop theoretical models or conceptual frameworks, design or provide feedback on research methodology or experiments, implement methods, assist with translation, clean and reformat datasets. We have not used generative AI tools for the following tasks: support qualitative and thematic data analysis, interpret results and following tasks: formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, and propose or refine hypotheses are not applicable to this work. Additionally, we used generative AI tools for the following tasks: creating or editing software code, creating artifacts, summarizing or analysing existing literature. We have reviewed all AI-assisted work. We checked LLM-generated refinement for potential plagiarism. LLM-generated code was verified and tested for correctness. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work uses publicly available task-oriented dialogue datasets and LLM-generated conversations. It does not collect real user interactions or execute real-world operations. Three NLP researchers reviewed sampled annotations for quality, without providing personal or sensitive information. We will release the derived data with appropriate attribution and in accordance with the licenses of the source datasets. Although ARDSE is evaluated in simulated settings, proactive execution may cause unintended or premature actions in real deployments. Practical systems should therefore require appropriate user authorization, privacy safeguards, restricted tool permissions, and human oversight for consequential operations.

REPRODUCIBILITY STATEMENT

We have provided our generated benchmark in the anonymous link in the abstract and supplemental material. All prompts and implementation details are presented in the paper.

REFERENCES

Jacob Andreas, John Bufe, David Burkett, Charles Chen, Josh Clausman, Jean Crawford, Kate Crim, Jordan DeLoach, Leah Dorner, Jason Eisner, Hao Fang, Alan Guo, David Hall, Kristin Hayes, Kellie Hill, Diana Ho, Wendy Iwaszuk, Smriti Jha, Dan Klein, Jayant Krishnamurthy, Theo Lanman, Percy Liang, Christopher H. Lin, Ilya Lintsbakh, Andy McGovern, Aleksandr Nisnevich, Adam Pauls, Dmitrij Petters, Brent Read, Dan Roth, Subhro Roy, Jesse Rusak, Beth Short, Div Slomin, Ben Snyder, Stephon Striplin, Yu Su, Zachary Tellman, Sam Thomson, Andrei Vorobev, Izabela Witoszko, Jason Wolfe, Abby Wray, Yuchen Zhang, and Alexander Zotov. Task-oriented dialogue as dataflow synthesis. *Transactions of the Association for Computational Linguistics*, 8:556–571, 2020. doi: 10.1162/tacl_a_00333. URL https://aclanthology.org/2020.tacl-1.36/.

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.

Wenjie Dong, Sirong Chen, and Yan Yang. ProTOD: Proactive task-oriented dialogue system based on large language model. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert (eds.), *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 9147–9164, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.614/.

Zixi Jia, Qinghua Liu, Hexiao Li, Yuyan Chen, and Jiqiang Liu. Evaluating the long-term memory of large language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics:*

ACL 2025, pp. 19759–19777, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1014. URL https://aclanthology.org/2025.findings-acl.1014/.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory OS of AI agent. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 25961–25970, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1318. URL https://aclanthology.org/2025.emnlp-main.1318/.

Genglin Liu and Saadia Gabriel. Pm-bench: Evaluating prospective memory of llm agents. In *Third Conference on Language Modeling*, 2026.

Tianjian Liu, Fanqi Wan, Jiajian Guo, and Xiaojun Quan. ProactiveEval: A unified evaluation framework for proactive dialogue agents. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 41068–41100, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1906. URL https://aclanthology.org/2026.acl-long.1906/.

Xingyu Bruce Liu, Shitao Fang, Weiyan Shi, Chien-Sheng Wu, Takeo Igarashi, and Xiang 'Anthony' Chen. Proactive conversational agents with inner thoughts. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, CHI '25, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400713941. doi: 10.1145/3706598.3713760. URL https://doi.org/10.1145/3706598.3713760.

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 *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=sRIU6k2TcU.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.747. URL https://aclanthology.org/2024.acl-long.747/.

Johannes E. M. Mosig, Shikib Mehri, and Thomas Kober. Star: A schema-guided dialog dataset for transfer learning, 2020. URL https://arxiv.org/abs/2010.11853.

Kai Tzu-iunn Ong, Namyoung Kim, Minju Gwak, Hyungjoo Chae, Taeyoon Kwon, Yohan Jo, Seung-won Hwang, Dongha Lee, and Jinyoung Yeo. Towards lifelong dialogue agents via timeline-based memory management. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8631–8661, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.435. URL https://aclanthology.org/2025.naacl-long.435/.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Andrea L. Patalano and Colleen M. Seifert. Opportunistic planning: Being reminded of pending goals. *Cognitive Psychology*, 34(1):1–36, 1997. doi: 10.1006/cogp.1997.0655. URL https://doi.org/10.1006/cogp.1997.0655.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8689–8696, 2020.

Jan Rummel and Lia Kvavilashvili. Current theories of prospective memory and new directions for theory development. *Nature Reviews Psychology*, 2(1):40–54, 2023. ISSN 2731-0574. doi: 10.1038/s44159-022-00121-4. URL https://doi.org/10.1038/s44159-022-00121-4.

Yiting Shen, Kun Li, Wei Zhou, and Songlin Hu. Mem2ActBench: A benchmark for evaluating long-term memory utilization in task-oriented autonomous agents. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8173–8190, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.370. URL https://aclanthology.org/2026.acl-long.370/.

Zhen Tan, Jun Yan, I-Hung Hsu, Rujun Han, Zifeng Wang, Long Le, Yiwen Song, Yanfei Chen, Hamid Palangi, George Lee, Anand Rajan Iyer, Tianlong Chen, Huan Liu, Chen-Yu Lee, and Tomas Pfister. In prospect and retrospect: Reflective memory management for long-term personalized dialogue agents. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8416–8439, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.413. URL https://aclanthology.org/2025.acl-long.413/.

Bowen Wu, Wenqing Wang, Haoran Li, Yunhan Deng, Ying Li, Jingsong Yu, and Baoxun Wang. Interpersonal memory matters: A new task for proactive dialogue utilizing conversational history. In Gemma Boleda and Michael Roth (eds.), *Proceedings of the 29th Conference on Computational Natural Language Learning*, pp. 47–67, Vienna, Austria, July 2025a. Association for Computational Linguistics. ISBN 979-8-89176-271-8. doi: 10.18653/v1/2025.conll-1.4. URL https://aclanthology.org/2025.conll-1.4/.

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. Transferable multi-domain state generator for task-oriented dialogue systems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 808–819, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1078. URL https://aclanthology.org/P19-1078/.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 86809–86836, 2025b. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/d813d324dbf0598bbdc9c8e79740ed01-Paper-Conference.pdf.

Heng-Da Xu, Xian-Ling Mao, Puhai Yang, Fanshu Sun, and Heyan Huang. Rethinking task-oriented dialogue systems: From complex modularity to zero-shot autonomous agent. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2748–2763, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.152. URL https://aclanthology.org/2024.acl-long.152/.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for LLM agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=FiM0M8gcct.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schuetze, Volker Tresp, and Yunpu Ma. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12805–12825, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.583. URL https://aclanthology.org/2026.acl-long.583/.

Xiaoxue Zang, Abhinav Rastogi, Srinivas Sunkara, Raghav Gupta, Jianguo Zhang, and Jindong Chen. Multiwoz 2.2: A dialogue dataset with additional annotation corrections and state tracking baselines. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI, ACL 2020*, pp. 109–117, 2020.

Yifei Zhang, Hooshang Nayyeri, Rinat Khaziev, Emine Yilmaz, Gokhan Tur, Dilek Hakkani-Tür, and Hari P Thadakamalla. ATOD: An evaluation framework and benchmark for agentic task-oriented dialogue systems. In *ICLR 2026 Workshop on Memory for LLM-Based Agentic Systems*, 2026. URL https://openreview.net/forum?id=1L7cY1x2zp.

Zeyu Zhang, Quanyu Dai, Rui Li, Xiaohe Bo, Xu Chen, and Zhenhua Dong. Learn to memorize: Optimizing llm-based agents with adaptive memory framework. *arXiv preprint arXiv:2508.16629*, 2025.

Jeffrey Zhao, Yuan Cao, Raghav Gupta, Harrison Lee, Abhinav Rastogi, Mingqiu Wang, Hagen Soltau, Izhak Shafran, and Yonghui Wu. AnyTOD: A programmable task-oriented dialog system. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 16189–16204, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.1006. URL https://aclanthology.org/2023.emnlp-main.1006/.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 19724–19731, 2024.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Bryan Kian Hsiang Low, and Paul Pu Liang. MEM1: Learning to synergize memory and reasoning for efficient long-horizon agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=XY8AaxDSLb.

A Preliminary Study Details

The preliminary study controls the availability of explicit task cues while holding the underlying conversational context fixed. We select 50 complete conversations from STAR that cover 13 target APIs. Each conversation contains a recurring target operation separated by at least one distinct operation. Each evaluation point occurs between the 21st and 43rd turn of the conversation, with 10 to 34 off-task turns separating it from the earlier mention of the target task. When necessary, the final user utterance is minimally rewritten to remove an explicit reference to the target operation while preserving its original semantics.

From each conversation, we construct three input settings, as summarized in Table 4. **Explicit Baseline** appends a standardized request to resume the earlier target task, whereas **Active Recall** uses the same conversational context without this explicit request. **Missing-State** removes one required information value with its corresponding turns that provide it, and then appends the same explicit task request. The operation itself is never removed, and no explicit mask token is introduced. This controlled construction varies task-cue availability or task-state completeness, enabling comparisons of proactive recall and incomplete-state recognition.

For Explicit Baseline and Active Recall, a response is considered correct only if it performs the target operation and explicitly states or unambiguously applies all gold required information. For Missing-State, responses are categorized as *correct clarification*, *overconfident execution*, or *other*. A response is considered a correct clarification only if it refrains from executing the task and correctly identifies the missing information. Any response that claims, promises, or reports task execution despite the incomplete state is classified as overconfident execution.

Table 4: Construction of the three preliminary-study settings from the same underlying conversation.

Setting	Conversation	Final input	Intended observation
Explicit Baseline	Complete conversation	Explicit request to resume the target task	Task execution under explicit reinstatement
Active Recall	Complete conversation	No explicit request to resume the target task	Active recall of the pending task
Missing-State	Turns providing one required-information item removed	Explicit request to resume the target task	Missing-information recognition without premature execution

B Additional Related Work

Beyond predefined memory-management rules, Learn to Memorize (Zhang et al., 2025) introduces learnable retrieval and memory aggregation together with task-specific reflection, optimizing memory storage and utilization within an interaction cycle. Memory-R1 (Yan et al., 2026) separates memory management from answer generation: a memory manager learns when to add, update, delete, or preserve entries, while an answer agent selects relevant memories and reasons over them. MEM1 (Zhou et al., 2026) instead integrates memory consolidation into the agent's reasoning process. At each interaction step, it combines previous memory with new observations into a compact internal state, retaining useful information while discarding redundant content. These methods explore distinct ways to learn memory behavior, ranging from explicit operations over external records to the joint optimization of reasoning and internal-state consolidation.

Existing proactive-dialogue benchmarks also provide insights. MapDia (Wu et al., 2025a) introduces the Chinese memory-aware dialogue dataset ChMapData and a framework comprising historical-topic summarization, topic retrieval, and proactive topic-shift detection and generation. This formulation explicitly connects the selection of historical topics with decisions about when to introduce them. ProactiveEval (Liu et al., 2026) separates target planning from dialogue guidance, distinguishing the selection of conversational objectives from the ability to steer an interaction toward them across multiple domains. ATOD (Zhang et al., 2026) provides a synthetic dialogue-generation pipeline and an evaluation framework, ATOD-Eval, that assesses task completion, agentic capabilities, and response quality through offline and online evaluation protocols. These evaluation designs examine different components of proactive behavior rather than treating conversational success as a single undifferentiated outcome.

C Benchmark Construction Algorithm and Data Format

Algorithm 1 formalizes the construction pipeline. For each seed conversation d_i, the pipeline builds upon a task $\tau_i = (o_i, R_i)$, where o_i is the target operation and $R_i = \{(r_{ij}, v^*_{ij})\}_{j=1}^{K_i}$ is its required-information set. It then derives a temporally ordered task trajectory, validates the first executable point, inserts state-preserving off-task exchanges, and produces one executable instance together with earlier incomplete-state instances. Cross-session and conditional variants are generated only for eligible tasks and are subjected to the same validation and annotation procedure. For If-Then tasks, the triggering condition is included as a required information item, such as the fact that a paper has been accepted. This item is marked as filled only when the conversation establishes that the condition holds. All trajectories in this subset eventually satisfy their triggering conditions. PTE-Bench is used exclusively for evaluation. All instances in the benchmark belong to the test set, and no training or development split is created. We do not use these data for model training or fine-tuning.

At each evaluation instance, a required information item remains missing until the conversation update unambiguously confirms its value for the target operation. We preserve the original decision-making structure of the SGD and MultiWOZ seed data: candidate mentions, comparisons between alternatives, and tentative values remain in the conversation but do not fulfill the requirements. For example, several restaurants may be discussed before the user selects one for the reservation. Information revision in these trajectories therefore refines tentative information or resolves alternatives,

Algorithm 1 Construction of PTE-Bench evaluation instances

Require: Seed conversations $\mathcal{D}^{\text{seed}}$, service schemas Σ
Ensure: Active Recall instances $\mathcal{B}_{\text{AR}}$ and Task State Recognition instances $\mathcal{B}_{\text{SR}}$

$\mathcal{B}_{\text{AR}} \leftarrow \varnothing$; $\mathcal{B}_{\text{SR}} \leftarrow \varnothing$
for all $d_i \in \mathcal{D}^{\text{seed}}$ **do**
  $\tau_i = (o_i, R_i) \leftarrow \textsc{ExtractTask}(d_i, \Sigma)$, where $R_i = \{(r_{ij}, v^*_{ij})\}_{j=1}^{K_i}$
  $H_i = (h_{i1}, \ldots, h_{iJ_i}) \leftarrow \textsc{SegmentAndRewrite}(d_i, \tau_i)$
  $a_i(r) \leftarrow \textsc{AlignFinalValue}(r, v^*_i(r), H_i)$ for every $r \in R_i$
  $S_{i,m} \leftarrow \{(r, v^*_i(r)) \in R_i \mid a_i(r) \le m\}$ for $m = 1, \ldots, J_i$
  $m^*_i \leftarrow \min\{m \mid S_{i,m} = R_i\}$
  if $m^*_i \neq J_i$ or any $a_i(r)$ is undefined **then**
    $(H_i, a_i, m^*_i) \leftarrow \textsc{RepairTrajectory}(H_i, \tau_i)$
  end if
  if $m^*_i \neq J_i$ or the repaired trajectory is inconsistent **then**
    continue
  end if
  for $m = 1$ to $J_i - 1$ **do**
    $L_{i,m} \leftarrow \max(20, \lceil 5\bar{\ell}_i \rceil)$
    $z_{i,m} \leftarrow \textsc{GenerateOffTaskBridge}(h_{i,m}, h_{i,m+1}, L_{i,m})$
      ▷ $z_{i,m}$ must not reveal future values, execute o_i, or change $S_{i,m}$
  end for
  $C_i \leftarrow h_{i1} \oplus z_{i,1} \oplus \cdots \oplus z_{i,J_i-1} \oplus h_{iJ_i}$
  $\mathcal{C}_i \leftarrow \{C_i\} \cup \textsc{ComposeEligibleVariants}(C_i, \textsc{CrossSession}, \textsc{Conditional})$
  for all $C \in \mathcal{C}_i$ **do**
    $\mathcal{E}_{ij} \leftarrow \textsc{AnnotateMinimalEvidence}(C, r_{ij}, v^*_{ij})$ for every $(r_{ij}, v^*_{ij}) \in R_i$
    $\mathcal{B}_{\text{AR}} \leftarrow \mathcal{B}_{\text{AR}} \cup \{(C_{1:m^*_i}, o_i, R_i, \{\mathcal{E}_{ij}\})\}$
    for $m = 1$ to $m^*_i - 1$ **do**
      $\mathcal{M}_{i,m} \leftarrow R_i \setminus S_{i,m}$
      $\mathcal{B}_{\text{SR}} \leftarrow \mathcal{B}_{\text{SR}} \cup \{(C_{1:m}, o_i, R_i, \mathcal{M}_{i,m}, \{\mathcal{E}_{ij}\})\}$
    end for
  end for
end for
return $(\mathcal{B}_{\text{AR}}, \mathcal{B}_{\text{SR}})$

rather than replacing an already confirmed required information value, such as changing a confirmed Monday to Friday. To this end, missing-information labels reflect unresolved requirements at the current evaluation point, rather than retrospectively treating earlier valid values as missing because they differ from later ones.

Table 5 gives an abridged example of one MultiWOZ-derived conversation. It retains the operation, gold required information, task stages, acquisition mapping, first executable stage, and minimal evidence while omitting repetitive off-task turns. The released JSONL records retain every turn and all provenance metadata.

D Annotation and Human Verification

As shown in Figure 5, we developed a web-based interface for annotation, which jointly displays the target operation, required information, stage-to-information mapping, earliest executable stage, inserted exchanges, and turn-level evidence. Three NLP researchers examined randomly sampled instances using a shared protocol. They verified that (i) the operation and required-information set match the source conversation, (ii) each gold value is grounded in the minimum sufficient evidence and bound to the correct task instance, (iii) tentative, superseded, or redundant values are excluded, (iv) the annotated executable stage is the first stage at which all required information values are unambiguously established for execution, and (v) inserted exchanges neither reveal future values nor alter or execute the target task. Samples that violated any criterion were corrected or removed before evaluation.

Table 5: Abridged JSON example from a PTE-Bench conversation. The omission marker replaces intervening off-task turns only.


```
{
  "task": "book_restaurant for service restaurant",
  "required_info": [
    {"name":"restaurant-food",       "value":"indian"},
    {"name":"restaurant-pricerange", "value":"expensive"},
    {"name":"restaurant-area",       "value":"centre"},
    {"name":"restaurant-bookday",    "value":"saturday"},
    {"name":"restaurant-bookpeople", "value":"6"},
    {"name":"restaurant-booktime",   "value":"19:30"},
    {"name":"restaurant-name",       "value":"saffron brasserie"}
  ],
  "conversation": [
    {"speaker":"user", "text":"I'd really like to go to a nice Indian restaurant.", "
     segment_id":1},
    {"speaker":"user", "text":"I'd prefer something a bit on the expensive side.", "
     segment_id":1},
    {"speaker":"assistant", "text":"[state-preserving off-task turns omitted]", "segment_id
     ":1000},
    {"speaker":"user", "text":"I need it to be in the centre of town.", "segment_id":2},
    {"speaker":"user", "text":"I'm thinking Saturday evening. How about 19:30?", "
     segment_id":4},
    {"speaker":"user", "text":"It'll be for six people.", "segment_id":4},
    {"speaker":"user", "text":"Let's go with Saffron Brasserie.", "segment_id":4},
    {"speaker":"assistant", "text":"To recap: Saffron Brasserie for six people at 19:30 on
     Saturday?", "segment_id":5},
    {"speaker":"user", "text":"Yes, that's right!", "segment_id":5}
  ],
  "segment_required_info_mapping": {
    "1":["restaurant-food","restaurant-pricerange"],
    "2":["restaurant-area"],
    "3":[],
    "4":["restaurant-bookday","restaurant-bookpeople","restaurant-name"],
    "5":["restaurant-name","restaurant-bookpeople","restaurant-booktime","restaurant-
     bookday"]
  },
  "earliest_resolvable_segment": 5,
  "segment_required_info_evidence": {
    "2":[{"required_info":"restaurant-area","local_turn_ids":[1],
          "evidence_context":["I need it to be in the centre of town."]}],
    "4":[{"required_info":"restaurant-name","local_turn_ids":[4],
          "evidence_context":["Let's go with Saffron Brasserie."]}]
  }
}
```


Additionally, evaluation is grounded in explicit gold annotations for the target operation, required-information values, and missing-information set, rather than subjective response quality. Semantically equivalent expressions are accepted (e.g., *price is high* and *expensive*), but the underlying information must match the references. For example, if the gold answer specifies Monday and a particular restaurant name, the response must preserve both the day and the restaurant identity, providing clear factual criteria for evaluation. We additionally manually review sampled model responses and their GPT-4o Mini judgments. The reviewed judgments are consistent with human assessments, supporting the reliability of the automatic evaluation.

E Additional Implementation Details

When all candidate tasks have the same value for a signal, its normalized value is set to zero for every candidate. Such a signal does not distinguish among candidates, so this convention avoids division by zero without affecting task ranking. The same rule applies when only one candidate task is available. The block size in ARDSE controls how the available conversation prefix is processed, rather than when execution decisions are made. Any remaining turns form a shorter final block and are processed before the execution decision. Qwen3-14B is evaluated with thinking disabled.

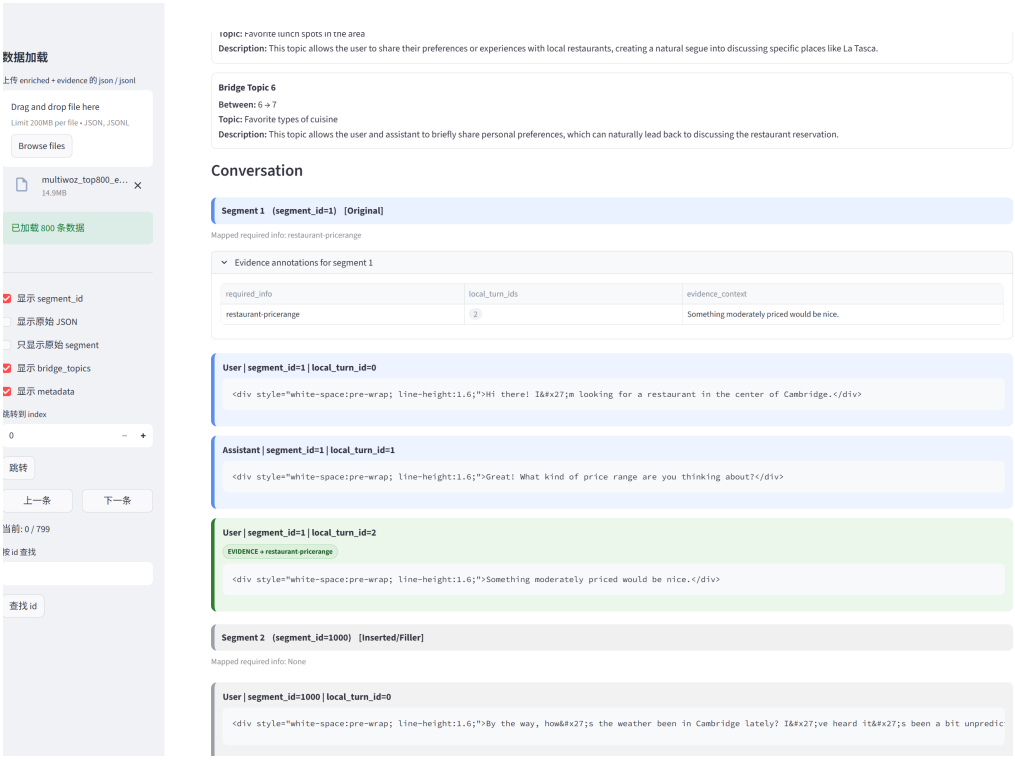

Figure 5: The annotation and human-verification interface.

Table 6: ARDSE performance across four benchmark subsets under the With Task and Without Task settings. Each cell reports ARAcc / RIC (%).

Setting	Subset	GPT-4o Mini	Claude Sonnet 4.6	LLaMA-3.1-8B	Qwen3-14B
With Task	SGD	65.96 / 87.23	69.24 / 92.78	47.89 / 78.15	57.49 / 81.98
With Task	MultiWOZ	33.90 / 76.41	55.93 / 90.46	32.20 / 66.75	30.51 / 67.51
With Task	Cross-Session	48.33 / 75.42	49.00 / 82.25	2.00 / 17.58	41.33 / 60.26
With Task	If–Then	52.14 / 76.04	88.57 / 97.26	50.71 / 79.54	60.71 / 84.85
	Overall	61.11 / 84.66	67.06 / 91.74	41.18 / 70.78	54.14 / 78.94
Without Task	SGD	23.45 / 61.57	28.25 / 57.79	20.29 / 65.16	16.20 / 55.12
Without Task	MultiWOZ	24.58 / 70.01	31.36 / 74.91	27.12 / 71.02	20.34 / 60.23
Without Task	Cross-Session	5.00 / 19.30	5.00 / 24.58	3.67 / 27.28	2.33 / 25.45
Without Task	If–Then	29.29 / 70.24	67.14 / 87.53	19.29 / 65.10	25.00 / 67.07
	Overall	21.43 / 57.73	27.73 / 56.69	18.39 / 61.17	15.12 / 52.73

F Detailed Experimental Results

Table 6 reports ARDSE's performance across four benchmark subsets. ARDSE achieves substantial required-information coverage on SGD, MultiWOZ, and If-Then, with Claude exceeding 90% RIC in all three subsets under With Task. On If-Then, Claude achieves 88.57% ARAcc and 97.26% RIC, retaining 67.14% and 87.53%, respectively, without explicit task specifications. These results demonstrate the effectiveness of using distributed evidence for conditional execution. Meanwhile, cross-Session exhibits greater backbone dependence: with task information, Claude achieves 49.00% ARAcc and 82.25% RIC, compared with LLaMA's 2.00% and 17.58%. These results indicate that proactive task execution remains challenging for smaller LLMs when task threads are deliberately interleaved with multiple unrelated sessions within substantially longer conversation histories.

Table 7: Detailed Active Recall results by task-graph size.

GPT						**Claude**					
		With Task		Without Task				With Task		Without Task	
$\lvert V^{\text{req}}\rvert$	N	ARAcc	RIC	ARAcc	RIC	$\lvert V^{\text{req}}\rvert$	N	ARAcc	RIC	ARAcc	RIC
1–2	41	56.10	64.29	53.66	61.43	1–2	41	82.93	90.00	41.46	51.43
3–4	293	75.43	88.19	37.54	57.49	3–4	293	73.72	85.58	44.37	58.23
5–6	958	67.12	87.01	27.97	61.21	5–6	958	70.04	91.05	33.19	62.58
7–8	564	55.50	85.27	12.77	55.77	7–8	564	65.25	92.71	21.28	55.58
9–10	199	54.27	86.68	4.52	63.38	9–10	199	62.31	93.05	14.07	55.84
11–13	139	46.76	84.64	2.16	53.26	11–13	139	58.27	94.27	7.91	45.16
14+	74	17.57	67.16	2.70	46.57	14+	74	36.49	91.48	6.76	50.54
LLaMA						**Qwen**					
		With Task		Without Task				With Task		Without Task	
$\lvert V^{\text{req}}\rvert$	N	ARAcc	RIC	ARAcc	RIC	$\lvert V^{\text{req}}\rvert$	N	ARAcc	RIC	ARAcc	RIC
1–2	41	43.90	48.57	46.34	61.43	1–2	41	56.09	57.25	51.22	47.14
3–4	293	43.34	58.33	36.52	58.98	3–4	293	66.55	80.44	28.67	48.00
5–6	958	47.39	69.75	22.86	62.87	5–6	958	57.93	80.36	16.81	52.86
7–8	564	33.16	69.81	11.52	62.51	7–8	564	53.01	80.12	10.64	55.03
9–10	199	48.24	79.88	0.50	62.37	9–10	199	51.25	80.47	4.02	57.80
11–13	139	31.65	75.56	2.16	57.89	11–13	139	33.81	77.12	5.04	50.76
14+	74	10.81	70.39	4.05	53.60	14+	74	9.46	68.62	2.70	43.51

Table 7 reports the exact values of performance across Task-Graph sizes in section 6.3, corresponding to Figure 3. Required-information coverage remains comparatively stable across the task sizes that contain most evaluation instances, while exact execution becomes more selective as the task graph contains more requirements. In particular, at 5 to 10 nodes, GPT-4o Mini maintains 85.27% to 87.01% RIC and Qwen3-14B maintains 80.12% to 80.47% RIC with task information. Meanwhile, Claude retains 91.48% RIC even beyond 14 nodes. The same separation between RIC and ARAcc without task information further shows that reconstructing substantial portions of a task state and actively recalling the correct task are distinct capabilities.

G Additional Case Study

Table 8 presents a complementary incomplete-state case in which the restaurant and city have already been established before the conversation shifts to live music, alcohol availability, and contact information. Each baseline fails for a distinct reason. Full-Context Prompting incorrectly requests the restaurant name again, despite its earlier presence in the conversation. A-MEM retrieves task-related evidence but fails to reconcile it with the current state of required information for the reservation, thus treating the known restaurant name as missing. Proactive Agent follows the local request and identifies alcohol availability and contact information instead of the unresolved reservation requirements. Inner Thoughts directly answers the local question while introducing an unsupported contact-number placeholder. AutoTOD reports that the requested local information is unavailable and offers generic assistance, without clearly identifying the pending reservation state. In contrast, ARDSE preserves the reservation state across the topic transition and correctly identifies all four missing items: date, time, party size, and price range. This case demonstrates that access to relevant context, local response generation, and proactive intervention do not ensure accurate reconstruction and identification of a pending task state.

H Efficiency Analysis

We compare ARDSE with the memory-based baseline A-MEM (Xu et al., 2025) using GPT-4o Mini on the same 2,268 Active Recall instances. We report average LLM calls per instance, including memory construction, intermediate inference, and final generation, and token usage. Under With Task and Without Task, ARDSE requires 10.09 and 9.35 calls, compared with 25.54 and 23.52 for

Table 8: Selected GPT-4o Mini responses for an incomplete restaurant-reservation state. Long off-task exchanges are omitted.

Method / context	Selected content	Observation
Task state	Operation: reserve a restaurant. Known: *8 Immortals Restaurant*, *San Francisco*. Final local query: "Do they serve alcohol? And what's their contact number?" Gold missing set: date, time, party size, and price range.	The current query is related to the restaurant but does not complete the reservation state.
Full-Context	"No. Missing: restaurant name, time, party size, date, price range."	Re-requests an already established value.
A-MEM	"No; Missing: restaurant name, time, party size, date, price range."	Retrieved context is not fully reconciled with the task requirements.
Proactive Agent	"No. Missing: contact number, alcohol availability."	Tracks the local question instead of the pending reservation.
Inner Thoughts	"Yes. 8 Immortals Restaurant does serve alcohol. Their contact number is [insert contact number here]."	Executes the local request and introduces an unsupported placeholder.
AutoTOD	"I don't have that information available. Would you like me to help with anything else?"	Responds to local information availability without reconstructing the reservation state.
ARDSE	"No. Missing: time, party size, date, price range."	Recovers the complete gold missing set without premature execution.

A-MEM, reducing calls by 60.5% and 60.2%, respectively, while improving ARAcc by 44.58% and 13.01%. ARDSE's auxiliary task-state context contains only 216 and 272 tokens, compared with 2,309 and 2,340 tokens of retrieved memory for A-MEM. Assuming comparable latency per call and serial execution, ARDSE offers an estimated 2.5× speedup over A-MEM in both settings. These results show that ARDSE achieves higher execution accuracy with higher efficiency: fewer LLM calls and more compact auxiliary context.

I PROMPTS

Tables 9 and 10 document the preliminary study: the former gives the common generation instruction and task-specific explicit request used to construct the three settings, while the latter defines the completion and Missing-State judges. Tables 11-18 present the benchmark-construction prompts in processing order. Table 11 extracts the target operation and minimum required information; Table 12 divides each seed conversation according to task progress; Table 13 rewrites the staged trajectory as a natural conversation; Table 14 reconstructs the segment-to-information mapping and earliest executable segment; Tables 15 and 16 propose and realize state-preserving off-task exchanges; Table 17 identifies the minimum supporting evidence; and Table 18 repairs trajectories whose terminal segment does not establish all required information. Repaired trajectories are checked again before contextual expansion.

Tables 19-22 provide the generalized prompts for ARDSE. Table 19 updates progressive task states with newly observed evidence; Table 20 reconciles the maintained states with the complete conversation; Table 21 refines task requirements and verifies their supporting evidence; and Table 22 generates the final response from the verified state. For the baselines, Table 23 reports the Full-Context Prompting and A-MEM prompts, Table 24 presents the Proactive Agent prompts, Table 25 presents the Inner Thoughts prompts, and Table 26 gives the AutoTOD prompt. Table 27 provides the semantic evaluation prompts for Active Recall and Task State Recognition. Bracketed variables are instantiated for each example, while deterministic parsing, score-based task activation, and output post-processing are omitted because they do not invoke an LLM. For each backbone and input setting, all methods receive the same available conversation and task information and are evaluated using the same criteria. We use generalized, method-specific prompts across evaluation

Table 9: Generation prompts used in the preliminary study.

Generation prompts used in the preliminary study

[Common system prompt]
You are a helpful conversational assistant. Continue the conversation naturally. Use only information provided in the conversation and do not invent facts.
[Explicit Baseline and Missing-State final request]
{task-specific explicit request}
The request is instantiated using the target-specific realization defined for each of the 13 STAR APIs. Active Recall receives no appended request; all preceding conversation turns remain identical.

Table 10: LLM-judge prompts used in the preliminary study.

LLM-judge prompts used in the preliminary study

[Completion judge]
Evaluate an assistant response to a conversational task using the response text only.
Target operation: {target_api}
Gold required information: {numbered_required_information}
Assistant response: {response}
First classify the target operation using exactly one label. Use `executed` when the response supplies the requested result, reports completion or its outcome, or makes an unconditional commitment to perform the operation. A negative result also counts when it demonstrates that the requested operation was performed. Use `not_executed` when the response only offers help, asks whether it should proceed, requests information or confirmation, refuses, discusses the task, or performs a different operation.
Judge every gold information item independently. An item is covered only when the response explicitly states the correct value or unambiguously applies it to the target operation. Equivalent wording is acceptable; omitted, incorrect, guessed, vague, or unrelated values are not covered. Return one decision for every listed index and no additional indices.
Return JSON only:
```
{ "operation": "executed",
"information": [{ "index": 0, "covered": true },
{ "index": 1, "covered": false }] }
```
[Missing-State judge]
Classify an assistant response when one essential task value is unavailable.
Target operation: {target_api}
Removed required information: {field}: {gold_expression}
Assistant response: {response}
Choose exactly one mutually exclusive label in the following order: `overconfident_execution` when the response claims, promises, or reports execution despite the unavailable value, including invented values or task-specific results; `correct_clarification` only when it does not execute and explicitly requests or identifies the removed information type; and `other` otherwise. If a response both executes and asks for information, use `overconfident_execution`. Performing a different operation is `other`.
Return JSON only: `{ "label": "overconfident_execution" }`.

instances. The baseline prompts explicitly instruct models to recall earlier requests and, for Task State Recognition, identify missing information before execution. Prompt structure follows each method's processing pipeline. ARDSE's additional prompts implement progressive state monitoring and evidence verification, which are evaluated as components of the proposed framework.

Table 11: Task and required-information extraction prompt.

Task and required-information extraction prompt

Given a task-oriented conversation, extract one concise task operation and the minimum information required to complete it. Include only information explicitly supported by the conversation, retain final corrected values, and keep independently fillable inputs as separate fields. Do not add optional preferences, inferred facts, assistant capabilities, execution outcomes, or explanations.
Return JSON only:
`{ "task": "concise operation", "required_info": [{ "name": "field", "value": "gold value" }] }`
Task context: {task}
Conversation: {conversation_json}

Table 12: Task-progress segmentation prompt.

Task-progress segmentation prompt

You are given a task, its required information, and a conversation. Segment the conversation into a small number of temporally ordered stages that reflect meaningful changes in task progress, including task introduction, partial information acquisition, revision, and the first stage at which all necessary information is available. Each turn must belong to exactly one contiguous stage. Multiple related fields may be established in the same stage; do not mechanically create one stage per field.
Return JSON only:
`{ "segments": [ { "turn_index": 0, "segment_id": 1 }, { "turn_index": 1, "segment_id": 1 }] }`
The turn index is 0-based, every turn must appear exactly once, and positive segment IDs must be non-decreasing. Prefer three to six stages unless conversation length requires otherwise.
Task: {task}
Required information: {required_info_names}
Conversation: {conversation_json}

Table 13: Conversation rewriting prompt.

Conversation rewriting prompt

You are given a task-oriented conversation. Rewrite it as a natural and realistic interaction while preserving the task progression and stage structure indicated by `segment_id`.
Reveal information gradually rather than front-loading the task, avoid rigid slot filling and repetitive acknowledgements, and let the user remain the primary source of task-specific information. The assistant may guide, react, or summarize, and may supply system-derived information only when supported by the seed. Preserve the original language, task intent, final required-information values, and coarse stage order. A stage may introduce multiple related values, but early stages must not contain the complete task specification. Turns may be added or removed when needed for coherence. Do not use backend, tool, database, or system-query language.
Return JSON only:
`{ "conversation": [ { "speaker": "user", "text": "...", "segment_id": 1 }, { "speaker": "assistant", "text": "...", "segment_id": 1 }] }`
Every turn must contain exactly `speaker`, `text`, and a positive, non-decreasing `segment_id`.
Task: {task}
Required Information: {required_info_json}
Original Conversation: {conversation_json}

Table 14: Task-state trajectory analysis prompt.

Task-state trajectory analysis prompt

You are given a task-oriented conversation with segment annotations. For every required-information field, identify the segment in which its final correct value, or an equivalent expression, is established. Tentative, partial, incorrect, or later-revised mentions must not be treated as final. Also identify the earliest segment after which the task can be completed using all final values.
Return JSON only:
```
{ "segment_required_info_mapping":
{ "1": ["..."], "2": ["..."] },
"earliest_resolvable_segment": 3 }
```
Task: {task}
Required Information: {required_info_json}
Conversation: {conversation_json}

Table 15: Off-task bridge-topic proposal prompt.

Off-task bridge-topic proposal prompt

You are given a task and the conversation segments immediately before and after an insertion point. Propose a concise intermediate topic, but do not generate the intervening conversation. The topic must clearly deviate from the task: it cannot continue, reformulate, clarify, summarize, or advance the task, and it cannot repeat required information. It should nevertheless be loosely motivated by the local context so that a temporary side conversation and a natural return are plausible. Prefer everyday tangents, situational remarks, passing associations, minor concerns, or anecdotes, using the language of the surrounding conversation.
Return JSON only: `{ "bridge_topic": "...", "short_description": "..." }`.
Task: {task}
Previous Segment: {prev_segment_json}
Next Segment: {next_segment_json}

Table 16: Off-task conversational expansion prompt.

Off-task conversational expansion prompt

Given a task, a bridge topic, the segments before and after an insertion point, and a target minimum length, generate a realistic intervening conversation driven primarily by the bridge topic. Its opening should connect naturally to the preceding segment and its ending should return smoothly to the following segment. Keep the target task in the background: do not collect its required information, advance or execute its operation, or reveal, paraphrase, or imply information that appears only later. Generate at least {target_turns} turns with strict user–assistant alternation, add genuinely new content, and preserve the surrounding language.
Return JSON only:
```
{ "conversation": [
{ "speaker": "user", "text": "..." },
{ "speaker": "assistant", "text": "..." }] }
```
Task: {task}
Bridge Topic: {bridge_topic}
Bridge Topic Description: {bridge_topic_desc}
Previous Segment: {prev_segment_json}
Next Segment: {next_segment_json}
Target Length: {target_turns}

Table 17: Minimal evidence annotation prompt.

Minimal evidence annotation prompt

You are given a task, one local conversation segment, and the required-information items known to be established in that segment. For each supplied item, identify the minimum local turn or turns that establish its final usable value. Work strictly within the supplied segment and annotate only the supplied items. Prefer later evidence when an earlier mention is tentative, partial, or superseded; exclude weak implications and redundant turns.
Return JSON only:
`{ "annotations": [ { "required_info": "...", "local_turn_ids": [0], "evidence_context": ["..."] }] }`
The `required_info` value must exactly match an input item, `local_turn_ids` are 0-based indices in the supplied segment, and `evidence_context` contains the corresponding exact turn texts.
Task: {task}
Required Information Assigned to This Segment: {segment_required_info_json}
Local Segment: {local_segment_json}

Table 18: GPT-based task-state trajectory repair prompt.

Task-state trajectory repair prompt

You are given a task-oriented conversation with segment annotations. Rewrite it into a natural and realistic conversational form while preserving the original task intent and overall stage structure.
Input.
The input contains the task, its required information, the missing required-information fields that must be newly established in the final segment, and the original conversation with `speaker`, `text`, and `segment_id`.
Rewriting requirements.
Keep the conversation natural and human-like. Important task-related information should mainly come from the user. Every missing required-information field must be newly and naturally established in the final segment, preferably by the user. Preserve the overall conversation flow and segment structure; turns may be added or revised only when necessary.
Return JSON only:
`{ "conversation": [ { "speaker": "user", "text": "...", "segment_id": 1 }, { "speaker": "assistant", "text": "...", "segment_id": 1 }] }`
Each turn must contain exactly `speaker`, `text`, and `segment_id`. The speaker must be either `user` or `assistant`, and positive segment IDs must be non-decreasing. Do not output explanations or additional metadata.
Task: {task}
Required Information: {required_info_json}
Missing Required Information: {missing_required_info_json}
Original Conversation: {conversation_json}

Table 19: Generalized progressive task-state update prompt used by ARDSE.

ARDSE progressive task-state update prompt

Maintain the progressive state of unresolved user tasks while processing a conversation chronologically. Given the current task state and one new conversation block, identify only evidence-supported changes introduced by that block.
Current task state: {current_task_state}
Current conversation block with turn IDs: {current_block}
Preserve an initialized operation and its required-information names when supplied; otherwise retain the strongest explicit unresolved user operation and its coordinated sub-operations. Bind every value to the correct task, entity, and request instance. Record newly supplied values, explicit corrections, withdrawals, and conflicts together with their supporting turn IDs. Later explicit corrections supersede earlier values, whereas unrelated turns leave the previous state unchanged. Preserve natural date and time expressions, treat negative boolean values as supplied, and never invent placeholders, future actions, missing fields, completeness decisions, or response text.
Return JSON only:

```
{ "operation": "", "sub_operations": [],
"requirement_updates": [
{ "name": "", "action": "set | remove | conflict",
"value": "", "evidence_turn_ids": [0] }] }
```

Table 20: Generalized global task-state reconciliation prompt used by ARDSE.

ARDSE global task-state reconciliation prompt

Reconcile the progressively maintained task state with the complete conversation prefix. This is an internal state-repair step rather than a final-answer step.
Current task state: {current_task_state}
Complete conversation prefix with turn IDs: {conversation_prefix}
Treat the conversation as authoritative. Preserve supported operations and values, and repair omitted, stale, invalid, conflicting, or wrongly bound items without inventing information. Every filled or corrected value must copy the conversational expression, cite its supporting turn IDs, and remain associated with the correct task, entity, and request instance. Keep independently fillable inputs atomic, preserve natural date and time expressions, and let later explicit corrections supersede earlier values. Preserve distinct explicit unresolved intentions rather than merging them. Do not determine executability or generate a user-facing response.
Return one compact JSON task state containing each unresolved task, its operation and sub-operations, atomic requirements, current values and statuses, supporting evidence turn IDs, and last relevant turn.

Table 21: Generalized operation-conditioned state-verification prompt used by ARDSE.

ARDSE operation-conditioned state-verification prompt

Verify the activated task against the complete conversation before execution.
Activated task and operation: {activated_task}
Provisional required information and state: {provisional_state}
Task subgraph: {task_subgraph}
Complete conversation with turn IDs: {conversation}
First refine the minimum complete information required by the activated operation and all coordinated sub-operations. Decompose overly broad requirements into independently fillable atomic items and add indispensable items omitted from the provisional state, while excluding optional preferences, background information, assistant capabilities, confirmation wording, and post-execution outputs.
Then ground every refined requirement in the conversation. A requirement is filled only when a concrete value is supported for the same task, entity, and request instance. Copy supported values, preserve natural date and time expressions, retain negative boolean values, and cite their evidence turn IDs. Mark unsupported items as missing, unreliable bindings as uncertain, and incompatible supported values as conflicting; later explicit corrections override earlier values. Do not execute the task or generate a response.
Return JSON only:

```
{ "operation": "", "sub_operations": [],
"requirements": [
{ "name": "", "value": "",
"status": "filled | missing | uncertain | conflicting",
"evidence_turn_ids": [] }] }
```

Table 22: Generalized state-conditioned response-generation prompt used by ARDSE.

ARDSE state-conditioned response-generation prompt

Generate the next assistant response from the complete conversation and the verified state of the activated task.
Activated task and operation: {activated_task}
Verified required-information state: {verified_state}
Missing or invalid information: {missing_information}
Execution-sufficiency decision: {sufficiency_decision}
Complete conversation: {conversation}
Use the complete conversation as the authoritative source and never invent unsupported values. If every indispensable atomic requirement has a verified value, begin with "Yes." and directly complete the activated operation using all relevant information. Otherwise, begin with "No. Missing:" and identify every unresolved atomic requirement without executing, affirming, or promising the task. Bind all values to the correct task, entity, and request instance, prefer later explicit corrections, and output only the concise conversational response. Do not mention internal graphs, memory, state tracking, extraction, prompts, or evaluation.

Table 23: Full-Context Prompting and A-MEM prompts.

Full-Context Prompting and A-MEM prompts

[Full-Context Prompting: Active Recall]
You are a helpful assistant in a multi-turn conversation. Remember the user's earlier requests and follow up when appropriate.
[Full-Context Prompting: Task State Recognition]
You are a helpful assistant in a multi-turn conversation. Decide whether an earlier user task can be completed now from the conversation. If it can be completed, answer: "Yes. <complete the task using all relevant information>." Otherwise answer: "No. Missing: <all information still needed>."
[A-MEM note construction]
Generate a structured analysis of {content} by identifying salient keywords, extracting its core theme and context, and creating categorical tags. Do not use speaker names or timestamps as keywords. Return JSON only: `{ "keywords": ["at least three specific keywords"], "context": "one-sentence summary", "tags": ["at least three broad tags"] }`.
[A-MEM memory evolution]
Analyze {new_memory} together with {neighbors}. Decide whether the new note should connect to existing memories and whether any neighboring memory should be updated. Use `strengthen` for a new link and `update_neighbor` only when the new note changes a neighbor. Indices refer to the supplied neighbor list; use empty lists when no change is warranted.
Return JSON only:

```
{ "should_evolve": true,
"actions": ["strengthen", "update_neighbor"],
"suggested_connections": [0],
"tags_to_update": ["tag"],
"new_context_neighborhood": ["updated context"],
"new_tags_neighborhood": [["updated tag"]] }
```

[A-MEM retrieval-query generation]
Given {question}, generate several retrieval keywords using `cosmos` as the separator. Return JSON only: `{ "keywords": "keyword1 cosmos keyword2 cosmos keyword3" }`.
[A-MEM final response]
Continue the conversation naturally. Remember earlier requests and, when an earlier task can now be completed, complete it using all relevant information. For Task State Recognition, return either "Yes: <answer using all relevant information>" or "No; Missing: <all missing information>."

Table 24: Proactive Agent prompts.

Proactive Agent prompts

[Proactive detector]
Given a conversation, decide whether the assistant should take the initiative in the next response. Initiative is appropriate when the conversation contains an unfinished user intention and enough detail is available to provide useful assistance. Use only the conversation and do not assume hidden task labels. Return JSON only: `{ "should_act": true, "proactive_task": "concrete task or empty string", "reason": "brief reason" }`.
[Proactive generator]
Continue the conversation by directly helping with the predicted user need when it is concrete enough. Use only the conversation and include relevant details. Otherwise, continue naturally and request the missing information.
[State detector]
Given a conversation, decide the next action for an unresolved earlier task. If enough information is available, complete it using all relevant information; otherwise identify every missing item. Use only the conversation and do not assume hidden task labels. Return JSON only: `{ "can_complete": true, "missing_information": ["..."], "reason": "brief reason" }`.
[State generator]
Use the controller diagnosis and conversation. If the task can be completed, answer: "Yes. <complete the task using all relevant information>." Otherwise answer: "No. Missing: <all information still needed>."

Table 25: Inner Thoughts prompts.

Inner Thoughts prompts
[System 1] Produce one quick, intuitive private thought prompted by the recent conversation. Do not address the user or write a final response. Return JSON only: { `"thought": "one concise inner thought"` }. **[System 2]** Deliberately form two useful private thoughts from the recent conversation and retrieved memories. A thought may connect a current detail to an earlier unresolved intention, assess whether action is appropriate, or identify an information gap. Do not write the final response. Return JSON only: { `"thoughts": [{ "content": "first thought", "stimuli": ["memory or context id"] }, { "content": "second thought", "stimuli": ["memory or context id"] }]` }. **[Single-thought evaluation]** Score the supplied thought's intrinsic motivation to be expressed from 1.0 to 5.0. Consider its relevance to an unresolved intention, closure of an information gap, expected usefulness, urgency, coherence, and non-redundancy. Return JSON only: { `"score": 1.0, "reason": "brief reason"` }. **[Articulation]** Turn the selected private thought into the next assistant response. Directly help with the unresolved earlier intention when it is concrete enough, using only the provided context. For Task State Recognition, return exactly either "Yes. <complete the task using all relevant information>" or "No. Missing: <all information still needed>." Do not mention thoughts, retrieval, memories, scores, or hidden labels.

Table 26: AutoTOD prompt.

AutoTOD prompt
The AI Assistant is an intelligent agent that helps the user complete tasks. A task may contain multiple subtasks; first determine the relevant subtasks, then complete the request according to their instructions. Be friendly and polite, use only information supplied in the conversation, and never invent parameter values. Available tools: - Task Query: inspect or organize task information already available. It cannot provide facts absent from the conversation. - Task Operation: simulate a concrete user-requested operation using every relevant parameter available in the conversation. Determine the relevant task from the conversation. If necessary information is missing, request or identify it rather than invoking a tool with invented values. When information is sufficient, use Task Operation and summarize the operation and parameters. Tool-call format: Thought: Do I need to use a tool? Yes Action: [Task Query | Task Operation] Action Input: [input] Observation: Final-response format: Thought: Do I need to use a tool? No AI Assistant: [response] For Active Recall, continue the conversation by remembering earlier requests and following up when appropriate. For Task State Recognition, the content after "AI Assistant:" must be either "Yes. <complete the task using all relevant information>" or "No. Missing: <all information still needed>."

Table 27: Semantic evaluation prompts used for PTE-Bench.

Semantic evaluation prompts used for PTE-Bench

[Active Recall judge]
Check whether a model response correctly completes a task. Judge only from the task, gold required information, and response. Equivalent natural wording is acceptable; wrong, vague, unsupported, or inconsistent values are missing.
Task: {task}
Gold required information: {required_info_json}
Model response: {model_response}
Judge whether the response performs the operation implied by the task and check every gold information item independently. An item is correctly used only when the response includes the gold value or an unambiguous equivalent. Record every missing item together with its field, gold answer, and corresponding response content or "not mentioned." Treat equivalent time expressions as correct and be permissive about date format unless it is clearly inconsistent with the gold date. Require exact or clearly equivalent values for names, locations, quantities, service slots, and other factual arguments. Complete proactive resolution is true only when the operation is correct and no required information is missing.
Return JSON only:

```
{ "operation_correct": true,
"all_required_information_provided": false,
"missing_required_info": [
{ "name": "...", "gold_answer": "...",
"model_response_content": "not mentioned" }],
"complete_proactive_resolution": false,
"explanation": "short explanation" }
```

[Task State Recognition semantic alignment]
Judge whether a response correctly recognizes an incomplete task and identifies exactly the gold missing information.
Task: {task}
Gold missing information: {gold_missing_json}
Model response: {model_response}
Extract every information item that the response explicitly identifies or requests as missing. Match these items to the gold set using strict semantic equivalence.
Return matched gold items in `covered`, using their exact gold names. Return all additional predicted missing items in `extra_missing`. Do not discard extra predictions or credit vague requests for "details."
Set `incomplete_without_execution` to `true` only if the response recognizes the target task as incomplete and neither executes it, claims completion, nor unconditionally commits to executing it. A response that both executes and requests missing information must receive `false`.
Return JSON only:

```
{
  "covered": [],
  "extra_missing": [],
  "incomplete_without_execution": false
}
```