AutoPersona: Proactively Resolving Implicit Requirements in Tasks through Structured Personalized Memory

Qijia Zhuang,[1] **Zihan Shen,**[1] **Rui Liu,**[2] **Yuxiang Ren**[1]*

[1]Nanjing University
[2]Northeastern University

Abstract

As agents increasingly undertake long-horizon personal assistant tasks in concrete application scenarios, personalized memory should support more than passive storage and retrieval. It should also proactively recover implicit requirements in tasks, determine whether prior experience is applicable to the current task, and identify information gaps before they affect task execution. Existing memory systems typically preserve user–agent interaction trajectories, but they lack joint modeling of environment–user–agent interactions and do not learn how memory should influence future actions. We present AutoPersona, a proactive personalized memory framework for resolving implicit requirements during task execution. AutoPersona organizes continual interactions into three complementary memory layers: Trajectory Memory preserves traceable interaction records and task execution histories; Workspace Memory captures user–environment interactions, application states, and task-specific information; and Persona Memory stores user information, preferences, and personalized assistance strategies. A dedicated PersonaAgent proactively utilizes these memories by completing evidence-supported implicit constraints, asking targeted clarification questions when existing memory is insufficient for reliable inference, and injecting personalized guidance into relevant task nodes. Building upon supervised fine-tuning, we use reinforcement learning to jointly optimize personalized memory summarization and clarification of implicit requirements. Experiments on Persona2Web, PersonaMem-v2, and Π-Bench demonstrate that AutoPersona's structured personalized memory and proactive memory utilization effectively improve task execution performance, personalization consistency, and execution efficiency.

Introduction

Agents are evolving from single-turn question-answering systems into personal assistants capable of carrying out long-horizon, user-facing tasks (Park et al. 2023; Garg, Nitin, and Huang 2026). Systems such as OpenClaw and Claude Code demonstrate the potential of agents to operate across applications, tools, files, and long-running workflows (OpenClaw Contributors 2026; Anthropic 2025). Long-horizon personal assistants must understand not only the user's current request, but also user-specific information, historical task states, and implicit conditions that persist across tasks. Memory is therefore not merely an auxiliary component for preserving history, but a foundation for long-term personalization and task continuity (Zhong et al. 2024; Packer et al. 2023; Li et al. 2025; Wu et al. 2024; Maharana et al. 2024; He et al. 2026).

However, existing memory frameworks still face fundamental limitations in user-facing agent settings.

Incomplete modeling and organization of environment–user–agent interactions. Existing methods often preserve only partial information from user conversations or agent trajectories (Chhikara et al. 2025; Xu et al. 2026; Lin et al. 2026; Tan et al. 2025; Sun, Zhang, and Zeng 2026), without fully modeling the interactions among the environment, the user, and the agent in on-device tasks. In practice, user goals, agent actions, and environment states are interdependent; together, they determine how a task is executed and which constraints should persist into subsequent tasks. Moreover, different applications depend on different types of memory: web tasks emphasize user preferences and prior choices, whereas development tasks rely on project conventions, environment configurations, and failure experience. Placing all such information into a flat store-and-retrieve module can obscure information provenance and introduce redundant context. On-device personalized memory should therefore model complete environment–user–agent interactions and organize user information, task states, and execution experience in a differentiated manner.

Limited support for implicit requirements and proactive assistance. In long-horizon personal-assistant tasks, users often state only an explicit goal while omitting implicit requirements such as preferences, constraints, historical states, and execution criteria. Simply retrieving relevant memories and appending them to the context does not ensure that the information will be used at the appropriate stage of task execution. The agent must also determine which requirements can be reliably inferred from memory, which information remains missing, and at which task nodes the resulting personalization constraints should be injected. When memory evidence is insufficient, the system should avoid blindly assuming the user's intent and instead ask a targeted clarification (Huang et al. 2025; Kim, Lee, and Lee 2026; Shen et al. 2026; Zhang et al. 2026c). Figure 1 illustrates this distinction: traditional memory methods often stop at semantic retrieval, whereas proactive personalized memory must identify implicit requirements, detect information gaps, and transform

*Corresponding author.

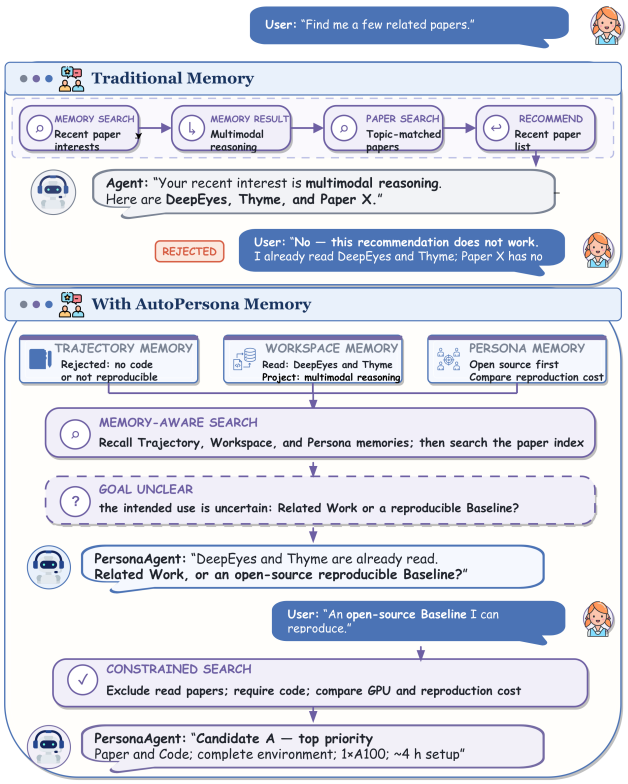

Figure 1: A motivating example comparing traditional memory with AutoPersona. Traditional memory primarily performs semantic retrieval over related history, whereas AutoPersona identifies implicit requirements and information gaps, asks a targeted clarification when necessary, and produces task outcomes consistent with the user's history, workspace state, and personalization constraints.

memory into personalized strategies that influence task planning and execution.

To address these challenges, we propose **AutoPersona**, a proactive personalized memory framework. AutoPersona uses structured personalized memory to identify and resolve implicit requirements, while a dedicated **PersonaAgent** performs memory retrieval, proactive clarification, and personalized intervention. At its core is a structured memory bank composed of three complementary memory types. **Trajectory Memory** preserves traceable interaction records and task execution histories. **Workspace Memory** captures user–environment interactions, application states, and task-specific information. **Persona Memory** stores user information, preferences, and personalized assistance strategies. By separating trajectory evidence, workspace state, and user persona, AutoPersona preserves information provenance while providing compact and actionable personalized context for the current task.

We first use supervised fine-tuning to teach PersonaAgent to retrieve relevant personalized memories and generate responses from the retrieved evidence. We then apply reinforcement learning to jointly optimize the summarization of task-required personalized memory points and clarification of missing implicit requirements (Ouyang et al. 2022; Chen et al. 2023; Shao et al. 2024; Gao et al. 2025). This training rewards correct and necessary memory points and clarification questions while penalizing omissions, incorrect information, unnecessary questions, and irrelevant memory. At inference time, AutoPersona employs collaborative personalized task-graph execution (Chen et al. 2024; Zhang et al. 2025; Ruan et al. 2026). It decomposes a task into dependent subtask nodes and repeatedly injects user preferences, historical information, and personalized strategies, allowing personalization constraints to persist throughout task planning and execution.

Our main contributions are summarized as follows:

- We propose **AutoPersona**, a proactive personalized memory framework that uses three structured memory layers—Trajectory Memory, Workspace Memory, and Persona Memory—to provide traceable personalized context for resolving implicit task requirements.
- We design PersonaAgent to decouple personalized memory use from the main agent and train it through supervised fine-tuning and reinforcement learning to perform memory retrieval, personalized memory summarization, and proactive clarification when information is missing.
- We systematically evaluate AutoPersona on Persona2Web, PersonaMem-v2, and Π-Bench (Kim, Lee, and Lee 2026; Jiang et al. 2025; Zhang et al. 2026a), demonstrating its effectiveness in task performance, personalization consistency, proactive clarification, and execution efficiency.

Related Work

Personalized Memory. Personalized memory enables agents to preserve user information across sessions and recover relevant preferences and historical context in subsequent tasks (Zhong et al. 2024; Huang et al. 2026a; Packer et al. 2023; Li et al. 2025; Tan et al. 2025; Huang et al. 2026b). Early approaches typically write dialogue snippets or extracted user facts to external storage and retrieve them through semantic similarity (Park et al. 2023). Mem0 (Chhikara et al. 2025) provides a scalable long-term memory layer for production environments, maintaining user, session, and agent states through information extraction, update, and retrieval. A-MEM (Xu et al. 2026) draws on the Zettelkasten method to organize memories as dynamic notes with contextual descriptions, keywords, and tags, continuously updating the memory network through link generation and memory evolution. LightMem (Fang et al. 2025) further improves efficiency by separating memory into perceptual, short-term, and long-term stages and moving long-term consolidation offline to reduce online inference overhead. Me-Agent (Wang et al. 2026; Lin et al. 2026) distinguishes long-term user habits from application-specific preferences to handle ambiguous instructions on mobile devices. These methods improve memory storage and retrieval, yet most still treat memory as auxiliary external context and provide limited support for using memory to resolve implicit

requirements and inject personalization during task execution (Zhang et al. 2026c; Sun, Zhang, and Zeng 2026; Huang et al. 2026b). In contrast, AutoPersona explicitly models user–agent–environment interactions and workspace states, and uses structured personalized memory to identify and resolve implicit requirements while continually injecting personalization during task execution.

Agent Optimization. Large language model agents are commonly initialized through supervised fine-tuning (SFT) to learn instruction following, tool use, and basic action formats (Ouyang et al. 2022; Chen et al. 2023), and then further optimized through reinforcement learning based on task outcomes (Jung et al. 2025; Gao et al. 2025). With the development of Group Relative Policy Optimization (GRPO) and related methods, reinforcement learning based on task rewards has become an important paradigm for optimizing the decision-making and execution capabilities of large language model agents (Shao et al. 2024). In the memory domain, Memory-R1 (Yan et al. 2025; Yu et al. 2026; Wang et al. 2025) trains a Memory Manager and an Answer Agent with PPO and GRPO to learn memory addition, deletion, update, selection, and use. DeltaMem (Zhang et al. 2026b) combines operation-level supervision and reinforcement learning to optimize persona-centric memory management. These methods primarily focus on general agent execution, memory operations, or answer-generation quality. However, how to train agents to use personalized memory for implicit-requirement identification or proactive clarification remains underexplored. AutoPersona trains PersonaAgent through SFT and reinforcement learning, jointly optimizing personalized memory summarization and clarification of implicit requirements so that structured memory can directly participate in task planning and node-level personalized decisions.

Methodology

AutoPersona resolves implicit task requirements through structured personalized memory. It decouples personalization from general task execution: PersonaAgent retrieves relevant information from the three memory types, summarizes personalized constraints, and proactively asks the user for clarification when information is insufficient, while the main LLM decomposes and executes the task.

Through supervised fine-tuning and reinforcement learning, PersonaAgent learns to extract personalized memories and inject personalized constraints into relevant nodes throughout task-graph execution. Figure 2 presents the overall architecture, with further implementation details provided in Appendix B.

Structured Personalized Memory Bank Construction

Given a sequence of historical user interactions $\mathcal{H} = \{h_j\}_{j=1}^{N}$, each history record is represented as

$$h_j = (q_j,\ \tau_j,\ \mathcal{A}_j,\ f_j)\,, \tag{1}$$

where q_j is the initial user query, τ_j is the raw execution trajectory, $\mathcal{A}_j$ is the set of task-related artifacts, and f_j is feedback, a success signal, or an evaluation result. AutoPersona does not directly preserve lengthy raw histories. Instead, it first refines task trajectories and then extracts Workspace Memory and Persona Memory from the refined trajectories. The resulting structured personalized memory bank is

$$\mathcal{M} = (\mathcal{M}_\tau,\ \mathcal{M}_W,\ \mathcal{M}_P)\,, \tag{2}$$

where $\mathcal{M}_\tau$, $\mathcal{M}_W$, and $\mathcal{M}_P$ denote Trajectory Memory, Workspace Memory, and Persona Memory, respectively.

Trajectory Memory Trajectory Memory is AutoPersona's traceable evidence layer. It preserves compact records of user requests, agent actions, environment responses, and task outcomes, supporting historical recall, provenance verification, and conflict resolution.

Trajectory Refinement. AutoPersona constructs Trajectory Memory from records of user–agent–environment interactions in historical tasks. For the j-th historical task, the refinement step removes redundant intermediate artifacts and environment responses produced during execution, retaining only the user instruction, key agent actions, and final task state:

$$\Delta\mathcal{M}_\tau^{(j)} = (t_j, \tau_j, m_j)\,, \tag{3}$$

where t_j is the task identifier, τ_j denotes the retained key interaction record, and m_j contains metadata such as the user ID, task type, timestamp, dependencies, and success status. Trajectory Memory also serves as the basis for subsequent extraction of Workspace Memory and Persona Memory, but is not fully injected into every prompt by default.

Workspace Memory Workspace Memory records interactions between the user and the task environment as well as the current working state. It consists of three parts:

- **Interaction Log** records the user's historical operations in the workspace, such as opening files, changing settings, invoking tools, and visiting pages.
- **State History** records environment state changes caused by these operations, such as file-content updates, tool-configuration changes, page-state transitions, and the creation of task artifacts.
- **Task State** records the current task, execution progress, completed steps, pending steps, and cross-task dependencies.

Accordingly, Workspace Memory is represented as

$$\mathcal{M}_W = \{\mathcal{M}_{\text{log}}, \mathcal{M}_{\text{state}}, \mathcal{M}_{\text{task}}\}\,. \tag{4}$$

AutoPersona extracts this information from historical task trajectories and updates Workspace Memory as

$$\Delta\mathcal{M}_W^{(j)} = \text{Extract}_W(\tau_j). \tag{5}$$

Workspace Memory allows the agent to understand how the user previously operated within the workspace, which environment changes occurred, and how far the current task has progressed, thereby providing reusable workspace context for subsequent tasks.

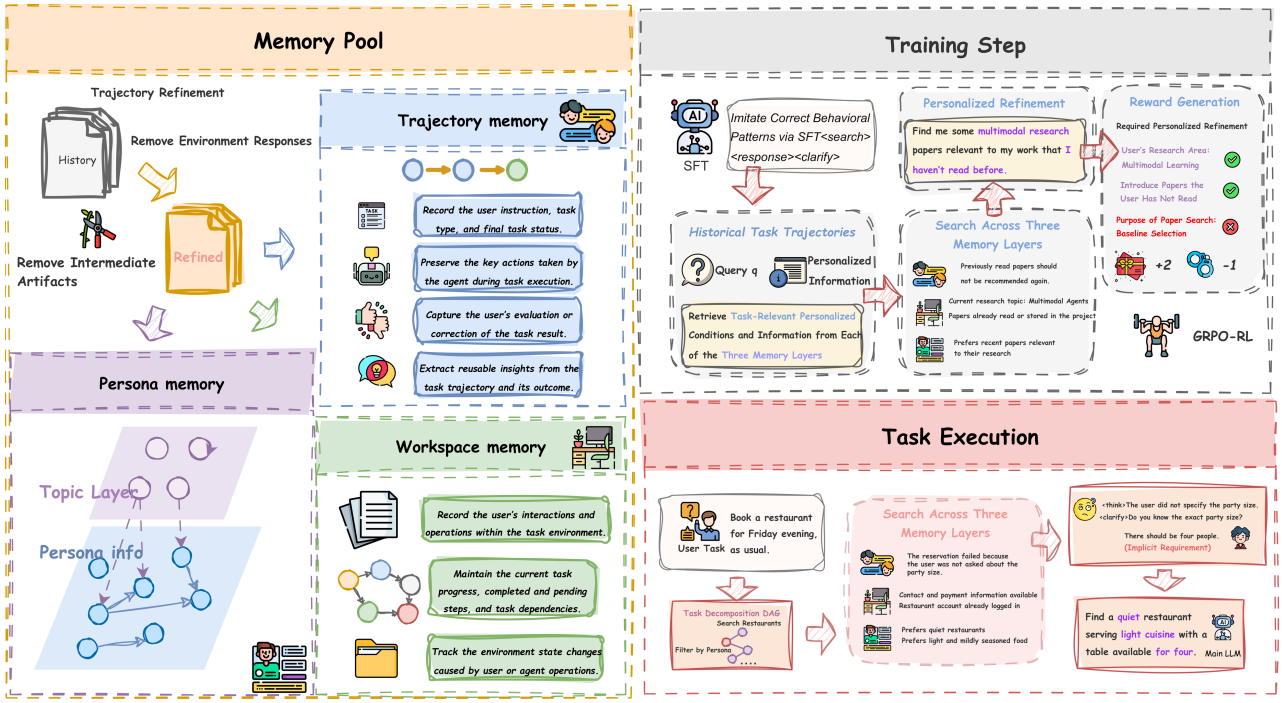

Figure 2: The overall framework of AutoPersona. Historical interactions are first organized into three layers of structured memory. Trajectory Memory preserves key task evidence from interactions among the user, agent, and environment. Workspace Memory records user interaction logs, environment state changes, and current task state. Persona Memory stores applicable tasks, user preferences, personalized assistance strategies, and their historical evidence. PersonaAgent first learns memory retrieval and personalized response generation through supervised fine-tuning, and then jointly optimizes personalized memory summarization and clarification of implicit requirements through reinforcement learning. During task execution, the main LLM decomposes a user request into a dependency-aware task graph. For nodes that may involve personalized requirements, PersonaAgent retrieves relevant memories, asks the user when information is insufficient, and injects the resulting personalized constraints before the main LLM executes the node.

Persona Memory AutoPersona further extracts user preferences and personalized assistance strategies from Trajectory Memory, Workspace Memory, and task feedback to form Persona Memory. Each persona item is represented as

$$p_l = (k_l,\ g_l,\ \pi_l,\ \mathcal{X}_l)\,, \tag{6}$$

where k_l denotes the applicable task or task type, g_l denotes a user preference, constraint, or personalization objective, π_l denotes the corresponding reusable assistance strategy, and $\mathcal{X}_l$ denotes the related historical interactions and task evidence. Persona Memory is extracted as

$$\Delta\mathcal{M}_P^{(j)} = \mathcal{F}_P\left(\tau_j, \Delta\mathcal{M}_W^{(j)}, f_j\right). \tag{7}$$

Persona Memory stores not only preferences explicitly stated by the user but also user-specific assistance patterns summarized from historical tasks and their feedback. For example, in a paper-recommendation task, it may record that the user prefers open-source and readily reproducible work and derive an assistance strategy that proactively checks code links, reproduction difficulty, and experimental resources.

Based on new task histories and user feedback, AutoPersona adds, deletes, or updates entries in Persona Memory:

$$\mathrm{op} = \mathcal{F}_{\mathrm{op}}(p_l, p, f_j), \qquad \mathrm{op} \in \{\textsc{Add}, \textsc{Delete}, \textsc{Update}\}. \tag{8}$$

New user preferences or assistance strategies are added to Persona Memory; entries that become invalid or are contradicted by subsequent behavior are deleted; and existing entries are updated when their preference, task scope, or assistance strategy changes.

Through this process, AutoPersona organizes user preferences, applicable tasks, assistance strategies, and their historical evidence into a continuously updated Persona Memory. The agent can therefore adapt its future personalized assistance according to user behavior and task feedback.

PersonaAgent Training

AutoPersona trains PersonaAgent using supervised fine-tuning followed by reinforcement learning. Supervised fine-tuning teaches it to retrieve personalized memories and generate responses, while reinforcement learning jointly optimizes memory summarization and clarification of implicit requirements.

Supervised Fine-Tuning PersonaAgent is first supervised-fine-tuned to retrieve relevant memories given a user query and to generate structured personalized guidance from the

retrieved results:

$$\mathcal{L}_{\mathrm{SFT}}(\theta) = -\log P_\theta\left(\mathcal{R}^* \mid q, \mathcal{M}\right) - \log P_\theta\left(g^* \mid q, \mathcal{R}^*\right), \tag{9}$$

where $\mathcal{M}$ is the structured memory bank, $\mathcal{R}^*$ is the target memory relevant to the current query, and g^* is the target personalized-guidance output, including the memory summary, personalized constraints, and, when necessary, a clarification decision or question. It is not the final task response, which is generated by the main LLM.

Reinforcement Learning Starting from the SFT initialization, AutoPersona applies reinforcement learning to jointly optimize personalized memory-point summarization and clarification of implicit requirements. For the same task, the policy samples G candidate outputs d_i, where $\hat{A}_i$ denotes the group-normalized advantage computed from task-level rewards.

The reward is computed directly from PersonaAgent's output behavior. The agent receives $+1$ for every required personalized memory point that it correctly summarizes. A correct and necessary clarification question receives $+\lambda_{\mathrm{ask}}$, where λ_{ask} is the clarification reward weight. Omitting a required memory point, summarizing incorrect information, asking an incorrect or unnecessary question, and producing an irrelevant or non-critical memory point each receive a penalty of -1:

$$R(d) = N_{\mathrm{correct}}(d) + \lambda_{\mathrm{ask}} N_{\mathrm{ask}}(d) - N_{\mathrm{error}}(d) - N_{\mathrm{irrelevant}}(d). \tag{10}$$

Here, N_{correct} is the number of required personalized memory points summarized correctly, N_{ask} is the number of correct and necessary clarification questions, N_{error} counts omissions, incorrect summaries, and incorrect or unnecessary questions, and $N_{\mathrm{irrelevant}}$ counts additional irrelevant or non-critical memory points. We treat λ_{ask} as a hyperparameter and set it to 0.4 based on the analysis in Figure 3. PersonaAgent is updated with the GRPO objective. Let x denote the input context and $d_i = (d_{i,1}, \ldots, d_{i,|d_i|})$ denote the i-th sampled output. The token-level importance ratio is $\rho_{i,t}(\theta) = \pi_\theta(d_{i,t} \mid x, d_{i,<t}) / \pi_{\mathrm{old}}(d_{i,t} \mid x, d_{i,<t})$, where π_{old} is the policy used to sample the output group. Let $\bar{\rho}_{i,t} = \mathrm{clip}(\rho_{i,t}, 1-\epsilon_{\mathrm{clip}}, 1+\epsilon_{\mathrm{clip}})$. PersonaAgent is updated with

$$\mathcal{J}_{\mathrm{RL}}(\theta) = \mathbb{E}\Bigg[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|d_i|}\sum_{t=1}^{|d_i|}\min\Big(\rho_{i,t}\hat{A}_i, \bar{\rho}_{i,t}\hat{A}_i\Big) - \beta\mathbb{D}_{\mathrm{KL}}\left[\pi_\theta \| \pi_{\mathrm{ref}}\right]\Bigg]. \tag{11}$$

Here, π_{ref} is a fixed reference policy used only for the KL regularization term. Through this training, PersonaAgent learns to summarize the personalized memory points needed by a task and to proactively ask the user when existing memory cannot resolve an implicit requirement.

Personalized DAG Execution

Given a user query q, the main LLM first decomposes it into a task DAG with dependency relations:

$$\mathcal{G} = (V, E) = \mathrm{Decompose}(q), \tag{12}$$

where each node $v_i \in V$ denotes a subtask and each edge $(v_i, v_j) \in E$ indicates that v_j depends on v_i. The main LLM executes the nodes in topological order.

For the current node v_i, PersonaAgent first retrieves relevant personalized memories from the structured memory bank $\mathcal{M}$:

$$\mathcal{R}_i = \mathrm{Retrieve}(\mathcal{M}, v_i). \tag{13}$$

If the retrieved memories are insufficient to complete an implicit requirement that affects node execution, PersonaAgent asks the user a targeted clarification question and obtains supplementary information a_i:

$$a_i = \mathrm{Clarify}(v_i, \mathcal{R}_i). \tag{14}$$

No clarification is needed when the existing memories are sufficient. PersonaAgent then personalizes the current node using the retrieved memories and, when necessary, the user's answer:

$$v_i^+ = \mathrm{Personalize}(v_i, \mathcal{R}_i, a_i), \tag{15}$$

where $a_i = \varnothing$ indicates that the current node does not require clarification. Node refinement may include user preferences, workspace conditions, and other personalized constraints. Finally, the main LLM completes the current subtask using the personalized node and the outputs of its predecessors:

$$o_i = \mathcal{L}_{\mathrm{main}}\left(v_i^+, \{o_j \mid v_j \in \mathrm{pre}(v_i)\}\right). \tag{16}$$

Thus, for every task node that may involve personalized requirements, AutoPersona follows the process of memory retrieval, clarification when necessary, node refinement, and task execution. Personalized constraints are therefore injected continuously throughout DAG execution rather than appended only once to the original query.

After all nodes have been executed, the main LLM generates the final response from the node outputs:

$$r^+ = \mathcal{F}_{\mathrm{resp}}\left(\{o_i\}_{v_i \in V}\right). \tag{17}$$

Experiments

We evaluate AutoPersona on personalized-memory task benchmarks. The experiments focus on two questions: whether AutoPersona improves personalized retrieval, personalized responses, and proactive task completion; and whether AutoPersona reduces the token and end-to-end time overhead of personalization. Additional experimental details and the evaluation protocol are provided in Appendix A.

Experimental Setup

Evaluation Benchmarks. **Persona2Web** (Kim, Lee, and Lee 2026) evaluates whether an agent can recover website choices and user-specific constraints, such as size and budget, from user history and use them to complete queries. We report Website Exact Match (Web EM), the exact-match rate between the predicted and gold websites, and Preference Recall (Pref R), the per-task recall of annotated preference values macro-averaged across tasks. **PersonaMem-v2** (Jiang et al. 2025) evaluates whether a model can infer implicit and evolving personalized memories from long-term conversations and generate responses and choices aligned with user

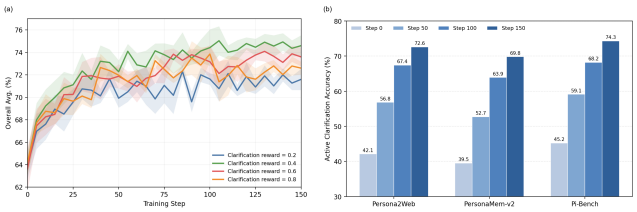

Figure 3: Clarification hyperparameter analysis. (a) Overall average performance under different clarification-reward weights. (b) Proactive clarification-decision accuracy.

preferences. Π**-Bench** (Zhang et al. 2026a) evaluates personalized behavior of proactive personal assistants in long-horizon workflows containing hidden intents and task dependencies.

Baselines and Executors. We compare AutoPersona with six memory methods: Base+RAG, which uses generic retrieval-augmented generation (Lewis et al. 2020); A-MEM, which constructs an agentic memory network through dynamic memory linking and organization (Xu et al. 2026); Mem0 (Entity), which builds structured memory from entities and their relations (Chhikara et al. 2025); LangMem, which supports long-term interaction through memory extraction, consolidation, and retrieval (LangChain 2025); LightMem, which uses lightweight memory compression and retrieval (Fang et al. 2025); and PersonaMem-v2, which uses reinforcement learning to infer implicit user preferences from long-term interactions and iteratively maintains user memory for personalized responses (Jiang et al. 2025).

We use Qwen3-8B (Yang et al. 2025) as the PersonaAgent. For Persona2Web and PersonaMem-v2, Qwen3-8B is also used as the task-execution model. For Π-Bench, we use GPT-5-mini (OpenAI 2025a) as the API executor. All methods use the same model within each evaluation setting. GPT-5.2-Codex (OpenAI 2025b) is used as the judge model.

Evaluation Metrics. For Persona2Web, we report Web EM, Pref R, input **Tokens**, and execution time. For PersonaMem-v2, we report multiple-choice accuracy, open-ended accuracy, input **Tokens**, and execution time. For Π-Bench, we report **Checklist**, **Proactiveness**, input **Tokens**, and execution time. Token denotes the average number of input tokens consumed per task, including retrieved memory context and model inputs during task execution. Checklist measures satisfaction of explicit task criteria, whereas Proactiveness measures coverage of hidden requirements.

Implementation Details. Detailed implementation settings are provided in Appendix B.

PersonaAgent is initialized through SFT and subsequently optimized by reinforcement learning with a joint reward: it summarizes the required personalized memory points and asks targeted clarification questions when information is missing. All methods use the same task order, tool permissions, and evaluation split.

Main Results

Tables 1 and 2 report the main results. AutoPersona achieves the best performance on all three benchmarks. It outperforms the strongest baseline by 10.0 points on Persona2Web, 9.5 points on PersonaMem-v2, and 6.4 points on Π-Bench in the corresponding aggregate metric. It also achieves the fastest end-to-end execution time on all three benchmarks.

Personalized Retrieval and Reasoning. On Persona2Web, AutoPersona achieves a P-Avg score of 78.8 with Qwen3-8B, outperforming the strongest baseline, PersonaMem-v2, by 10.0 points. The simultaneous improvements in Web EM and Pref R show that structured memory helps recover the intended website and evidence-supported user constraints from history. On PersonaMem-v2, AutoPersona reaches an Overall Accuracy of 56.6, exceeding PersonaMem-v2 by 9.5 points. Existing approaches commonly focus on preserving dialogue fragments, constructing local memory associations, or performing similarity retrieval. AutoPersona further separates historical evidence, workspace state, and personalized user information, and PersonaAgent determines which memories apply to the current task. When the existing memory supports reliable inference, the system directly produces node-level personalized refinements, allowing personalized constraints to remain active throughout task execution and improving both concrete preference recovery and implicit persona reasoning.

Proactive Personalized Assistance. Table 2 reports the Π-Bench results. AutoPersona obtains 67.2 Checklist, 71.3 Proactiveness, and 69.3 Overall, improving over the strongest baseline in each metric by 7.9, 3.9, and 6.4 points, respectively. The Checklist gain indicates that AutoPersona more completely satisfies explicit delivery criteria. Workspace Memory continually preserves application state, task artifacts, and cross-task dependencies, enabling subsequent tasks to continue from the established workspace state instead of relying only on local context. The Proactiveness gain shows that Persona Memory and targeted clarification identify unstated user requirements during execution and obtain necessary information when existing evidence is insufficient, reducing rework or failure caused by missing implicit conditions.

Token and Time Efficiency. AutoPersona exhibits strong online efficiency across the benchmarks. It uses 301.4K, 3.187K, and 131.2K input tokens on Persona2Web, PersonaMem-v2, and Π-Bench, respectively, which is the second-lowest token consumption on each benchmark. Its corresponding execution times are 44.2, 49.3, and 199.7 seconds, the fastest among all methods. Relative to the next-fastest method, these values reduce end-to-end time by 23.3%, 19.4%, and 12.9%, respectively.

This reduction is consistent with the design of lightweight memory units and the three-layer memory bank's separation of different memory emphases. This organization lets the system retrieve memory units required by the current task without repeatedly injecting complete historical trajectories. Meanwhile, PersonaAgent supplies only the person-

Method	Persona2Web					PersonaMem-v2				
	Web EM	Pref R	P-Avg	Token (K)	Time (s)	MCQ Acc.	Open-Ended Acc.	Overall Acc.	Token (K)	Time (s)
Executor: Qwen3-8B										
Base+RAG	38.2	52.8	45.5	**186.4**	57.6	20.2	17.9	19.1	**2.638**	61.2
A-Mem	40.4	54.3	47.4	415.9	72.9	24.3	22.1	23.2	3.942	76.1
Mem0 (entity)	59.6	69.5	64.6	542.2	69.3	37.3	38.2	37.8	4.159	73.8
LangMem	56.4	60.4	58.4	533.1	67.7	33.7	35.0	34.4	3.911	70.6
LightMem	34.7	54.2	44.5	512.3	62.1	42.1	42.4	42.3	3.929	66.4
PersonaMem-v2	60.6	76.9	68.8	376.8	75.5	47.7	46.5	47.1	3.556	78.2
AutoPersona	**71.7**	**85.8**	**78.8**	301.4	**44.2**	**55.9**	**57.2**	**56.6**	3.187	**49.3**

P-Avg is the arithmetic mean of Web EM and Pref R, and Overall Acc. is the arithmetic mean of MCQ and open-ended accuracy. Token is the average number of input tokens consumed per task, including retrieved memory context and model inputs during task execution, and is reported in thousands. Time is the average end-to-end wall-clock time per task in seconds.

Table 1: Main results on Persona2Web and PersonaMem-v2. Best results are shown in bold and second-best results are underlined. Lower Token and Time are better.

Method	Check.	Proact.	Avg.	Tok. (K)	Time (s)
Executor: GPT-5-mini					
Base+RAG	54.3	65.9	60.1	**124.7**	239.5
A-Mem	49.1	55.6	52.4	149.7	243.2
Mem0 (entity)	50.5	67.4	59.0	138.5	242.1
LangMem	51.6	57.3	54.5	147.0	277.3
LightMem	59.3	66.5	62.9	187.8	229.4
PersonaMem-v2	56.4	65.1	60.8	192.9	303.9
AutoPersona	**67.2**	**71.3**	**69.3**	131.2	**199.7**

Overall is the arithmetic mean of Checklist and Proactiveness. Token is the average number of input tokens consumed per task, including retrieved memory context and model inputs during task execution, and is reported in thousands. Time is the average end-to-end wall-clock time per task in seconds.

Table 2: Main results on Π-Bench. Best results are shown in bold and second-best results are underlined. Lower Token and Time are better.

Variant	P2W	PM-v2	Π-Bench	Overall
Memory-layer additions				
Trajectory Memory	42.8	24.7	51.9	39.8
+ Workspace Memory	56.9	36.5	57.8	50.4
+ Persona Memory	67.9	46.4	60.3	58.2
Training-stage additions				
Only SFT	72.6	50.8	64.7	62.7
Full AutoPersona	**78.8**	**56.6**	**69.3**	**68.2**

P2W Avg. is Persona2Web P-Avg; PM-v2 Avg. is PersonaMem-v2 Overall Accuracy; Π-Bench Avg. is the arithmetic mean of Checklist and Proactiveness. Overall Avg. is the macro-average over the three benchmark averages.

Table 3: Incremental component study of the three memory layers and training stages. Best results are shown in bold and second-best results are underlined. Higher is better for all columns.

alized constraints needed by the current task node, limiting additional context and processing overhead. AutoPersona therefore improves task effectiveness while maintaining competitive online token consumption and low execution time.

Ablation Study

Contributions of the Memory Layers. Table 3 incrementally adds the memory layers and training stages. Using Trajectory Memory alone obtains an Overall Avg. of 39.8. Adding Workspace Memory raises it to 50.4, a gain of 10.6 points, demonstrating the importance of persistent environment conditions and task continuity. Adding Persona Memory further raises the score to 58.2, a gain of 7.8 points, showing the value of structured user preferences, constraints, and personalized assistance strategies. Together, the three layers provide traceable evidence, workspace context, and user-specific guidance.

Contributions of the Training Stages. The configuration trained only with SFT improves Overall Avg. from 58.2 to 62.7, indicating that stable memory retrieval, personalized information summarization, and clarification behavior benefit from supervised demonstrations. The full configuration, which additionally applies RL, reaches 68.2. This further gain confirms that task-level rewards improve memory summarization and clarification policies beyond supervised training alone.

Clarification-Reward Weight. We compare clarification-reward weights of 0.2, 0.4, 0.6, and 0.8 over 150 training steps. As shown in Figure 3, a weight of 0.4 achieves the highest and most stable Overall Avg. This indicates that an appropriate clarification reward encourages PersonaAgent to proactively complete missing information while avoiding excessive or unnecessary clarification. We therefore use 0.4 as the clarification-reward weight in subsequent experiments.

Under this reward weight, we further compare clarification-decision accuracy after 0, 50, 100, and 150 training steps. This metric measures whether PersonaAgent correctly initiates clarification when information is insufficient and avoids unnecessary questions when information is

sufficient. Performance consistently improves across all three datasets, indicating that reinforcement learning progressively strengthens PersonaAgent's ability to identify information gaps and make clarification decisions.

Conclusion

We presented AutoPersona, a proactive personalized memory framework that uses structured personalized memory to resolve implicit task requirements. AutoPersona organizes task histories, workspace states, and user persona through a three-layer memory architecture and trains PersonaAgent to perform personalized memory retrieval and clarification for implicit requirements, allowing memory to support node-level personalization throughout task execution. Experimental results on Persona2Web, PersonaMem-v2, and Π-Bench demonstrate that AutoPersona improves task performance, personalization consistency, and execution efficiency.

References

Anthropic. 2025. Claude Code: Best Practices for Agentic Coding. https://www.anthropic.com/engineering/claude-code-best-practices. Published April 18, 2025.

Chen, B.; Shu, C.; Shareghi, E.; Collier, N.; Narasimhan, K.; and Yao, S. 2023. FireAct: Toward Language Agent Fine-tuning. *arXiv preprint arXiv:2310.05915*.

Chen, W.; Su, Y.; Zuo, J.; Yang, C.; Yuan, C.; Chan, C.-M.; Yu, H.; Lu, Y.; Hung, Y.-H.; Qian, C.; et al. 2024. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *International Conference on Learning Representations*, volume 2024, 20094–20136.

Chhikara, P.; Khant, D.; Aryan, S.; Singh, T.; and Yadav, D. 2025. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*.

Fang, J.; Deng, X.; Xu, H.; Jiang, Z.; Tang, Y.; Xu, Z.; Deng, S.; Yao, Y.; Wang, M.; Qiao, S.; Chen, H.; and Zhang, N. 2025. LightMem: Lightweight and Efficient Memory-Augmented Generation. *arXiv preprint arXiv:2510.18866*.

Gao, H.; Sun, Z.; Min, E.; Cai, H.; Wang, S.; Yin, D.; and Chen, X. 2025. Solving the Granularity Mismatch: Hierarchical Preference Learning for Long-Horizon LLM Agents. *arXiv preprint arXiv:2510.03253*.

Garg, S.; Nitin, V.; and Huang, Y. 2026. Terminus-4B: Can a Smaller Model Replace Frontier LLMs at Agentic Execution Tasks? *arXiv preprint arXiv:2605.03195*.

He, Z.; Wang, Y.; Zhi, C.; Hu, Y.; Chen, T.-P.; Yin, L.; Chen, Z.; Wu, T. A.; Ouyang, S.; Wang, Z.; et al. 2026. Memoryarena: Benchmarking agent memory in interdependent multi-session agentic tasks. *arXiv preprint arXiv:2602.16313*.

Huang, T.; Chen, S.; Chen, M.; May, J.; Yang, L.; Wan, M.; and Zhou, P. 2025. Teaching Language Models to Gather Information Proactively. *arXiv preprint arXiv:2507.21389*.

Huang, Z.; Dai, Q.; Wu, G.; Wu, X.; Li, X.; Ge, T.; Wang, W.; and Jin, Q. 2026a. Mem-pal: Towards memory-based personalized dialogue assistants for long-term user-agent interaction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, 31229–31237.

Huang, Z.; Tian, M.; Wang, X.; Xu, J.; Guo, Z.; Qian, Q.; Song, K.; Yuan, J.; Lv, C.; and Zheng, X. 2026b. Controllable Memory Usage: Balancing Anchoring and Innovation in Long-Term Human–Agent Interaction. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*.

Jiang, B.; Yuan, Y.; Shen, M.; Hao, Z.; Xu, Z.; Chen, Z.; Liu, Z.; Vijjini, A. R.; He, J.; Yu, H.; et al. 2025. Personamem-v2: Towards personalized intelligence via learning implicit user personas and agentic memory. *arXiv preprint arXiv:2512.06688*.

Jung, S.; Lee, D.; Lee, S.; Seo, G.; Lee, D.; Ko, B.; Cho, J.; Kim, K.; Kim, E.; and Shin, M. 2025. Diatool-dpo: Multi-turn direct preference optimization for tool-augmented large language models. In *Proceedings of the 26th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, 397–416.

Kim, S.; Lee, S.; and Lee, D. 2026. Persona2Web: Benchmarking Personalized Web Agents for Contextual Reasoning with User History. In *International Conference on Machine Learning*. ArXiv:2602.17003.

LangChain. 2025. LangMem: Long-Term Memory for Language Model Agents. https://langchain-ai.github.io/langmem/. Accessed July 28, 2026.

Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.-t.; Rocktäschel, T.; Riedel, S.; and Kiela, D. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems*, volume 33, 9459–9474.

Li, Z.; Song, S.; Wang, H.; Niu, S.; Chen, D.; Yang, J.; Xi, C.; Lai, H.; Zhao, J.; Wang, Y.; et al. 2025. MemOS: An Operating System for Memory-Augmented Generation in Large Language Models. *arXiv preprint arXiv:2505.22101*.

Lin, Z.; Li, J.; Xu, D.; Pan, S.; Shi, Y.; Liu, Y.; Min, Y.; and Yao, Y. 2026. Mobile GUI Agent Privacy Personalization with Trajectory Induced Preference Optimization. *arXiv preprint arXiv:2604.11259*.

Maharana, A.; Lee, D.-H.; Tulyakov, S.; Bansal, M.; Barbieri, F.; and Fang, Y. 2024. Evaluating Very Long-Term Conversational Memory of LLM Agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 13851–13870.

OpenAI. 2025a. Introducing GPT-5 for Developers. https://openai.com/index/introducing-gpt-5-for-developers/. Published August 7, 2025.

OpenAI. 2025b. Introducing GPT-5.2-Codex. https://openai.com/index/introducing-gpt-5-2-codex/. Published December 18, 2025.

OpenClaw Contributors. 2026. OpenClaw: Your Own Personal AI Assistant. https://github.com/openclaw/openclaw. Accessed July 28, 2026.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. 2022. Training Language Models to Follow Instructions with Human Feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744.

Packer, C.; Wooders, S.; Lin, K.; Fang, V.; Patil, S. G.; Stoica, I.; and Gonzalez, J. E. 2023. MemGPT: Towards LLMs as Operating Systems. *arXiv preprint arXiv:2310.08560*.

Park, J. S.; O'Brien, J.; Cai, C. J.; Morris, M. R.; Liang, P.; and Bernstein, M. S. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, 1–22.

Ruan, J.; Xu, Z.; Peng, Y.; Ren, F.; Yu, Z.; Liang, X.; Xiang, J.; Chen, Y.; Liu, B.; Wu, C.; et al. 2026. Aorchestra: Automating sub-agent creation for agentic orchestration. *arXiv preprint arXiv:2602.03786*.

Shao, Z.; Wang, P.; Zhu, Q.; Xu, R.; Song, J.; Bi, X.; Zhang, H.; Zhang, M.; Li, Y. K.; Wu, Y.; and Guo, D. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300*.

Shen, Y.; Li, K.; Zhou, W.; and Hu, S. 2026. Mem2ActBench: A Benchmark for Evaluating Long-Term Memory Utilization in Task-Oriented Autonomous Agents. *arXiv preprint arXiv:2601.19935*.

Sun, H.; Zhang, Z.; and Zeng, S. 2026. Preference-Aware Memory Update for Long-Term LLM Agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, 783–793.

Tan, Z.; Yan, J.; Hsu, I.-H.; Han, R.; Wang, Z.; Le, L.; Song, Y.; Chen, Y.; Palangi, H.; Lee, G.; et al. 2025. In Prospect and Retrospect: Reflective Memory Management for Long-term Personalized Dialogue Agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 8416–8439.

Wang, S.; Liu, C.; Loo, G.; Zheng, L.; Wei, K.; Zeng, X.; Zhang, J.; and Tian, Y. 2026. Me-Agent: A Personalized Mobile Agent with Two-Level User Habit Learning for Enhanced Interaction. *arXiv preprint arXiv:2601.20162*.

Wang, Y.; Takanobu, R.; Liang, Z.; Mao, Y.; Hu, Y.; McAuley, J.; and Wu, X. 2025. Mem-α: Learning Memory Construction via Reinforcement Learning. *arXiv preprint arXiv:2509.25911*.

Wu, D.; Wang, H.; Yu, W.; Zhang, Y.; Chang, K.-W.; and Yu, D. 2024. LongMemEval: Benchmarking Chat Assistants on Long-Term Interactive Memory. *arXiv preprint arXiv:2410.10813*.

Xu, W.; Liang, Z.; Mei, K.; Gao, H.; Tan, J.; and Zhang, Y. 2026. A-mem: Agentic memory for llm agents. *Advances in Neural Information Processing Systems*, 38: 17577–17604.

Yan, S.; Yang, X.; Huang, Z.; Nie, E.; Ding, Z.; Li, Z.; Ma, X.; Bi, J.; Kersting, K.; Pan, J. Z.; Schütze, H.; Tresp, V.; and Ma, Y. 2025. Memory-R1: Enhancing Large Language Model Agents to Manage and Utilize Memories via Reinforcement Learning. *arXiv preprint arXiv:2508.19828*.

Yang, A.; Li, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Gao, C.; Huang, C.; Lv, C.; et al. 2025. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*.

Yu, Y.; Yao, L.; Xie, Y.; Tan, Q.; Feng, J.; Li, Y.; and Wu, L. 2026. Agentic Memory: Learning Unified Long-Term and Short-Term Memory Management for Large Language Model Agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, 21457–21483.

Zhang, H.; Xu, L.; Wang, Z.; Gui, R.; Zhang, S.; Lei, H.; He, Z.; He, B.; Qin, C.; Zhu, T.; Qu, X.; Yang, Y.; Cheng, Y.; and Li, Y. 2026a. π-Bench: Evaluating Proactive Personal Assistant Agents in Long-Horizon Workflows. *arXiv preprint arXiv:2605.14678*.

Zhang, J.; Xiang, J.; Yu, Z.; Teng, F.; Chen, X.; Chen, J.; Zhuge, M.; Cheng, X.; Hong, S.; Wang, J.; et al. 2025. Aflow: Automating agentic workflow generation. In *International Conference on Learning Representations*, volume 2025, 34040–34077.

Zhang, Q.; Huang, S.; Liu, C.; Yang, S.; Zhao, J.; Wang, H.; and Xie, P. 2026b. DeltaMem: Towards Agentic Memory Management via Reinforcement Learning. *arXiv preprint arXiv:2604.01560*.

Zhang, W.; Zhang, X.; Zhang, C.; Yang, L.; Shang, J.; Wei, Z.; Zou, H. P.; Huang, Z.; Wang, Z.; Gao, Y.; et al. 2026c. PersonaAgent: Bridging Memory and Action for Personalized LLM Agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, 26421–26439.

Zhong, W.; Guo, L.; Gao, Q.; Ye, H.; and Wang, Y. 2024. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, 19724–19731.

A Experimental Details and Supplementary Experiments

A.1 Experimental Setup

Training Data We use only the official PersonaMem-v2 text training split to construct the PersonaAgent training data. The text version contains 18,549 training queries, 2,061 validation queries, and 5,000 benchmark queries, with disjoint `persona_id`s across splits. Training samples are generated only from the training split; the validation split is used for checkpoint and hyperparameter selection, and the benchmark split is excluded from both training and model selection. Persona2Web and Π-Bench are used only for cross-dataset evaluation.

Each PersonaMem-v2 source example provides a user query, a target answer, a target preference, a related conversation snippet, and the full dialogue history. We match the related snippet to its source turn and construct evidence-sufficient, key-evidence-masked, and explicitly specified variants. In particular, 30% of the training examples have their key preference evidence masked and are converted into clarification examples. This construction teaches PersonaAgent to retrieve and answer when the history is sufficient, ask a targeted question when a task-critical condition is missing, and avoid asking again when the current request already states that condition.

SFT Training PersonaAgent uses Qwen3-8B as its backbone and is first supervised fine-tuned with LoRA in LLaMA-Factory. The SFT set contains 1,000 examples: 400 memory-search examples constructed by GPT-5.2-Codex from PersonaMem-v2 training queries and histories, 300 direct-answer examples, and 300 clarification examples. The three subsets respectively supervise the `<search>`, `<final>`, and `<clarify>` actions. Table A1 reports the configuration.

Setting	Value
PersonaAgent backbone	Qwen/Qwen3-8B
SFT framework	LLaMA-Factory
Fine-tuning method	LoRA
LoRA rank / alpha	8 / 16
Per-device batch size	1
Gradient accumulation	8
Learning rate	1×10^{-6}
Search / answer / clarification	400 / 300 / 300
Training examples	1,000
Search-data constructor	GPT-5.2-Codex
Epochs	2
Precision	BF16
Checkpoint interval	50 steps

Table A1: LLaMA-Factory LoRA SFT configuration for PersonaAgent.

RL Training The RL stage initializes PersonaAgent from the SFT checkpoint and uses GRPO in VERL 0.7.0, with vLLM for rollout generation. Its 1,600 examples are all constructed from the PersonaMem-v2 training split, and 30% have their key evidence masked to require clarification. Whereas SFT establishes the basic search, answer, and clarification actions, RL adjusts memory selection and clarification timing using task-level rewards. Table A2 gives the RL configuration.

Setting	Value
RL framework / algorithm	VERL 0.7.0 / GRPO
Hardware	8× NVIDIA H100-80GB
Rollout engine	vLLM
GRPO group size	8
Rollouts per query	8
Learning rate	1×10^{-6}
Clarification weight λ_{ask}	0.4
Training examples	1,600
Training steps	150
Evaluation / checkpoint interval	5 / 20 steps
Random seeds	42, 43, 44

Table A2: VERL GRPO configuration for PersonaAgent.

For a candidate output d, the reward is

$$R(d) = N_{\text{correct}}(d) + \lambda_{\text{ask}} N_{\text{ask}}(d) - N_{\text{error}}(d) - \lambda_{\text{irr}} N_{\text{irrelevant}}(d). \tag{18}$$

N_{correct} counts necessary memory points whose values are correct, N_{ask} counts useful questions asked when a key condition is truly missing, N_{error} covers omitted or incorrect memory, missed or incorrect clarification, and unnecessary questions, and $N_{\text{irrelevant}}$ counts extra memory unrelated to the current task node. We use $\lambda_{\text{ask}} = 0.4$.

Task Execution and Evaluation Persona2Web and PersonaMem-v2 use Qwen3-8B as the task executor, whereas Π-Bench uses GPT-5-mini. PersonaAgent uses Qwen3-8B on all three benchmarks. Within each benchmark, all methods share the executor, task order, history, tool permissions, maximum interaction steps, and evaluation split. PersonaMem-v2 open-ended questions are the only results evaluated with an LLM-as-Judge, for which we use GPT-5.2-Codex. PersonaMem-v2 multiple-choice questions, Persona2Web, and Π-Bench are evaluated using their gold labels, rules, or official procedures.

End-to-end time covers online memory retrieval, PersonaAgent and executor inference, tool calls, clarification, and retries, but excludes offline indexing and judge scoring. Token usage includes all online input tokens consumed by the executor and PersonaAgent. Method names are hidden and candidate outputs are randomized during judge evaluation.

A.2 Datasets and Memory Context

Persona2Web Persona2Web contains 50 user profiles, each accompanied by approximately one year of browsing history. Its 150 base tasks are instantiated at three levels of preference explicitness, yielding 450 queries. We extract memory from browsing and purchase records that precede each test task. We report Web Exact Match, Preference Recall, their arithmetic mean P-Avg, average input tokens, and end-to-end execution time.

PersonaMem-v2 PersonaMem-v2 covers 1,000 personas and 335 query topics, with histories up to 128K tokens. We use its text-32K version and extract memory from the dialogue history for the current personalized question. MCQ Accuracy is computed from the gold option, Open-Ended Accuracy is judged by GPT-5.2-Codex, and Overall Accuracy is their arithmetic mean.

Π-Bench Π-Bench contains 100 multi-turn tasks under five occupational personas, with 20 tasks per persona. We extract memory from history-dependent trajectories and persistent workspace states to complete subsequent dependent tasks. We report the official Checklist, Proactiveness, and their mean Overall score.

A.3 Baseline Settings

We compare Base+RAG, A-Mem, Mem0, LangMem, LightMem, and PersonaMem-v2. These methods respectively represent retrieval over raw history, dynamically linked memory, entity-relationship memory, long-term memory extraction and consolidation, lightweight compression and retrieval, and RL-based preference learning from long interactions. Each baseline follows its public memory-write, organization, and retrieval procedure. To control for access to proactive interaction, we equip every method with the same prompt-based clarification mechanism: a targeted question may be asked only when the current request and retrieved results cannot determine a personalized condition that materially changes task execution. The prompt, observable input, and interaction limit are identical for all methods.

A.4 Supplementary Experiments

Detailed Ablation Study The ablation progressively varies memory structure, node-level execution, and training. *Only T* retains only Trajectory Memory; *Only W+T* adds Workspace Memory; *Only W+P* uses Workspace and Persona Memory without the original trajectory; *Flat Memory* stores the same history, workspace state, and preferences in one unified index; *T+W+P* uses all three layers without DAG-based node personalization; *+DAG* adds task decomposition, node-level retrieval, and personalized injection; *+SFT* initializes PersonaAgent with LoRA SFT; *+RL ($\lambda_{\text{ask}} = 0$)* trains memory selection without rewarding correct clarification; and the full model jointly optimizes memory selection and clarification with $\lambda_{\text{ask}} = 0.4$. All variants use the same backbone, data split, history, tools, interaction budget, and memory-token budget. Flat and structured memory receive exactly the same historical content.

Statistical Experiment We independently run PersonaMem-v2 and AutoPersona with random seeds 42, 43, and 44 and report the mean and standard deviation on all three benchmarks. Figure A1 shows that AutoPersona obtains 78.8, 56.6, and 69.3 on Persona2Web, PersonaMem-v2, and Π-Bench, respectively, compared with 68.8, 47.1, and 60.8 for PersonaMem-v2. The consistently small error bars indicate that the gains remain stable across seeds.

Human Evaluation of Judge Agreement We sample 200 PersonaMem-v2 examples that do not require clarification and 200 that do, for a total of 400 examples. Human annotators independently apply the same scoring criterion used by GPT-5.2-Codex. Method names are hidden and candidate outputs are randomly ordered. Table A4 reports the mean and standard deviation of agreement between the automatic and human scores.

Clarification Stability We evaluate clarification stability on all 100 multi-turn Π-Bench tasks, comprising five personas with 20 tasks each. All tasks contain hidden intents. Because an intent may either be inferred and directly satisfied or obtained through a targeted question, we use the official hidden-intent and targeted-followup judgments. Effective clarification coverage is the fraction of hidden intents obtained by targeted follow-up. A question unrelated to any remaining unsatisfied intent is unnecessary, and an intent that is neither directly satisfied nor targeted by a question is missed.

We compare $\lambda_{\text{ask}} \in \{0, 0.2, 0.4, 0.6, 0.8\}$ using seeds 42, 43, and 44. Figure A2 groups the five weights for each metric; bars show means and error bars show standard deviations. Increasing the weight from 0 to 0.4 raises Proactiveness from 65.1 to 73.5, reduces the unnecessary-clarification rate from 13.8% to 6.9%, and reduces the missed-intent rate from 34.9% to 26.5%. Larger weights further improve effective coverage but substantially increase unnecessary questions and reduce Proactiveness. Thus, $\lambda_{\text{ask}} = 0.4$, selected on the PersonaMem-v2 validation split, provides the best balance between intent coverage and over-clarification.

B Implementation Details of AutoPersona

AutoPersona decouples personalized memory management from general task execution. It maintains Trajectory, Workspace, and Persona Memory. Before a task node is executed, an independent PersonaAgent selects relevant memory or asks a clarification question, while the main agent performs task decomposition, tool use, and final result generation.

B.1 Memory Update and Retrieval

Trajectory Memory Trajectory Memory stores reusable process information from a historical task. Each item contains only four fields:

```
{
  "query": "the user's original task",
  "trace": ["key execution steps"],
  "outcome": "the final task result",
  "insight": "reusable information
    summarized from the task"
}
```

The trace retains only operations that affect the result; routine logs, duplicated tool returns, and irrelevant intermediate steps are omitted.

Workspace Memory Workspace Memory contains three types of information. *Task* records the organization and dependencies of work in the current workspace. For example, a paper-reproduction project may be represented as the DAG "select a paper → prepare the environment and

Variant	P2W Avg.	PM-v2 Avg.	Π-Bench Avg.	Overall Avg.
Only T	42.8	24.7	51.9	39.8
Only W+T	56.9	36.5	57.8	50.4
Only W+P	57.9	38.0	58.5	51.5
Flat Memory	59.2	39.1	59.0	52.4
T+W+P	67.9	46.4	60.3	58.2
+DAG	70.4	48.6	62.0	60.3
+SFT	72.6	50.8	64.7	62.7
+RL ($\lambda_{\text{ask}} = 0$)	75.4	53.7	67.0	65.4
+RL (Full AutoPersona)	**78.8**	**56.6**	**69.3**	**68.2**

Table A3: Detailed ablation of AutoPersona's memory structure, DAG execution, SFT, and RL. P2W Avg. is Persona2Web P-Avg; PM-v2 Avg. is PersonaMem-v2 Overall Accuracy; Π-Bench Avg. is the mean of Checklist and Proactiveness; Overall Avg. is the macro-average of the three benchmark averages.

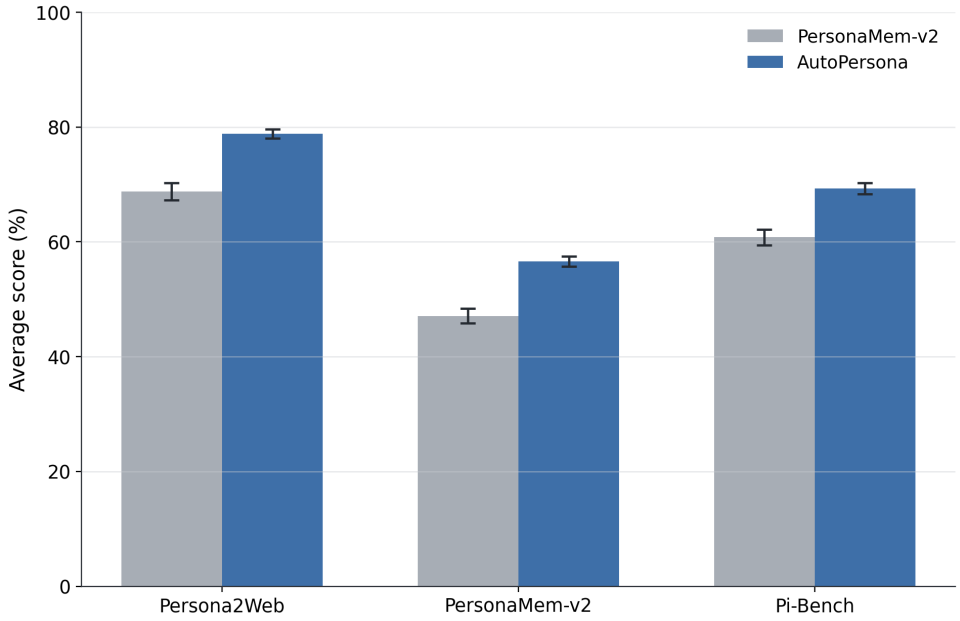

Figure A1: Cross-seed comparison between PersonaMem-v2 and AutoPersona. Bars show the mean of three runs and error bars show the standard deviation.

Sample type	No.	Agreement	Std.
No clarification needed	200	97.33%	0.58%
Clarification required	200	95.83%	0.76%
Overall	400	96.58%	0.55%

Table A4: Agreement between GPT-5.2-Codex and human scores on PersonaMem-v2 examples with different clarification requirements.

data → run SFT → run GRPO → evaluate." *Interaction* records important user or agent operations in the environment. In Persona2Web, this may be a user logging into a shopping site and purchasing a product; in Π-Bench, it may be creating a Todoist project named `Temporary Launch Rehearsal Board - Beacon`, adding a `Tonight Rehearsal` section, and creating its tasks. *State* records files, application states, and task states that remain valid. For example, a Π-Bench research workspace may contain `paper_list.txt` with ICLR 2026 papers and the results of a previous think-with-image paper search.

```
{
  "task": "current task structure and
    dependencies",
  "interaction": "important user
  /agent environment operation",
  "state": "currently valid workspace state"
}
```

Persona Memory Persona Memory stores cross-task preferences and corresponding strategies. Each item contains only a topic, preference, and strategy:

```
{
  "topic": "scientific literature reading",
```

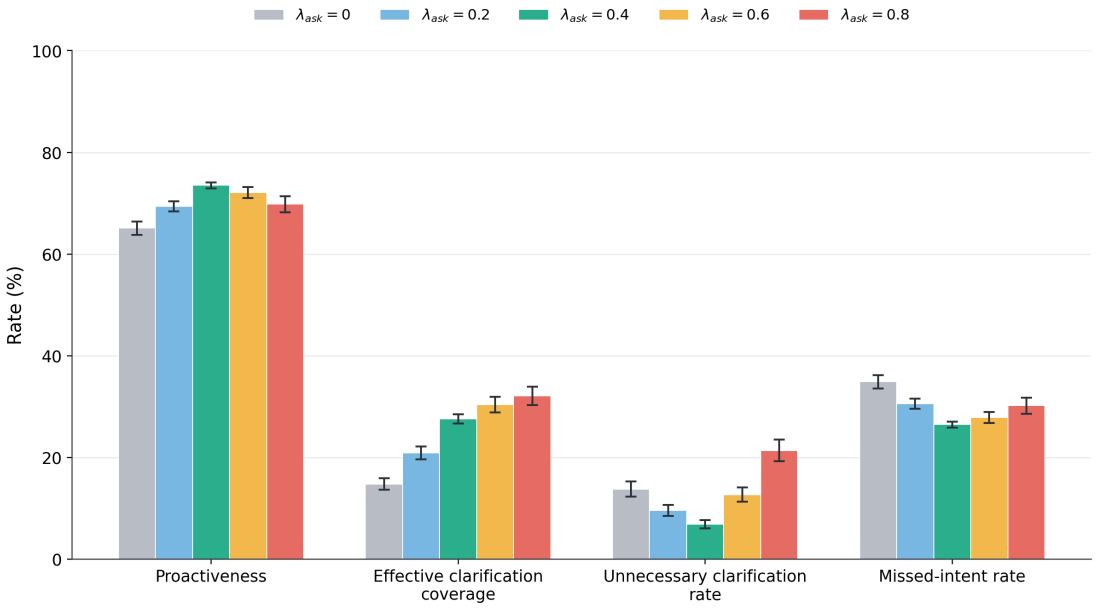

Figure A2: Clarification stability under different reward weights on Π-Bench. Bars show the mean of three runs and error bars show the standard deviation.

```
  "preference": "the user prefers
  reading papers in Chinese",
  "strategy": "translate key English content
     into Chinese before summarizing"
}
```

One-off task requirements are not converted into long-term Persona Memory.

Updating and Retrieving Memory For each update, the system retrieves similar memories by embedding and asks an LLM to decide whether the new information should be added, used to update an existing item, or used to delete an obsolete item. If no mergeable memory is retrieved, a new item is added. For a new task, the system retrieves from all three memory types using the user request, current node, and predecessor results. PersonaAgent keeps only information needed by the current node and asks for clarification instead of guessing when a key condition is missing.

B.2 Execution Flow

Historical Memory Construction After each historical task, the system stores its query, key trace, outcome, and insight in Trajectory Memory, extracts task, interaction, and state for Workspace Memory, and finally extracts reusable topic, preference, and strategy for Persona Memory. Similar items are then retrieved by embedding and resolved by the LLM before the memories are updated.

DAG Decomposition and Node Scheduling Given a user request q, the main agent decomposes it into a directed acyclic graph

$$\mathcal{G} = (V, E) = \text{Decompose}(q). \tag{19}$$

Each node contains an instruction, current requirements, and predecessor dependencies. Nodes are executed in topological order, and ready nodes without dependencies may run in parallel. PersonaAgent does not decompose tasks or call tools; it selects personalized memory for the current node or determines that a clarification is needed.

Node Personalization and Execution PersonaAgent adds selected personalized conditions to the current node without changing its original objective. The main agent then executes the personalized node with its predecessor outputs:

$$o_i = \mathcal{L}_{\text{main}}(v_i^+, \{o_j \mid v_j \in \text{pre}(v_i)\}). \tag{20}$$

The node's key process, result, and summary update the three memory layers. Once all nodes are complete, the main agent aggregates their outputs.